第 3 章: 基本物件
3: Basic Objects

3.1 資料物件與運算式 Data Objects and Expression

R 與 S 都是以物件導向爲主的程式語言, R 中, 每一樣 "東西", 都叫做 "**物件**", 物件可以是向量 (vector), 矩陣 (matrix), 陣列 (array), 列表 (Lists), 或 資料框架 (data frames) 等. 在 R 中, 資料分析基本上是 產生資料物件, 命名, 使用函式對物件運算操作. 透過指令, 很容易地對物件進行統計分析與統計繪圖.

3.1.1 運算式語言 Expression Language

R 基本介面是一個互動式指令視窗, 指令可分成 **運算式 expression** 如 `1+2` 或 **指派運算 (賦值運算) assignment**, 如 `x<-1+2`. 當一個 R 程式需要你輸入指令時, 它會顯示指令提示號, 指令提示符號通常是一個 `>` (大於符號), 完整 **運算式** 指令輸入後的結果, 馬上顯示在指令下方. **指派運算** 同樣會做運算式, 並且把結果 (值) 傳給變數, 但結果不會自動顯示在視窗螢幕上. 指派運算符號通常是 "`<-`", 一個小於符號和一個短線符號組成, 如 `x<-1+2`, 讀成 `x` "得到" $(1+2)$.

如果一條指令在一行結束的時候, 在語法上還不完整, R 會給出另一個不同的提示符號, 通常是 +, 該提示符號 +, 會出現在第二行, 和隨後的行中, 持續等待輸入指令, 當一指令在語法上是完整的時候, 才執行指令. 不同的完整指令再同一行時, 可用 ; (分號) 隔開, 或是另起一新輸入行. 指令可以放入大括弧內, { }, 放在一起, 構成一個複合運算式 (compound expression). **注釋 (commands)** 幾乎可以放在任何地方, 任何一行中, 注釋從 # (井號) 開始, 到句子收尾之間的語句就是是注釋. 基本指派運算之指令符號見表 3.1.

R 是一種運算式語言 (expression language), 運算式由含有指令形式的物件名稱 (object name), 算數操作符號 (operators), 函式 (function) 組成. R 的最基本物件是向量, 利用基本算數符號, 如 `+`, `-`, `*`, `/`, `%/%`, `%%`, `%*%`, `^`, `==`, `!`, `!=`, `<`, `>`, `<=`, `>=`, `&`, `&&`, `|`, `||`, `xor` 等, 或簡單的函式, 如 `sum()`, `mean()`, `sqrt()`, `log()`, `exp()`, `abs()`, `sin()` 等, 對向量做運算, 基本運算符號與簡單的函也可以對物件做運算.

表 3.1: 指派運算之指令符號

Basic Assignment	
`?`	Help
`<-`	Left assignment, binary
`->`	Right assignment, binary
`$`	List subset, binary
`:`	Sequence, binary
`:`	In model formulae: interaction
`~`	Tilde, used for model formulae, can be either uniary or binary

```
> x <- 3; x + 5
[1] 8
> y <- 1:10
> #
> { x <- 3
+     x + 5
+     }
[1] 8
```

3.1.2 物件模式與結構

R 的最基本物件是向量, 向量是由包含相同 "模式" 的元素 (element) 組成, 向量物件的基本元素 "**模式**" (**basic mode**) 主要分成

1. `"numeric"`, 實數型, 含 `"integer"`, 整數向量 (有時需特別指定), 與 `"double"`, 倍精確度型.
2. `"logical"`, 邏輯型 (true or false), 以 `TRUE` (`T`) 或 `FALSE` (`F`) 呈現, (也可以是 1 (`T`) 與 0 (`F`).
3. `"complex"`, 複數型
4. `"character"`, 文字型 (或字串), 通常輸入時, 在文字或字串兩側加上雙引號 (`"`).

向量物件的複雜元素模式, 包含 `"list"`, `"expression"`, `"name"`, `"symbol"` 與 `"function"` 等.

物件結構可簡單分成 "**原型**" (**atomic**) 與 "**遞迴型**" (**recursive**). "**原型**" 結構物件如向量, 矩陣, 陣列等; **遞迴型** 結構物件如列表, 資料框架, 函式等.

3.1.3 物件命名

須特別注意, 在 R 指令的英文大小寫有差異, `s` 與 `S` 是不同的, R 也保留一些物件與指令名稱, 如 `c`, `s`, `C`, `T`, `F` 等, 這些叫做 "**保留名字** ("**reserved names**")

```
FALSE Inf NA NaN NULL TRUE
break else for function if in next repeat while
F T
```

另外一些系統常用的指令名稱, 如

```
c q s t C D F I T diff mean pi range rank var
```

對物件命名時盡量避免定義一個物件, 與現有的物件同名. 所以命名時要避免重覆, 以免後來引起錯亂. 對物件命名時, 物件名稱起始位置須以文字或 “.” (句點), 若物件名稱以 “.” (句點) 爲起始, 名稱第二個位置需爲文字, 物件名稱其餘位置, 以文字 (A-Z 或 a-z) , 數字 (0-9), “/”, “.”, 或 “-”, 皆可. 中間不可有空格或 “_” (underscore).

3.2 基本向量 Basic Operator

R 的最基本物件是向量, “**向量**” 是指包含相同 “模式” (mode) 的元素 (element) 組成, 主要有 6 種基本模式 (mode), `logical, integer, double, single, complex, and character` (邏輯, 整數, 倍精準度, 單精準度, 複數, 文字), 同一向量內的元素不可混合. 單一數值 (scalar), 可視爲僅具有單一元素的向量, 標準的向量是倍精準度 (double) 的 “實數” (**numerical**) 向量. 向量可以用 `c()` 函式產生, 如下

```
> # numerical
> x<-c(1/1, 1/2, 1/3, 1/4, 1/5)
> x
[1] 1.0000000 0.5000000 0.3333333 0.2500000 0.2000000
> # character
> y<-c("Hello", "What's your name?", "Your email?")
> y
[1] "Hello"             "What's your name?" "Your email?"
> # logicl
> z<-c(F, T, T, F, F)
> z
[1] FALSE  TRUE  TRUE FALSE FALSE
> # complex
> x.complex<-8+3i
> x
[1] 1.0000000 0.5000000 0.3333333 0.2500000 0.2000000
```

R 的向量物件其基本操作符號 (operators), 如同 `C` 語言, 可以分成算數操作 (arithmetic operator), 相關比較操作 (relation/comparison operator), 邏輯操作 (logical operator).

3.2.1 算數操作 Arithmetic Operator

算數操作 (arithmetic operator) 符號包含 `+, -, *, /, ^, %%, %*%, %/%, %x%` 等. 如同

一般算數運用在向量. 主要的算數操作符號, 建表 3.2.

