第 4 章: 資料結構

4: Data Structure

在 R 中產生資料和控制運算的實體稱爲 “物件”, 物件包含儲存資料的向量 (vector), 矩陣 (matrix), 陣列 (array), 列表 (Lists), 資料框架 (data frames) 或執行特定運算指令的函式 (function) 等. R 對資料物件的特徵製作統一的資料結構, R 物件的 資料結構 (data structure) 主要由 維度 (dimensionality), 1-D, 2-D, or p-D 等) 組成元素的 模式 (mode) 爲同質性或異質性 (所有元素都是相同的基本模式).

4.1 資料結構 Data Structure

物件的 結構 (structure) 可簡單分成 原型 (atomic) 與 遞迴型 (recursive). 原型物件 (atomic object) 有向量, 矩陣, 陣列等. R 的最基本物件是向量, 向量是由包含相同 模式 (mode) 的元素組成, 矩陣 (marix) 與 陣列 (array) 也由包含相同模式的元素組成, 同一向量, 矩陣 與 陣列 內的元素不可混合, 單一數值 (scalar), 可視爲僅具有單一元素的向量.

遞迴型物件 (recursive object) 有列表 (list), 資料框架 (data frame), 函式 (function) 等. 遞迴型物件含有複雜 (混合) 元素模式, 其他如, `"expression"`, `"name"`, `"symbol"` 等也是遞迴型物件. 使用函式指令 `mode()` 可以用來查看物件的模式. 參

見表 4.1.

表 4.1: R 資料結構

維度 Dimensionality	同質性 Homogeneous	異質性 Heterogeneous
1-D	vector	list
2-D	matrix	data frame
p-D	array	

4.2 原型物件 Atomic Object

R 的最基本物件是向量, 向量 (vector) 是由包含相同模式 (mode) 的元素組成, 但向量 沒有維度 (no dimension) 的分別, 主要的基本模式 (basic mode) 有 `numeric` (`single` 與 `double`), `integer`, `logical`, `complex` 與 `character`, 等. 當 R 顯示向量物件, 若向量長度超過 R Console 螢幕設定出現的字元數時 `[number]` 出現在某特定元素左方時, 表示某特定元素在該向量物件的位置 (position), 例如, `[k]` 表示在在該向量的第 k 個位置之元素.

```
> ## atomic object
> ## vector
> x.vec <- 1:30 # numerical
> x.vec
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
[21] 21 22 23 24 25 26 27 28 29 30
> ## left side with [20]
> ## means x.vec[20] the 20th element
> flavors.vec <- c("chocolate", "vanilla", "strawberry") # character
> flavors.vec
[1] "chocolate"  "vanilla"    "strawberry"
> z.vec <- c(T, F, F) # logic
> z.vec
[1]  TRUE FALSE FALSE
```

矩陣 (matrix) 物件由包含相同的元素組成的 二維 (2-dimension) 資料物件, 也可以將矩陣視爲一個向量具二維結構. R 顯示矩陣物件, `[m, ]` 出現在某特定元素左

方時, 表示某特定元素在該矩陣物件之第 m 列 (row) 的位置; `[ , n]` 出現在某特定元素上方時, 表示某特定元素在該矩陣物件之第 n 欄 (column) 的位置.

```
> ## atomic object
> ## matrix
> x.vec <- c("a", "b", "c", "d", "e", "f")
> x.vec
[1] "a" "b" "c" "d" "e" "f"
> y.mat <- matrix(x.vec, nrow = 2, ncol = 3)
> y.mat
     [,1] [,2] [,3]
[1,] "a"  "c"  "e"
[2,] "b"  "d"  "f"
> #
> y.mat <- matrix(x.vec, nrow = 2, ncol = 3, byrow = T)
> y.mat
     [,1] [,2] [,3]
[1,] "a"  "b"  "c"
[2,] "d"  "e"  "f"
> #
> x.mat <- matrix(1:6, nrow = 2, ncol = 3)
> dimnames(x.mat) <- list(c("A1", "A2"),
                          c("B1", "B2", "B3"))
> x.mat
   B1 B2 B3
A1  1  3  5
A2  2  4  6
> #
> m.dat <- matrix(c(1, 2, 3, 11, 12, 13),
                  nrow = 2, ncol = 3, byrow = TRUE,
                  dimnames = list(c("row1", "row2"),
                                  c("C.1", "C.2", "C.3")))
> m.dat
     C.1 C.2 C.3
row1   1   2   3
row2  11  12  13
```

