

- R程式變數(Variables)命名法及語法
- Rounding of Numbers
- 向量 (Vectors)
 - 向量運算、規律的序列、邏輯向量、字元向量、遺失值
- 因子 (Factors)、陣列 (Arrays)、矩陣 (Matrices)
- 表列 (List)、資料框 (Data Frame)
- 物件的模式(Mode)、類別 (Class)、屬性 (Attributes)
- 查詢物件之模式、類別、屬性、結構及可取用的元素。
- 存取物件內之元素

R變數命名法

- Case sensitive
 - `A` and `a` are different
- All alphanumeric symbols are allowed (`A-Z`, `a-z`, `0-9`)
 - “.”, “_”.
- Name must start with “.” or a letter.
 - 錯誤命名
 - `3x`
 - `3_x`
 - `3-x`
 - `3.x`
 - `.3variable`
 - 正確命名
 - `x_3`
 - `x3`
 - `x.3`
 - `taiwan.taipei.x3`
 - `.variable`

R 語法

- Assignments

```
> x <- 5
> x = 5
> x
> (x <- 5)
[1] 5
> assign("x", 2)
> 2 -> x
> a <- b <- c <- 6
```

Assignment: 建議使用 `<-`, 而不是 `=`
<https://google.github.io/styleguide/Rguide.xml>
 Google's R Style Guide

- `x <-` expressions

```
> x <- 3+5
```

- `;` or new line

```
> x <- 5 ; y <- 7
or
> x <- 5
> y <- 7
```

- Comment

```
> # how are you?
■ "+" : If a comment is not complete at the end of a
line, R will give a different prompt.
> x <-
+ 5
```

提示符號(prompt)

```
> (y <- 1:5)
[1] 1 2 3 4 5
> sum(y)
[1] 15
> options(prompt = 'hmwu> ', continue = "+ ")
hmwu> sum(y^2)
[1] 55
hmwu> options(prompt = '> ', continue = "+ ")
>
```

物件 (Objects)

- Variables, arrays of numbers, character strings, functions,...

```
> x <- 3+5  
> y <- 7  
> objects()  
[1] "x" "y"
```

```
> ls()  
[1] "x" "y"
```

```
> rm(x, y)  
> rm(list = ls())
```

```
> objects()  
character(0)
```

儲存R物件所佔用的記憶體估計:

```
object.size(x)  
print(object.size(x), units = "Mb")
```

```
> n <- 10000  
> p <- 200  
> myData <- matrix(rnorm(n*p), ncol = p, nrow=n)  
> print(object.size(myData), units = "Mb")  
15.3 Mb
```

```
> os <- function(x){  
+   print(object.size(x), units = "Mb")  
+ }  
> os(myData)  
15.3 Mb
```

物件 (Objects)

```
> x <- 2 # x is a vector of length 1
> x <- vector() # x is a vector of 0 length
> x <- matrix() # x is a matrix of 1 column, 1 row
> x <- 'Hello Dolly' # x is a vector containing 1 string
> x <- c('Hello', 'Dolly') # x is a vector with 2 strings
> x <- function(){ } # x is a function that does nothing
```

- The vectors are atomic objects—all of their elements must be of the same mode.
- In most cases, we work with vectors of modes logical, numeric or character.
- Most other types of objects in R are more complex than vectors. They may consist of collections of vectors, matrices, data frames and functions. When an object is created (for example with the assignment <-), R must allocate memory for the object. The amount of memory allocated depends on the mode of the object.

Rounding of Numbers

- **ceiling**: the **smallest integers** not less than the corresponding elements of x .
- **floor**: the **largest integers** not greater than the corresponding elements of x .
- **trunc**: the **integers** formed by truncating the values in x toward 0.
- **round**: rounds the values in its first argument to the specified number of decimal places (default 0).
- **signif**: rounds the values in its first argument to the specified number of significant digits.

```
> (x <- c(pi, 1/3, -1/3, -pi))
[1] 3.1415927 0.3333333 -0.3333333 -3.1415927
> ceiling(x)
[1] 4 1 0 -3
> floor(x)
[1] 3 0 -1 -4
> trunc(x)
[1] 3 0 0 -3
> round(x, 2)
[1] 3.14 0.33 -0.33 -3.14
> round(x, 5)
[1] 3.14159 0.33333 -0.33333 -3.14159
> signif(x, 2)
[1] 3.10 0.33 -0.33 -3.10
> signif(x, 5)
[1] 3.14160 0.33333 -0.33333 -3.14160
```

設定數值顯示位數 (1)

```
> getOption("digits") # digits: controls the number (1~22, default 7) of
digits to print when printing numeric values.
[1] 7
> x1 <- c(-10, -0.00001, 0, 0.00001, 10)
> x2 <- c(-10, -0.00001, 0, 0.00001, 10, pi)
> x3 <- c(-12, -0.12345, 0, 0.12345, 12)
> x4 <- c(1.810032e+09, 1.810032e-09, 10, pi, 0.0001, -0.0000005)
> x1 # same as print(x1)
[1] -1e+01 -1e-05 0e+00 1e-05 1e+01
> x2
[1] -10.000000 -0.000010 0.000000 0.000010 10.000000 3.141593
> x3
[1] -12.00000 -0.12345 0.00000 0.12345 12.00000
> x4
[1] 1.810032e+09 1.810032e-09 1.000000e+01 3.141593e+00 -5.000000e-07
> cat(x1, "\n")
-10 -1e-05 0 1e-05 10
> cat(x2, "\n")
-10 -1e-05 0 1e-05 10 3.141593
> cat(x3, "\n")
-12 -0.12345 0 0.12345 12
> cat(x4, "\n")
1810032000 1.810032e-09 10 3.141593 -5e-07
```


設定數值顯示位數 (2)

```

> op <- options();
> str(op) # nicer printing
List of 72
 $ add.smooth          : logi TRUE
 ...
> options(digits = 3) # try options(digits = 1) or options(digits = 5)
> x1 # same as print(x1)
[1] -1e+01 -1e-05  0e+00  1e-05  1e+01
> x2
[1] -10.00000  -0.00001  0.00000  0.00001  10.00000  3.14159
> x3
[1] -12.000  -0.123  0.000  0.123  12.000
> x4
[1] 1.81e+09  1.81e-09  1.00e+01  3.14e+00 -5.00e-07
> cat(x1, "\n")
-10 -1e-05 0 1e-05 10
> cat(x2, "\n")
-10 -1e-05 0 1e-05 10 3.14
> cat(x3, "\n")
-12 -0.123 0 0.123 12
> cat(x4, "\n")
1.81e+09 1.81e-09 10 3.14 -5e-07
> options(op) # reset (all) initial options
> options("digits") # or getOption("digits")
$digits
[1] 7

```

See also:

```

> getOption("scipen")
# (penalty) print numeric values in
fixed (scipen = positive integer)
notation or exponential (negative
integer) notation. Fixed notation
will be preferred unless it is more
than scipen digits wider.
> options(scipen = 100, digits = 4)