表 3.2: 算數操作 (Arithmetic Operator)

符號	定義
-	Substraction, can be unary or binary
+	Addition, can be unary or binary
!	Unary not
*	Multiplication, binary
/	Division, binary
^	Exponentiation, binary
%%	Modulus, binary
%/%	Integer divide, binary
%*%	Matrix product, binary
%o%	Outer product, binary
%x%	Kronecker product, binary
%in%	Matching operator, binary (in model formulae: nesting)

```
> 1+2
[1] 3
> 2*3+4
[1] 10
> 2*(3+4)
[1] 14
> (3+11*2)/4
[1] 6.25
>
> x.complex<-(8+3i)+(1+2i)
> x.complex
[1] 9+5i
> #
> x<-1:5
> y<-6:10
> z<-c(2,2,3,3,4)
> x+y
[1]  7  9 11 13 15
> x-y
[1] -5 -5 -5 -5 -5
>
> x*2
[1]  2  4  6  8 10
> x*y
[1]  6 14 24 36 50
>
```

```
> x/2
[1] 0.5 1.0 1.5 2.0 2.5
> x/y
[1] 0.1666667 0.2857143 0.3750000 0.4444444 0.5000000
>
> x^2
[1]  1  4  9 16 25
> x^z
[1]   1   4  27  64 625
>
> y/2
[1] 3.0 3.5 4.0 4.5 5.0
> y/x
[1] 6.000000 3.500000 2.666667 2.250000 2.000000
>
> x%*%y
     [,1]
[1,]  130
> y%*%x
     [,1]
[1,]  130
>
> x%x%y
 [1]  6  7  8  9 10 12 14 16 18 20 18 21 24 27 30 24 28 32 36 40 30 35 40 45 50
> y%x%x
 [1]  6 12 18 24 30  7 14 21 28 35  8 16 24 32 40  9 18 27 36 45 10 20 30 40 50
>
> y%%3
[1] 0 1 2 0 1
> y%/%3
[1] 2 2 2 3 3
> y%/%x
[1] 6 3 2 2 2
```

3.2.2 關係比較操作 Relation/Comparison Operator

關係比較操作 (relation/comparison operator) 符號包含常見的 `==`, `!=`, `<`, `<=`, `>`, `>=`, 主要用來比較大小, 回傳一個邏輯模式的向量, 見表 3.3.

表 3.3: 關係比較操作 Relation/Comparison Operator

符號	定義
<	Less than, binary
>	Greater than, binary
==	Equal to, binary
!=	Not equal to
>=	Greater than or equal to, binary
<=	Less than or equal to, binary

```
> x<-1:5
> y<-6:10
>
> x<2
[1]  TRUE FALSE FALSE FALSE FALSE
> x<=2
[1]  TRUE  TRUE FALSE FALSE FALSE
> x==2
[1] FALSE  TRUE FALSE FALSE FALSE
> x!=2
[1]  TRUE FALSE  TRUE  TRUE  TRUE
> x<y
[1] TRUE TRUE TRUE TRUE TRUE
> x<(y-7)
[1] FALSE FALSE FALSE FALSE FALSE
>
> x<=y
[1] TRUE TRUE TRUE TRUE TRUE
> x<=(y-7)
[1] FALSE FALSE FALSE FALSE FALSE
>
> x==y
[1] FALSE FALSE FALSE FALSE FALSE
> x==(y-7)
[1] FALSE FALSE FALSE FALSE FALSE
> x!=y
[1] TRUE TRUE TRUE TRUE TRUE
> x!=(y-7)
[1] TRUE TRUE TRUE TRUE TRUE
```

3.2.3 操作 Logical Operator

邏輯操作 (logical operator) 符號包含常見的 !，&，&&，|，||, 主要用來做布林 (boolean) 運算, 見表 3.3.

表 3.4: 邏輯操作 Logical Operator

符號	定義
!	Logical NOT
&	Logical AND, binary, vectorized
&&	Logical AND, binary, not vectorized
|	Logical OR, binary, vectorized
||	Logical OR, binary, not vectorized

```
> (x>0) & (y>0)
[1] FALSE FALSE  TRUE  TRUE  TRUE
> ((x-2)>0) & ((y-7)>0)
[1] FALSE FALSE FALSE FALSE FALSE
>
> (x>0) && (y>0)
[1] FALSE
> ((x-2)>0) && ((y-7)>0)
[1] FALSE
>
> (x>0) | (y>0)
[1] TRUE TRUE TRUE TRUE TRUE
> ((x-2)>0) | ((y-7)>0)
[1] FALSE FALSE  TRUE  TRUE  TRUE
>
> (x>0) || (y>0)
[1] TRUE
> ((x-2)>0) || ((y-7)>0)
[1] FALSE
>
> xor((x>0), (y>0))
[1]  TRUE  TRUE FALSE FALSE FALSE
> xor(((x-2)>0), ((y-7)>0))
[1] FALSE FALSE  TRUE  TRUE  TRUE
>
> xx<-x<=1
> yy<-x>4
> xor(xx, yy)
[1]  TRUE FALSE FALSE FALSE  TRUE
> xx | yy
[1]  TRUE FALSE FALSE FALSE  TRUE
```

3.3 物件 Objects

3.3.1 物件類形 (Type)

物件類型 (type) 主要是向量 (vector), 矩陣 (matrix), 陣列 (array), 因素 (factor), 列表 (list), 資料框架 (data frame), 函式 (function). `typeof()` 函式指令可以用來查看物件的類型.

3.3.2 物件模式與結構 (Mode)

物件基本元素之 "**模式**" (**basic mode**) 分成

1. `"numeric"`, 實數型, 含 `"integer"`, 整數型 (有時需特別指定), 與 `"double"`, 倍精確度型.
2. `"logical"`, 邏輯型 (true or false), 以 `TRUE` (`T`) 或 `FALSE` (`F`) 呈現, (也可以是 1 (`T`) 與 0 (`F`).
3. `"complex"`, 複數型
4. `"character"`, 文字型 (或字串), 通常輸入時, 在文字或字串兩側加上雙引號 (").

物件結構可簡單分成 "**原型**" (**atomic**) 與 "**遞迴型**" (**recursive**). "**原型**" (**atomic**) 結構物件如向量, 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" 的元素 (element) 組成, 矩陣與陣列也由包含相同模式的元素組成; **遞迴型** (**recursive**) 結構物件如列表, 資料框架, 函式等. 遞迴型結構物件含有複雜 (混合) 元素模式, 包擴 `"list"`, `"expression"`, `"name"`, `"symbol"` 與 `"function"` 等. `mode()` 函式指令可以用來查看物件的模式.

