109-1: EE4052

通識課程：計算機程式設計之旅

Computer Programming

Unit 07: 函數 - 計算與排序

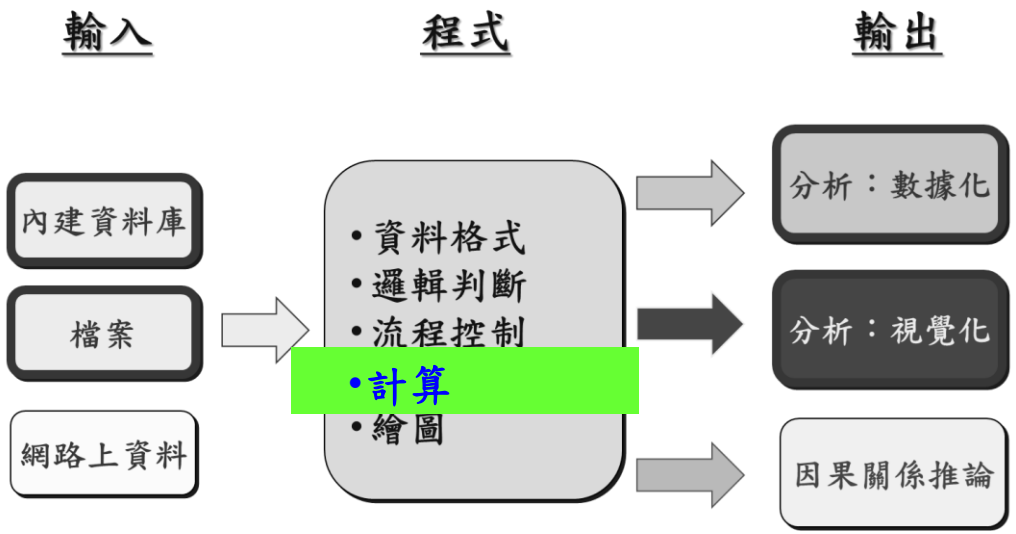
連 豊 力

臺大電機系

Sep 2020 - Jan 2021

課程主題進度

- **U01:** 課程介紹：討論主題，作業，報告，進行方式
- **U02:** 主題，案例，程式，演算法，資源
- **U03:** 設定軟體 R 與 Rstudio
- **U04:** 數據處理與繪圖指令功能
- **U05:** 資料類別與基本運算
- **U06:** 邏輯判斷與流程控制
- **U07:** 函數：計算與排序
- **U08:** 多維度資料格式
- **U09:** 檔案資料輸入與輸出
- **U10:** 繪圖功能與文字
- **U11:** 多重繪圖與顏色
- **U12:** 函數：動畫與動作
- **U13:** 探索性資料分析
- **U14:** 資料間的相關性
- **U15:** 資料連結分析



大綱

- 常用函數
  - 基本操作，基本統計，排序
- 使用者自訂函數
  - 符號函數：sign
  - 計算總和，平均值，標準差
  - 數據排序
  - 找出：最大值，最小值
  - 找出：中位數，四分位數
- 自定數學函數

大綱

作業

HW05：函數 - 計算與排序

- 正規化 (Normalization) 的定義為：
  - The normalization of ratings means
    adjusting values measured on different scales
    to a notionally common scale, often prior to averaging.

- 正規化 (Normalization) 的公式為：

$$\frac{X - \overline{X}}{s}$$

- $\overline{X}$ 是平均值
- s 是標準差

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{1}{n}(x_1 + \cdots + x_n)$$

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}.$$

HW05：函數 - 計算與排序

- 請您寫一個函數 (function), myNorm( ),

  可以把一組數據進行正規化 (Normalization)
- 請同時用 myNorm( ) 與 scale( ) 兩個進行測試，看看結果是否一樣？
- 請測試比較下面幾個範例：
  - myNorm(iris[, 1])
  - scale(iris[, 1])
  - myNorm(CO2[, 5])
  - scale(CO2[, 5])