```

向量 (Vector)

- 向量(vector): 同樣屬性(數字或文字)的資料的集合
- `c()`: 串連多個字串 (combine values into a vector or list)

```
> x1 <- c(10, 5, 3, 6, 2.7)
> x1
[1] 10.0  5.0  3.0  6.0  2.7
>
> assign("x2", c(10, 5, 3, 6, 2.7))
> x2
[1] 10.0  5.0  3.0  6.0  2.7
>
> c(10, 5, 3, 6, 2.7) -> x3
> x3
[1] 10.0  5.0  3.0  6.0  2.7
>
> length(x1)
[1] 5
>
> c(1,7:9)
[1] 1 7 8 9
> c(1:5, 10.5, "next")
[1] "1" "2" "3" "4" "5" "10.5" "next"
```

```
> x1[4]
[1] 6
> x1[2:4]
[1] 5 3 6
> x1[c(4, 2, 1)]
[1] 6 5 10
> x1[-3]
[1] 10.0  5.0  6.0  2.7
> x1[x1<5]
[1] 3.0 2.7
> x1[10]
[1] NA
> x1[2] <- 32; x1
[1] 10.0 32.0  3.0  6.0  2.7
> x1[c(1, 3, 5)] <- c(1,2,3)
> x1
[1] 1 32 2 6 3
```

向量 (Vectors)

```
> (y1 <- 1/x1)
[1] 0.1000000 0.2000000 0.3333333 0.1666667 0.3703704
> length(y1)
[1] 5
>
> v1 <- x1 + y1+1; v1; length(v1)
[1] 11.100000  6.200000  4.333333  7.166667  4.070370
[1] 5
>
> (y2 <- c(x1, 0, x1)); length(y2)
[1] 10.0  5.0  3.0  6.0  2.7  0.0 10.0  5.0  3.0  6.0  2.7
[1] 11
>
> length(x1)
[1] 5
> (v2 <- x1 + y2 + 1); length(v2)
[1] 21.0 11.0  7.0 13.0  6.4 11.0 16.0  9.0 10.0  9.7 13.7
Warning message:
In x1 + y2 :
  longer object length is not a multiple of shorter object length
[1] 11
```

x1 repeated 2.2 times, y2 repeated one time, 1 repeated 11 times

The screenshot shows the RStudio interface with the following components:

- Source Editor:** Contains R code for vector operations:

```
1 ceiling(x)
2 floor(x)
3 floor(x)
4 trunc(x)
5 round(x, 2)
6 round(x, 5)
7 signif(x, 2)
8 signif(x, 5)
9
10 ## vectors
11 x1 <- c(10, 5, 3, 6, 2.7)
12 x1[4]
13 x1[2:4]
14 x1[c(4, 2, 1)]
15 x1[-3]
16 x1[x1<5]
17 x1[10]
18 x1[2] <- 32; x1
19 x1[c(1, 3, 5)] <- c(1,2,3)
```
- Console:** Shows the execution results:

```
> length(y1)
[1] 5
> v1 <- x1 + y1+1; v1; length(v1)
[1] 3.000000 33.031250 3.500000 7.166667 4.333333
[1] 5
> (y2 <- c(x1, 0, x1)); length(y2)
[1] 1 32 2 6 3 0 1 32 2 6 3
[1] 11
> length(x1)
[1] 5
> (v2 <- x1 + y2 +1); length(v2)
[1] 3 65 5 13 7 2 34 35 9 10 5
Warning message:
In x1 + y2 :
  longer object length is not a multiple of shorter ob
ject length
[1] 11
> ?round
> |
```
- Environment:** Lists variables in the Global Environment:

Variable	Value
iris	150 obs. of 5 variables
v1	num [1:5] 3 33.03 3.5 7.17 4.33
v2	num [1:11] 3 65 5 13 7 2 34 35 9 10 ...
x	num [1:4] 3.142 0.333 -0.333 -3.142
x1	num [1:5] 1 32 2 6 3
y1	num [1:5] 1 0.0312 0.5 0.1667 0.3333
y2	num [1:11] 1 32 2 6 3 0 1 32 2 6 ...
- Help:** Shows the documentation for the `round` function:

Rounding of Numbers

Description

`ceiling` takes a single numeric argument `x` and returns a numeric vector containing the smallest integers not less than the corresponding elements of `x`.

`floor` takes a single numeric argument `x` and returns a numeric vector containing the largest integers not greater than the corresponding elements of `x`.

注意: 打完一行程式，就立刻執行，查看結果

向量運算 (Vector Arithmetic)

一些簡單的數學計算：

- `+`, `-`, `*`, `/`, `^`
- `log(x)`, `logb(x, b)`
- `pi`, `exp(x)`,
- `sin(pi/2)`, `cos(pi)`,
`tan(pi/4)`,
- `abs(x)`, `sqrt(x)`,
- `length(x)`
- `prod(x)`
- `choose(n, k)`
- `factorial(x)`

一些簡單的統計運算：

- `max(x)`, `min(x)`
- `pmax(x)`, `pmin(x)`
- `range(x)`
 - `c(min(x), max(x))`
- `mean(x)`
 - `sum(x)/length(x)`
- `var(x)`, `cov(x)`
 - `sum((x-mean(x))^2)/(length(x)-1)`
- `sqrt(var(x))`
- `median(x)`
- `summary(x)`
- `cor(x, y)`

其它函式應用

- `sort(x)` #排序，由小到大。
- `rank(x)` #排序等級。
- `order(x)` #排序後，各個元素的原始所在位置。

```
> ?mean
```

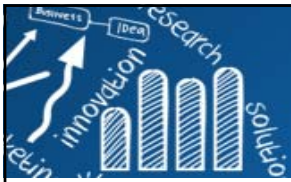
查看 `mean` 的用法

```
mean(x, trim = 0, na.rm = FALSE, ...)
```

課堂練習2

```
> x <- c(1.58, -0.29, 0.59, -0.38, 0.72)
> max(x)
[1] 1.58
> min(x)
[1] -0.38
> range(x)
[1] -0.38 1.58
> c(min(x), max(x))
[1] -0.38 1.58
> mean(x)
[1] 0.444
> sum(x)/length(x)
[1] 0.444
```

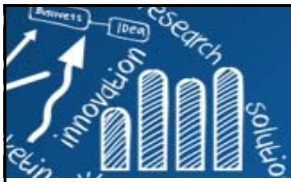
```
> var(x)
[1] 0.65143
> sum( (x-mean(x))^2)/(length(x)-1)
[1] 0.65143
> sqrt(var(x))
[1] 0.8071121
> median(x)
[1] 0.59
> summary(x)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-0.380 -0.290   0.590   0.444   0.720   1.580
> sort(x)
[1] -0.38 -0.29  0.59  0.72  1.58
> rank(x)
[1] 5 2 3 1 4
> order(x)
[1] 4 2 3 5 1
```



