中研院統計所2015統計研習營
2015/07/01~03

(4) 編寫R函式(Functions)

吳漢銘
淡江大學 數學系
資料科學與數理統計組

http://www.hmwu.idv.tw

本章大綱&學習目標

- 條件判別與執行: if else
- 撰寫自訂函式: function()
- 外顯迴圈: for, while, repeat
- 迴圈的控制: next, break, switch
- 隱含迴圈: apply, tapply, lapply, sapply
- 樣式比對、搜尋與替換
- which
- 集合運算
- 日期時間
- 排序: Rank, Sort and Order
- 其它

Grouped Expression (Block)

```
{expr_1; ...; expr_m}
```

or

```
{
    expr_1
    ...
    expr_m
}
```

```
{ a <- c(1,2,3); b <- 5 }

{
  a <- c(1,2,3)
  b <- 5
  c <- sum(a, b)
}
```

工作第一步，設定工作目錄

```
> getwd()
[1] "C:/Documents and Settings/user/My Documents“

> setwd("C:\\Program Files\\R\\working")
```

條件執行: `if`

```
if (expr.1) expr.2 else expr.3
```

- `expr.1` is evaluated to yield **value1**.
- If **value1** is a *logical vector*:
  - first element of **value1** is `TRUE` then `expr.2` is evaluated.
  - first element of **value1** is `FALSE` then `expr.3` is evaluated.
- If `expr.1` is a *numeric vector*:
  - first element of **value1** is `zero,` then `expr.3` is evaluated
  - first element of **value1** is `non-zero,` then `expr.2` evaluated.
- Only the first element of **value1** is used.
- If **value1** has any other type, an error is signaled.

課堂練習1: If value1 is a logical vector

```
> x <- 1
> if((x-2) < 0) cat("expr2 \n") else cat("expr3 \n")
expr2
>
> if((x-2) > 0) cat("expr2 \n") else cat("expr3 \n")
expr3
```

```
> x <- c(-1, 2, 3)
> if((x-2) < 0) cat("expr2 \n") else cat("expr3 \n")
expr2
Warning message:
In if ((x - 2) < 0) cat("expr2 \n") else cat("expr3 \n") :
  the condition has length > 1 and only the first element will be used

> if((x-2) > 0) cat("expr2 \n") else cat("expr3 \n")
expr3
Warning message:
In if ((x - 2) > 0) cat("expr2 \n") else cat("expr3 \n") :
  the condition has length > 1 and only the first element will be used
```

課堂練習2: If expr.1 is a numeric vector

```
> x <- 0
> if(x) cat("expr2 \n") else cat("expr3 \n")
expr3
> if(x+1) cat("expr2 \n") else cat("expr3 \n")
expr2
>
```

```
> x <- c(-1, 0, 1, 2,3)
> if(x) cat("expr2 \n") else cat("expr3 \n")
expr2
Warning message:
In if (x) cat("expr2 \n") else cat("expr3 \n") :
  the condition has length > 1 and only the first element will be used

> if(x+1) cat("expr2 \n") else cat("expr3 \n")
expr3
Warning message:
In if (x + 1) cat("expr2 \n") else cat("expr3 \n") :
  the condition has length > 1 and only the first element will be used
```

課堂練習3

```
> x <- c(-1, 2, 3)
> if(any(x <=0)) y <- log(1+x) else y <- log(x)
> y
[1]      -Inf 1.098612 1.386294
> z <- if(any(x<=0)) log(1+x) else log(x)
> z
[1]      -Inf 1.098612 1.386294
```

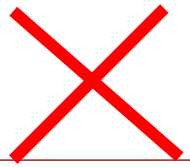
```
all() #return TRUE if all values are TRUE
any() #return TRUE if any values are TRUE
```

```
> x <- c(-1, 2, 3)
> if(any(x <=0)){
+      y <- log(1+x)
+ } else{
+     y <- log(x)
+ }
> y
[1]      -Inf 1.098612 1.386294
>
>
>
>
> x <- c(-1,2,3)
> if(any(x <=0)){
+      y <- log(1+x)
+ }
> else{
Error: unexpected 'else' in "else"
>     y <- log(x)
Warning message:
In log(x) : NaNs produced
> }
Error: unexpected '}' in "}"
> y
[1]       NaN 0.6931472 1.0986123
```

這種寫法比較好

(程式編輯器)

```
x <- c(-1,2,3)
if(any(x <=0)){
    y <- log(1+x)
} else{
   y <- log(x)
}
```



```
x <- c(-1,2,3)
if(any(x <=0)){
    y <- log(1+x)
}
else{
   y <- log(x)
}
```

條件判斷

- apply element-wise to vectors
  - `&: #and`
  - `|: #or`
- apply to vector
  - `&&: #and`
  - `||: #or`

- 若運算對象是一個數字變數，則 &&,|| 和 &, | 沒有差別。
- 使用&結合兩個條件，傳回真假值判別向量。
- 使用&&結合兩個條件，只傳回判別向量的第一個真假值元素。

- use "`==`" in `if` instead of "`=`"

```
if(cond1 && cond2){
        ...
}

if(cond1 || cond2){
        ...
}

if(cond1 & cond2){
        ...
}

if(cond1 | cond2){
        ...
}

if(expre2 == expre1){
        ...
}
```

課堂練習4.1

```
> x <- 3
> y <- 4
>
> x < 2
[1] FALSE
> y > 2
[1] TRUE
> x < 2 || y > 2
[1] TRUE

> x > 2
[1] TRUE
> y > x
[1] TRUE
> x > 2 && y > x
[1] TRUE
```

```
> x < 2 | y > 2
[1] TRUE

> x > 2 & y > x
[1] TRUE
```

```
> xv <- c(1,2,3)
> yv <- c(2,2,5)

> xv < 2
[1] TRUE FALSE FALSE
> yv > 2
[1] FALSE FALSE  TRUE

> xv < 2 || yv > 2
[1] TRUE
> (! xv < 2) || yv > 2
[1] FALSE
> xv < 2 || (! yv > 2)
[1] TRUE

> xv < 2 && yv > 2
[1] FALSE
> (! xv < 2) && yv > 2
[1] FALSE
> xv < 2 && (! yv > 2)
[1] TRUE
```

```
> xv < 2 | yv > 2
[1]  TRUE FALSE  TRUE
> (! xv < 2) | yv > 2
[1] FALSE  TRUE  TRUE
> xv < 2 | (! yv > 2)
[1]  TRUE  TRUE FALSE

> xv < 2 & yv > 2
[1] FALSE FALSE FALSE
> (! xv < 2) & yv > 2
[1] FALSE FALSE  TRUE
> xv < 2 & (! yv > 2)
[1]  TRUE FALSE FALSE
```

巢狀 if/else: Nested if/else

```
if(expr_1) expr_2
else if(expr_3) expr_4
else if(expr_5) expr_6
else expr_7
```

```
a <- 2.13

if( a > 10 ){
    cat("a > 10 \n")
}else if(a > 5){
    cat("5 < a < 10 \n")
}else if(a > 2.5){
    cat("2.5 < a < 5 \n")
}else if(a > 1.25){
    cat("1.25 < a < 2.5 \n")
}else{
    cat("a < 1.25")
}

1.25 < a < 2.5
```

ifelse(condition, a ,b)

