105-1: EE4052

計算機程式設計

Computer Programming

Unit 08: 函數與程式

連 豐 力

臺 大 電 機 系

Sep 2016 - Jan 2017

大綱

- 常用函數
- 矩陣運算函數
- 使用者自訂函數

大綱

常用函數

定義一個陣列

- ?array
- array(data = NA, dim = length(data), dimnames = NULL)
  - data: 陣列內容的資料，預設值為 NA
  - dim: 維度，陣列的一個屬性
  - dimnames: 維度的名稱
- array(1:12)
- array(, c(3, 4))
- array(1:12, c(3, 4))
- array(data = 1:12, dim = c(3, 4))
- array(data = 1:60, dim = c(3, 4, 5))
- args(array)

常用的函數 – 基本操作

- x <- c(1, 8, 5, 2, 3, 1)

- length(x)
- diff(x)
- sum(x)
- prod(x)
- max(x)
- min(x)
- which.max(x)
- which.min(x)
- range(x)

常用的函數 – 基本統計

- x <- c(1, 8, 5, 2, 3, 1)

- round(x * pi, 2)
- cumsum(x)
- cumprod(x)
- unique(x)
- mean(x)
- median(x)
- var(x)
- sd(x)
- summary(x)

常用的函數 – 排序

- x <- c(1.2, -3.4, 5.7, -6, 0, 3)

- sort(x)
- rank(x)
- order(x)

- x[order(x)]
- order(x)[3]

常用的函數 – 排序 – 逆向

- x <- c(1.2, -3.4, 5.7, -6, 0, 3)

- which(rank(x) == 3)
- sort(x)[3]
- x[order(x)[3]]

- sort(x, decreasing = TRUE)
- rev(sort(x))
- rev(rank(x))
- order(x, decreasing = TRUE)
- rev(order(x))

常用的函數 – 排序 – 平手

- x <- c(2.5, 2.5, 2.5, 2.5, 2.3, 4.7, -2.2, 4.6, 4.6)

- sort(x)
- rank(x, ties.method = "average")
- rank(x, ties.method = "first")
- rank(x, ties.method = "random")
- rank(x, ties.method = "max")
- rank(x, ties.method = "min")

- rank(x)

大綱

矩陣運算函數

矩陣運算函數 – apply, sweep

- apply(X, MARGIN, FUN, ...)
  - X: 數據矩陣
  - FUN: 欲使用的函數
  - MARGIN: = 1, by row; = 2, by column

- sweep(X, MARGIN, STATS, FUN = "-", check.margin = TRUE, ...)
  - X: 數據矩陣
  - FUN: 欲使用的函數
  - MARGIN: = 1, by row; = 2, by column
  - STATS: FUN 函數的另一個運算元

矩陣運算函數 – apply()

- apply(X, MARGIN, FUN, ...)
  - X: 數據矩陣
  - FUN: 欲使用的函數
  - MARGIN: = 1, by row; = 2, by column

- A <- matrix(1:12, nrow = 4, ncol = 3)
- apply(A, MARGIN = 2, FUN = mean)
- apply(A, MARGIN = 1, FUN = mean)

- apply(A, MARGIN = 2, FUN = sum)
- apply(A, MARGIN = 1, FUN = sum)

矩陣運算函數 – apply()

- apply(X, MARGIN, FUN, ...)
  - X: 數據矩陣
  - FUN: 欲使用的函數
  - MARGIN: = 1, by row; = 2, by column

- apply(A, MARGIN = 1, FUN = function(x) sd(x) / mean(x))

- myfunc <- function(x, c1, c2) c(mean(x[c1]), sum(x[c2]))

- apply(A, MARGIN = 1, FUN = myfunc)
- apply(A, MARGIN = 1, FUN = myfunc, c1 = c(1, 2), c2 = c(2, 3))

計算機程式設計 – 2016F
Chap 08:函數與程式
Feng-Li Lian @ NTU-EE

矩陣運算函數 – apply()

- apply(X, MARGIN, FUN, ...)
  - X: 數據矩陣
  - FUN: 欲使用的函數
  - MARGIN: = 1, by row; = 2, by column

- A <- array(1:24, dim = c(4, 3, 2))

- apply(A, MARGIN = 1, FUN = sum)
- apply(A, MARGIN = 2, FUN = sum)
- apply(A, MARGIN = c(1, 2), FUN = sum)

矩陣運算函數 – sweep()

- sweep(X, MARGIN, STATS, FUN = "-", check.margin = TRUE, …)
  - X: 數據矩陣
  - FUN: 欲使用的函數
  - MARGIN: = 1, by row; = 2, by column
  - STATS: FUN 函數的另一個運算元