R 的最基本物件是向量, 由包含相同的元素組成, 但無**維度** (**dimension**), 主要有 6 種基本模式 (mode), `logical, integer, double, single, complex, and character` (邏輯, 整數, 倍精準度, 單精準度, 複數, 文字). 當 R 顯示向量物件, `[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).

```
> x<-1:30
> x
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
[20] 20 21 22 23 24 25 26 27 28 29 30
> flavors<-c("chocolate", "vanilla", "strawberry")
> flavors
[1] "chocolate"  "vanilla"    "strawberry"
> z<-c(T, F, F)
> z
[1]  TRUE FALSE FALSE
```

矩陣由包含相同的元素組成的二維 (2-dimension) 資料物件, 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 mth 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 nth 欄 (column) 的位置.

```
> x<-c("a", "b", "c", "d")
> x
[1] "a" "b" "c" "d"
> y<-matrix(x,2,2)
> y
     [,1] [,2]
[1,] "a"  "c"
[2,] "b"  "d"
> y<-matrix(x,2,2,byrow=T)
> y
     [,1] [,2]
[1,] "a"  "b"
[2,] "c"  "d"
```

陣列也由包含相同模式的元素組成的 n-維資料物件, 可以將陣列視爲一個向量具 n-維結構. R 顯示陣列物件, `[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 mth 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 nth 欄 (column) 的位置, 依此類推.

```
> a<-1:24
> a
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
[20] 20 21 22 23 24

> b<-array(a, dim=c(2,3,4), dimnames=c("x", "y", "z"))
> b
, , 1

     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6

, , 2

     [,1] [,2] [,3]
[1,]    7    9   11
[2,]    8   10   12

, , 3

     [,1] [,2] [,3]
[1,]   13   15   17
[2,]   14   16   18

, , 4

     [,1] [,2] [,3]
```

```
[1,]   19   21   23
[2,]   20   22   24
```

當有人想要將不同的基本元素放入原型模式 (atomic mode) 物件如向量, 矩陣, 陣列等, R 會將資料中不同的基本元素融合成相同的基本元素, 其融合的依據是階層結構: `character > complex > numeric > logical`. 也就是說, 若向量中包含有任一文字元素, 則向量內所有的元素都將轉換融合成文字基本元素.

```
> x<-c("hellp", 3+6i, 5.42, FALSE) # coerce to character
> x
[1] "hellp" "3+6i"  "5.42"  "FALSE"
> y<-c(3+6i, 5.42, FALSE) # coerce to complex
> y
[1] 3.00+6i 5.42+0i 0.00+0i
> z<-c(5.42, FALSE) # coerce to numerical
> z
[1] 5.42 0.00
```

遞迴型 (**recursive**) 結構物件如列表 (list), 資料框架 (data frame), 函式 (function) 等. 遞迴型結構物件含有複雜 (混合) 元素模式, 包擴 `"list"`, `"expression"`, `"name"`, `"symbol"` 與 `"function"` 等. 列表物件與資料框架物件的模式都是 `"list"`, 函式物件的模式是 `"function"`. `mode()` 函式指令可以用來查看物件的模式.

列表物件是由資料物件組成, 列表物中的成份 (component) 是物件, 是有順序的 (order sequence), 成份物件的元素模式, 沒有任合限制, 每一個別成份的物件之原型模式可以不相同.

```
> # List
> x<-1:4
> y<-c("Male", "Female")
> z<-matrix(1:9,3,3)
> xyz.list<-list(x, y, z)
> 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(X=x, Y=y, Z=z)
> xyz.list
$X
[1] 1 2 3 4

$Y
[1] "Male"   "Female"

$Z
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

> xyz.list$X
[1] 1 2 3 4
> xyz.list$Y
[1] "Male"   "Female"
> xyz.list$Z
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

資料框架物件是一個 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.

```
> subjectid<-c(1, 2, 3, 4)
> age<-c(35, 55, 45, 25)
> sex<-c("Male", "Male", "Female", "Female")
> disease<-c("Yes", "No", "No", "Yes")
> x.data <- data.frame(subjectid, age, sex, disease)
>
```

```
> mode(x.data)
[1] "list"
> class(x.data)
[1] "data.frame"
```

3.3.3 物件長度 (Length)

物件的 **長度** (**length**) 與物件的 **模式** (**mode**) 是物件兩個基本的 **特徵** (**property**), 模式和長度又叫做一個物件的 "**內在屬性**" (**intrinsic attributes**). 物件長度的定義, 與物件的模式有關, 陣列的長度是所有元素的數目, 所以 2-維度, 或 3-維度的陣列, 其陣列長度是各維度素的數目之乘積 (product). 列表物件其長度的定義, 是列表中的物件成份 (component) 的數目, 但是資料框架物件其長度的定義, 是 "欄位數目" (number of columns). `length()` 指令可以查看物件長度. 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.

3.3.4 物件的屬性 (Attribute)

物件的"**屬性**" (指非內在屬性) (**non-intrinsic attributes**), 如列位名, 欄位名 (row name, column name), 維度 (dimension) 等. 函式 `attributes(object)` 回傳物件當前定義的非內在屬性的列表. 若要讀取資料框架的 "列位名" (row name), 可以用 `row.names()` 函式, 要讀取資料框架的 "欄位名" (column name), 可以用 `names()` 函式. 使用 `str()` 可以得知更詳細的物件摘要.

```
> # vector
> z<-1:100
> mode(z)
[1] "numeric"
> length(z)
[1] 100
> attributes(z)
NULL
>
> # matrix
> # matrix
> x<-c("a", "b", "c", "d")
> y<-matrix(x,2,2)
> mode(x)
[1] "character"
> length(x)
[1] 4
> attributes(x)
NULL
>
> # array
> a<-1:24
> b<-array(a, dim=c(2,3,4), dimnames=c("x", "y", "z"))
```

```
> mode(b)
[1] "numeric"
> length(b)
[1] 24
> dim(b)
[1] 2 3 4
> attributes(b)
$dim
[1] 2 3 4
>
> # List
> x<-1:4
> y<-c("Male", "Female")
> z<-matrix(1:9,3,3)
> xyz.list<-list(x, y, z)
> mode(xyz.list)
[1] "list"
> length(xyz.list)
[1] 3
> attributes(xyz.list)
NULL
>
> # data frame
> subjectid<-c(1, 2, 3, 4)
> age<-c(35, 55, 45, 25)
> sex<-c("Male", "Male", "Female", "Female")
> disease<-c("Yes", "No", "No", "Yes")
> x.data <- data.frame(subjectid, age, sex, disease)
> mode(x.data)
[1] "list"
> length(x.data)
[1] 4
> ncol(x.data)
[1] 4
> nrow(x.data)
[1] 4
> dim(x.data)
[1] 4 4
> row.names(x.data)
[1] "1" "2" "3" "4"
> names(x.data)
[1] "subjectid" "age"       "sex"       "disease"
>
> attributes(x.data)
$names
[1] "subjectid" "age"       "sex"       "disease"
```

```
$row.names
[1] "1" "2" "3" "4"

$class
[1] "data.frame"
>
> dimnames(x.data)
[[1]]
[1] "1" "2" "3" "4"

[[2]]
[1] "subjectid" "age"       "sex"       "disease"
>
> str(x.data)
'data.frame':   4 obs. of  4 variables:
 $ subjectid: num  1 2 3 4
 $ age      : num  35 55 45 25
 $ sex      : Factor w/ 2 levels "Female","Male": 2 2 1 1
 $ disease  : Factor w/ 2 levels "No","Yes": 2 1 1 2
```

屬性包含列位名 (row name), 欄位名 (column name), 維度 (dimension) 等, 函式 `attr(object, name)` 可以用來選擇特定的屬性. 這些函數很少使用. 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.