規律的序列 (Regular Sequences)

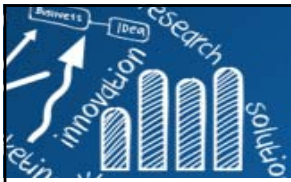
15/50

```
> x <-c(1,2,3,4,5,6,7,8,9,10)
> x <- 1:10
> y <- 10:2
> 2*1:10 #The colon operator has high priority with an expression
[1] 2 4 6 8 10 12 14 16 18 20
>
> n <- 10
> 1:n-1
[1] 0 1 2 3 4 5 6 7 8 9
> 1:(n-1)
[1] 1 2 3 4 5 6 7 8 9
>
> width <- 1
> seq(from=2, to=5, by=width)
[1] 2 3 4 5
>
> 2:5
[1] 2 3 4 5
> s1 <- seq(-5, 5, by=0.2)
> s2 <- seq(length=51, from=-5, by=0.2)
```



規律的序列 (Regular Sequences)

```
> rep(x, times=5)
[1] 1 2 3 4 5 6 7 8 9 10 1 2 3 4 5 6 7 8
[19] 9 10 1 2 3 4 5 6 7 8 9 10 1 2 3 4 5 6
[37] 7 8 9 10 1 2 3 4 5 6 7 8 9 10
>
> rep(x, each=5)
[1] 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4
[19] 4 4 5 5 5 5 5 6 6 6 6 6 7 7 7 7 8
[37] 8 8 8 8 9 9 9 9 9 10 10 10 10 10
>
> rep(1:4, each=5)
[1] 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4
>
> rep(LETTERS[1:4], 3)
[1] "A" "B" "C" "D" "A" "B" "C" "D" "A" "B" "C" "D"
>
> rep(LETTERS[1:4], length.out=3)
[1] "A" "B" "C"
```

課堂練習3: 造出規律的序列

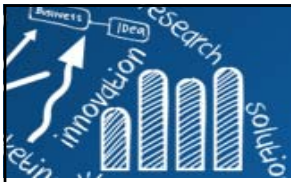
17/50

- 1 2 2 3 3 3 4 4 4 4 5 5 5 5 5
- 5 4 4 3 3 3 2 2 2 2 1 1 1 1 1
- 1 2 3 1 2 3 1 2 3
- Fibonacci number (可能需要用到**for**):
0 1 1 2 3 5 8 13 21 34 55 ...
- 1 2 3 4 5 2 3 4 5 3 4 5 4 5 5
- 1 6 13 22 33 46 ...
- 1 2 3 4 9 8 27 16 ...

Ex: 將 $[0, 2]$ 分成20等份的子區間。

- 取左端點
- 取右端點
- 取子區間之中點

($[a, b]$ 之partition, 用於Riemann Sum)



邏輯向量 (Logical Vectors)

- TRUE, FALSE
- T, F
- Logical operators
 - <, <=, >, >=, ==, !=
- c1,c2: logical expression
 - c1&c2: intersection “and”
 - c1|c2: union “or”

```
> x <- c(12, 4, 7, 20, 13)
> x < 15
[1] TRUE TRUE TRUE FALSE TRUE
> x <= 15
[1] TRUE TRUE TRUE FALSE TRUE
> x > 13
[1] FALSE FALSE FALSE TRUE FALSE
> x >= 10
[1] TRUE FALSE FALSE TRUE TRUE
> x == 12
[1] TRUE FALSE FALSE FALSE FALSE
> x != 20
[1] TRUE TRUE TRUE FALSE TRUE
> (x = 3)
[1] 3
```

```
> (x > 10)
[1] TRUE FALSE FALSE TRUE TRUE
> x != 20
[1] TRUE TRUE TRUE FALSE TRUE
> (x > 10) & (x != 20)
[1] TRUE FALSE FALSE FALSE TRUE
> (x > 10) | (x != 20)
[1] TRUE TRUE TRUE TRUE TRUE
```

```
> (x >= 10)
[1] TRUE FALSE FALSE TRUE TRUE
> 1*(x >= 10)
[1] 1 0 0 1 1
> (x >= 15)
[1] FALSE FALSE FALSE TRUE FALSE
> 2*(x >= 15)
[1] 0 0 0 2 0
```

遺失值 (Missing Values)

- **NA**: not available, missing values

```
> z <- c(1:3, NA)
> z
[1] 1 2 3 NA
> ind <- is.na(z)
> ind
[1] FALSE FALSE FALSE TRUE
```

- A vector of the same length as x all of whose values are NA

```
> x == NA
[1] NA NA NA NA NA
```

- **NaN**: not a number, missing values

```
> 0/0
[1] NaN
> Inf-Inf
[1] NaN
```

See also: `na.fail(x)`, `na.pass(x)`,
`na.omit(x)`, `na.exclude(x)`

`is.na(xx)` is **TRUE** both for **NA** and **NaN** values
`is.nan(xx)` is only **TRUE** for **NaNs**

練習: NA, NaN and Inf

```

> x <- c(NA, 0 / 0, Inf - Inf, Inf, 5) # Inf is a number.
> x
[1] NA NaN NaN Inf 5
> y <- data.frame(x, is.na(x), is.nan(x), x == Inf, x == 5)
> y
  x is.na.x. is.nan.x. x....Inf x....5
1  NA      TRUE      FALSE      NA      NA
2 NaN      TRUE      TRUE      NA      NA
3 NaN      TRUE      TRUE      NA      NA
4 Inf     FALSE     FALSE     TRUE     FALSE
5  5      FALSE     FALSE     FALSE    TRUE
> colnames(y) <- c("x", "is.na(x)", "is.nan(x)", "x == Inf", "x == 5")
> y
  x is.na(x) is.nan(x) x == Inf x == 5
1  NA      TRUE      FALSE      NA      NA
2 NaN      TRUE      TRUE      NA      NA
3 NaN      TRUE      TRUE      NA      NA
4 Inf     FALSE     FALSE     TRUE     FALSE
5  5      FALSE     FALSE     FALSE    TRUE

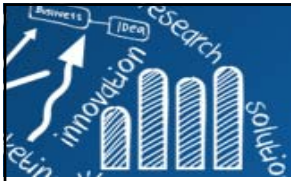
```

字元向量 (Character Vectors)

- Character strings are entered using either
 - double (") quotes or
 - single (') quotes
- Character strings are printed using double quotes.

```
> (x <- "x-values")
[1] "x-values"
> (y <- "New iteration results")
[1] "New iteration results"
> (answer1 <- c("a1", "a2", "b1", "b3"))
[1] "a1" "a2" "b1" "b3"
> (answer2 <- c('a1', 'a2', 'b1', 'b3'))
[1] "a1" "a2" "b1" "b3"
> (answer3 <- c('a', "a2", 3))
[1] "a" "a2" "3"
```

```
> paste("A", 1:6, sep = "")
[1] "A1" "A2" "A3" "A4" "A5" "A6"
> paste("Today is", date())
[1] "Today is Wed Sep 24 13:26:20 2028"
> labs <- paste(c("X", "Y"), 1:10, sep="")
> labs
[1] "X1" "Y2" "X3" "Y4" "X5" "Y6" "X7" "Y8" "X9" "Y10"
```