陣列 (array) 物件也由包含相同模式的元素組成的 p-維資料物件, 也可以將陣列視爲一個向量具 p-維結構. R 顯示 3-維陣列物件 $m \times n \times k$, `[m, , ]` 出現在某特定元素之前時, 表示某特定元素在該陣列物件之第 m 列 (row) 的位置; `[ , n, ]` 出現在某特定元素之前時, 表示某特定元素在該陣列物件之第 n 欄 (column) 的位置, 依此類推. `[ , ,k]` 表示 3-維陣列的第 1, 2-維度之矩陣.

```
> ## atomic object
> ## array
> a.vec <- 1:24
```

```
> a.vec
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
[22] 22 23 24
> b.arr <- array(a.vec, dim = c(4, 3, 2),
                 dimnames = list(c("x1", "x2", "x3", "x4"),
                                 c("y1", "y2", "y3"),
                                 c("z1", "z2")))
> b.arr
, , z1
   y1 y2 y3
x1  1  5  9
x2  2  6 10
x3  3  7 11
x4  4  8 12
, , z2
   y1 y2 y3
x1 13 17 21
x2 14 18 22
x3 15 19 23
x4 16 20 24
```

4.3 遞迴型物件 Recursive Object

遞迴型物件 (recursive object) 有列表 (list), 資料框架 (data frame), 函式 (function) 等. 遞迴型物件含有複雜 (混合) 元素模式, 其他如, `"expression"`, `"name"`, `"symbol"` 等也是遞迴型物件. 列表物件 與 資料框架物件 的模式 (mode) 都是 `"list"`, 函式物件 的模式是 `"function"`. 使用函式指令 `mode()` 可以用來查看物件的模式.

列表 (list) 是由資料物件組成的一種物件, lsit 是一個特殊的向量型式, 每個 list 的元素稱爲 成份 (component). 而組成列表中的個別成份 (component) 本身是物件, 且個別成份是有順序的 (order sequence), 個別成份物件的元素模式, 沒有任合限制, 每一個別成份物件的結構可以不相同. 但是, 若某一個別成份物件是原型物件, 則此個別成份內的元素必須符相同的基本模式.

```
> ##### recursive object
> ## list
> x.vec <- 1:4
> y.vec <- c("Male", "Female")
```

```
> z.mat <- matrix(1:9, nrow = 3, ncol = 3)
> xyz.list <- list(x.vec, y.vec, z.mat)
> xyz.list
[[1]]
[1] 1 2 3 4
[[2]]
[1] "Male"   "Female"
[[3]]
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
>
> ## get the elements in a list
> xyz.list[1]
[[1]]
[1] 1 2 3 4
> xyz.list[2]
[[1]]
[1] "Male"   "Female"
> xyz.list[3]
[[1]]
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
>
> ## give some names
> xyz.list <- list(A.name = x.vec, B.name = y.vec, C.name = z.mat)
> xyz.list
$A.name
[1] 1 2 3 4
$B.name
[1] "Male"   "Female"
$C.name
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> xyz.list$A.name
[1] 1 2 3 4
> xyz.list$B.name
[1] "Male"   "Female"
> xyz.list$C.name
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
>
> ## function
```

```
> mode(mean)
[1] "function"
```

資料框架 (data frame) 物件是一個 2-維度 的列表, 統計分析常用資料框架物件, 通常每一 “列 (row)” 表示研究的個體 (subject), 每一 “欄位 (column)” 表示研究的變數.

```
> ## data frame = list
> id.vec <- c(1, 2, 3, 4)
> age.vec <- c(35, 55, 45, 25)
> sex.vec <- c("Male", "Male", "Female", "Female")
> disease.vec <- c("Yes", "No", "No", "Yes")
> x.df <- data.frame(id = id.vec,
                     age = age.vec,
                     sex = sex.vec,
                     disease = disease.vec)
> x.df
  id age    sex disease
1  1  35   Male     Yes
2  2  55   Male      No
3  3  45 Female      No
4  4  25 Female     Yes
> x.df$age
[1] 35 55 45 25
> x.df$disease
[1] Yes No  No  Yes
Levels: No Yes
>
```

4.4 物件模式 Mode of Object

R 物件元素的 模式 (mode) 分成 基本模式 (basic mode) 與 複雜模式 (complex mode). R 的最基本物件是 向量 (vector). 向量的元素是由包含相同 基本模式 (basic mode) 的元素組成, 且 矩陣 (matrix), 陣列 (array) 的元素也都必須優單一的基本模式組成. 在 R 中可以使用函式 `mode()` 來查看物件的類型.