- A <- matrix(1:12, nrow = 4, ncol = 3)
  - 第一直行至第三直行之數字分別依序行加上1, 2, 3
- sweep(A, MARGIN = 2, STATS = 1:3, FUN = "+")
- sweep(A, MARGIN = 2, STATS = c(100,200,300), FUN = "+")

計算機程式設計 – 2016F
Chap 08:函數與程式
Feng-Li Lian @ NTU-EE

矩陣運算函數 – sweep()

- sweep(X, MARGIN, STATS, FUN = "-", check.margin = TRUE, …)
  - X: 數據矩陣
  - FUN: 欲使用的函數
  - MARGIN: = 1, by row; = 2, by column
  - STATS: FUN 函數的另一個運算元

- A <- matrix(1:12, nrow = 4, ncol = 3)

- u <- apply(A, MARGIN = 2, FUN = mean)
- w <- sweep(A, MARGIN = 2, STATS = u, FUN = "-")
  - 每一行減去該行的樣本平均值

矩陣運算函數 – 標準化程序

- 給定一個數字向量，
  先將其每個分量減去樣本平均值，
  再除以樣本標準差

- u <- apply(x, MARGIN = 2, FUN = mean)
- v <- apply(x, MARGIN = 2, FUN = sd)
- w <- sweep(x, MARGIN = 2, STATS = u, FUN = "-")
- z <- sweep(w, MARGIN = 2, STATS = v, FUN = "/")

- scale(A)
  - 內建函數scale(): 提供標準化程序

矩陣運算函數 – 有用函數

- 介紹幾個有用的函數：
  - with( ) with(DATA, EXPRESSION, ...)
  - apply( ) apply(X, MARGIN, FUN, ...)
  - tapply( ) tapply(X, INDEX, FUN = NULL, ...)
  - sapply( ) sapply(X, FUN, ..., simplify = TRUE, ...)
  - lapply( ) lapply(X, FUN, ...)

植物 對 二氧化碳之攝取 (CO2)

- 資料的筆數為：84筆
- 共有五個欄位（變數）：
  1. Plant：植物辨識碼，12 株植物。
  2. Type：植物品種發源地，2種。
  3. Treatment ：處理方式，2種。
  4. conc：周遭的二氧化碳濃度，數字。
  5. uptake：二氧化碳攝取量， 數字。

```
> summary(CO2)
```

```
    Plant             Type         Treatment       conc          uptake
Qn1    : 7   Quebec      :42   nonchilled:42   Min.   :  95   Min.   : 7.70
Qn2    : 7   Mississippi:42   chilled   :42   1st Qu.: 175   1st Qu.:17.90
Qn3    : 7                                    Median : 350   Median :28.30
Qc1    : 7                                    Mean   : 435   Mean   :27.21
Qc3    : 7                                    3rd Qu.: 675   3rd Qu.:37.12
Qc2    : 7                                    Max.   :1000   Max.   :45.50
(Other):42
```


使用函數處理數據

- 數據：

```
   Plant        Type  Treatment conc uptake
1    Qn1      Quebec nonchilled   95   16.0
2    Qn1      Quebec nonchilled  175   30.4
3    Qn1      Quebec nonchilled  250   34.8
…
35   Qc2      Quebec    chilled 1000   42.4
36   Qc3      Quebec    chilled   95   15.1
37   Qc3      Quebec    chilled  175   21.0
38   Qc3      Quebec    chilled  250   38.1
39   Qc3      Quebec    chilled  350   34.0
40   Qc3      Quebec    chilled  500   38.9
41   Qc3      Quebec    chilled  675   39.6
42   Qc3      Quebec    chilled 1000   41.4
43   Mn1 Mississippi nonchilled   95   10.6
44   Mn1 Mississippi nonchilled  175   19.2
45   Mn1 Mississippi nonchilled  250   26.2
46   Mn1 Mississippi nonchilled  350   30.0
47   Mn1 Mississippi nonchilled  500   30.9
…
```

使用函數處理數據

- 不同品種發源地(type)對二氧化碳攝取量 (uptake) 之平均值：
  - with(CO2, tapply(uptake, INDEX = Type, FUN = mean))

```
  Quebec Mississippi
33.54286    20.88333
```

- 不同處理方式(Treatment)對二氧化碳攝取量 (uptake) 之平均值：
  - with(CO2, tapply(uptake, INDEX = Treatment, FUN = mean))

```
nonchilled    chilled
  30.64286   23.78333
```