```
> ## attr
> # create a 2 by 5 matrix
> x <- 1:10
> attr(x,"dim") <- c(2, 5)
> x
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10
>
> y.str<-structure(matrix(c(1,2,3,4),nr=2,byrow=T),
     dimnames=list(c("row1","row2"),c("col1", "col2")))
> y.str
     col1 col2
row1    1    2
row2    3    4
>
> dimnames(y.str)
[[1]]
[1] "row1" "row2"

[[2]]
[1] "col1" "col2"
```

3.3.5 物件的分類 (Class)

一個特殊的物件屬性, 稱做物件的 "**類別**" ("**class**"), 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, 指令 `plot()` 會做出特別的圖形等等. 相關的"**通用函式 (generic function)**, 如 `summary()`, 對某個特定物件類別, 會有一樣的操作與運算. 可以用函數式 `unclass()` 刪除一個物件的類別. 表 3.5 顯示資料物件的各種性質, 表 3.6 顯示查看資料物件常用的各種函式.

表 3.5: R 資料物件的各種性質

Data type	Data Object	Possible mode	Default 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 3.6: R 函式: 用來操作資料物件的性質

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
資料物件特別訊息	
函式	說明
`mode()`	回傳資料物件模式
`length()`	回傳資料物件長度
`dim()`	回傳資料物件維度
`nrow()`	回傳資料物件列位數
`ncol()`	回傳資料物件欄位數
`dimnames()`	回傳資料物件列與欄位名
`row.names()`	回傳資料物件列位名
`names()`	回傳資料物件欄位名

3.4 向量物件 Vector

R 的最基本物件是向量, "**向量**" 是指包含相同 "模式" 的元素 (element) 組成序列, 主要有 6 種基

本模式 (mode), `logical, integer, double, single, complex, and character` (邏輯, 整數, 倍精準度, 單精準度, 複數, 文字). 向量是具有相同基本類型的元素序列, 大體相當於其他語言中的 1-維度數列, 在 R 中, 單一數值 (**scalar**) 也可看成是長度爲 1 的向量.

3.4.1 向量的產生 c()

向量的產生最常用辦法是使用函式 `c()`, 它把若干個數值或字串組合爲一個向量,

```
> flavors<-c("chocolate", "vanilla", "strawberry")
> flavors
[1] "chocolate"  "vanilla"    "strawberry"
> 口味<-c("巧克立", "香草", "草莓")
> 口味
[1] "巧克立" "香草"   "草莓"
> scores<-c(5,5,5,3,6,2,4,5,3,4)
> scores
 [1] 5 5 5 3 6 2 4 5 3 4
> y<-c(scores,0,0,scores)
> y
 [1] 5 5 5 3 6 2 4 5 3 4 0 0 5 5 5 3 6 2 4 5 3 4
```

3.4.2 數列向量 seq() 與 sequence()

在統計運算中, 常需要產生數列向量, 如 `[1,3,5,7,9]`等, 可以用 `:`, `seq()` 函式.

```
> 1:5
[1] 1 2 3 4 5
> 5:1
[1] 5 4 3 2 1
> seq(1, 2,0.3)
[1] 1.0 1.3 1.6 1.9
> 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,length=3)
[1] 1 3 5
> z<-c("a", "b", "c", "d", "e")
> seq(along=z)
[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)
```

3.4.3 向量含重複元素 rep() 與 gl()

另一個與 seq() 類似的函式 rep() 可以產生相同的元素.

```
> rep(1,5)
[1] 1 1 1 1 1
> rep(0, times=10)
 [1] 0 0 0 0 0 0 0 0 0 0
> x<-c(1,2,3)
> rep(x,each=2)
[1] 1 1 2 2 3 3
> rep(x,times=2)
[1] 1 2 3 1 2 3
> rep(x,times=c(2,2,2))
[1] 1 1 2 2 3 3
> x<-c(1,2,3)
> rep(x, times=c(1,2,3))
[1] 1 2 2 3 3 3
> help(rep)
```

另一個與 rep() 類似的函式 gl(x). 可以有許多不同的變化.

```
> #  gl(n, k, length = n*k, labels = 1:n, ordered = FALSE)
> ## First control, then treatment:
> gl(2, 3, label = c("Control", "Treat"))
[1] Control Control Control Treat   Treat   Treat
Levels: Control Treat
> ## 20 alternating 1s and 2s
> gl(2, 1, 5)
[1] 1 2 1 2 1
Levels: 1 2
> ## alternating pairs of 1s and 2s
> gl(2, 2, 5)
[1] 1 1 2 2 1
Levels: 1 2
```

3.4.4 文字向量操作 paste()

函式 paste() 可以將各種向量融合或強制轉換成文字與字串向量, 並且可以把它們一個接一個連成字串. paste() 中的引數, 內設的分隔符號是一個空白字元.

```
> paste("X", 1)
[1] "X 1"
> paste("X", "Y", 1)
[1] "X Y 1"
> paste("X", "Y", 1, sep="")
```

```
[1] "XY1"
> paste("X", "Y", 1:3, sep="")
[1] "XY1" "XY2" "XY3"
>
> paste(c("X", "Y"), 1)
[1] "X 1" "Y 1"
> paste(c("X", "Y"), 1, sep="")
[1] "X1" "Y1"
> paste(c("X", "Y"), 1:4, sep="")
[1] "X1" "Y2" "X3" "Y4"
> paste(c("X", "Y"), 1:3, sep="")
[1] "X1" "Y2" "X3"
>
> paste(c("X", "Y"), 1:4, sep="", collapse=" + ")
[1] "X1 + Y2 + X3 + Y4"
```

3.4.5 文字向量操作 substr() 與 strsplit()

函式 substr() 可以將文字 (與字串) 向量中, 文字 (與字串) 元素抽取出部份文字 (與字串). 另一個函式 strsplit() 可以根據某一特定字元, 分離文字 (與字串) 向量.