- 在 .R 的程式碼中，註解您所加註的程式碼的意義或想法。

HW05：函數 - 計算與排序

- 繳交下面檔案，檔案名稱：HW05_學號_關鍵字.xxx
  - 函數檔案：　　HW05_B01921001_myNormFunc.R
  - 測試程式檔案：　HW05_B01921001_myNormTest.R
  - 報告檔案：　　HW05_B01921001_myNormRpt.pdf

- 繳交方式與期限：
  - 上傳檔案到：https://cool.ntu.edu.tw
  - 繳交期限：11/2 (Mon), 11pm 以前

- 學習方式：
  - 請至下面網址輸入此次的學習方式所花的時間：
  - https://forms.gle/TGYXj2uLoL4HwqLHA

HW05：函數 - 計算與排序

- HW05 的主要目的：
  - 是讓您們練習使用 function 的指令來建立一個函數，然後可以被方便重複使用。
  - 因此，只要有建立下面的架構，以及測試過程即可：
    - myNorm <- function(x){
    - ....
    - y <- ...
    - return(y)
    - }
  - 至於，{}之間的功能要自己寫，或者利用現成的函數或指定等，都可以。
  - 但是，不要很無理頭的，只有下面的程式碼：
    - myNorm <- function(x){
    - y <- scale(x)
    - return(y)
    - }
  - 上面的程式碼，是最簡潔的方法，
  - 但是，您可能都沒練習到什麼設計程式的過程。

HW05：函數 - 計算與排序

- HW05 的主要目的：
  - 非常建議您自己盡量練習寫一些過程，
  - 熟悉一下程式設計的感覺。
  - 因為程式設計的功力是來自於多多的練習，
  - 在錯誤中學習到經驗，
  - 未來，當您真正用程式來幫助您的課業或工作的時候，
  - 才能夠發揮實際的效用。

大綱

常用函數

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, -3, 5, -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, -3, 5, -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, 2, 2, 2, 1, 4, -2, 6, 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

使用者自訂函數

定義一個函數

```
my_function_name <- function( arg_1, arg_2, ... ){
    statements
    return( object )
}
```

- arg_1, arg_2, …,: argument, 引數, 輸入值
- return: 傳回 object 值 或傳回 object資料

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

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(2)
- mySign(-3)
- mySign(0)

自定函數 – 計算總和

- 給定一組數據，計算所有數據的總和
- 1, 2, 3, 4, 5

- sum = 0 = (0)
- sum = (0) + 1 = (0 + 1)
- sum = (0 + 1) + 2 = (0 + 1 + 2)
- sum = (0 + 1 + 2) + 3 = (0 + 1 + 2 + 3)
- sum = (0 + 1 + 2 + 3) + 4 = (0 + 1 + 2 + 3 + 4)
- sum = (0 + 1 + 2 + 3 + 4) + 5 = (0 + 1 + 2 + 3 + 4 + 5)
- sum = (0 + 1 + 2 + 3 + 4 + 5)

$$\sum_{i=1}^{n} x_i = \quad (x_1 + \cdots + x_n)$$

自定函數 – 計算總和

- 給定一組數據，計算所有數據的總和

$$\sum_{i=1}^{n} x_i = (x_1 + \cdots + x_n)$$

```
mySum <- function( x ) {
        Num <- length( x )
        temp_sum <- 0
        for( k in 1:Num ) {
                temp_sum <- temp_sum + x[ k ]
        }
        return( temp_sum )
}
```

- mySum(1:10)
- sum(1:10)

自定函數 – 計算平均值

- 給定一組數據，計算所有數據的平均值

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{1}{n}(x_1 + \cdots + x_n)$$

```
myMean <- function( x ) {
        Num <- length( x )
        temp_sum <- 0
        for( k in 1:Num ) {
                temp_sum <- temp_sum + x[ k ]
        }
        return( temp_sum / Num )
}
```

- myMean (1:10)
- mean(1:10)

自定函數 – 計算平均值

- 給定一組數據，計算所有數據的平均值
- myMean <- function(x) {

```
    temp_Num <- length( x )
    temp_Sum <- mySum( x )
    return( temp_Sum / temp_Num )
}
```