Return a vector of the length of its longest argument, with elements a[i] if condition[i] is true, otherwise b[i].

```
> (x <- c(2:-1))
[1]  2  1  0 -1

> sqrt(x)
[1] 1.414214 1.000000 0.000000      NaN
Warning message:
In sqrt(x) : NaNs produced

> sqrt(ifelse(x >= 0, x, NA))
[1] 1.414214 1.000000 0.000000       NA

> ifelse(x >= 0, sqrt(x), NA)
[1] 1.414214 1.000000 0.000000       NA
Warning message:
In sqrt(x) : NaNs produced
```

```
> (yes <- 5:6)
[1] 5 6
> (no <- pi^(0:2))
[1] 1.000000 3.141593 9.869604
>
> ifelse(NA, yes, no)
[1] NA
> ifelse(TRUE, yes, no)
[1] 5
> ifelse(FALSE, yes, no)
[1] 1
>
> typeof(ifelse(NA, yes, no))
[1] "logical"
> typeof(ifelse(TRUE, yes, no))
[1] "integer"
> typeof(ifelse(FALSE, yes, no))
[1] "double"
```

```
> x
 [1] 24 13 26 21  7  9  2  1 30 14 20 16  6  4 12  8 11 22 18  3
> ifelse(x <= 10, 1, ifelse(x <= 20, 2, 3))
 [1] 3 2 3 3 1 1 1 1 3 2 2 2 1 1 2 1 2 3 2 1
```

課堂練習4.2

- 將年齡資料轉換為年齡群組1~20, 21~40, 41~60, 61歲以上，並編碼為A, B, C, D。

```
> set.seed(12345)
> age <- sample(1:100, 20)
> age
 [1] 73 87 75 86 44 16 31 48 67 91  4 14 65  1 34 40 33 97 15 78
```

- 將”A”與”E”編碼為1，”C”編碼為2，”B”與”D”編碼為3。

```
> set.seed(12345)
> code <- sample(LETTERS[1:5], 20, replace=T)
> code
 [1] "D" "E" "D" "E" "C" "A" "B" "C" "D" "E" "A" "A" "D" "A" "B" "C"
[17] "B" "C" "A" "E"
```

see also: **cut(), recode{car}**

撰寫自訂函式: function()

```
function.name <- function(input.var1, input.var2){
   output.var1 <- expre.1
   command1
   ...
   output
}

function.name <- function(input.var1, input.var2=value){
   output.var1 <- expre.1
   command1
   ...
   output.var2 <- expre.2
   return(list(output.name1=output.var1, output.name2=output.var2))
}
```

函式呼叫: call function

```
> function.name(input.var1, input.var2)
```

引數及內定值

引數arguments: "name = object"

```
fun1 <- function(data, data.frame, is.graph, limit){…}

> ans <- fun1(data=d, data.frame=df, is.graph=TRUE, limit=20)
> ans <- fun1(d, df, TRUE, 20)
> ans <- fun1(d, df, is.graph=TRUE, limit=20)
> ans <- fun1(data=d, limit=20, is.graph=TRUE, data.frame=df)
```

內定值 (Defaults)

```
fun1 <- function(data, data.grame, is.graph=TRUE, limit=20){…}

> ans <- fun1(d, df)
> ans <- fun1(d, df, limit=10)
```

函式之回傳值

```
> min(5:1, pi)
[1] 1
> pmin(5:1, pi)
[1] 3.141593 3.141593 3.000000 2.000000 1.000000
```

```
parmax <- function(a, b){
    c <- pmax(a,b)
    median(c)
}
> x <- c(1,9,2,8,3,7)
> y <- c(9,2,8,3,7,2)
> parmax(x,y)
[1] 8
```

```
data.ratio <- function(x){
    x.number <- length(x)
    x.up <- mean(x) + sd(x)
    x.down <- mean(x) - sd(x)
    x.n <- length(x[x.down < x & x < x.up])
    x.p <- x.n/x.number
    list(number=x.n, percent=x.p)
}


> data.ratio(iris[,1])
$number
[1] 90

$percent
[1] 0.6
```

課堂練習5

```
compute <- function(a, b=0.5){

    sum <- a+b
    diff <- a-b
    prod <- a*b
    if(b!=0){
      div <- a/b
    }else{
      div <- "divided by zero"
    }
    return(list(sum=sum, diff=diff, product=prod, divide=div))
}
```

```
> norm <- function(x) sqrt(x%*%x)
> norm(1:4)
         [,1]
[1,] 5.477226
```

```
> compute(2, 5)
$sum
[1] 7

$diff
[1] -3

$product
[1] 10

$divide
[1] 0.4
```

```
> compute(2)
$sum
[1] 2.5

$diff
[1] 1.5

$product
[1] 1

$divide
[1] 4
```

```
> compute(2, 0)
$sum
[1] 2

$diff
[1] 2

$product
[1] 0

$divide
[1] "divided by zero"
```

課堂練習6: 兩樣本之t檢定

```
two.sample.test <- function(y1, y2){
    n1 <- length(y1); n2 <- length(y2)
    m1 <- mean(y1); m2 <- mean(y2)
    s1 <- var(y1); s2 <- var(y2)
    s <- ((n1-1)*s1 + (n2-1)*s2)/(n1+n2-2)
    stat <- (m1-m2)/sqrt(s*(1/n1+1/n2))
    return(list(means=c(m1, m2), pool.var=s, stat=stat))
}

> t.stat <- two.sample.test(iris[,1], iris[,2]);
> t.stat
> t.stat
$means
[1] 5.843333 3.057333

$pool.var
[1] 0.4378365

$stat
[1] 36.46328
```

函式內的變數

- Any ordinary assignments done within the function are local and temporary and are lost after exit from the function.

```
> rm(list=ls())
> my.sqrt.sum <- function(x, y){
    a <- sqrt(x)
    b <- sqrt(y)
    c <- a+b
    return(c)
}

> a <- 4
> b <- 9
> my.sqrt.sum(a, b)
[1] 5
> a
[1] 4
> b
[1] 9
```

```
> rm(list=ls())
> my.sqrt.sum <- function(x, y){
    a <- sqrt(x)
    b <- sqrt(y)
    c <- a+b
    return(c)
}
> my.sqrt.sum(4, 9)
[1] 5
> a
Error: object "a" not found
> b
Error: object "b" not found
```

函式內的變數

```
> rm(list=ls())
> y <- 9
> my.sqrt.sum <- function(x){
    a <- sqrt(x)
    b <- sqrt(y)
    y <- sqrt(y)
    c <- a+b
    return(c)
}
> my.sqrt.sum(4)
[1] 5
> a
Error: object "a" not found
> b
Error: object "b" not found
> y
[1] 9
```

```
rm(list=ls())
Y.VALUE <- 9
my.sqrt.sum <- function(x){
    a <- sqrt(x)
    b <- sqrt(Y.VALUE)
    c <- a+b
    return(c)
}
my.sqrt.sum(4)
[1] 5
```

```
rm(list=ls())
my.sqrt.sum <- function(x, y){
    x <- sqrt(x)
    y <- sqrt(y)
    c <- x+y
    return(c)
}

>  x <- 4
>  y <- 9
>  x <- my.sqrt.sum(x, y)
>  x
[1] 5
>  y
[1] 9
```