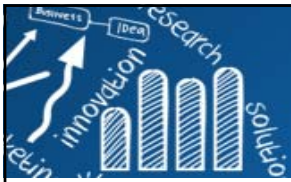


跳脫字元 (Escape Character)

- **Escape Character:**
 - `\n`: new line.
 - `\t`: tab.
 - `\b`: backspace.

```
> cat("How are you?", "\n", "I'm fine.", "\n")
How are you?
I'm fine.
> cat("How are you?", "\t", "I'm fine.", "\n")
How are you?      I'm fine.
> cat("How are you?", "\b\b\b", "I'm fine.")
How are yo I'm fine.>
```

NOTE: "`\`" is entered and printed as "`\\`"
> `setwd("c:\\temp\\mydata")`

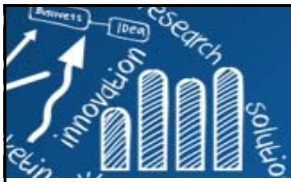


■ A logical vector

```
> x <- c(7, 2, 4, 9, NA, 4)
> x[2]
[1] 2
> x[5]
[1] NA
> x[0]
numeric(0)
> x[10]
[1] NA
> y <- x[!is.na(x)]
> y
[1] 7 2 4 9 4
> (x+1)[(!is.na(x))&(x>0)] -> z
> z
[1] 8 3 5 10 5
```

■ A vector of positive integral quantities

```
> rep(c(1,2,2,1), times=3)
[1] 1 2 2 1 1 2 2 1 1 2 2 1
> c("x","y")[rep(c(1,2,2,1), times=3)]
[1] "x" "y" "y" "x" "x" "y" "y" "x" "x" "y" "y" "x"
```



- A vector of negative integral quantities

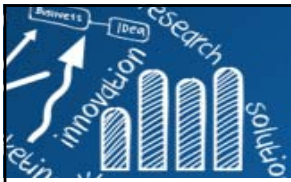
```
> x <- c(7, 2, 4, 9, NA, 4)
> x[-2]
[1] 7 4 9 NA 4
> x[-(1:3)]
[1] 9 NA 4
```

More example

```
> x <- c(7, 2, 4, 9, NA, 4)
> x[is.na(x)] <- 0
> x
[1] 7 2 4 9 0 4
> y <- c(-7, 2, 4, 9, 0, -4)
> abs(y)
[1] 7 2 4 9 0 4
> y[y<0] <- -y[y<0]
> y
[1] 7 2 4 9 0 4
```

- A vector of character strings

```
> 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
```

- 練習「字元向量，跳脫字元，向量索引」

```
> x <- c(A=5, B=3, third=10)
> x
      A      B third
      5      3     10

> x[1]
A
5

> x["A"]
A
5

> x[c("third", "B")]
third      B
      10      3

> x[c(3, 1)]
third      A
      10      5

> names(x)
[1] "A"      "B"      "third"

> names(x) <- c("AA", "BB", "CC")
> x
AA BB CC
  5  3 10
```

因子 (Factors)

- The levels of factors are stored in alphabetical order.

```
> scores <- c(60, 49, 90, 54, 54, 48, 61, 61, 51, 49, 49)
> gender <- c("f", "f", "m", "f", "m", "m", "m", "m", "m", "f", "f", "m")
> levels(gender)
NULL
> gender.f <- factor(gender)
> gender.f
[1] f f m f m m m m f f m
Levels: f m
> levels(gender.f)
[1] "f" "m"
> table(gender.f)
gender.f
f m
5 6
```

```
> levels(gender.f) <- c("女", "男")
> gender.f
[1] 女 女 男 女 男 男 男 男 女 女 男
Levels: 女 男
> (scores.mean <- tapply(scores, gender.f, mean))
  女  男
52.6 60.5
```

```
> grade <- as.factor(c("B", "F", "A", "C", "A", "C", "B", "A", "F", "D"))
> levels(grade)
[1] "A" "B" "C" "D" "F"
> grade2 <- ordered(grade, levels=rev(levels(grade)))
> grade2
[1] B F A C A C B A F D
Levels: F < D < C < B < A
```

```
> grade2[which(grade2 >= "B")]
[1] B A A B A
Levels: F < D < C < B < A
```

因子 (Factors)

```
> MyLetter <- c("C", "D", "A", "K", "A", "I", "J", "I", "K", "H", "A", "K",
"K", "B", "E", "H", "G", "L", "H", "H", "I", "K", "B", "D")
> MyLetter.factor <- factor(MyLetter)
> MyLetter.factor
 [1] C D A K A I J I K H A K K B E H G L H H I K B D
Levels: A B C D E G H I J K L
> table(MyLetter.factor)
MyLetter.factor
A B C D E G H I J K L
3 2 1 2 1 1 4 3 1 5 1
>
> MyLetter.ordered <- factor(MyLetter, levels=c("A", "B", "C", "D", "E", "F",
"G", "H", "I", "J", "K", "L"), ordered=TRUE)
> MyLetter.ordered[1] < MyLetter.ordered[2]
 [1] TRUE
> table(MyLetter.ordered)
MyLetter.ordered
A B C D E F G H I J K L
3 2 1 2 1 0 1 4 3 1 5 1
```

陣列 (Arrays)

- An array is a multiply subscripted collection of data entries.

```
> z <- 1:30
> z
[1] 1 2 3 4 5 6 7 8 9 10 11
12 13 14 15 16 17 18 19 20 21 22 23 24
25 26 27 28
[29] 29 30
>
> dim(z) <- c(3,5,2)
> z
, , 1

      [,1] [,2] [,3] [,4] [,5]
[1,]    1    4    7   10   13
[2,]    2    5    8   11   14
[3,]    3    6    9   12   15

, , 2

      [,1] [,2] [,3] [,4] [,5]
[1,]   16   19   22   25   28
[2,]   17   20   23   26   29
[3,]   18   21   24   27   30
```

```
> z[1,3,2]
[1] 22
>
> z[1,1,]
[1] 1 16
>
> z[1,,2]
[1] 16 19 22 25 28
>
> z[1,1:2,1]
[1] 1 4
>
> z[-1,,]
, , 1

      [,1] [,2] [,3] [,4] [,5]
[1,]    2    5    8   11   14
[2,]    3    6    9   12   15

, , 2

      [,1] [,2] [,3] [,4] [,5]
[1,]   17   20   23   26   29
[2,]   18   21   24   27   30
```