- 不同品種發源地(type)及不同處理方式(Treatment)對二氧化碳攝取量 (uptake) 之平均值：
  - with(CO2, tapply(uptake, INDEX = list(Type, Treatment), FUN = mean))

```
            nonchilled  chilled
Quebec         35.33333 31.75238
Mississippi    25.95238 15.81429
```


使用函數處理數據

- 二氧化碳濃度 (conc) 與 攝取量 (uptake) 之平均值：
  - apply(CO2[, 4:5], MARGIN = 2, FUN = mean)

```
> apply( CO2[, 4:5], MARGIN = 2, FUN = mean )
    conc   uptake
435.0000  27.2131
```

  - sapply(CO2[, 4:5], FUN = mean)

```
> sapply( CO2[, 4:5], FUN = mean )
    conc   uptake
435.0000  27.2131
```

  - lapply(CO2[, 4:5], FUN = mean)

```
> lapply( CO2[, 4:5], FUN = mean )
$conc
[1] 435

$uptake
[1] 27.2131
```

• lapply ():回傳之物件為列表(list)

大綱

使用者自訂函數

定義一個函數

```
my_func_name <- function( arg-1, arg-2, ... ){
    statements
    return( object )
}
```

矩陣運算函數 – 標準化程序

- 給定一個數字向量，
  先將其每個分量減去樣本平均值，
  再除以樣本標準差

- u <- apply(x, MARGIN = 2, FUN = mean)
- v <- apply(x, MARGIN = 2, FUN = sd)
- w <- sweep(x, MARGIN = 2, STATS = u, FUN = "-")
- z <- sweep(w, MARGIN = 2, STATS = v, FUN = "/")

- scale(A)
  - 內建函數scale(): 提供標準化程序

矩陣運算函數 – 標準化程序

- myfunc <- function(x) {
- u <- apply(x, MARGIN = 2, FUN = mean)
- v <- apply(x, MARGIN = 2, FUN = sd)
- w <- sweep(x, MARGIN = 2, STATS = u, FUN = "-")
- z <- sweep(w, MARGIN = 2, STATS = v, FUN = "/")
- return(z)
- }
- A <- matrix(1:12, nrow = 4, ncol = 3)
- myfunc(A)

- scale(A)
  - 內建函數scale(): 提供標準化程序

自定函數 – 標準化函數

- 給定一個數字向量，
  先將其每個分量減去樣本平均值，
  再除以樣本標準差

```
myNormalization <- function( x ) {
    u <- apply( x, MARGIN = 2, FUN = mean )
    v <- apply( x, MARGIN = 2, FUN = sd )
    w <- sweep( x, MARGIN = 2, STATS = u, FUN = "-" )
    z <- sweep( w, MARGIN = 2, STATS = v, FUN = "/" )
    return(z)
}
```


計算機程式設計 – 2016F
Chap 08:函數與程式
Feng-Li Lian @ NTU-EE

自定函數 – 標準化函數

```
A <- matrix(1:12, nrow = 4, ncol = 3)
myNormalization( A )
```

```
scale( A )
```

- 內建函數scale(): 提供標準化程序

自定函數 – 符號函數

- sgn(x) = -1, x < 0
  0, x = 0
  1, x > 0

- sgn(2) ➔ 1
- sgn(-3) ➔ -1
- sgn(0) ➔ 0

計算機程式設計 – 2016F
Chap 08:函數與程式
Feng-Li Lian @ NTU-EE

自定函數 – 符號函數

- sgn(x) = -1, x < 0
  0, x = 0
  1, x > 0

-
```
mySign <- function( x ) {
    if ( x < 0 ) {
        value <- -1
    } else if ( x == 0 ) {
        value <- 0
    } else {
        value <- 1
    }
    return(value)
}
```

自定函數 – 符號函數

- mySign(c(2, -3, 0))
- mySignVec <- function(x) {

```
    n <- length( x )
    value <- integer( n )
    for ( i in 1:n ) {
        if ( x[i] < 0 ) {
            value[i] <- -1
        } else if ( x[i] == 0 ) {
            value[i] <- 0
        } else {
            value[i] <- 1
        }
    }
    return( value )
}
```


自定函數 – 符號函數

- mySign(c(2, -3, 0))
- mySignVec(c(2, -3, 0))

自定函數 – 兩個以上回傳值

- 給定一組數列，計算出與回傳最大值與最小值的函數
- x <- c(-2, -5, -7, -9, 12, 17, -18)
- myRange(x)
- myData <- myRange(x)
- c(myData$mymin, myData$mymax)
- -18, 17
- c(min(x), max(x))
- -18, 17