課堂練習7.1: <<-

```
myfun1 <- function(x){
   y <- x + 5
   cat("y: ", y, "\n")
}

myfun2 <- function(x){
   y <<- x + 5
   cat("y: ", y, "\n")
}

y <- 5; cat("y: ", y, "\n")
myfun1(3)
cat("y: ", y, "\n")

y <- 5; cat("y: ", y, "\n")
myfun2(3)
cat("y: ", y, "\n")
```

```
> myfun1 <- function(x){
+     y <- x + 5
+     cat("y: ", y, "\n")
+ }
>
> myfun2 <- function(x){
+     y <<- x + 5
+     cat("y: ", y, "\n")
+ }
>
> y <- 5; cat("y: ", y, "\n")
y:  5
> myfun1(3)
y:  8
> cat("y: ", y, "\n")
y:  5
>
>
> y <- 5; cat("y: ", y, "\n")
y:  5
> myfun2(3)
y:  8
> cat("y: ", y, "\n")
y:  8
```

課堂練習7.2

計算數列x的個數，平均及標準差。

```
my.stat <- function(x){
    x.number <- length(x)
    x.mean <- mean(x)
    x.sd <- sd(x)
    list(number=x.number, mean=x.mean,
sd=x.sd)
}

> my.stat(iris[,1])
$number
[1] 150

$mean
[1] 5.843333

$sd
[1] 0.8280661
```

'...'參數

- Allows the function to accept additional arguments of unspecified name and number.
- If a function has '...' as a formal argument then any actual arguments that do not match a formal argument are matched with '...'.
- '...' is used in the argument list to specify that an arbitrary number of arguments are to be passed to the function.

```
> lm
function (formula, data, subset, weights, na.action, method = "qr",
    model = TRUE, x = FALSE, y = FALSE, qr = TRUE, singular.ok = TRUE,
    contrasts = NULL, offset, ...)
```

```
> myfun <- function(x, ...){
+     y <- mean(...) + x
+     y
+ }
>
> data <- rnorm(40)
> myfun(6, data)
[1] 5.997225
```

課堂練習7.3

Here is a function that takes any number of vectors and calculates their means and variances.

```
many.means <- function(...){

    #use [[]] subscripts in addressing its elements.
    data <- list(...)
    n <- length(data)
    means <- numeric(n)
    vars  <- numeric(n)
    for(i in 1:n){
        means[i] <-  mean(data[[i]])
       vars[i] <-  var(data[[i]])
    }

    print(means)
    print(vars)
}

> x <- rnorm(100); y <- rnorm(200); z <- rnorm(300)
> many.means(x,y,z)
[1] -0.007530678  0.031621030  0.026945631
[1] 0.8479211 0.9526169 1.1456980
```

課堂練習8

k為一常數，計算數列x在[mean(x)-k*sd(x), mean(x)+k*sd(x)]間的個數和比例。

```
data.kratio <- function(x, k=1){
   x.number <- length(x)
   x.mean <- mean(x)
   x.sd <- sd(x)
   x.up <- x.mean + k*x.sd;
   x.down <- x.mean - k*x.sd;
   x.n <- length(x[(x.down < x) & (x < x.up)])
   x.p <- x.n/x.number
   list(number=x.n, percent=x.p)
}
library(MASS)
data.kratio(drivers, 1)
data.kratio(drivers, 2)
data.kratio(drivers, 3)
```

```
> library(MASS)
> data.kratio(drivers, 1)
$number
[1] 134

$percent
[1] 0.6979167

> data.kratio(drivers, 2)
$number
[1] 185

$percent
[1] 0.9635417

> data.kratio(drivers, 3)
$number
[1] 191

$percent
[1] 0.9947917
```

迴圈 (Looping)

- 外顯迴圈(Explicit looping):
  `for, while, repeat`

- 隱含迴圈(Implicit looping):
  `apply, tapply, lapply, sapply`

for 迴圈

```
> for (name in expr_1) expr.2
```

- **name**: loop variable.
- **expr.1**: can be either a vector or a list.
- for each element in **expr.1** the variable name is set to the value of that element and **expr.2** is evaluated.

執行「多次有規律性的指令」

```
for(i in 1:5){
    cat("loop: ", i, "\n")
}
```

```
loop:  1
loop:  2
loop:  3
loop:  4
loop:  5
```

```
for(k in c(1, 17, 3, 56, 2)){
    cat(k, "\t")
}

for(bloodType in c("A", "AB", "B", "O")){
    cat(bloodType, "\t")
}
```

for 迴圈

```
rm(list=ls())
y <- round(rnorm(10), 2)
z <- y
y
i
for(i in 1:length(y)){
        if(y[i] < 0)
        y[i]<-0
}
y
i

z[z<0] <- 0
z
```

- side effect: the variable name still exists after the loop has concluded and it has the value of the last element of vector that the loop was evaluated for.

```
> rm(list=ls())
> y <- round(rnorm(10), 2)
> z <- y
> y
 [1]  1.04  1.74 -0.05 -0.44 -0.71 -0.57  0.11 -0.06  0.32 -0.76
> i
Error: object "i" not found
> for(i in 1:length(y)){
+ if(y[i] < 0)
+ y[i]<-0
+ }
> y
 [1] 1.04 1.74 0.00 0.00 0.00 0.00 0.11 0.00 0.32 0.00
> i
[1] 10
>
> z[z<0] <- 0
> z
 [1] 1.04 1.74 0.00 0.00 0.00 0.00 0.11 0.00 0.32 0.00
```

for雙迴圈

- 單一迴圈

```
a <- numeric(5)
for(i in 1:5){
  a[i]<- i^2
}
> a
[1]  1  4  9 16 25
```

- 雙迴圈

```
m <- 3
n <- 4
for(i in 1:m){
   for(j in 1:n){
     cat("loop: (", i, ",", j, ")\n")
   }
}
```

```
loop: ( 1 , 1 )
loop: ( 1 , 2 )
loop: ( 1 , 3 )
loop: ( 1 , 4 )
loop: ( 2 , 1 )
loop: ( 2 , 2 )
loop: ( 2 , 3 )
loop: ( 2 , 4 )
loop: ( 3 , 1 )
loop: ( 3 , 2 )
loop: ( 3 , 3 )
loop: ( 3 , 4 )
```

```
a <- matrix(0,2,4)
for(i in 1:2){
    for(j in 1:4){
       a[i,j]<- i+j
    }
}
> a
     [,1] [,2] [,3] [,4]
[1,]    2    3    4    5
[2,]    3    4    5    6
```

NOTE: 寫R程式，應儘量避免使用for雙迴圈。

迴圈的控制: next

- **next**: immediately causes control to return to the start of the loop.
  - The next iteration of the loop is then executed.
  - No statement below **next** in the current loop is evaluated.

```
m <- 3
n <- 4
for(i in 1:m){
   for(j in 1:n){

     if(i==2){
       cat("before next:", i,",",j, "\n")
       next
       cat("after next:", i,",",j, "\n")
     }else{
       cat("loop: (", i, ",", j, ")\n")
     }
   }
}
```

```
loop: ( 1 , 1 )
loop: ( 1 , 2 )
loop: ( 1 , 3 )
loop: ( 1 , 4 )
before next: 2 , 1
before next: 2 , 2
before next: 2 , 3
before next: 2 , 4
loop: ( 3 , 1 )
loop: ( 3 , 2 )
loop: ( 3 , 3 )
loop: ( 3 , 4 )
```

迴圈的控制: break

- **break**: causes an exit from the innermost loop that is currently being executed.

```
m <- 3
n <- 4
for(i in 1:m){
   for(j in 1:n){

     if(i==2){
       cat("before break:", i,",",j, "\n")
       break
       cat("after break:", i,",",j, "\n")
     }else{
       cat("loop: (", i, ",", j, ")\n")
     }
   }
}
```

```
loop: ( 1 , 1 )
loop: ( 1 , 2 )
loop: ( 1 , 3 )
loop: ( 1 , 4 )
before break: 2 , 1
loop: ( 3 , 1 )
loop: ( 3 , 2 )
loop: ( 3 , 3 )
loop: ( 3 , 4 )
```

repeat and while

```
> repeat{expr.1}
```

- repeat: causes repeated evaluation of the body until a break is specifically requested.

```
> while(condition) expr.1
```

- condition is evaluated and if its values is TRUE than expr.1 is evaluated.
- This process continues until expr.1 evaluates to FALSE.
- If expr.1 is never evaluated then while returns NULL and otherwise it returns the value of the last evaluation of expr.1.