R 物件元素的 基本模式 (basic mode) 主要分成爲

1. `"numeric"`, 數值型 (實數型), 含 `"single"` 單精準度型 與 `"double"` 倍精準度型.

2. "integer", 整數向量 (有時需特別指定).
3. "logical", 邏輯型 (true or false), 以 TRUE (T) 或 FALSE (F) 呈現, (也可以是 1 與 0 (分別代表 T 與 F).
4. "complex", 複數型.
5. "character", 文字型 或 字串型, 通常輸入時在文字兩側加上雙引號 (").

R 物件元素的 複雜模式 (complex mode), 則包含 "list", "expression", "name", "symbol" 與 "function" 等. 矩陣 (matrix), 陣列 (array), 可視爲向量的組合, 且列表 (lists), 函式 (function) 或資料框架 (data frames) 等. 任何物件都可視爲向量的組合.

```
> ## mode of object
> # vector
> x.vec <- c(1/1, 1/2, 1/3, 1/4, 1/5)
> mode(x.vec) # check mode
[1] "numeric"
> y.vec <- c("Hello", "What's your name?", "Your email?")
> mode(y.vec)
[1] "character"
> z.vec <- c(F, T, T, F, F)
> mode(z.vec)
[1] "logical"
> x.complex <- 8+3i
> mode(x.complex)
[1] "complex"
> # matrix
> x.mat <- matrix(c(1, 5, 3, 7, 4, 9), nrow = 2)
> mode(x.mat)
[1] "numeric"
> # array
> a.vec <- c(1:24)
> b.arr <- array(a.vec, dim = c(4, 3, 2),
                 dimnames = list(c("x1", "x2", "x3", "x4"),
                                 c("y1", "y2", "y3"),
                                 c("z1", "z2")))
> mode(b.arr)
[1] "numeric"
> # list
> x.num <- c(1, 3, 6)
> y.char <- c("chocolate", "vanilla", "strawberry")
> xy.list <- list(x.num.name = x.num, y.char.name = y.char)
> mode(xy.list)
```

```
[1] "list"
> # data frame
> data(Puromycin)   # load Puromycin data frame
> mode(Puromycin) # # check Puromycin mode
[1] "list"
> # function
> mode(mean)
[1] "function"
> mode(var)
[1] "function"
```

4.5 物件類型 Type of Object

物件 類型 (type) 主要是分成 向量 (vector), 矩陣 (matrix), 陣列 (array), 因子 (factor), 列表 (list), 函式 (function) 與 資料框架 (data frame) 等. 向量, 矩陣 與 陣列通常是由包含相同的 基本模式 (basic mode) 的元素組成. 向量, 矩陣 與 陣列 的 類型 又可分細分成 `numeric` (`single` 與 `double`), (數值, 單精準度 與 倍精準度), `integer` (整數), `logical` (邏輯), `complex` (複數), 與 `character` (文字) 等, 在 R 中可以使用函式 `typeof()` 來查看物件的類型, `typeof()` 與 `mode()` 功能相似.

```
> ## type of object
> # vector
> x.vec <- c(1/1, 1/2, 1/3, 1/4, 1/5)
> typeof(x.vec)
[1] "double"
> y.vec <- c("Hello", "What's your name?", "Your email?")
> typeof(y.vec)
[1] "character"
> z.vec <- c(F, T, T, F, F)
> typeof(z.vec)
[1] "logical"
> x.complex <- 8+3i
> typeof(x.complex)
[1] "complex"
> # matrix
> x.mat <- matrix(c(1, 5, 3, 7, 4, 9), nrow = 2)
> typeof(x.mat)
[1] "double"
> # array
> a.vec <- c(1:24)
> b.arr <- array(a.vec, dim = c(4, 3, 2),
```

```
                dimnames = list(c("x1", "x2", "x3", "x4"),
                                c("y1", "y2", "y3"),
                                c("z1", "z2")))
> typeof(b.arr)
[1] "integer"
> # list
> x.num <- c(1, 3, 6)
> y.char <- c("chocolate", "vanilla", "strawberry")
> xy.list <- list(x.num.name = x.num, y.char.name = y.char)
> typeof(xy.list)
[1] "list"
> # data frame
> data(Puromycin)   # load Puromycin data frame
> typeof(Puromycin)
[1] "list"
> typeof(mean)
[1] "closure"
> typeof(var)
[1] "closure"
```