- myMean (1:10)
- mean(1:10)

自定函數 – 計算標準差

- 給定一組數據，計算所有數據的標準差
- 母體的標準差

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(X_i - \mu)^2}.$$

- 樣本的標準差

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2}.$$

- Data: **x[1], x[2], x[3], ...**
- mean: **x_mean** <- myMean(**x**)
- diff:

  x_diff_1[i] <- **x**[i] - **x_mean**

  x_diff_2[i] <- (**x_diff_1**[i])^2

  x_diff_3[i] <- (**x**[i] - **x_mean**)^2
- sum: **x_sum** <- mySum(**x_diff_2**)

  x_sum <- mySum(**x_diff_3**)
- sqrt: sqrt (**x_sum / (N – 1)**)

丁村成，從標準差除以 n 或除以 n – 1 談起
http://web.math.sinica.edu.tw/math_media/d291/29102.pdf

自定函數 – 計算標準差

- 給定一組數據，計算所有數據的標準差
- mySD <- function(x) {

```
    Num       <- length( x )
    temp_sum  <- 0
    temp_mean <- 0
    temp_diff <- rep(0, length.out = Num)
    for( k in 1:Num ) {
        temp_sum <- temp_sum + x[ k ]
    }
    temp_mean <- temp_sum/Num
    for( k in 1:Num ) {
        temp_diff[ k ] <- ( x[ k ] - temp_mean )^2
    }
    temp_sum_diff <- mySum( temp_diff )
    temp_SD <- sqrt( temp_sum_diff / ( Num - 1 ) )
    return( temp_SD )
}
```

- mySD(1:10)
- sd(1:10)

自定函數 – 計算標準差

```
mySD <- function( x ) {
        Num       <- length( x )
        temp_sum  <- 0
        temp_mean <- 0
        temp_diff <- rep(0, length.out = Num)
        for( k in 1:Num ) {
                temp_sum <- temp_sum + x[ k ]
        }
        temp_mean <- temp_sum/Num
        print( temp_mean )
        for( k in 1:Num ) {
                temp_diff[ k ] <- ( x[ k ] - temp_mean )^2
        }
        print( temp_diff )
        temp_sum_diff <- mySum( temp_diff )
        print( temp_sum_diff )
        temp_SD <- sqrt( temp_sum_diff / ( Num - 1 ) )
        return( temp_SD )
}
```

- mySD(1:10)
- sd(1:10)

自定函數 – 數據排序

- 給定一組數據，按照大小順序排序

```
> mySort( c( 3, 5, 2, 1 ) )
[1] 1 2 3 5
> sort( c( 3, 5, 2, 1 ) )
[1] 1 2 3 5

> mySort( c( -3, 5, 2, 1, -2, 4 ) )
[1] -3 -2  1  2  4  5
> sort( c( -3, 5, 2, 1, -2, 4 ) )
[1] -3 -2  1  2  4  5
```

自定函數 – 數據排序

- 0 9 8 7 6 5 4 3 2 1
- 1 9 8 7 6 5 4 3 2 1
- 1 9 8 7 6 5 4 3 2 1
- 1 8 9 7 6 5 4 3 2 1
- 2 8 9 7 6 5 4 3 2 1
- 2 8 9 7 6 5 4 3 2 1
- 2 8 7 9 6 5 4 3 2 1
- 3 8 7 9 6 5 4 3 2 1
- 3 8 7 9 6 5 4 3 2 1
- 3 8 7 6 9 5 4 3 2 1
- 4 8 7 6 9 5 4 3 2 1
- 4 8 7 6 9 5 4 3 2 1
- 4 8 7 6 5 9 4 3 2 1
- 5 8 7 6 5 9 4 3 2 1
- 5 8 7 6 5 9 4 3 2 1
- 5 8 7 6 5 4 9 3 2 1
- 6 8 7 6 5 4 9 3 2 1
- 6 8 7 6 5 4 9 3 2 1
- 6 8 7 6 5 4 3 9 2 1
- 7 8 7 6 5 4 3 9 2 1
- 7 8 7 6 5 4 3 9 2 1
- 7 8 7 6 5 4 3 2 9 1
- 8 8 7 6 5 4 3 2 9 1
- 8 8 7 6 5 4 3 2 9 1
- 8 8 7 6 5 4 3 2 1 9
- 9 8 7 6 5 4 3 2 1 9