課堂練習9

```
a <- 5
while(a>0){
    a <- a-1
    cat(a,"\n")
    if(a==2){
        cat("before next:", a, "\n")
        next
        cat("after next:", a, "\n")
    }
}
```

```
4
3
2
before next: 2
1
0
```

```
a <- 5
while(a>0){

    if(a==2){
        cat("before break:", a, "\n")
        break
    }
    a <- a-1
    cat(a,"\n")
}
```

```
4
3
2
before break: 2
```

```
a <- 5
while(a>0){

    if(a==2){
        cat("before break:", a, "\n")
        next
        cat("after break:", a, "\n")
    }
    a <- a-1
    cat(a,"\n")
}
```

無窮迴圈

課堂練習10：計算n!

```
factorial.for <- function(n){
    f <- 1
    if(n<2) return(1)
    for(i in 2:n){
        f <- f*i
    }
    f
}
factorial.for(5)
```

```
factroial.repeat <- function(n){
    f <- 1
    t <- n
    repeat{
        if(t<2) break
        f <- f*t
        t <- t-1
    }
    return(f)
}
factroial.repeat(5)
```

```
factorial.while <- function(n){
    f <- 1
    t <- n
    while(t>1){
        f <- f*t
        t <- t-1
    }
    return(f)
}
factorial.while(5)
```

```
factorial.call <- function(n, f){

    if(n <= 1){
      return(f)
    }
    else{
      factorial.call(n-1, n*f)
    }
}
factorial.call(5, 1)
```

```
factorial.cumprod <- function(n) max(cumprod(1:n))
factorial.cumprod(5)
```

switch(expr.1, list)

- **expr.1** is evaluated and the result value obtained.
- If value is a number between 1 and the length of list then the corresponding element list is evaluated and the result returned.
- If value is too large or too small **NULL** is returned.
- if value is a character vector then the element of '...' with a name that exactly matches value is evaluated.
- If there is no match **NULL** is returned.

```
> x <- 3
> switch(x,
         cat("2+2\n"),
         cat("mean(1:10)\n"),
         cat("sd(1:10)\n"))
sd(1:10)
> switch(x, 2+2, mean(1:10), sd(1:10))
[1] 3.027650
> switch(2, 2+2, mean(1:10), sd(1:10))
[1] 5.5
> switch(6, 2+2, mean(1:10), sd(1:10))
NULL
```

```
my.lunch <- function(y){
    switch(y,
           fruit="banana",
           vegetable="broccoli",
           meat="beef")
}
> my.lunch("fruit")
[1] "banana"
> my.lunch(fruit)
Error in switch(y, fruit = "banana",
vegetable = "broccoli", meat = "beef") :
  object "fruit" not found
```

課堂練習11: 計算中心程度

```
x.center <-  function(x, type){
    switch(type,
           mean = mean(x),
           median = median(x),
           trimmed = mean(x, trim = 0.1),
           stop("Measure is not included!"))
}

> x <- rnorm(20)
> x.center(x, "mean")
[1] 0.1086806
> x.center(x, "median")
[1] 0.2885969
> x.center(x, "trimmed")
[1] 0.2307617
> x.center(x, "mode")
Error in switch(type, mean = mean(x), median = median(x), trimmed
= mean(x,  :
  Measure is not included!
```

課堂練習12: 計算median

```
my.median.1 <- function(x){
    odd.even <- length(x)%%2
    if(odd.even==0){
        (sort(x)[length(x)/2] + sort(x)[1+length(x)/2])/2
    }else{
        sort(x)[ceiling(length(x)/2)]
    }
}
```

```
my.median.2 <- function(x){
    odd.even <- length(x)%%2
    s.x <- sort(x)
    n <- length(x)
    if(odd.even==0){
        median <- (s.x[n/2] + s.x[1+n/2])/2
    }else{
        median <- s.x[ceiling(n/2)]
    }
    return(median)
}
```

```
> x <- rnorm(30)
> my.median.1(x)
[1] -0.06110589
> my.median.2(x)
[1] -0.06110589
> median(x)
[1] -0.06110589
```

apply: Apply Functions Over Array Margins

```
> (x <- matrix(1:24, nrow=4))
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    5    9   13   17   21
[2,]    2    6   10   14   18   22
[3,]    3    7   11   15   19   23
[4,]    4    8   12   16   20   24
>
> #1: rows, 2:columns
> apply(x, 1, sum)
[1] 66 72 78 84
> apply(x, 2, sum)
[1] 10 26 42 58 74 90
>
> #apply function to the individual elements
>
> apply(x, 1, sqrt)
         [,1]     [,2]     [,3]     [,4]
[1,] 1.000000 1.414214 1.732051 2.000000
[2,] 2.236068 2.449490 2.645751 2.828427
[3,] 3.000000 3.162278 3.316625 3.464102
[4,] 3.605551 3.741657 3.872983 4.000000
[5,] 4.123106 4.242641 4.358899 4.472136
[6,] 4.582576 4.690416 4.795832 4.898979
> apply(x, 2, sqrt)
         [,1]     [,2]     [,3]     [,4]     [,5]     [,6]
[1,] 1.000000 2.236068 3.000000 3.605551 4.123106 4.582576
[2,] 1.414214 2.449490 3.162278 3.741657 4.242641 4.690416
[3,] 1.732051 2.645751 3.316625 3.872983 4.358899 4.795832
[4,] 2.000000 2.828427 3.464102 4.000000 4.472136 4.898979
```

apply自定函式

將某班三科成績，皆以開根號乘以10重新計分。

```
> # generate score data
> math <- sample(1:100, 50, replace=T)
> english <- sample(1:100, 50, replace=T)
> algebra <- sample(1:100, 50, replace=T)
> ScoreData <- cbind(math, english, algebra)
> head(ScoreData, 5)
     math english algebra
[1,]    7      52      93
[2,]    7      17       9
[3,]   57      89      69
[4,]   69      21      97
[5,]   20      64      64
>
> myfun <- function(x){
+     sqrt(x)*10
+ }
> sdata1 <- apply(ScoreData, 2, myfun)
> head(sdata1, 5)
         math  english  algebra
[1,] 26.45751 72.11103 96.43651
[2,] 26.45751 41.23106 30.00000
[3,] 75.49834 94.33981 83.06624
[4,] 83.06624 45.82576 98.48858
[5,] 44.72136 80.00000 80.00000
```

```
> head(apply(ScoreData, 2, function(x) sqrt(x)*10), 5)
         math  english  algebra
[1,] 26.45751 72.11103 96.43651
[2,] 26.45751 41.23106 30.00000
[3,] 75.49834 94.33981 83.06624
[4,] 83.06624 45.82576 98.48858
[5,] 44.72136 80.00000 80.00000
>
> myfun2 <- function(x, attend){
+     y <- sqrt(x)*10 + attend
+     ifelse(y > 100, 100, y)
+ }
> sdata2 <- apply(ScoreData, 2, myfun2, attend=5)
> head(sdata2, 5)
         math  english   algebra
[1,] 31.45751 77.11103 100.00000
[2,] 31.45751 46.23106  35.00000
[3,] 80.49834 99.33981  88.06624
[4,] 88.06624 50.82576 100.00000
[5,] 49.72136 85.00000  85.00000
```