4.6 物件長度 Length of Object

物件的 長度 (length) 與物件的 模式 (mode) 是物件兩個基本 特徵 (property), 模式 和 長度 又稱爲一個物件的 "內在屬性" (intrinsic attributes). 物件長度 的定義, 與 物件模式 有關, 矩陣 與 陣列 的長度是所有元素的數目, 所以 2-維度 矩陣, 或 3-維度 陣列, 其長度是各維度素的數目之乘積 (product). 列表 之長度的定義是列表中的物件成份 (component) 的數目, 但是 資料框架物件 之長度的定義是 "欄位數目" (number of columns). 使用函式指令 `length()` 可以查看物件長度. 長度爲 0 的物件有如一個空盒子內無物品, 長度爲 NULL 指沒有盒子.

```
> ## length of object
> # vector
> x.vec <- 1:10
> length(x.vec)
[1] 10
> # matrix
> x.vec <- c("a", "b", "c", "d", "e", "f")
> y.mat <- matrix(x.vec , nrow = 2, ncol = 3)
> length(y.mat)
[1] 6
```

```
> # array
> a.vec <- 1:24
> z.arr <- array(a.vec, dim = c(4, 3, 2))
> length(b.arr)
[1] 24
> # list
> x.vec <- 1:4
> y.vec <- c("Male", "Female")
> z.vec <- matrix(1:9, 3, 3)
> xyz.list <- list(x.vec, y.vec, z.vec)
> length(xyz.list)
[1] 3
> # data frame = list
> id.vec <- c(1, 2, 3, 4)
> age.vec <- c(35, 55, 45, 25)
> sex.vec <- c("Male", "Male", "Female", "Female")
> disease.vec <- c("Yes", "No", "No", "Yes")
> x.df <- data.frame(id = id.vec,
                     age = age.vec,
                     sex = sex.vec,
                     disease = disease.vec)
> x.df
  id age    sex disease
1  1  35   Male     Yes
2  2  55   Male      No
3  3  45 Female      No
4  4  25 Female     Yes
> length(x.df)
[1] 4
> # data frame
> data(Puromycin)   # load Puromycin data frame
> head(Puromycin)
  conc rate   state
1 0.02   76 treated
2 0.02   47 treated
3 0.06   97 treated
4 0.06  107 treated
5 0.11  123 treated
6 0.11  139 treated
> length(Puromycin)
[1] 3
```

4.7 物件的類別 Class

一個的物件屬性或特徵, 稱做物件的 類別 (class), 列表 (list) 與 資料框架 (data

frame) 的 type 都是 list, 但 列表 與 資料框架 的 class 卻不同, 資料框架 統計常用的資料型式, 是一個非常有自己特徵的 list. 物件的類別方便 R 進行程式寫作. 類別可以讓 R 得知物件的特殊性, 使用特別的方法進行操作. 例如, 有一個物件其類別是資料框架, 則此物件會以特別形式列印, 例如使用函式指令 `print()` 會對 任一資料框架 做出特別的列印方式都相同, 但不同於 list 的列印方式. 相關的 "通用函式 (generic function), 對某個特定 物件類別, 會有一樣的操作與運算. 例如, 函式 `summary()` 對 一般線性模型 (lm()) 的操作都相同, 但 函式 `summary()` 對 一般線性模型 (lm()) 與 廣線性模型義 `glm()` 的操作不太相同. 可以使用 函式 `class()` 指定物件的類別, 使用 函式 `unclass()` 刪除一個物件的類別.

```
> #### class
> # vector
> x.vec <- 1:10
> class(x.vec)
[1] "integer"
> # matrix
> x.vec <- c("a", "b", "c", "d", "e", "f")
> y.mat <- matrix(x.vec , nrow = 2, ncol = 3)
> class(y.mat)
[1] "matrix"
> # array
> a.vec <- 1:24
> z.arr <- array(a.vec, dim = c(4, 3, 2))
> class(z.arr)
[1] "array"
> # list
> x.vec <- 1:4
> y.vec <- c("Male", "Female")
> z.vec <- matrix(1:9, 3, 3)
> xyz.list <- list(x.vec, y.vec, z.vec)
> class(xyz.list)
[1] "list"
> # data frame
> data(Puromycin)   # load Puromycin data frame
> class(Puromycin)
[1] "data.frame"
> # function
> class(mean)
[1] "function"
> class(glm)
[1] "function"
```