自定函數－數據排序

- 0 8 7 6 5 4 3 2 1 9
- 1 8 7 6 5 4 3 2 1 9
- 1 8 7 6 5 4 3 2 1 9
- 1 7 8 6 5 4 3 2 1 9
- 2 7 8 6 5 4 3 2 1 9
- 2 7 8 6 5 4 3 2 1 9
- 2 7 6 8 5 4 3 2 1 9
- 3 7 6 8 5 4 3 2 1 9
- 3 7 6 8 5 4 3 2 1 9
- 3 7 6 5 8 4 3 2 1 9
- 4 7 6 5 8 4 3 2 1 9
- 4 7 6 5 8 4 3 2 1 9
- 4 7 6 5 4 8 3 2 1 9
- 5 7 6 5 4 8 3 2 1 9
- 5 7 6 5 4 8 3 2 1 9
- 5 7 6 5 4 3 8 2 1 9
- 6 7 6 5 4 3 8 2 1 9
- 6 7 6 5 4 3 8 2 1 9
- 6 7 6 5 4 3 2 8 1 9
- 7 7 6 5 4 3 2 8 1 9
- 7 7 6 5 4 3 2 8 1 9
- 7 7 6 5 4 3 2 1 8 9
- 8 7 6 5 4 3 2 1 8 9

自定函數－數據排序

- 0 8 7 6 5 4 3 2 1 9
- …
- 8 7 6 5 4 3 2 1 8 9

- 0 7 6 5 4 3 2 1 8 9
- …
- 7 6 5 4 3 2 1 7 8 9

- 0 6 5 4 3 2 1 7 8 9
- …
- 6 5 4 3 2 1 6 7 8 9

- 0 5 4 3 2 1 6 7 8 9
- …
- 5 4 3 2 1 5 6 7 8 9

- 0 4 3 2 1 5 6 7 8 9
- …
- 4 3 2 1 4 5 6 7 8 9

- 0 3 2 1 4 5 6 7 8 9
- …
- 3 2 1 3 4 5 6 7 8 9

- 0 2 1 3 4 5 6 7 8 9
- …
- 2 1 2 3 4 5 6 7 8 9

自定函數 – 數據排序

- 0 3 5 1 2 4

- 2 3 5 1 2 4
- 2 3 1 5 2 4

- 3 3 1 5 2 4
- 3 3 1 2 5 4

- 4 3 1 2 5 4
- 4 3 1 2 4 5

- 0 3 1 2 4 5

- 1 3 1 2 4 5
- 1 1 3 2 4 5

- 2 1 3 2 4 5
- 2 1 2 3 4 5

- 0 3 2 5 4 1

- 1 3 2 5 4 1
- 1 2 3 5 4 1

- 3 2 3 5 4 1
- 3 2 3 4 5 1

- 4 2 3 4 5 1
- 4 2 3 4 1 5

- 3 2 3 4 1 5
- 3 2 3 1 4 5

- 2 2 3 1 4 5
- 2 2 1 3 4 5

- 1 2 1 3 4 5
- 1 1 2 3 4 5

自定函數 – 數據排序

```
mySort <- function( x ) {
        itemCount <- length( x )
        repeat {
                hasChanged <- FALSE
                itemCount <- itemCount - 1
                if ( itemCount >= 1 ){
                for( k in 1 : itemCount ) {
                        if ( x[ k ] > x[ k+1 ] ) {
                                t <- x[ k ]
                                x[ k ] <- x[ k+1 ]
                                x[ k+1 ] <- t
                                hasChanged <- TRUE
                        }
                }
                }
                if ( !hasChanged ) break;
        }
        return( x )
}
```