tapply: Apply a Function Over a "Ragged" Array

```
> tapply(iris$Sepal.Length, iris$Species, sum)
    setosa versicolor  virginica
     250.3      296.8      329.4
> tapply(iris$Sepal.Width, iris$Species, mean)
    setosa versicolor  virginica
     3.428      2.770      2.974
```

```
> set.seed(12345)
> scores <- sample(0:100, 50, replace=T)
> grade <- as.factor(sample(c("大一", "大二", "大三", "大四"), 50, replace=T))
> bloodtype <- as.factor(sample(c("A","AB","B","O"), 50, replace=T))
> tapply(scores, grade, mean)
    大一     大二     大三     大四
51.69231 55.87500 35.06667 59.42857
> tapply(scores, bloodtype, mean)
       A       AB        B        O
68.88889 43.12500 54.18750 37.94118
> tapply(scores, list(grade, bloodtype), mean)
         A       AB    B        O
大一 96.00       NA 65.5 31.14286
大二 97.00 50.33333 71.0 42.66667
大三 47.25 13.00000 39.0 25.66667
大四 71.00 56.00000 60.0 55.50000
```

tapply: Apply a Function Over a "Ragged" Array

```
> n <- 20
> (my.factor <- factor(rep(1:3, length = n), levels = 1:5))
 [1] 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2
Levels: 1 2 3 4 5
> table(my.factor)
my.factor
1 2 3 4 5
7 7 6 0 0
> tapply(1:n, my.factor, sum)
 1  2  3  4  5
70 77 63 NA NA
```

```
> tapply(1:n, my.factor, range)
$`1`
[1]  1 19
$`2`
[1]  2 20
$`3`
[1]  3 18
$`4`
NULL
$`5`
NULL
```

```
> tapply(1:n, my.factor, quantile)
$`1`
  0%  25%  50%  75% 100%
 1.0  5.5 10.0 14.5 19.0

$`2`
  0%  25%  50%  75% 100%
 2.0  6.5 11.0 15.5 20.0

$`3`
   0%   25%   50%   75%  100%
 3.00  6.75 10.50 14.25 18.00

$`4`
NULL

$`5`
NULL
```

lapply: Apply a Function over a List or Vector

lapply returns a list of the same length as X, each element of which is the result of applying FUN to the corresponding element of X.

```
> a <- c("a", "b", "c", "d")
> b <- c(1, 2, 3, 4, 4, 3, 2, 1)
> c <- c(T, T, F)
>
> list.object <- list(a,b,c)
>
> my.la1 <- lapply(list.object, length)
> my.la1
[[1]]
[1] 4

[[2]]
[1] 8

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

```
> my.la2 <- lapply(list.object, class)
> my.la2
[[1]]
[1] "character"

[[2]]
[1] "numeric"

[[3]]
[1] "logical"
```

Apply a Function over ....

- **sapply**
  - a user-friendly version of lapply by default returning a vector or matrix if appropriate.
- **mapply**
  - for applying a function to multiple arguments.
- **rapply**
  - for a recursive version of **lapply()**.
- **eapply**
  - for applying a function to each entry in an environment.
- **aggregate**
  - Splits the data into subsets, computes summary statistics for each, and returns the result in a convenient form.

See also: scale {base}
Scaling and Centering of Matrix-like Objects
sweep which allows centering (and scaling) with arbitrary statistics.

課堂練習13

```
> (select.num <- sapply(iris, is.numeric))  #return vector
Sepal.Length  Sepal.Width Petal.Length  Petal.Width       Species
        TRUE         TRUE         TRUE         TRUE         FALSE
> iris[1:2, select.num]
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1          5.1         3.5          1.4         0.2
2          4.9         3.0          1.4         0.2
> select.fac <- sapply(iris, is.factor)
> select.fac
Sepal.Length  Sepal.Width Petal.Length  Petal.Width       Species
       FALSE        FALSE        FALSE        FALSE          TRUE
> iris[1:5, select.fac]
[1] setosa setosa setosa setosa setosa
Levels: setosa versicolor virginica

# don't use apply(iris, 2, is.numeric)
> apply(iris, 2, is.numeric)
Sepal.Length  Sepal.Width Petal.Length  Petal.Width       Species
       FALSE        FALSE        FALSE        FALSE         FALSE

> unique(iris$Species)
[1] setosa     versicolor virginica
Levels: setosa versicolor virginica
> table(iris$Species)
    setosa versicolor  virginica
        50         50         50
```

樣式比對: Pattern Matching

```
> wf <- read.table("worldfloras.txt", header=TRUE)
> attach(wf)
> names(wf)
> dim(wf)
[1] 161   7
```

Data Editor

	Country	Latitude	Area	Population	Flora	Endemism	Continent
1	Afghanistan	30	636	14.3	3000	0.27	Asia
2	Albania	42	29	3	3200	0.008	Europe
3	Algeria	35	2382	21.3	3139	0.08	N.Africa
4	Andorra	42	0.5	0.034	1000	0	Europe
5	Angola	25	1247	8.5	5000	0.25	Africa
6	Antarctica	85	14000	0	2	0	Antarctica
7	Argentina	45	2777	30.1	9000	0.27	S.America
8	Australia	25	7682	15.5	23000	0.8	Australia
9	Austria	48	84	7.5	3000	0.012	Europe
10	Bahrain	26	0.66	0.4	175	0	Asia
11	Balearic Islands	40	5	0.62	1400	0.067	Europe
12	Bangladesh	23	144	98.5	5000	-1	Asia
13	Belgium	52	31	9.9	1700	0	Europe
14	Belize	14	23	0.16	3240	0.046	C.America
15	Benin	5	113	3.9	2000	0.006	Africa
16	Bhutan	30	47	1.4	5000	0.12	Asia
17	Bolivia	16	1099	6.2	16500	0.23	S.America
18	Botswana	25	575	1	2015	0.008	Africa
19	Brazil	10	8512	132.6	55000	0.5	S.America
20	Brunei	5	5.7	0.27	6000	-1	SE.Asia
21	Bulgaria	42	111	9.2	3600	0.015	Europe
22	Burkina Faso	15	274	6.8	1096	0	Africa
23	Burma	20	678	38.5	7000	0.153	SE.Asia
24	Burundi	0	28	4.5	2500	0.04	Africa
25	Cameroon	5	475	9.5	9000	0.017	Africa
26	Canada	55	9922	25.3	3220	0.045	N.America
27	Central African Republic	0	625	2.5	3600	0.025	Africa
28	Chad	10	1284	4.9	1600	-1	Africa
29	Chile	40	752	11.9	5500	0.5	S.America
30	China	35	9597	1051.6	30000	0.1	Asia
31	Colombia	5	1139	28.1	45000	0.05	S.America
32	Congo	0	342	1.7	4000	0.2	Africa
33	Corsica	42	8.7	0.23	2200	0.014	Europe
34	Costa Rica	12	51	2.5	8000	0.17	C.America
35	Crete	35	8.3	0.46	1700	0.09	Europe
36	Cuba	20	114	10	7000	0.5	C.America
37	Cyprus	35	9.2	0.7	2000	0.058	Europe

樣式比對: grep

Select subsets of countries on the basis of specified patterns.

#contain "R"