計算機程式設計 – 2016F
Chap 08:函數與程式
Feng-Li Lian @ NTU-EE

自定函數 – 兩個以上回傳值

```
myRange <- function( x ) {
    if ( length(x) == 1 ) {
        mymin <- x[1]; mymax <- x[1]
    } else {
        if ( x[1] < x[2] ) {
            mymin <- x[1]; mymax <- x[2]
        } else {
            mymin <- x[2]; mymax <- x[1]
        }
        if ( length(x) > 2 ) {
            for ( i in 3:length(x) ) {
                if ( x[i] < mymin )   mymin  <- x[i]
                if ( x[i] > mymax )   mymax <- x[i]
            }
        }
    }
    return( list( mymin = mymin, mymax = mymax ) )
}
```

條件分支函數 – switch()

```
myCenter <- function( x, type ) {
        switch( type,
                mean = mean( x ),
                median = median( x ),
                trimmed.mean = mean( x, trim = 0.2 ) )
 }
set.seed(1)
x <- rcauchy(10)
myCenter( x, "mean")
myCenter( x, "median")
myCenter( x, "trimmed.mean")
```


自定數學函數 – f(x)

```
f(x)  =  x^3 * cos(x) - 2 * x^2 * sin(x) + 5 * x - 1

fx <- function(x) x^3 * cos(x) - 2 * x^2 * sin(x) + 5 * x - 1

fx( -5 )
fx( 0 )
fx( 5 )
fx( c( -5, 0, 5 ) )
```

自定數學函數 – f(x, y)

- f(x, y) = x^3 * cos(y) - 2 * x^2 * sin(y) + 5 * x - 1
- fxy <- function(x, y) x^3 * cos(y) - 2 * x^2 * sin(y) + 5 * x - 1
- fxy(-5, pi)
- fxy(1, pi)
- fxy(1, pi/2)
- fxy(0, pi/6)

自定數學函數 – f(x, y, z)

- f(x, y, z) = x^3 * cos(y) - 2 * x^2 * sin(z) + 5 * x - 1
- fxyz <- function(x, y, z) x^3 * cos(y) - 2 * x^2 * sin(z) + 5 * x - 1
- fxyz(-5, pi, pi)
- fxyz(1, pi, pi/2)
- fxyz(1, pi/2, pi/2)
- fxyz(0, pi/6, 3*pi)

為什麼要設計程式？

- 100人的資料
- 如何找出 最大值 或 最小值？
- 寫一個程式，先把資料依照大小排序：

```
BubbleSort <- function(array) {
  count <- 0
  while(1) {
        count_swaps <- 0
        for (j in 1 : (length(array) - 1 - count)) {
              if (array[j] > array[j + 1]) {
                 s <- array[j]
                 array[j] <- array[j+1]
                 array[j+1] <- s
                 count_swaps <- count_swaps + 1
              }
        }
        count <- count + 1
        if(count_swaps == 0) break
  }
  array
}
```

- Math_sort <- sort(Math)

	A	B	C	D
1	Name	Chinese	Enghlish	Math
2	Agnes	70	19	65
3	Aiolos	84	71	56
4	Alan	39	79	19
5	Alexis	32	76	33
6	Alice	60	90	82
7	Alina	31	81	71
8	Ambrose	78	10	84
9	Amelia	77	82	76
10	Angela	61	48	76
11	Ann	79	60	46
12	Anna	81	29	79
13	April	33	60	65
14	Arabella	76	60	38
15	Arnold	69	33	83
16	Arthur	71	39	83
17	Aveza	70	56	33
18	Avis	66	21	78
19	Bethany	83	57	19
20	Bill	67	13	69
21	Bridget	75	39	70
22	Betsy	39	81	83
23	Charlie	0	0	59
24	Caroline	87	69	71
25	Chanceller	82	44	57
26	Cindy	19	28	39
27	Christopher	62	10	45
28	Daniel	71	46	69
29	Debbie	60	36	84
30	David	83	78	30

7 人 100 人 工作表3

查成績的設計

- HW1 - HW6 繳交記錄: (updated on 11/12/16)
  - 請執行下面兩個指令或函數，
    即可獲得您的作業是否有繳交的記錄：
    - myfunc <- lapply("http://goo.gl/VoH06K", source)
    - HomeworkYesNo("B01921001", 3)
  - 第二個指令中：HomeworkYesNo，
    - 第一個資料是您的學號，
    - 第二個資料是作業編號。
  - 結果有三種：Yes, Yes-Late, No
    - 分別是：有交（期限之前），有交（期限之後），沒交