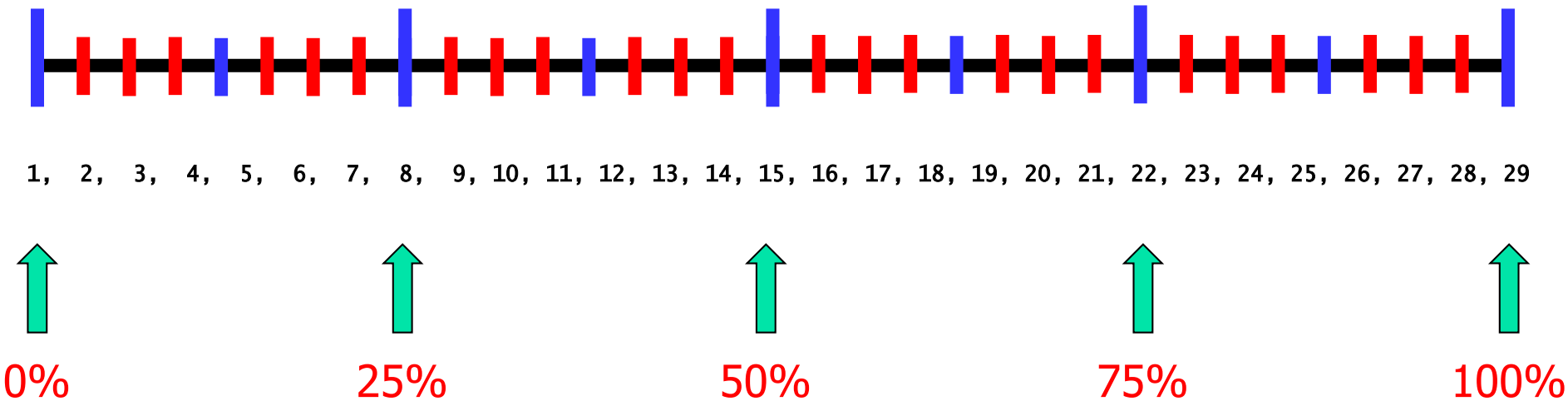
自定函數 – 最大值，最小值

```
myMax <- function( x ) {
        Num <- length( x )
        temp <- mySort( x )
        return( temp[ Num ] )
}

myMin <- function( x ) {
        Num <- length( x )
        temp <- mySort( x )
        return( temp[ 1 ] )
}

myRange <- function( x ) {
        temp_max <- myMax( x )
        temp_min <- myMin( x )
        temp <- c( temp_min, temp_max )
        return( temp )
}
```