```
> index <- grep("R", as.character(Country))
[1]  27  34  40 116 118 119 120 152
> as.vector(Country[index])
```

#begin with "R"

```
> as.vector(Country[grep("^R", as.character(Country))])
```

" R" with multiple name

```
> as.vector(Country[grep(" R", as.character(Country))])
```

two or more names

```
> as.vector(Country[grep(" ", as.character(Country))])
```

ending by "y"

```
> as.vector(Country[grep("y$", as.character(Country))])
```

樣式比對: grep

select countries with names containing C to E

```
> my.pattern <- "[C-E]"
> index <- grep(my.pattern, as.character(Country))
> as.vector(Country[index])
```

select countries with names containing C to E in the first

```
> as.vector(Country[grep("^[C-E]", as.character(Country))])
```

select countries that do not end with a letter between ‘a’ and ‘t’.

```
> as.vector(Country[-grep("[a-t]$", as.character(Country))])
```

select countries that do not end with a letter between ‘a’’A’ and ‘t’’T’.

```
> as.vector(Country[-grep("[A-T a-t]$", as.character(Country))])
```

樣式比對: grep

'.' means anything

y is the second character
```
> as.vector(Country[grep("^.y", as.character(Country))])
```

y is the third character
```
> as.vector(Country[grep("^..y", as.character(Country))])
```

y is the sixth character
```
> as.vector(Country[grep("^.{5}y", as.character(Country))])
```

{,4} means 'repeat up to four' anything before $
```
> as.vector(Country[grep("^.{,4}$", as.character(Country))])
```

all the countries with 15 or more characters in their name
```
> as.vector(Country[grep("^.{15,}$", as.character(Country))])
```

See also:
- strtrim{base}, substr{base}, substring{base}
- strsplit{base}

搜尋與替換: sub, gsub

- replaces only the first occurrence of a pattern within a character string: `sub(pattern, replacement, x)`
- replace all occurrences: `gsub(pattern, replacement, x)`

```
> text <- c("arm", "leg", "head", "foot", "hand", "hindleg", "elbow")
> text
[1] "arm"     "leg"     "head"    "foot"    "hand"    "hindleg" "elbow"
> gsub("h", "H", text)
[1] "arm"     "leg"     "Head"    "foot"    "Hand"    "Hindleg" "elbow"

> gsub("o", "O", text)
[1] "arm"     "leg"     "head"    "fOOt"    "hand"    "hindleg" "elbOw"

> sub("o", "O", text)
[1] "arm"     "leg"     "head"    "fOot"    "hand"    "hindleg" "elbOw"

> gsub("^.", "O", text)
[1] "Orm"     "Oeg"     "Oead"    "Ooot"    "Oand"    "Oindleg" "Olbow"
```

樣式的位置: regexpr

> regexpr(pattern, text)

- match location: if the pattern does not appear within the string, return -1

```
> text <- c("arm", "leg", "head", "foot", "hand", "hindleg", "elbow")
> regexpr("o", text)
[1] -1 -1 -1  2 -1 -1  4
attr(,"match.length")
[1] -1 -1 -1  1 -1 -1  1

#which elements of text contained an "o"
> grep("o", text)
[1] 4 7

#extract the character string
> text[grep("o", text)]
[1] "foot"  "elbow"

#how many "o"s there are in each string
> gregexpr("o", text)
[[1]]
[1] -1
attr(,"match.length")
[1] -1
...
```

```
#multiple match return 0
> charmatch("m", c("mean","median","mode"))
[1] 0

#unique match return index
> charmatch("med", c("mean","median","mode"))
[1] 2
```

which: Which indices are TRUE?

```
> stock <- c("car", "van")
> requests <- c("truck", "suv", "van", "sports", "car", "waggon", "car")
> requests %in% stock
[1] FALSE FALSE  TRUE FALSE  TRUE FALSE  TRUE
> index <- which(requests %in% stock)
> requests[index]
[1] "van" "car" "car"
```

```
> x <- round(rnorm(10), 2)
> x
 [1] -1.17 -0.05  0.57  0.72 -1.79  0.55  0.03  0.09 -1.81  0.04
> index <- which(x < 0)
> index
[1] 1 2 5 9
> x[index]
[1] -1.17 -0.05 -1.79 -1.81
> x[x<0]
[1] -1.17 -0.05 -1.79 -1.81
```

which.max() #locates first maximum of a numeric vector
which.min() #locates first minimum of a numeric vector

課堂練習14

```
> x <- matrix(sample(1:12), ncol=4, nrow=3)
> x
     [,1] [,2] [,3] [,4]
[1,]   12    4    2    8
[2,]    6   10    5    9
[3,]    1    3    7   11
> which(x %% 3 == 0)
[1]  1  2  6 11
> which(x %% 3 == 0, arr.ind = T)
     row col
[1,]   1   1
[2,]   2   1
[3,]   3   2
[4,]   2   4
```

See also:

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

```
> x <- c(45, 3, 50, 41, 14, 50, 3)
> which.min(x)
[1] 2
> which.max(x)
[1] 3
> x[which.min(x)]
[1] 3
> x[which.max(x)]
[1] 50
> which(x == max(x))
[1] 3 6
```

```
> match(1:10, 4)
 [1] NA NA NA  1 NA NA NA NA NA NA
> match(1:10, c(4, 2))
 [1] NA  2 NA  1 NA NA NA NA NA NA
> x
[1] 45  3 50 41 14 50  3
> match(x, c(50, 3))
[1] NA  2  1 NA NA  1  2
```

集合運算

```
> setA <- c("a","b","c", "d", "e")
> setB <- c("d", "e", "f", "g")

> union(setA, setB)
[1] "a" "b" "c" "d" "e" "f" "g"

> intersect(setA, setB)
[1] "d" "e"

> setdiff(setA, setB)
[1] "a" "b" "c"

> setdiff(setB, setA)
[1] "f" "g"

> setA %in% setB
[1] FALSE FALSE FALSE  TRUE  TRUE

> setB %in% setA
[1]  TRUE  TRUE FALSE FALSE

> setA[setA %in% setB] #intersect(setA, setB)
[1] "d" "e"
```

日期時間

```
> Sys.time()
[1] "2028-10-14 21:16:07 台北標準時間"
#extract date
> substr(as.character(Sys.time()), 1, 10)
[1] "2028-10-14"
#extract time
> substr(as.character(Sys.time()), 12, 19)
[1] "21:16:07"
> date()
[1] "Tue Oct 14 21:16:09 2028"
```

sec, min, hour,
mday (# day number within the month),
mon (#January=0),
year (#+1900),
wday (#day of the week starting at 0=sunday),
yday (#day of the year after 1 january=0)

```
> my.date <- as.POSIXlt(Sys.time())
> my.date
[1] "2028-10-14 21:18:31 台北標準時間"
> my.date$sec
[1] 31.304
> my.date$min
[1] 18
> my.date$hour
[1] 21
> my.date$mday
[1] 14
```

```
> my.date$mon
[1] 9
> my.date$year+1900
[1] 2028
> my.date$wday
[1] 2
> my.date$yday
[1] 287
```

日期時間