```
> #  substr(x, start, stop)
> substr("abcdef",2,4)
[1] "bcd"
> substring("abcdef",1:6,1:6)
[1] "a" "b" "c" "d" "e" "f"
>
> substr(rep("abcdef",4),1:4,4:5)
[1] "abcd" "bcde" "cd"   "de"
> x <- c("asfef", "qwerty", "xyz   ", "ab", "a")
> substr(x, 2, 5)
[1] "sfef" "wert" "yz  " "b"    ""
> substring(x, 2, 4:6)
[1] "sfe"   "wert"  "yz   " "b"     ""
>
> substring(x, 2) <- c("..", "+++")
> x
[1] "a..ef"  "q+++ty" "x..   " "a+"     "a"
> #
> help(strsplit)
> # split x on the letter e
> x <- c(as = "asfef", qu = "qwerty", "xyz   ", "ab", "a")
> strsplit(x,"e")
$as
[1] "asf" "f"
```

```
$qu
[1] "qw"  "rty"

[[3]]
[1] "xyz   "

[[4]]
[1] "ab"

[[5]]
[1] "a"
```

3.4.6 向量列印輸出

函式 `print()` 可以用來列印 R 物件, 有時候, 可以用 `cat()` 列印輸出. 列印文字向量的時候, 採用雙引號 (有時根本不用引號). 採用 C 語言形式的轉義控制序列, 用 \ 表示轉義字元, 所以輸入 \\ 將會得到 \ 的輸出, 而想插入雙引號, ", 則要輸入 \". 其他有用的轉義字元如 \n (換行), \t (跳位字元), 和 \b (倒退鍵) 等等.

3.4.7 向量的算數操作

算數操作 (arithmetic operator) 符號包含 `+`, `-`, `*`, `/`, `^`, `%%`, `%*%`, `%/%`, `%x%` 等. 通常其含意是對向量的每一個元素進行運算的 "單元運算子" (unary) 或 "二元運算子" (binary), 如同一般算數運用在向量. 主要的算數操作符號, 建表 3.7.

表 3.7: 算數操作 (Arithmetic Operator)

符號	定義
-	減法 Substraction, can be unary or binary
+	加法 Addition, can be unary or binary
*	乘法 Multiplication, binary
/	除法 Division, binary
^	乘冪 Exponentiation, binary
!	否定 Unary not
%%	Modulus, binary
%/%	Integer divide, binary
%*%	Matrix product, binary
%o%	Outer product, binary
%x%	Kronecker product, binary
%in%	Matching operator, binary (in model formulae: nesting)

```
> x<-c(0:4)
> a<-1
> y<-2 * (a+log(x))
> y
[1]     -Inf 2.000000 3.386294 4.197225 4.772589
> y<-2 * (log(a+x))
> y
[1] 0.000000 1.386294 2.197225 2.772589 3.218876
```

3.4.8 邏輯向量的操作

邏輯向量元素的值有 TRUE, FALSE 和 NA. 可以分別簡寫爲 T 和 F. 注意 T 和 F 不是系統的保留字 (reserved word), 可以被覆寫. 應該儘量使用 TRUE 和 FALSE. 邏輯向量通常可以由 條件式產生.

```
> x<-1:5
> y<-(x>2)
> y
[1] FALSE FALSE  TRUE  TRUE  TRUE
```

y 是一個長度和 x 一樣的向量, 它的元素 TRUE 表示 x 的元素符合控制條件 x > 2, 它的元素 TRUE 則相反.

R 邏輯運算符號是 <, <=, >, >= 以及判斷相等的 == 和判斷不等的 !=.

此外, 如果 X 和 Y 表示邏輯的不等式, 那麼 X & Y 是它們的交集運算 (**AND**), X | Y 是聯集運算 (OR); !X 是 X 的否定運算 (X 與 非 X). 在算術運算中採用邏輯變數, 會將邏輯變數強制轉換成數值變數, FALSE 變成 0, TRUE 變成 1. 邏輯運算符號參見表 3.8.

表 3.8: 邏輯向量的運算操作符號

符號	定義
<	Less than, binary
>	Greater than, binary
==	Equal to, binary
!=	Not equal to
>=	Greater than or equal to, binary
<=	Less than or equal to, binary
!	Logical NOT
&	Logical AND, binary, vectorized
&&	Logical AND, binary, not vectorized
|	Logical OR, binary, vectorized
||	Logical OR, binary, not vectorized

```
> x<-1:5
```

```
> y<-6:10
>
> x<2
[1]  TRUE FALSE FALSE FALSE FALSE
> x<=2
[1]  TRUE  TRUE FALSE FALSE FALSE
> x==2
[1] FALSE  TRUE FALSE FALSE FALSE
> x!=2
[1]  TRUE FALSE  TRUE  TRUE  TRUE
>
> x<y
[1] TRUE TRUE TRUE TRUE TRUE
> x<(y-7)
[1] FALSE FALSE FALSE FALSE FALSE
>
> x<=y
[1] TRUE TRUE TRUE TRUE TRUE
> x<=(y-7)
[1] FALSE FALSE FALSE FALSE FALSE
>
> x==y
[1] FALSE FALSE FALSE FALSE FALSE
> x==(y-7)
[1] FALSE FALSE FALSE FALSE FALSE
> x!=y
[1] TRUE TRUE TRUE TRUE TRUE
> x!=(y-7)
[1] TRUE TRUE TRUE TRUE TRUE
>
> #######
> (x>0) & (y>0)
[1] TRUE TRUE TRUE TRUE TRUE
> ((x-2)>0) & ((y-7)>0)
[1] FALSE FALSE  TRUE  TRUE  TRUE
>
> (x>0) && (y>0)
[1] TRUE
> ((x-2)>0) && ((y-7)>0)
[1] FALSE
>
> (x>0) | (y>0)
[1] TRUE TRUE TRUE TRUE TRUE
> ((x-2)>0) | ((y-7)>0)
[1] FALSE FALSE  TRUE  TRUE  TRUE
>
> (x>0) || (y>0)
```

```
[1] TRUE
> ((x-2)>0) || ((y-7)>0)
[1] FALSE
>
> xor((x>0), (y>0))
[1] FALSE FALSE FALSE FALSE FALSE
> xor(((x-2)>0), ((y-7)>0))
[1] FALSE FALSE FALSE FALSE FALSE
>
> xx<-x<=1
> yy<-x>4
> xor(xx, yy)
[1]  TRUE FALSE FALSE FALSE  TRUE
> xx | yy
[1]  TRUE FALSE FALSE FALSE  TRUE
```

3.4.9 向量的模式與改變

不同模式的向量做運算, 其模式會有改變. 其模式改變融合的依據是階層結構: `character > complex > numeric > logical`. 也就是說, 若向量中包含有任一文字元素, 則向量內所有的元素都將轉換融合成文字基本元素.

```
> x<-3+5i
> mode(x)
[1] "complex"
> y<-2
> mode(y)
[1] "numeric"
> mode(c(x,y))
[1] "complex"
> x+y
[1] 5+5i
> mode(x+y)
[1] "complex"
>
> x<-c("hellp", 3+6i, 5.42, FALSE) # coerce to character
> mode(x)
[1] "character"
> y<-c(3+6i, 5.42, FALSE) # coerce to complex
> mode(y)
[1] "complex"
> z<-c(5.42, FALSE) # coerce to numerical
> mode(z)
[1] "numeric"
```

3.4.10 向量的長度改變與元素重覆使用 (Recycle)

不同長度的向量做運算, 其結果的向量長度會與最長的向量長度相同. 長度較短的向量, 其元素會重覆使用 (**rcycle**), 直到長度與長的向量長度相同. 同長度的向量做運算, R 會運算並得到結果, R 會提出警告, 使用者須小心.