陣列 (Arrays)

```

> x <- array(1:20, dim=c(4,5))
> x
      [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
[2,]    2    6   10   14   18
[3,]    3    7   11   15   19
[4,]    4    8   12   16   20
>
> i <- array(c(1:3, 3:1), dim=c(3,2))
> i
      [,1] [,2]
[1,]    1    3
[2,]    2    2
[3,]    3    1
>
> x[i] <- 0
> x
      [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    0   13   17
[2,]    2    0   10   14   18
[3,]    0    7   11   15   19
[4,]    4    8   12   16   20

```

```

> x <- c(1,2,3,4,5)
> x
[1] 1 2 3 4 5
> z <- array(x, dim=c(3,4))
> z
      [,1] [,2] [,3] [,4]
[1,]    1    4    2    5
[2,]    2    5    3    1
[3,]    3    1    4    2
>
> t(z) #transpose
      [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    1
[3,]    2    3    4
[4,]    5    1    2

```

課堂練習5: interval data

```
> temperature
      January.a January.b February.a February.b
AnQing         1.8         7.1          2.1         7.2
BaoDing        -7.1         1.7         -5.3         4.8
BeiJing        -7.2         2.1         -5.9         3.8
BoKeTu        -23.4        -15.5        -24.0        -14.0
ChangChun     -16.9         -6.7        -17.6         -6.8
> tempArray <- array(0, dim=c(5,2,2))
> tempArray[,,1] <- as.matrix(temperature[,c(1,3)])
> tempArray[,,2] <- as.matrix(temperature[,c(2,4)])
> tempArray
, , 1
     [,1] [,2]
[1,]  1.8  2.1
[2,] -7.1 -5.3
[3,] -7.2 -5.9
[4,] -23.4 -24.0
[5,] -16.9 -17.6

, , 2
     [,1] [,2]
[1,]  7.1  7.2
[2,]  1.7  4.8
[3,]  2.1  3.8
[4,] -15.5 -14.0
[5,]  -6.7  -6.8
```

```
> colnames(tempArray) <- c("January", "February")
> rownames(tempArray) <- rownames(temperature)
> dimnames(tempArray)[[3]] <- c("min", "max")
> dimnames(tempArray)
[[1]]
[1] "AnQing"      "BaoDing"      "BeiJing"      "BoKeTu"
"ChangChun"

[[2]]
[1] "January" "February"

[[3]]
[1] "min" "max"
> tempArray
, , min
     January February
AnQing         1.8         2.1
BaoDing        -7.1        -5.3
BeiJing        -7.2        -5.9
BoKeTu        -23.4       -24.0
ChangChun     -16.9       -17.6

, , max
     January February
AnQing         7.1         7.2
BaoDing         1.7         4.8
BeiJing         2.1         3.8
BoKeTu        -15.5       -14.0
ChangChun      -6.7         -6.8
```

matrix(): 矩陣

- A matrix is an array with two subscripts.

```
> x <- 1:20
> A <- matrix(x, ncol=4)
> A
      [,1] [,2] [,3] [,4]
[1,]    1    6   11   16
[2,]    2    7   12   17
[3,]    3    8   13   18
[4,]    4    9   14   19
[5,]    5   10   15   20
> A.1 <- matrix(x, ncol=4, byrow=TRUE)
> A.1
      [,1] [,2] [,3] [,4]
[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
> nrow(A)
[1] 5
> ncol(A)
[1] 4
```

```
> dim(A)
[1] 5 4
> diag(A)
[1]  1  7 13 19
>
> B <- matrix(x+2, ncol=4)
> A * B #element by element product
      [,1] [,2] [,3] [,4]
[1,]    3   48  143  288
[2,]    8   63  168  323
[3,]   15   80  195  360
[4,]   24   99  224  399
[5,]   35  120  255  440
> A %**% t(B) #matrix product
      [,1] [,2] [,3] [,4] [,5]
[1,]  482  516  550  584  618
[2,]  524  562  600  638  676
[3,]  566  608  650  692  734
[4,]  608  654  700  746  792
[5,]  650  700  750  800  850
```

```
> x <- 4
> diag(x) #identity matrix
```

矩陣 (Matrices)

```

> apply(mat, 1, mean) # row means
[1] 9 10 11 12
> apply(mat, 2, mean) # column means
[1] 2.5 6.5 10.5 14.5 18.5
> apply(mat, 1, var) # row variances
[1] 40 40 40 40
> apply(mat, 2, var) # column variances
[1] 1.666667 1.666667 1.666667 1.666667 1.666667
>
> mean(mat)
[1] 10.5
> var(mat)
      [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 1.666667 1.666667 1.666667 1.666667 1.666667
[2,] 1.666667 1.666667 1.666667 1.666667 1.666667
[3,] 1.666667 1.666667 1.666667 1.666667 1.666667
[4,] 1.666667 1.666667 1.666667 1.666667 1.666667
[5,] 1.666667 1.666667 1.666667 1.666667 1.666667
> summary(mat)
      V1          V2          V3          V4          V5
Min.   :1.00   Min.   :5.00   Min.    : 9.00   Min.   :13.00   Min.   :17.00
1st Qu.:1.75   1st Qu.:5.75   1st Qu.: 9.75   1st Qu.:13.75   1st Qu.:17.75
Median :2.50   Median :6.50   Median :10.50   Median :14.50   Median :18.50
Mean   :2.50   Mean   :6.50   Mean   :10.50   Mean   :14.50   Mean   :18.50
3rd Qu.:3.25   3rd Qu.:7.25   3rd Qu.:11.25   3rd Qu.:15.25   3rd Qu.:19.25
Max.   :4.00   Max.   :8.00   Max.   :12.00   Max.   :16.00   Max.   :20.00

```

```

> mat <- matrix(1:20, ncol=5)
> mat
      [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
[2,]    2    6   10   14   18
[3,]    3    7   11   15   19
[4,]    4    8   12   16   20
> id <- mat[, 2] > 5
> id
[1] FALSE  TRUE  TRUE  TRUE
> mat[id, ]
      [,1] [,2] [,3] [,4] [,5]
[1,]    2    6   10   14   18
[2,]    3    7   11   15   19
[3,]    4    8   12   16   20

```


矩陣的索引

```

> (y <- array(1:15, dim=c(3, 5)))
> dim(y)
[1] 3 5

> x <- matrix(1:15, 3, 5)
> x
      [,1] [,2] [,3] [,4] [,5]
[1,]    1    4    7   10   13
[2,]    2    5    8   11   14
[3,]    3    6    9   12   15
> x[1]
[1] 1
> x[6]
[1] 6
>
> x <- matrix(1:15, 3, 5, byrow=TRUE)
> x
      [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    6    7    8    9   10
[3,]   11   12   13   14   15
> x[1]
[1] 1
> x[6]
[1] 12