4.8 物件屬性 Attributes of Object

物件的 屬性 (attribute) 是指 非內在屬性 (non-intrinsic attribute), 物件屬性用來說明物件的特徵. 物件屬性有物件的類別 (class), matrix 或 array 的維度 (dimension, dim) matrix 或 array 的維度名 (dimnames), matrix 或 array 列位名 (row name), 欄位名 (column name), 向量元素名 或 列表成分名 (names()), 資料框架變數名 (row.names()), 注釋 (comment) 等.

使用函式指令 `attributes()` R 回傳一個 list, 此 list 是由物件當前定義的非內在屬性所形成的列表. 或使用簡寫函式指令 `attr()`, 且須同時指定所要查看的屬性, R 回傳一個 list 由物件指定所要查看的屬性所形成的列表.

```
attr(obj)
attr(x, which, exact = FALSE)
```

函式指令 `attr()` 的引數爲

- `x`: 物件名.
- `which`: 輸入 class, comment, dim, dimnames, names, row.names and tsp.

使用函式指令 `attr(object, name)` 可以用來選擇特定的屬性. 對屬性進行改變和刪除必須小心, 因爲物件的屬性是 R 物件系統的重要部分. 表 4.2 顯示資料物件的各種性質, 表 4.3 顯示查看資料物件常用的各種函式.

一些函式指令可以取得物件的某一特定屬性. 例如若要讀取或命名 vector, matrix, array, list 的所有個別維度或成分名稱, `dimnames()`.

```
> #### attribute of object
> ## vector
> x.vec <- 1:3
> names(x.vec)
NULL
> x.vec <- c(a = 1, b = 2, c = 3)
> names(x.vec)
[1] "a" "b" "c"
```

```
>
> ## assign/change just one name
> names(x.vec)[2] <- "B"
> x.vec
a B c
1 2 3
> class(x.vec)
[1] "numeric"
>
> ## vector
> x.vec <- 1:10
> attr(x.vec, "my_attribute") <- "This is a vector"
> attr(x.vec, "my_attribute")
[1] "This is a vector"
> attributes(x.vec)
$my_attribute
[1] "This is a vector"

> names(x.vec)
NULL
> str(x.vec)
 atomic [1:10] 1 2 3 4 5 6 7 8 9 10
 - attr(*, "my_attribute") =  chr "This is a vector"
> str(attributes(x.vec))
List of 1
 $ my_attribute: chr "This is a vector"
> attributes(x.vec[1])
NULL
>
> ## vector
> structure(1:3, my_attribute = "This is a 3-element vector")
[1] 1 2 3
attr(,"my_attribute")
[1] "This is a 3-element vector"
```

若要對讀取或命名 矩陣 的 列位名 (row name) 或 欄位名 (column name), 可以用函式指令 `rownames()` 與 `colnames()`.

```
> ## matrix
> x.vec <- c("a", "b", "c", "d", "e", "f")
> y.mat <- matrix(x.vec , nrow = 2, ncol = 3)
> class(y.mat)
[1] "matrix"
> attr(y.mat, "dim")
[1] 2 3
> attributes(y.mat)
$dim
[1] 2 3