```
> x<-1:3
> 2*x      # 2 recycle 3 times
[1] 2 4 6
> y<-6:10
> x+y      # x recycle  1 2 3 1 2
[1]  7  9 11 10 12
Warning message:
長的目的物件長度不是短的目的物件長度的整數倍 in: x + y
```

遺失值 (缺失值) Missing Values

研究資料, 通常會有 **遺失值 (缺失值)** (**missing value**), 在 R 中, 輸入或輸出遺失值, 通常以 NA 表示, (NA = "not available"), R 還有另外有 NaN = "Not a Number", 以及 NULL 是指物件的長度是 0. 任何對遺失值 (NA) 的算數操作, 會得到遺失值 (NA) 結果. 函式 is.na(), is.nan() 可查看向量內那些元素是遺失值. 回傳一個邏輯向量. 對遺失值作比較大小運算須非常小心. 要移除遺失值, 可以用 na.omit(), na.fail(), na.exclude(), na.action() 等指令.

```
> z<-c(1:2,NA)
> is.na(z)
[1] FALSE FALSE  TRUE
> log(z)
[1] 0.0000000 0.6931472        NA
> z/0
[1] Inf Inf  NA
> 0/0
[1] NaN
> Inf-Inf
[1] NaN
>
> is.na(z)
[1] FALSE FALSE  TRUE
> is.na(0/0)
[1] TRUE
> is.nan(z)
[1] FALSE FALSE FALSE
> is.nan(0/0)
[1] TRUE
```

3.4.11 複數向量

R 具備複數運算, 複數常量只要用 2.3+4.6.1i 這樣的格式即可. 複數向量的每一個元素都是複數. Re() 計算實部, Im() 計算虛部, Mod() 計算複數極座標的模數, Arg() 計算複數極座標的角度. 若

$$
\begin{aligned}
z &= x + y\mathrm{i} \\
r &= \sqrt{x^2 + y^2} \\
x &= r \star \cos(\phi) \\
y &= r \star \sin(\phi)
\end{aligned}
$$

實部爲 x=Re(z), 虛部爲 y=Im(z), 極座標的模數爲 r=Mod(z), 極座標的角度爲 phi = Arg(z).

```
> z<-c(3+5i, 5+1i,6+7i)
> Re(z)
[1] 3 5 6
> Im(z)
[1] 5 1 7
> Mod(z)
[1] 5.830952 5.099020 9.219544
> Arg(z)
[1] 1.0303768 0.1973956 0.8621701
```

3.4.12 向量的元素命名

向量的元素命名, 可以在輸入元素時給予命名, 或另外使用 names() 函式給予命名.

```
> x<-c(age=50, chol=220, dbp=84, sbp=132)
> x
 age chol  dbp  sbp
  50  220   84  132
> names(x)
[1] "age"  "chol" "dbp"  "sbp"
>
> x<-c(55, 236, 80, 140)
> names(x)<-c("age", "chol", "sbp", "dbp")
> x
 age chol  sbp  dbp
  55  236   80  140
>
> y.name<-names(x)
> y<-c(60, 214, 90, 144)
> names(y)<-y.name
> y
```

```
 age chol  sbp  dbp
  60  214   90  144
```

3.4.13 向量的下標與索引

一個向量的元素可以向量的下標下標取得, 向量的 **下標 / 索引 (index)** 是在向量名稱後面加中括號內放入下標數目 (或向量) 得到. 向量的下標可以採用下面四種方式的任何一種.

1. **正整數下標向量** (positive integral quantities). 這種情況下, 下標向量必須是 `1, 2, ..., length(x)` 的子向量. 這種下標向量可以是任意長度, 下標向量中對應的元素會被取得.

   ```
   > # positive integer
   > x<-1:50
   > x[7]
   [1] 7
   > x[11:15]
   [1] 11 12 13 14 15
   > y<-x[11:15]
   > y
   [1] 11 12 13 14 15
   > c("a","b")[rep(c(1,2,2,1), times=2)]
   [1] "a" "b" "b" "a" "a" "b" "b" "a"
   ```

2. **負整數下標向量 (negative integral quantities)**. 取值下標向量在 `- 1, 2, ..., length(x)`, 表示刪除相應位置的元素.

   ```
   > z<-6:10
   > z[-c(2,4)]
   [1]  6  8 10
   ```

3. **文字 / 字串 下標向量 (character strings)**. 在定義向量時可以給元素加上名字, 以 `names()` 加上屬性, 這種情況下 文字 / 字串 下標向量可以如上使用. 文字 / 字串 下標向量的好處就是容易記 (但不能打錯字).

   ```
   > fruit <- c(5, 10, 1, 20)
   > fruit
   [1]  5 10  1 20
   > names(fruit) <- c("orange", "banana", "apple", "peach")
   > fruit
   orange banana  apple  peach
        5     10      1     20
   > lunch <- fruit[c("apple","orange")]
   > lunch
    apple orange
        1      5
   ```

4. **邏輯下標向量 (logical vector)**. 邏輯下標向量必須和原有向量長度一致, 才能挑選元素, 向量中對應的下標向量爲 TRUE 的元素將會得到, 而那些對應 FALSE 的元素則被刪除. 如果下標都是 FALSE, 則結果是一個 0 長度的向量, 顯示爲 numeric(0).

```
> x<-c(NA, seq(-2,2), NA, seq(-2,2))
> x
 [1] NA -2 -1  0  1  2 NA -2 -1  0  1  2
> y<-x[!is.na(x)]
> y
 [1] -2 -1  0  1  2 -2 -1  0  1  2
> z<-x[x>0 & !is.na(x)]
> z
[1] 1 2 1 2
> x[x < (-5)]
[1] NA NA
> y[y < (-5)]
numeric(0)
```

3.5 因素 / 因子 物件 (Factor) 與 類別變數

因數物件 (factor) 爲處理類別資料, 提供的一種有效的方法. 因爲統計中的離散變數 (discrete variable) 包含 **名義變數 (nominal variable)** 與 **有序變數 (ordinal variable)**, 有各種不同表示方法, 在 R 中方便起見, 使用 因素/因子 (factor) 來表示類別變數. 因素 / 因子 是一種特殊的文字向量, 文字向量中每一個元素, 取一個離散值, 因素物件有一個特殊屬性 "**層次 / 水平 / 水準 levels** 表示這組所有的離散值. 因數可以簡單地用函式 factor() 產生. 因素 / 因子 是用 "文字/字串" 輸入, 一但設定爲因素 / 因子向量, R 列印時, 並不會加上雙引號 ".

3.5.1 無序因素 / 名義變數 (Unordered Factor)

無序因子 (名義變數) 中的離散值併無大小順序的關係, 如性別的男與女. R 列印無序因素物件時, "層次" 內建式依照文字字母順序, 查看無序因素物件的層次, 以用 levels() 指令, 若要設定列印無序因素物件 "層次" 列印順序, 也可以用 levels() 指令. 函式 relevel() 可以改變無序因子層次的順序, 令某一個層次爲參考的第一層次 (reference level).