```

```

> y[2, 4]
[1] 11
> y[1,]
[1] 1 4 7 10 13
> y[,1]
[1] 1 2 3
> y[2:3, ]
      [,1] [,2] [,3] [,4] [,5]
[1,]    2    5    8   11   14
[2,]    3    6    9   12   15
> y[-2,]
      [,1] [,2] [,3] [,4] [,5]
[1,]    1    4    7   10   13
[2,]    3    6    9   12   15
> y[, -2]
      [,1] [,2] [,3] [,4]
[1,]    1    7   10   13
[2,]    2    8   11   14
[3,]    3    9   12   15
> dimnames(y)
NULL
> rownames(y)
NULL
> colnames(y)
NULL

```

矩陣結合 (Forming Partitioned Matrices)

■ cbind() and rbind()

```
> x <- c(1, 2, 3, 4, 5)
> y <- c(0.5, 0.4, 0.3, 0.2, 0.1)
> (z1 <- cbind(x,y))
```

```
      x   y
[1,] 1 0.5
[2,] 2 0.4
[3,] 3 0.3
[4,] 4 0.2
[5,] 5 0.1
```

```
> (z2 <- rbind(x,y))
      [,1] [,2] [,3] [,4] [,5]
x  1.0  2.0  3.0  4.0  5.0
y  0.5  0.4  0.3  0.2  0.1
```

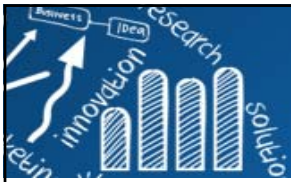
```
> (A <- rbind(x,y))
      [,1] [,2] [,3] [,4]
[1,]  1   6  11  16
[2,]  2   7  12  17
[3,]  3   8  13  18
[4,]  4   9  14  19
[5,]  5  10  15  20
[6,]  3   5   7   9
[7,]  4   6   8  10
```

```
> (x <- matrix(1:20, ncol=4, nrow=5))
      [,1] [,2] [,3] [,4]
[1,]  1   6  11  16
[2,]  2   7  12  17
[3,]  3   8  13  18
[4,]  4   9  14  19
[5,]  5  10  15  20
```

```
> (y <- matrix(3:10, ncol=4))
      [,1] [,2] [,3] [,4]
[1,]  3   5   7   9
[2,]  4   6   8  10
```

```
> (z <- matrix(rep(1:5, 2),nrow=5))
      [,1] [,2]
[1,]  1   1
[2,]  2   2
[3,]  3   3
[4,]  4   4
[5,]  5   5
```

```
> (B <- cbind(x,z))
      [,1] [,2] [,3] [,4] [,5] [,6]
[1,]  1   6  11  16   1   1
[2,]  2   7  12  17   2   2
[3,]  3   8  13  18   3   3
[4,]  4   9  14  19   4   4
[5,]  5  10  15  20   5   5
```



Some matrix functions

t	Transpose
diag	Diagonal
.*%	Inner (dot) product of two vectors $x^t y$, matrix multiplication
o%	Outer product of two vectors xy^t
crossprod, tcrossprod	Cross products $x^t y$ and xy^t of matrices
det	Determinant
solve	Inverse
eigen	Eigenvalues and eigenvectors
svd	Singular value decomposition
qr	QR decomposition
chol	Choleski decomposition

list(): 表列

- List is an object consisting of an ordered collection of objects known as its components.
- A list could consist of a numeric vector, a logical value, a matrix, a complex vector, a character array, a function, and so on.
- 許多 R 統計函式回傳值皆是 list。

```
> my.list <- list(name="George",
                  wife="Mary",
                  no.children=3,
                  child.ages=c(4,7,9))

> my.list
$name
[1] "George"

$wife
[1] "Mary"

$no.children
[1] 3

$child.ages
[1] 4 7 9
```

Construct list:

```
my.list <- list(name_1=object_1,...,name_m=object_m)
lst.ABC <- list(list.A, list.B, list.C)
```

see also: <http://faculty.nps.edu/sebuttre/home/R/lists.html>

表列 (List)

`lst[1]` VS `lst[[1]]`

- `[]`: a general subscripting.
- `[[]]`: the operator used to select **a single element**.
- `[1]`: a sublist of the **list** consisting of the first entry. (name are included in the sublist).
- `[[1]]`: first object in the list, **exclude name**.

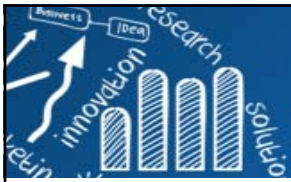
```
> my.list
$name
[1] "George"

$wife
[1] "Mary"

$no.children
[1] 3

$child.ages
[1] 4 7 9
```

```
> my.list[[1]] #傳回向量
[1] "George"
> my.list[[2]]
[1] "Mary"
> my.list[[4]][1]
[1] 4
> > getElement(mylist, "name")
>
> my.list$name #my.list[[1]]
#my.list[["name"]]
[1] "George"
> my.list$wife #my.list[[2]]
[1] "Mary"
> my.list$child.ages[1] #my.list[[4]][1]
[1] 4
> x <- "name"
> my.list[[x]]
[1] "George"
>
> my.list[1] #傳回list
$name
[1] "George"
> my.list[2]
$wife
[1] "Mary"
```



課堂練習6

```
my.list <- list(name=c("George", "John", "Tom"),
               wife=c("Mary", "Sue", "Nico"),
               no.children=c(3, 2, 0),
               child.ages=list(c(4,7,9), c(2, 5), NA))
```

```
> my.list$name
[1] "George" "John"   "Tom"
> my.list$wife
[1] "Mary" "Sue"  "Nico"
> my.list$no.children
[1] 3 2 0
> my.list$name[3]
[1] "Tom"
> my.list$name == "John"
[1] FALSE TRUE FALSE
> my.list$child.ages
[[1]]
[1] 4 7 9

[[2]]
[1] 2 5

[[3]]
[1] NA
```

```
> my.list$child.ages[2]
[[1]]
[1] 2 5
> my.list$child.ages[[2]]
[1] 2 5
> my.list$child.ages[2][1]
[[1]]
[1] 2 5
> my.list$child.ages[[2]][1]
[1] 2
> my.list$child.ages[[2]][2]
[1] 5
> length(my.list)
[1] 4
```

```
> my.list[[c(2, 3)]]
[1] "Nico"
> my.list[c(2, 3)]
$wife
[1] "Mary" "Sue"  "Nico"

$no.children
[1] 3 2 0
```

資料框 (Data Frame)