```

```
> str(y.mat)
 chr [1:2, 1:3] "a" "b" "c" "d" "e" "f"
>
> ## matrix
> x.mat <- matrix(1:6, nrow = 2, ncol = 3)
> dimnames(x.mat) <- list(c("A1", "A2"),
                          c("B1", "B2", "B3"))
> dim(x.mat)
[1] 2 3
> length(x.mat)
[1] 6
> names(x.mat)
NULL
> dimnames(x.mat)
[[1]]
[1] "A1" "A2"

[[2]]
[1] "B1" "B2" "B3"

> rownames(x.mat)
[1] "A1" "A2"
> colnames(x.mat)
[1] "B1" "B2" "B3"
> class(x.mat)
[1] "matrix"
> attr(x.mat, "dim")
[1] 2 3
> attr(x.mat, "dimnames")
[[1]]
[1] "A1" "A2"

[[2]]
[1] "B1" "B2" "B3"

> attributes(x.mat)
$dim
[1] 2 3

$dimnames
$dimnames[[1]]
[1] "A1" "A2"

$dimnames[[2]]
[1] "B1" "B2" "B3"

> str(x.mat)
 int [1:2, 1:3] 1 2 3 4 5 6
 - attr(*, "dimnames") = List of 2
  ..$ : chr [1:2] "A1" "A2"
```

```
  ..$ : chr [1:3] "B1" "B2" "B3"
>
> ## matrix
> (x.mat <- matrix(1:12, nrow = 2, ncol = 6, byrow = FALSE))
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    3    5    7    9   11
[2,]    2    4    6    8   10   12
> class(x.mat)
[1] "matrix"
> attributes(x.mat)
$dim
[1] 2 6
```

若要讀取或命名 array 的第 1 維度的 列位名 (row name) 或 第 2 維度的 欄位名 (column name), 可以用函式 `rownames()` 與 `colnames()`. 若要讀取或命名高維度 matrix 或 array 的所有維度名稱, 可以用函式 `dimnames()`.

```
> ## array
> a.vec <- 1:24
> b.arr <- array(a.vec, dim = c(4, 3, 2),
                 dimnames = list(c("x1", "x2", "x3", "x4"),
                                 c("y1", "y2", "y3"),
                                 c("z1", "z2")))
> names(b.arr)
NULL
> rownames(b.arr)
[1] "x1" "x2" "x3" "x4"
> colnames(b.arr)
[1] "y1" "y2" "y3"
> class(b.arr)
[1] "array"
> attr(b.arr, "dim")
[1] 4 3 2
> attr(b.arr, "dimnames")
[[1]]
[1] "x1" "x2" "x3" "x4"

[[2]]
[1] "y1" "y2" "y3"

[[3]]
[1] "z1" "z2"

> attributes(b.arr)
$dim
[1] 4 3 2

$dimnames
```

```
$dimnames[[1]]
[1] "x1" "x2" "x3" "x4"

$dimnames[[2]]
[1] "y1" "y2" "y3"

$dimnames[[3]]
[1] "z1" "z2"

> str(b.arr)
 int [1:4, 1:3, 1:2] 1 2 3 4 5 6 7 8 9 10 ...
 - attr(*, "dimnames") = List of 3
  ..$ : chr [1:4] "x1" "x2" "x3" "x4"
  ..$ : chr [1:3] "y1" "y2" "y3"
  ..$ : chr [1:2] "z1" "z2"
```

使用函式 `names()`. 讀取或命名 list 的所有個別成分名稱. 使用函式 `str()` 可以得知 list 更詳細的物件摘要.

```
> ## list
> x.num <- c(1, 3, 6)
> y.char <- c("chocolate", "vanilla", "strawberry")
> xy.list <- list(x.num.name = x.num, y.char.name = y.char)
> mode(xy.list)
[1] "list"
> typeof(xy.list)
[1] "list"
> length(xy.list)
[1] 2
> names(xy.list)
[1] "x.num.name"  "y.char.name"
> class(xy.list)
[1] "list"
> attr(xy.list, "dim")
NULL
> attr(xy.list, "names")
[1] "x.num.name"  "y.char.name"
> attributes(xy.list)
$names
[1] "x.num.name"  "y.char.name"

> str(xy.list)
List of 2
 $ x.num.name : num [1:3] 1 3 6
 $ y.char.name: chr [1:3] "chocolate" "vanilla" "strawberry"
> ## list
> x.num <- c(1, 3, 6)
> y.char <- c("chocolate", "vanilla", "strawberry")
> xy.list <- list(x.num.name = x.num, y.char.name = y.char)
```

```
> mode(xy.list)
[1] "list"
> typeof(xy.list)
[1] "list"
> length(xy.list)
[1] 2
> names(xy.list)
[1] "x.num.name"  "y.char.name"
> class(xy.list)
[1] "list"
> attr(xy.list, "dim")
NULL
> attr(xy.list, "names")
[1] "x.num.name"  "y.char.name"
> attributes(xy.list)
$names
[1] "x.num.name"  "y.char.name"

> str(xy.list)
List of 2
 $ x.num.name : num [1:3] 1 3 6
 $ y.char.name: chr [1:3] "chocolate" "vanilla" "strawberry"
>
> ## list
> x.vec <- 1:4
> y.vec <- c("Male", "Female")
> z.mat <- matrix(1:9, nrow = 3, ncol = 3)
> xyz.list <- list(x.vec, y.vec, z.mat)
> mode(xyz.list)
[1] "list"
> length(xyz.list)
[1] 3
> names(xyz.list)
NULL
> class(xyz.list)
[1] "list"
> attr(xyz.list, "names")
NULL
> attributes(xyz.list)
NULL
> str(xyz.list)
List of 3
 $ : int [1:4] 1 2 3 4
 $ : chr [1:2] "Male" "Female"
 $ : int [1:3, 1:3] 1 2 3 4 5 6 7 8 9
> attributes(xyz.list)
NULL
```