```
> # FACTOR OBJECT
> # UNORDERED
> sex <- factor(c("男", "女", "男", "男", "女"))
> sex
[1] 男 女 男 男 女
Levels: 女 男
> gender <- factor(c("Male", "Female", "Male", "Male", "Female"))
> gender
[1] Male   Female Male   Male   Female
```

```
Levels: Female Male
> levels(gender)
[1] "Female" "Male"
>
> gender <- factor(c("M", "F", "M", "M", "F"), levels=c("M", "F"))
> gender
[1] M F M M F
Levels: M F
>
> gender <- factor(c("M", "F", "M", "M", "F"))
> gender
[1] M F M M F
Levels: F M
> levels(gender<-c("F", "M"))
NULL
> gender
[1] "F" "M"
>
> income <- factor(c("Lo", "Mid", "Hi", "Mid", "Lo", "Hi", "Lo"))
> income
[1] Lo  Mid Hi  Mid Lo  Hi  Lo
Levels: Hi Lo Mid
> relevel(income, ref="Lo")
[1] Lo  Mid Hi  Mid Lo  Hi  Lo
Levels: Lo Hi Mid
> income
[1] Lo  Mid Hi  Mid Lo  Hi  Lo
Levels: Hi Lo Mid
```

3.5.2 有序因素 / 順序變數 (Ordered Factor)

有時候因素的層次 (水準) 有自己的自然大小順序的順序變數 (ordinal variable), 並且有可能用於進一步的統計分析. 函式 `ordered()` 就是用來產生這樣的有序因子物件. 函式 `ordered()` 與 `levels()` 基本完全一樣. R 內建大小的順序是依照文字字母排大小順序, 可以用 `levels()` 指令, 改變順序. 多數情況下, 有序和無序因子的唯一差別, 在於有序因子顯示的時候, 應了各水準的順序, 在線性模型統計分析的時候, 這種差異的意義更明顯.

```
> # ORDERED
> income <- ordered(c("Mid", "Hi", "Lo", "Mid", "Lo", "Hi", "Lo"))
> income
[1] Mid Hi  Lo  Mid Lo  Hi  Lo
Levels: Hi < Lo < Mid
>
> inc <- ordered(c("Mid", "Hi", "Lo", "Mid", "Lo", "Hi", "Lo"),
+     levels = c("Lo", "Mid", "Hi"))
```

```
> inc
[1] Mid Hi  Lo  Mid Lo  Hi  Lo
Levels: Lo < Mid < Hi
```

3.5.3 連續型變數轉換成類別變數與 cut() 函式

函式 cut() 可以將連續型變數分割成類別型變數. 配合 facor() 或 ordered, 可以各種類別轉換.

```
> ### cut()
> x<-1:12
> y<-cut(x, breaks=4)
> y
 [1] (0.989,3.74] (0.989,3.74] (0.989,3.74] (3.74,6.5]
 [5] (3.74,6.5]   (3.74,6.5]   (6.5,9.26]   (6.5,9.26]
 [9] (6.5,9.26]   (9.26,12]    (9.26,12]    (9.26,12]
Levels: (0.989,3.74] (3.74,6.5] (6.5,9.26] (9.26,12]
>
> y<-cut(x, breaks=4, include.lowest=T, right=F, dig.lab=0)
> y
 [1] [1,4)     [1,4)     [1,4)     [4,6)     [4,6)
 [6] [4,6)     [6,9)     [6,9)     [6,9)     [9,1e+01]
[11] [9,1e+01] [9,1e+01]
Levels: [1,4) [4,6) [6,9) [9,1e+01]
>
> y<-cut(x, breaks=seq(0,12,by=4))
> y
 [1] (0,4]  (0,4]  (0,4]  (0,4]  (4,8]  (4,8]  (4,8]  (4,8]
 [9] (8,12] (8,12] (8,12] (8,12]
Levels: (0,4] (4,8] (8,12]
>
> y<-cut(x, breaks=c(0,4,8,12))
> y
 [1] (0,4]  (0,4]  (0,4]  (0,4]  (4,8]  (4,8]  (4,8]  (4,8]
 [9] (8,12] (8,12] (8,12] (8,12]
Levels: (0,4] (4,8] (8,12]
>
> y<-ordered(y, labels=levels(y))
> y
 [1] (0,4]  (0,4]  (0,4]  (0,4]  (4,8]  (4,8]  (4,8]  (4,8]
 [9] (8,12] (8,12] (8,12] (8,12]
Levels: (0,4] < (4,8] < (8,12]
>
> #
> x<-1:12
> y<-cut(x, breaks=5, include.lowest=T)
```

```
> y
 [1] [0.989,3.19] [0.989,3.19] [0.989,3.19] (3.19,5.4]
 [5] (3.19,5.4]   (5.4,7.6]    (5.4,7.6]    (7.6,9.81]
 [9] (7.6,9.81]   (9.81,12]    (9.81,12]    (9.81,12]
5 Levels: [0.989,3.19] (3.19,5.4] (5.4,7.6] ... (9.81,12]
> y<-ordered(y, labels=levels(y))
> y
 [1] [0.989,3.19] [0.989,3.19] [0.989,3.19] (3.19,5.4]
 [5] (3.19,5.4]   (5.4,7.6]    (5.4,7.6]    (7.6,9.81]
 [9] (7.6,9.81]   (9.81,12]    (9.81,12]    (9.81,12]
5 Levels: [0.989,3.19] < (3.19,5.4] < ... < (9.81,12]
```

3.6 矩陣物件 Matrix

3.6.1 矩陣物件的產生

矩陣由包含相同的元素組成的 2-維 (2-dimension) 資料物件, 並有 `dim()` 屬性. 維度向量 (dimensino vector) 是一個僅具有正整數的向量, 如果維度向量的長度爲 k, 則陣列 `dim()` 屬性爲 k-維度陣列. 可以將矩陣視爲一個向量具 2-維陣列結構. 輸入簡單的矩陣資料, (列 $\times$ 行), 或希望以矩陣形式儲存, 可以用 `matrix()`指令, 設定列數 `nrow=2` 或欄 (行) 數 `ncol=2`, R 設定是以欄 (行) 位 (column) 優先填滿, 要改變設定, 可加入 `byrow=T`. `dim()` 指令回傳資料物件維度.

```
> x<-matrix(c(1, 5, 3, 7, 4, 9), nrow=2)
> x
     [,1] [,2] [,3]
[1,]    1    3    4
[2,]    5    7    9
> x<-matrix(c(1, 5, 3, 7, 4, 9), nrow=2, byrow=T)
> x
     [,1] [,2] [,3]
[1,]    1    5    3
[2,]    7    4    9
> y<-matrix(c(1, 5, 3, 7, 4, 9), ncol=2)
> y
     [,1] [,2]
[1,]    1    7
[2,]    5    4
[3,]    3    9
> y<-matrix(c(1, 5, 3, 7, 4, 9), ncol=2, byrow=T)
> y
     [,1] [,2]
[1,]    1    5
[2,]    3    7
[3,]    4    9
```

```
> z<-matrix(1:18, nrow=3)
> z
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    4    7   10   13   16
[2,]    2    5    8   11   14   17
[3,]    3    6    9   12   15   18
>
> x.mat<-matrix(1:12, 3,4)
> x.mat
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> class(x.mat)
[1] "matrix"
> dim(x.mat)
[1] 3 4
```

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 mth 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 nth 欄 (column) 的位置.