- A data frame is a list with class "data.frame".
- Regarded as a matrix with column possibly of differing modes and attributes.

```
> my.matrix <- matrix(1:15, ncol=3)
> my.matrix
      [,1] [,2] [,3]
[1,]    1    6   11
[2,]    2    7   12
[3,]    3    8   13
[4,]    4    9   14
[5,]    5   10   15
> my.data <- data.frame(my.matrix)
> my.data
  X1 X2 X3
1  1  6 11
2  2  7 12
3  3  8 13
4  4  9 14
5  5 10 15
```

```
> my.data[1, ]
  X1 X2 X3
1  1  6 11
> my.data[2, 3]
[1] 12
> my.data$X1
[1] 1 2 3 4 5
> my.data[, "X1"]
[1] 1 2 3 4 5
> my.data["X1"]
  X1
1  1
2  2
3  3
4  4
5  5
> rownames(my.data)
[1] "1" "2" "3" "4" "5"
> row.names(my.data)
[1] "1" "2" "3" "4" "5"
> colnames(my.data)
[1] "X1" "X2" "X3"
> names(my.data)
[1] "X1" "X2" "X3"
```

資料框 (Data Frame)

```
> rownames(my.data) <- c(paste("s", 1:5, sep="."))
> colnames(my.data) <- c("A1", "A2", "A3")
> my.data
  A1 A2 A3
s.1  1  6 11
s.2  2  7 12
s.3  3  8 13
s.4  4  9 14
s.5  5 10 15
```

```
> subjects <-c ('Chinese', 'Math', 'English')
> scores <- c(50, 90, 61)
> pass <- scores >= 60
> student <- data.frame(subjects, scores, pass)
> student
  subjects scores  pass
1  Chinese     50 FALSE
2    Math     90  TRUE
3  English     61  TRUE
> student["2",] # use row names to extract records for no.2
  subjects scores pass
2    Math     90  TRUE
> student[, "scores"] # use column names to extract values for "scores"
[1] 50 90 61
```

```
> attach(my.data)
> A1
[1] 1 2 3 4 5
> A2
[1] 6 7 8 9 10
> A3
[1] 11 12 13 14 15
```

```
> detach()
```

```
> A1
```

```
Error: object "A1" not found
```


列資料選取

```
> index.1 <- iris[, "Species"] == "virginica"
> iris[index.1, ]
      Sepal.Length Sepal.Width Petal.Length Petal.Width Species
101           6.3         3.3         6.0         2.5 virginica
102           5.8         2.7         5.1         1.9 virginica
...

> iris[Species == "virginica",]
      Sepal.Length Sepal.Width Petal.Length Petal.Width Species
101           6.3         3.3         6.0         2.5 virginica
102           5.8         2.7         5.1         1.9 virginica
...

> iris[!(Species == "virginica"),]
      Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1             5.1         3.5         1.4         0.2   setosa
2             4.9         3.0         1.4         0.2   setosa
...

> m <- mean(iris$Sepal.Length)
> index.3 <- iris[, "Sepal.Length"] > m
> iris[index.3, ]
      Sepal.Length Sepal.Width Petal.Length Petal.Width Species
51             7.0         3.2         4.7         1.4 versicolor
52             6.4         3.2         4.5         1.5 versicolor
...
```

mode(object): 物件的模式

- Mode: "logical", "integer", "double", "complex", "raw", "character", "list", "expression", "name", "symbol" and "function".

```
> mode(object)
```

- Vector must have their values all of the same mode.
 - Empty character string vector: `character(0)`.
 - Empty numeric vector: `numeric(0)`.

NOTE: Lists are of mode list. There are ordered sequences of objects which individually can be of any mode.

```
> (z <- 0:9)
[1] 0 1 2 3 4 5 6 7 8 9
> mode(z)
[1] "numeric"
> (digits <- as.character(z))
[1] "0" "1" "2" "3" "4" "5" "6" "7" "8" "9"
> mode(digits)
[1] "character"
> (d <- as.integer(digits))
[1] 0 1 2 3 4 5 6 7 8 9
> mode(d)
[1] "numeric"
> (x <- z[1:5]>3)
[1] FALSE FALSE FALSE FALSE TRUE
> mode(x)
[1] "logical"
```

class(object): 物件的類別

- All objects in R have a class.
 - For simple vector, mode=class: `numeric`, `logical`, `character`, `list`.
 - `matrix`, `array`, `factor`, `data.frame`

```
> x1 <- 10
> class(x1)
[1] "numeric"
> (x2 <- seq(1, 10, 2))
[1] 1 3 5 7 9
> class(x2)
[1] "numeric"
```

```
> my.f <- formula(iris$Sepal.Length ~ iris$Sepal.Width)
> class(my.f)
[1] "formula"
> class(lm(my.f))
[1] "lm"
> class(aov(my.f))
[1] "aov" "lm"
```

```
> class(iris)
[1] "data.frame"
> (iris.sub <- iris[5:10, 1:4])
  Sepal.Length Sepal.Width Petal.Length Petal.Width
5             5.0         3.6           1.4         0.2
6             5.4         3.9           1.7         0.4
7             4.6         3.4           1.4         0.3
8             5.0         3.4           1.5         0.2
9             4.4         2.9           1.4         0.2
10            4.9         3.1           1.5         0.1
> class(iris.sub)
[1] "data.frame"
>
> class(as.matrix(iris.sub))
[1] "matrix"
```

```
> as.list(iris.sub)
$Sepal.Length
[1] 5.0 5.4 4.6 5.0 4.4 4.9

$Sepal.Width
[1] 3.6 3.9 3.4 3.4 2.9 3.1

$Petal.Length
[1] 1.4 1.7 1.4 1.5 1.4 1.5

$Petal.Width
[1] 0.2 0.4 0.3 0.2 0.2 0.1

> class(as.list(iris.sub))
[1] "list"
```

class(object): 物件的類別

```
> ex1 <- expression(1 + 0:9) # expression object
> ex1
expression(1 + 0:9)
> eval(ex1)
 [1] 1 2 3 4 5 6 7 8 9 10
> class(ex1)
[1] "expression"
>
>
> hi <- function(){
+   cat("hello world!\n")
+ }
> hi()
hello world!
> class(hi) # function object
[1] "function"
```

"There is a special object called **NULL**. It is used whenever there is a need to indicate or specify that an object is absent. It should not be confused with a vector or list of zero length.

The **NULL** object has no type and no modifiable properties. There is only one **NULL** object in R, to which all instances refer. To test for **NULL** use `is.null`. You cannot set attributes on **NULL**. "

attributes(object): 物件的屬性

All objects except NULL can have one or more attributes attached to them.