若要讀取 資料框架 的 列位名 (row name), 可以用函式 `row.names()`, 要讀取

資料框架 的 欄位名 (column name) 或 變數名 (variable name), 可以用函式指令 `names()`. 如同讀取或命名 list 的所有個別成分名稱. 使用函式 `str()` 可以得知 data frame 詳細的物件摘要.

讀取 矩陣型式的資料, 如矩陣, 陣列, 資料框架 等命名, 可使用通用函示 `rownames()` 與 `colnames()` 分別讀取 第 1 維度 (列位名) 與 第 2 維度 (欄位名) 的命名. 因此, 讀取 資料框架 列位名 與 欄位名, 使用函式 `row.names()`, `names()` 與使用函式 `rownames()` `colnames()` 是相同的.

```
> ## data frame = list
> id.vec <- c(1, 2, 3, 4)
> age.vec <- c(35, 55, 45, 25)
> sex.vec <- c("Male", "Male", "Female", "Female")
> disease.vec <- c("Yes", "No", "No", "Yes")
> x.df <- data.frame(id = id.vec,
                     age = age.vec,
                     sex = sex.vec,
                     disease = disease.vec)
> mode(x.df)
[1] "list"
> length(x.df)
[1] 4
> ncol(x.df)
[1] 4
> nrow(x.df)
[1] 4
> dim(x.df)
[1] 4 4
> rownames(x.df)
[1] "1" "2" "3" "4"
> colnames(x.df)
[1] "id"      "age"     "sex"     "disease"
> row.names(x.df)
[1] "1" "2" "3" "4"
> names(x.df)
[1] "id"      "age"     "sex"     "disease"
> dimnames(x.df)
[[1]]
[1] "1" "2" "3" "4"

[[2]]
[1] "id"      "age"     "sex"     "disease"

> class(x.df)
[1] "data.frame"
> attr(x.df, "dim")
```

```
NULL
> attr(x.df, "dimnames")
NULL
> attributes(x.df)
$names
[1] "id"      "age"     "sex"     "disease"

$row.names
[1] 1 2 3 4

$class
[1] "data.frame"

> str(attributes(x.df))
List of 3
 $ names    : chr [1:4] "id" "age" "sex" "disease"
 $ row.names: int [1:4] 1 2 3 4
 $ class    : chr "data.frame"
```

表 4.2: R 資料物件的各種性質

結構 (structure)	類型 (type)	模式 (mode)	類別 (class)
Atomic	vector	character, complex, numeric, logical	NULL
	matrix	character, complex, numeric, logical	NULL
	array	character, complex, numeric, logical	NULL
Recursive	list	list	NULL
	data frame	list	data frame
	function	function	NULL

表 4.3: R 函式: 用來操作資料物件的性質

A: 資料物件摘要訊息	
函式	說明
`str()`	顯示資料物件結構
`attributes()`	列表回傳資料物件屬性
`attr()`	指派或更改資料物件屬性

B: 資料物件特別訊息	
函式	說明
`mode()`	回傳資料物件模式
`typeof()`	回傳資料物件類型
`class()`	回傳資料物件類別
`length()`	回傳資料物件長度
`dim()`	回傳資料物件維度
`nrow()`	回傳 matrix, array 列位數
`ncol()`	回傳 matrix, array 欄位數
`dimnames()`	回傳資料物件列與欄位名
`rownames()`	回傳 matrix, array, data frame 列位名
`colnames()`	回傳 matrix, array, data frame 欄位名
`row.names()`	回傳 data frame 列位名
`names()`	回傳 list, data frame 欄位名