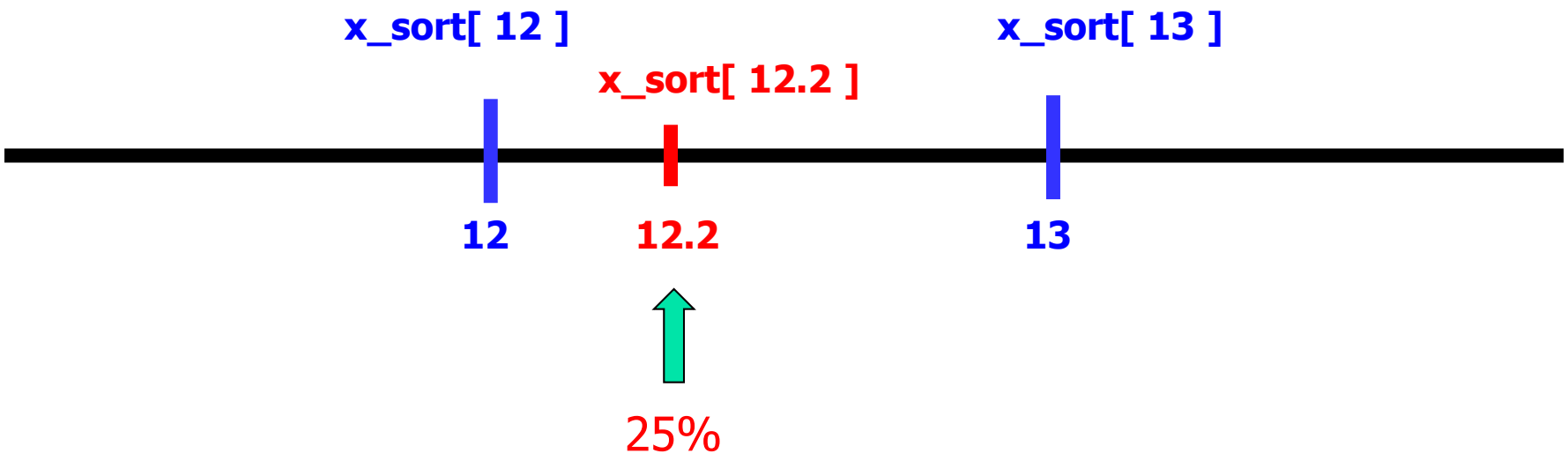
自定函數－中位數，四位數



- 如果有101個數據，也就是：100個間隔
- 50% 即是第51個 數據，25% 即是第26個 數據，75% 即是第76個 數據
- 如果不是剛好的個數，
  - 例如：25% 是 第 12.2 個 數據
  - 則這個 25% 的數據會是：第 12 個 數據 與 第 13 個 數據 的比例分配計算，
  - 也就是：

自定函數 – 中位數，四位數

- 也就是：



$$x\_sort[\ 25\%\ ] =$$

$$x\_sort[\ 12.2\ ] = x\_sort[\ 12\ ] + \frac{(\ x\_sort[\ 13\ ] - x\_sort[\ 12\ ]\ )}{13 - 12} * (\ 12.2 - 12\ )$$

自定函數 – 中位數，meidan

```
myMedian <- function( x ) {
        Num <- length( x )
        x_sort <- mySort( x )
        if( (Num-1)/2 == floor( (Num-1)/2 ) ){
                temp_median <- x_sort[ (Num-1)/2+1 ]
        }
        else {
                temp <- floor( Num/2 )
                temp_median <- ( x_sort[ temp ] + x_sort[ temp+1 ] )/2
        }
        return( temp_median )
}
```

自定函數 – 四位數 (25%, 75%)

```
my25p <- function( x ) {
        tp_num <- length( x )
        x_sort <- mySort( x )
        tp_max <- myMax( x )
        tp_min <- myMin( x )
        tp1 <- floor ( (tp_num-1) * 0.25 )  +1
        tp2 <- ( (tp_num-1) * 0.25 )   +1
        tp3 <- ceiling ( (tp_num-1) * 0.25 ) +1

        if( tp2 == tp1 ){
                tp_25p <- x_sort[ tp1 ]
        }
        else {

tp_25p <- x_sort[ tp1 ] + ( (tp2-tp1)/(tp3-tp1) )*( x_sort[tp3]-x_sort[tp1] )

        }
        return( tp_25p )
}
```

自定函數 – 四位數 (25%, 75%)

```
my75p <- function( x ) {
        tp_num <- length( x )
        x_sort <- mySort( x )
        tp_max <- myMax( x )
        tp_min <- myMin( x )
        tp1 <- floor ( (tp_num-1) * 0.75 )  +1
        tp2 <- ( (tp_num-1) * 0.75 )   +1
        tp3 <- ceiling ( (tp_num-1) * 0.75 ) +1

        if( tp2 == tp1 ){
                tp_75p <- x_sort[ tp1 ]
        }
        else {

tp_75p <- x_sort[ tp1 ] + ( (tp2-tp1)/(tp3-tp1) )*( x_sort[tp3]-x_sort[tp1] )

        }
        return( tp_75p )
}
```

自定函數 – mySummary

```
mySummary <- function( x ) {

tp_summary <- c( myMin( x ), my25p( x ), myMedian( x ), myMean( x ),
my75p( x ), myMax( x ) )

        return(  tp_summary )
}
```

大綱

自訂數學函數

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)

大綱

下課了