```
compute <- function(){
   a <- 1
    for(i in 1:1000){
        for(j in 1:1000){
             a <- a+i+j
        }
    }
}

> start.time <- as.POSIXlt(Sys.time())
> compute()
> end.time <- as.POSIXlt(Sys.time())
>
> time.diff <- end.time-start.time
> time.diff
Time difference of 2.36 secs
> difftime(end.time, start.time)
Time difference of 2.36 secs
```

?systems.time

```
#Assume reading data from Excel
> excel.dates <- c("27/02/2004", "27/02/2005", "14/01/2003", "28/06/2005", "01/01/1999")
> excel.dates
[1] "27/02/2004" "27/02/2005" "14/01/2003" "28/06/2005" "01/01/1999"

#Convert to R Date format
> strptime(excel.dates, format="%d/%m/%Y")
[1] "2004-02-27" "2005-02-27" "2003-01-14" "2005-06-28" "1999-01-01"
```

R程式執行時間

```
myFun <- function(n){
    for(i in 1:n){
        x <- x + i
    }
    x
}
```

```
> start.time <- Sys.time()
> ans <- myFun(10000)
> end.time <- Sys.time()
> end.time -start.time
Time difference of 0.0940001 secs
```

```
> system.time({
+     ans <- myFun(10000)
+ })
   user  system elapsed
   0.04    0.00    0.05
>
```

```
> start.time <- proc.time()
> for(i in 1:50) mad(runif(500))
> proc.time() - start.time
   user  system elapsed
   0.04    0.01    0.05
```

排序: Rank, Sort and Order

```
> city <- read.table("city.txt", header=TRUE, row.names=NULL, sep="\t")
> attach(city)
> names(city)
[1] "location" "price"
>
> rank.price <- rank(price)
> sorted.price <- sort(price)
> ordered.price <- order(price)
```

	A	B
1	location	price
2	Taipei	325
3	New York	201
4	Boston	157
5	Tokyo	162
6	Hong Kong	164
7	Shanghai	95
8	LA	117
9	Vancouver	188
10	Seoul	121
11	Seattle	101

- **order** returns an integer vector containing the permutation that will sort the input into ascending order.
- **order** is useful in sorting dataframes.
- **x[order(x)]** is the same as **sort(x)**

```
> sort(price, decreasing=TRUE)
 [1] 325 201 188 164 162 157 121 117 101  95
> rev(sort(price))
 [1] 325 201 188 164 162 157 121 117 101  95
```

排序: Rank, Sort and Order

```
> city
    location price
1      Taipei   325
2    New York   201
3      Boston   157
4       Tokyo   162
5   Hong Kong   164
6    Shanghai    95
7          La   117
8   Vancouver   188
9       Seoul   121
10    Seattle   101
```

```
> (view1 <- data.frame(location, price, rank.price))
    location price rank.price
1      Taipei   325         10
2    New York   201          9
3      Boston   157          5
4       Tokyo   162          6
5   Hong Kong   164          7
6    Shanghai    95          1
7          LA   117          3
8   Vancouver   188          8
9       Seoul   121          4
10    Seattle   101          2
```

```
> (view2 <- data.frame(sorted.price, ordered.price))
   sorted.price ordered.price
1            95             6
2           101            10
3           117             7
4           121             9
5           157             3
6           162             4
7           164             5
8           188             8
9           201             2
10          325             1
```

```
> (view3 <- data.frame(location[ordered.price],
price[ordered.price]))
   location.ordered.price. price.ordered.price.
1                 Shanghai                    95
2                  Seattle                   101
3                       LA                   117
4                    Seoul                   121
5                   Boston                   157
6                    Tokyo                   162
7                Hong Kong                   164
8                Vancouver                   188
9                 New York                   201
10                  Taipei                   325
```

See: multiple sorting, text sorting
http://rprogramming.net/r-order-to-sort-data/

x[order(x)] is the same as sort(x

資料處理、表格相關

Sampling without replacement

```
> y < 1:20
> sample(y)
> sample(y)
> sample(y, 5)
> sample(y, 5)
> sample(y, 5, replace=T)
```

Substrings

```
> substr("this is a test", start=1, stop=4)
> substr(rep("abcdef",4),1:4,4:5)

> x <- c("asfef", "qwerty", "yuiop[", "b", "stuff.blah.yech")
> substr(x, 2, 5)
> substring(x, 2, 4:6)
> substring(x, 2) <- c("..", "+++")
> x
```

See also:

- stack {utils}, reshape {stats}, melt{reshape}, cast{reshape}, merge {base}, sample {base}, subset {base}
- xtabs {stats}, table {base}, tabulate {base}, ftable {stats}, xtable{xtable}

Data Operation: by

```
> attach(iris)
> names(iris)

#by(): summary of the dataframe on the basis of factor levels

> by(iris[,1:4], Species, mean)
Species: setosa
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
       5.006        3.428        1.462        0.246
------------------------------------------------------------
Species: versicolor
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
       5.936        2.770        4.260        1.326
------------------------------------------------------------
Species: virginica
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
       6.588        2.974        5.552        2.026
```

物件屬性強制轉換 (Coercing)

Table 2.4. Functions for testing (is) the attributes of different categories of object (arrays, lists, etc.) and for coercing (as) the attributes of an object into a specified form. Neither operation changes the attributes of the object.

Type	Testing	Coercing
Array	is.array	as.array
Character	is.character	as.character
Complex	is.complex	as.complex
Dataframe	is.data.frame	as.data.frame
Double	is.double	as.double
Factor	is.factor	as.factor
List	is.list	as.list
Logical	is.logical	as.logical
Matrix	is.matrix	as.matrix
Numeric	is.numeric	as.numeric
Raw	is.raw	as.raw
Time series (ts)	is.ts	as.ts
Vector	is.vector	as.vector

```
> as.numeric(factor(c("a", "b", "c")))
> as.numeric(c("a", "b", "c")) #don't work
```

字串轉成變數名稱或指令

```
> (x <- sample(1:42, 6))
[1]  3  1 29 16 36 21
> (y <- letters)
> get("x")
[1]  3  1 29 16 36 21
> get("y")[1:5]
[1] "a" "b" "c" "d" "e"
>
> for(i in 1:5){
+     x.name <- paste("x", i, sep=".")
+     assign(x.name, 1:i)
+     cat(x.name, ": \t")
+     cat(get(x.name), "\n")
+ }
x.1 :    1
x.2 :    1 2
x.3 :    1 2 3
x.4 :    1 2 3 4
x.5 :    1 2 3 4 5
```

`eval()` evaluates an expression, but "5+5" is a string, not an expression. So, use parse() with text= to translate the string to an expression

```
> a <- 100
> (my.math <- c("3+4", "a/5"))
[1] "3+4" "a/5"
> eval(my.math)
[1] "3+4" "a/5"
> eval(parse(text=my.math[1]))
[1] 7
>
> plot.type <- c("plot", "hist", "boxplot")
> x <- rnorm(100)
> my.plot <- paste(plot.type, "(x)", sep="")
> eval(parse(text=my.plot[1]))
```

查看指令程式碼

```
> library(e1071)
> fclustIndex
function (y, x, index = "all")
{
    clres <- y
    gath.geva <- function(clres, x) {
        xrows <- dim(clres$me)[1]
        xcols <- dim(clres$ce)[2]
        ncenters <- dim(clres$centers)[1]
        scatter <- array(0, c(xcols, xcols, ncenters))
...
```

```
> plot
function (x, y, ...)
UseMethod("plot")
<bytecode: 0x000000000fdaaad8>
<environment: namespace:graphics>
> methods(plot)
 [1] plot.acf*           plot.agnes* ... plot.lm     ... plot.table ...