```
> x<-c("a", "b", "c", "d")
> x
[1] "a" "b" "c" "d"
> y<-matrix(x,2,2)
> y
     [,1] [,2]
[1,] "a"  "c"
[2,] "b"  "d"
> y<-matrix(x,2,2,byrow=T)
> y
     [,1] [,2]
[1,] "a"  "b"
[2,] "c"  "d"
```

3.6.2 矩陣的下標與索引

可用中括號 `matrix.name[i, j]` 可存取矩陣中的第 $[i, j]$ 元素; `matrix.name[i, ]`可存取矩陣中的第 i 列 (ith row), `matrix.name[ , j]`可存取矩陣中的第 j 欄 (ith column).

```
> ### index
> x.mat<-matrix(c(1:12), 3, 4)
> x
 age chol  sbp  dbp
  55  236   80  140
```

```
> x.mat[2,3]
[1] 8
> x.mat[2,3]<-30
> x.mat
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5   30   11
[3,]    3    6    9   12
> x.mat[2, ]
[1]  2  5 30 11
> x.mat[ ,3]
[1]  7 30  9
> x.mat[c(1,3), c(2,4)]
     [,1] [,2]
[1,]    4   10
[2,]    6   12
```

3.6.3 向量與矩陣的合併: rbind() 與 cbind() 函式

函式 rbind() 與 cbind() 可以依照 列 (row) 或 欄 (column), 用來合併向量與矩陣.

```
> # rbind(), cbind()
> x.vec<-c(1,2,3,4,5)
> y.vec<-c(6,7,8,9,10)
> rbind(x.vec, y.vec)
      [,1] [,2] [,3] [,4] [,5]
x.vec    1    2    3    4    5
y.vec    6    7    8    9   10
> cbind(x.vec, y.vec)
     x.vec y.vec
[1,]     1     6
[2,]     2     7
[3,]     3     8
[4,]     4     9
[5,]     5    10
>
> x.mat<-matrix(c(11:20), 2, 5)
> rbind(x.mat,x.vec)
      [,1] [,2] [,3] [,4] [,5]
        11   13   15   17   19
        12   14   16   18   20
x.vec    1    2    3    4    5
> cbind(x.mat,y.vec)
                     y.vec
[1,] 11 13 15 17 19     6
[2,] 12 14 16 18 20     7
```

```
Warning message:
    number of rows of result
    is not a multiple of vector length (arg 2) in: cbind(1, x.mat, y.vec)
```

3.7 陣列物件 Array

3.7.1 陣列的產生

陣列可以由包含相同模式的 “物件-元素” 組成的 n-維資料物件, 陣列中的 “元素” 是物件, 可以將陣列視爲一個向量具 n-維結構. 可以用 `matrix()`指令, 產生陣列. R 顯示陣列物件, `[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 mth 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 nth 欄 (column) 的位置, 依此類推.

```
> ## array
> a.vec<-1:24
> b.arr<-array(a.vec, dim=c(2,3,4), dimnames=c("x", "y", "z"))
> dimnames(b.arr)<-list(letters[1:2],
+       LETTERS[1:3],
+       c("i", "ii", "iii", "iv"))
> b.arr
, , i

  A B C
a 1 3 5
b 2 4 6

, , ii

  A  B  C
a 7  9 11
b 8 10 12

, , iii

   A  B  C
a 13 15 17
b 14 16 18

, , iv

   A  B  C
a 19 21 23
b 20 22 24
```

```
> mode(b.arr)
[1] "numeric"
> length(b.arr)
[1] 24
> dim(b.arr)
[1] 2 3 4
> attributes(b.arr)
$dim
[1] 2 3 4

$dimnames
$dimnames[[1]]
[1] "a" "b"

$dimnames[[2]]
[1] "A" "B" "C"

$dimnames[[3]]
[1] "i"   "ii"  "iii" "iv"

> class(b.arr)
[1] "array"
```

3.7.2 陣列的下標與索引

陣列的下標與索引與矩陣下標與索引類似, 可用中括號 `matrix.name[i, j, ...]` 可存取陣列中的第 $[i, j, ...]$ 元素; `matrix.name[i, , ]`可存取陣列中的第 i 列 (ith row), `matrix.name[ , j, ]`可存取矩陣中的第 j 欄 (ith column) 等等.

3.8 列表物件 List

3.8.1 列表的產生

列表是一個特殊的 "向量", 這特殊的向量中的元素是物件. 因此列表物件是由資料物件有順序組成, 列表物中的 "元素", 稱作 "**成份**" (**component**) 是物件本身, 是有順序的 (order sequence), 成份物件的元素模式, 沒有任合限制, 每一個別成份的物件之原型模式可以不相同.

```
> # List
> x<-1:4
> y<-c("Male", "Female")
> z<-matrix(1:9,3,3)
> xyz.list<-list(x, y, z)
> 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
```

3.8.2 列表的下標與索引

列表的下標與矩陣, 陣列不同, 若一個名叫 `List.Name` 的列表, 要取的其中的第 "i.number" 成份, 須用 `[[i.number]]`, 如 `List.Name[[3]]`. 注意, `[[i.number]]` 與 `[i.number]` 是不一樣的.

```
> # get the elements in a list
> xyz.list[1]
[[1]]
[1] 1 2 3 4

> xyz.list[[1]]
[1] 1 2 3 4
>
> xyz.list[2]
[[1]]
[1] "Male"   "Female"

> xyz.list[[2]]
[1] "Male"   "Female"
>
> xyz.list[3]
[[1]]
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

> xyz.list[[3]]
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

若列表中的成份有另外命名爲 comp.name, 可以 List.Name$comp.name 取得, 會與 List.Name[[comp.name]] 結果相同. 在 List.Name$comp.name 或 List.Name[[comp.name]] 加上中括號 [i,j] 等, 可以取得 List.Name$comp.name 中的元素.

```
> # give some names
> xyz.list<-list(class=x, gender=y, score=z)
> xyz.list
$class
[1] 1 2 3 4

$gender
[1] "Male"   "Female"

$score
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

> xyz.list$class
[1] 1 2 3 4
> xyz.list[["class"]]
[1] 1 2 3 4
> xyz.list[["class"]][2]
[1] 2
>
> xyz.list$gender
[1] "Male"   "Female"
> xyz.list[["gender"]][1]
[1] "Male"
>
> xyz.list$score
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> xyz.list[["score"]][2,3]
[1] 8
```