查成績的設計

```
HomeworkYesNo <- function( Student_ID, HW_ID ){
    DataIn <- read.csv( file =
    "http://cc.ee.ntu.edu.tw/~fengli/Teaching/Computer/HW_01_06.csv",
    header = TRUE )
    Index <- DataIn[ , 1 ] == Student_ID
    Homework_Status <- DataIn[ Index, ( HW_ID + 1 ) ]
    return( Homework_Status )
}
```


大綱

作業

HW07：函數與程式

- 經濟學（生物群體）上有一個有名的捕食者（例如：狐狸）與被捕食者（例如：兔子）的恐怖平衡的問題。
- 簡單介紹資料，可參考下面網頁：
  - https://en.wikipedia.org/wiki/Lotka%E2%80%93Volterra_equations
- 主要是探討補食者（狐狸）與被捕食者（兔子）之間的存活個數的問題。
- 假設目前狐狸的數量是 f，兔子的數量是 r。
- 因為，狐裡吃兔子，所以，兔子的數量會減少，
- 但是，兔子數量減少，狐裡吃不到兔子，就會餓死，
- 因此，狐裡的數量也會減少。
- 那麼，狐裡的數量減少之後，兔子就不容易被吃到，
- 所以，兔子的數量會增加。
- 就這樣，狐裡與兔子的數量，會來來回回增加或減少。
- 基本上，我們可以寫成下面的公式：t 代表的第 t 個單位時間或世代
- f(t+1) = f(t) + f(t) * (1 – 0.005 r(t)) * 0.01
- r(t+1) = r(t) + r(t) * (-1 + 0.005 f(t)) * 0.01

HW07：函數與程式

- 在這個作業中，
  首先，我們要寫一個函數: PreyPredatory <- function(f, r)，
  可以來計算這兩個群組的數量上變化。
- 接著，在寫一個程式，去呼叫這個函數，
  計算出連續 2000個 單位時間之後，分別的數量，以及繪製出數據圖。

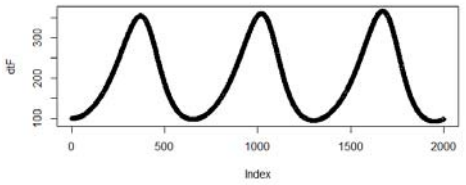
- 也就是，依據前一頁的兩個組群數量的關係式，先寫一個函數，
  同時，儲存成：PreyPredatory.R
- PreyPredatory <- function(f, r){

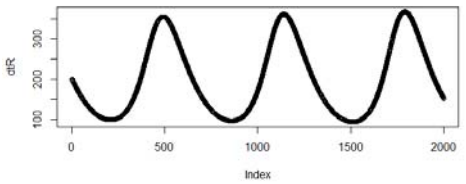
```
    f1 <- f
    r1 <- r
    f2 <-
    r2 <-
    NexNumber <- c( f2 , r2 )
    return( NextNumber )
}
```

HW07：函數與程式

- 然後，再去呼叫這個函數，
  計算出連續 2000 個單位時間之後，分別的數量，以及繪製出數據圖：

```
N <- 2000                                          # time step
myfunc <- lapply( "PreyPredatory.R", source)       # include function
dtF <- matrix( 0, nrow=N, ncol=1 )
dtR <- matrix( 0, nrow=N, ncol=1 )
dtF[ 1 ] <- 100; dtR[ 1 ] <- 200
for( t in 1:N ){
    out <- PreyPredatory( dtF[ t ], dtR[ t ] )
    dtF[ t+1 ] <- out[ 1 ]
    dtR[ t+1 ] <- out[ 2 ]
}
layout( matrix( c( 1, 2 ), byrow = F ) )
plot( dtF ); plot( dtR )
```





計算機程式設計 – 2016F
Chap 08:函數與程式
Feng-Li Lian @ NTU-EE
On 11/22, 2016

HW07：函數與程式

- 繳交兩個檔案：
- 一個程式碼：檔案名稱：PreyPredatory.R
- 以及一個報告檔案，檔案名稱：HW07_B01921001.pptx 或者 .pdf
  包含：
  - 一頁：學生基本資料，
  - 兩頁：兩個程式碼，與簡單解釋
  - 一頁：程式執行結果與說明

- 繳交方式與期限：
  - E-mail 檔案到：ntucp105f@gmail.com
  - E-mail 主旨：HW07_B01921001　(就是，作業編號_您的學號)
  - 繳交期限：11/23 (Wed), 2016, 11pm 以前
  - HW07，每位繳交自己的程式碼，
    程式碼中需要註解所加註的程式碼的意義或想法。