   Non-visible functions are asterisked
> plot.lm
function (x, which = c(1L:3L, 5L), caption = list("Residuals vs Fitted",
    "Normal Q-Q", "Scale-Location", "Cook's distance", "Residuals vs Leverage",
    expression("Cook's dist vs Leverage  " * h[ii]/(1 - h[ii]))),
```

查看指令程式碼

```
> plot.table
錯誤: 找不到物件 'plot.table'
> ?plot.table                          #plot.table {graphics}
> graphics:::plot.table
function (x, type = "h", ylim = c(0, max(x)), lwd = 2, xlab = NULL,
    ylab = NULL, frame.plot = is.num, ...)
{
    xnam <- deparse(substitute(x))
    rnk <- length(dim(x))
    if (rnk == 0L)
        stop("invalid table 'x'")
    if (rnk == 1L) {
    . . .
```

```
> anova
> methods(anova)
> stats:::anova.nls
> stats:::anova.loess
```

```
> svm
> ?svm
> methods(svm)
> e1071:::svm.default
. . .
   cret <- .C("svmtrain", as.double(if (sparse) x@ra else t(x)),
        as.integer(nr), as.integer(nco
. . .
```

Package source: e1071_1.6-3.tar.gz
Windows binaries: e1071_1.6-3.zip

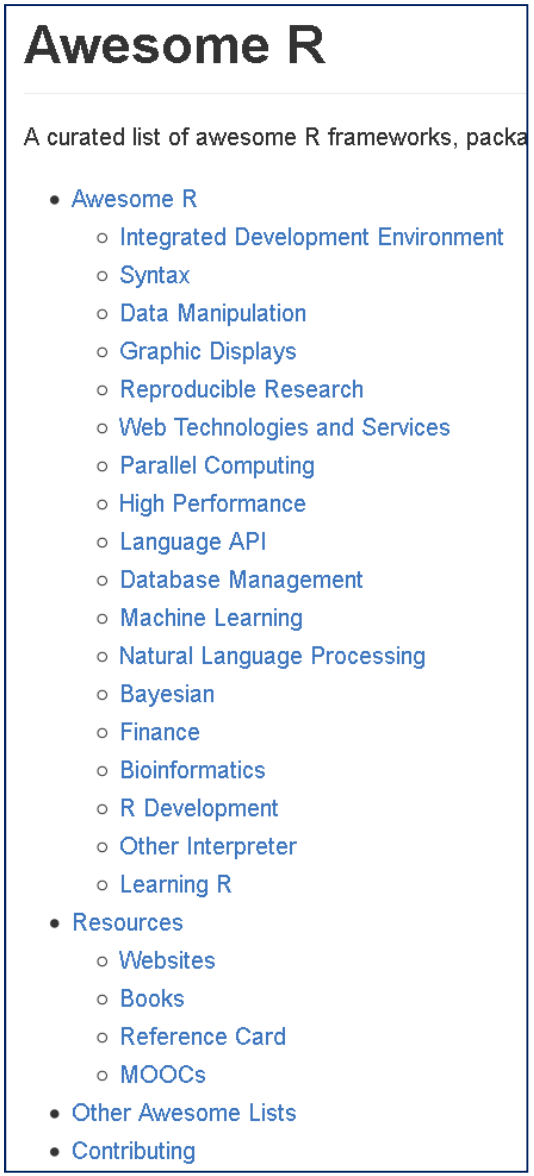
e1071_1.6-3\e1071\src
cmeans.c, cshell.c, floyd.c, Rsvm.c, svm.cpp, svm.h, ...
e1071_1.6-3\e1071\R
bclust.R, bincombinations.R, cmeans.R, fclustIndex.R, ...

Quick list of useful R packages

https://support.rstudio.com/hc/en-us/articles/201057987-Quick-list-of-useful-R-packages

- **To load data**: RODBC, RMySQL, RPostgresSQL, RSQLite, XLConnect, xlsx, foreign
- **To manipulate data**: dplyr, tidyr, stringr, lubridate
- **To visualize data**: ggplot2, ggvis, rgl, htmlwidgets, googleVis
- **To model data**: car, mgcv, lme4/nlme, randomForest, multcomp, vcd, glmnet, survival, caret
- **To report results**: shiny, RMarkdown, xtable
- **For Spatial data**: sp, maptools, maps, ggmap
- **For Time Series and Financial data**: zoo, xts, quantmod
- **To write high performance R code**: Rcpp, data.table, parallel
- **To work with the web**: XML, jsonlite, httr
- **To write your own R packages**: devtools, testthat, roxygen2

https://github.com/qinwf/awesome-R?files=1



Awesome R

A curated list of awesome R frameworks, packa

- Awesome R
  - Integrated Development Environment
  - Syntax
  - Data Manipulation
  - Graphic Displays
  - Reproducible Research
  - Web Technologies and Services
  - Parallel Computing
  - High Performance
  - Language API
  - Database Management
  - Machine Learning
  - Natural Language Processing
  - Bayesian
  - Finance
  - Bioinformatics
  - R Development
  - Other Interpreter
  - Learning R
- Resources
  - Websites
  - Books
  - Reference Card
  - MOOCs
- Other Awesome Lists
- Contributing

Reference Card

R Reference Card
(Version 2)

R Reference Card 2.0

Public domain, v2.0 2012-12-24.
V 2 by Matt Baggott, matt@baggott.net
V 1 by Tom Short, t.short@ieee.org
Material from *R for Beginners* by permission of Emmanuel Paradis.

Getting help and info

help(topic) documentation on topic
?topic same as above; special chars need quotes: for example **?'&&'**
help.search("topic") search the help system; same

Operators

<-	Left assignment, binary
->	Right assignment, binary
=	Left assignment, but not recommended
<<-	Left assignment in outer lexical scope; not for beginners
$	List subset, binary
-	Minus, can be unary or binary
+	Plus, can be unary or binary
~	Tilde, used for model formulae
:	Sequence, binary (in model formulae: interaction)
::	Refer to function in a package, i.e.,

R Reference Card for Data Mining
(2015)
http://www.rdatamining.com/docs/r-reference-card-for-data-mining

R Reference Card for Data Mining

Yanchang Zhao, RDataMining.com, January 8, 2015

- See the latest version at http://www.RDataMining.com
- The package names are in parentheses.
- Recommended packages and functions are shown in bold.
- Click a package in this PDF file to find it on CRAN.

Association Rules and Sequential Patterns

Functions

apriori() mine associations with APRIORI algorithm – a level-wise, breadth-first algorithm which counts transactions to find frequent itemsets (*arules*)
eclat() mine frequent itemsets with the Eclat algorithm, which employs equivalence classes, depth-first search and set intersection instead of counting (*arules*)
cspade() mine frequent sequential patterns with the cSPADE algorithm (*arulesSequences*)
seqefsub() search for frequent subsequences (*TraMineR*)

Packages

arules mine frequent itemsets, maximal frequent itemsets, closed frequent itemsets and association rules. It includes two algorithms, Apriori and Eclat.
arulesViz visualizing association rules
arulesSequences add-on for *arules* to handle and mine frequent sequences
TraMineR mining, describing and visualizing sequences of states or events

Classification & Prediction

Contributed Documentation
http://cran.r-project.org/other-docs.html