- Select a specific attribute

```
> attr(object, name)
```

- Set a specific attribute

```
> attr(z, "dim") <- c(10, 10)
```

```
> x <- matrix(1:10, ncol=2)
> x
      [,1] [,2]
[1,]    1    6
[2,]    2    7
[3,]    3    8
[4,]    4    9
[5,]    5   10
> attributes(x)
$dim
[1] 5 2

> attr(x, "dim")
[1] 5 2
> dim(x)
[1] 5 2
```

```
> x <- data.frame(matrix(1:10, ncol=2))
> x
  x1 x2
1  1  6
2  2  7
3  3  8
4  4  9
5  5 10
> attributes(x)
$names
[1] "x1" "x2"

$row.names
[1] 1 2 3 4 5

$class
[1] "data.frame"

> attr(x, "names")
[1] "x1" "x2"
> names(x)
[1] "x1" "x2"
```

```
> gender.f
[1] 女女男女男男男男女女男
Levels: 女 男
> str(gender.f)
Factor w/ 2 levels "女","男":
1 1 2 1 2 2 2 2 1 1 ...
> class(gender.f)
[1] "factor"
> attributes(gender.f)
$levels
[1] "女" "男"

$class
[1] "factor"
```

length(object): 物件的長度

```
> beta <- c(1, 3, 5, 2, 4, 6, 11, NA, NA, 22)
> length(beta)
[1] 10
> length(beta[!is.na(beta)])
[1] 8
>
> e <- numeric() # empty object
> e[3] <- 17
> length(e)
[1] 3
> e
[1] NA NA 17
>
> (alpha <- numeric(10))
[1] 0 0 0 0 0 0 0 0 0 0
> length(alpha)
[1] 10
> alpha <- alpha[2*1:5]
> length(alpha)
[1] 5
> length(alpha) <- 3
> alpha
[1] 0 0 0
```

```
> mye <- expression(x, {y <- x^2; y+2}, x^y)
> length(mye)
[1] 3
> str(mye)
expression(x, {      y <- x^2      y + 2 }, x^y)
```

```
> myf <- formula(y ~ x1 + x2 + x3)
> length(myf)
[1] 3
> str(myf)
Class 'formula' language y ~ x1 + x2 + x3
..- attr(*, ".Environment")=<environment: R_GlobalEnv>
> myf[1]
`~`()
> myf[2]
y()
> myf[3]
(x1 + x2 + x3)()
> myf[4]
Error in if (length(ans) == 0L ...
需要 TRUE/FALSE 值的地方有缺值
```

```
> mye[1]
expression(x)
> mye[2]
expression({
  y <- x^2
  y + 2
})
> mye[3]
expression(x^y)
> mye[4]
expression(NULL)
```

str(object): 物件之結構

```
> x <- 1:12
> str(x)
int [1:12] 1 2 3 4 5 6 7 8 9 10 ...
>
> ch <- letters[1:12]
> str(ch)
chr [1:12] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l"
>
> str(iris)
'data.frame': 150 obs. of 5 variables:
 $ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species      : Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 ...
>
> str(ls) # try > str(str)
function (name, pos = -1L, envir = as.environment(pos), all.names = FALSE,
pattern, sorted = TRUE)
> str(str)
function (object, ...)
>
> myp <- plot(iris[,1], iris[,2])
> str(myp)
NULL
```

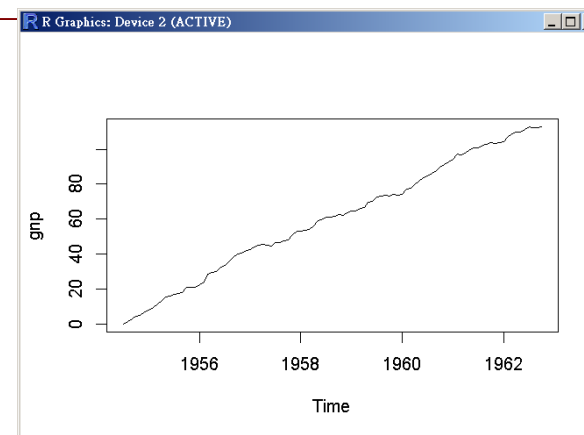
str(object): 物件之結構

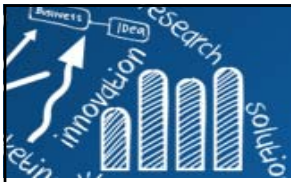
```
> my.f <- iris$Sepal.Length ~ iris$Sepal.Width
> str(my.f)
Class 'formula' language iris$Sepal.Length ~ iris$Sepal.Width
 ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
> my.lm <- lm(my.f)
> str(my.lm)
List of 12
 $ coefficients : Named num [1:2] 6.526 -0.223
 ..- attr(*, "names")= chr [1:2] "(Intercept)" "iris$Sepal.Width"
 ...
 .. .. .. ..- attr(*, "names")= chr [1:2] "iris$Sepal.Length" ...
 - attr(*, "class")= chr "lm"
> my.lm.s <- summary(my.lm)
> str(my.lm.s)
List of 11
 $ call          : language lm(formula = my.f)
 $ terms         :Classes 'terms', 'formula' language iris$Sepal.Length ~
iris$Sepal.Width
 ...
 .. ..$ : chr [1:2] "(Intercept)" "iris$Sepal.Width"
 - attr(*, "class")= chr "summary.lm"
>
> mye <- expression(x, {y <- x^2; y+2}, x^y)
> str(mye)
expression(x, {      y <- x^2      y + 2 }, x^y)
```


時間序列物件: `ts`

```
ts(data = NA, start = 1, end = numeric(), frequency = 1,
    deltat = 1, ts.eps = getOption("ts.eps"), class = , names = )
as.ts(x, ...)
is.ts(x)
```

```
> ts(1:10, frequency = 4, start = c(1959, 2))
      Qtr1 Qtr2 Qtr3 Qtr4
1959         1     2     3
1960         4     5     6     7
1961         8     9    10
> my.ts <- ts(1:10, frequency = 7, start = c(12, 2))
> class(my.ts)
[1] "ts"
> print(my.ts, calendar = TRUE)
      p1 p2 p3 p4 p5 p6 p7
12      1  2  3  4  5  6
13      7  8  9 10
> gnp <- ts(cumsum(1+round(rnorm(100), 2))), start = c(1954, 7), frequency = 12)
> gnp
      Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct   Nov   Dec
1954                -0.12  1.62  3.13  4.36  4.78  6.81
1955  7.98  9.62 11.26 12.80 15.25 15.88 17.13 17.72 18.04 20.77 21.01 21.22
. . .
1962 104.41 106.88 108.87 109.82 109.62 111.52 112.70 112.29 112.48 112.84
> plot(gnp)
```





Multivariate **ts**

```
> z <- ts(matrix(rnorm(300), 100, 3), start = c(1961, 1), frequency = 12)
> head(z, 3)
      Series 1 Series 2 Series 3
[1,] -0.008998503 0.5389408 -0.9403586
[2,] -0.750712987 0.3026561 -0.1112974
[3,] -2.086179305 0.6752907  0.8359952
> tail(z, 3)
      Series 1 Series 2 Series 3
[98,]  1.6249153 -0.8999009  0.12837969
[99,]  0.6174681 -0.8451825  0.86245135
[100,] 0.5894715 -0.2738029 -0.05433789
>
> class(z)
[1] "mts"      "ts"        "matrix"
> plot(z)
> plot(z, plot.type = "single", lty = 1:3)
```

