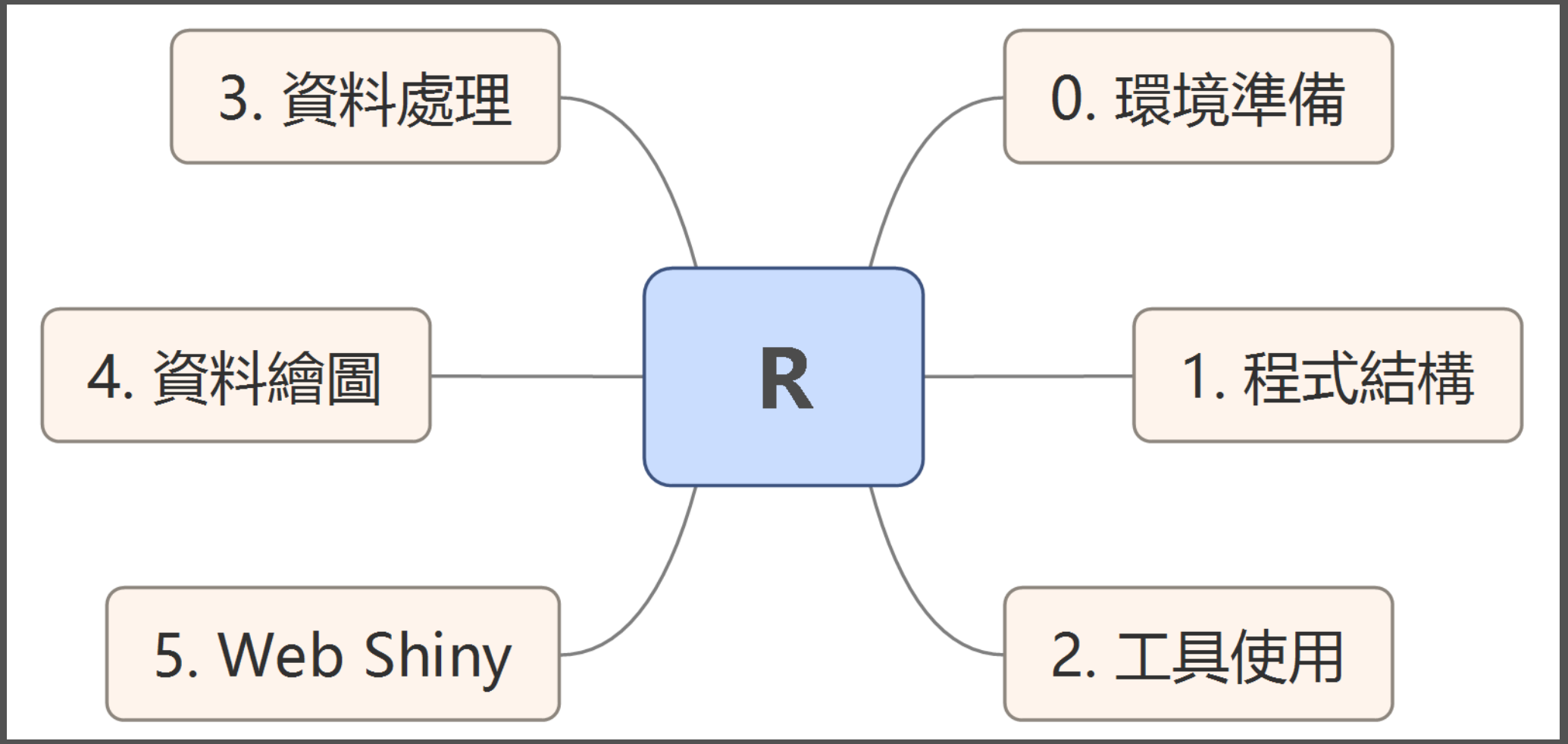
R 資料處理

目的

在此說明 R 資料清洗方案與分析方法。



R – Apply 系列函數

1. apply
2. tapply
3. lapply
4. sapply
5. mapply
6. rapply
7. vapply
8. eapply

R – Apply 系列函數 – apply

最常用於替代 for 的函數， 1 為橫列，2 為直行。

```
> # apply
>  (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(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
 (x = matrix(1:24, nrow=4))
#1: rows, 2:columns
apply(x, 1, sum)
apply(x, 2, sum)
apply(x, 1, sqrt)
apply(x, 2, sqrt)
```

apply {base} R Documentation

Apply Functions Over Array Margins

Description

Returns a vector or array or list of values obtained by applying a function to margins of an array or matrix.

Usage

```
apply(X, MARGIN, FUN, ...)
```

R – Apply 系列函數 – apply

```
theMatrix = matrix(1:9, nrow = 3)
apply(theMatrix, 1, sum)
apply(theMatrix, 2, sum)
rowSums(theMatrix)
colSums(theMatrix)
```

apply([所操作的物件與資料],[利用物件的行與列],[要套用的函數])

apply([所操作的物件與資料],[利用物件的行與列],[要套用的函數],[引數])

下面建立一個三列的矩陣

1 代表對列加總

1 + 4 + 7 = 12

2 + 5 + 8 = 15

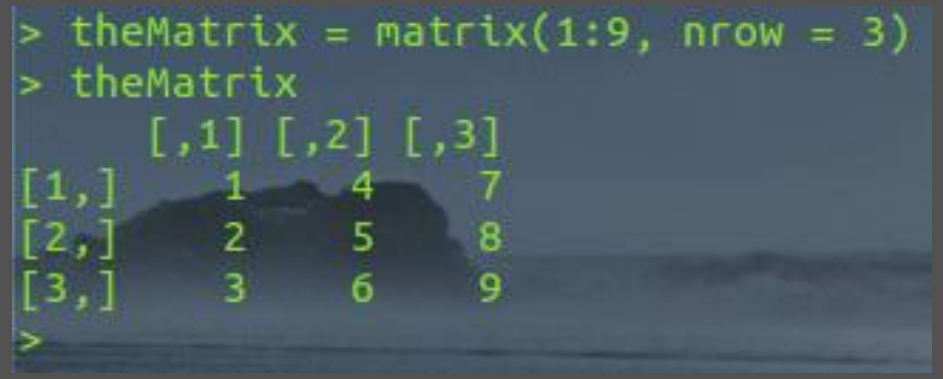
3 + 6 + 9 = 18

2 代表對行加總

1 + 2 + 3 = 6

4 + 5 + 6 = 15

7 + 8 + 9 = 24

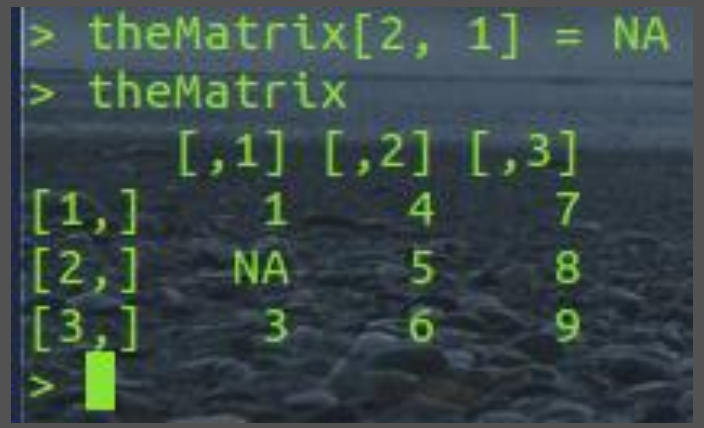



```
> theMatrix = matrix(1:9, nrow = 3)
>
> apply(theMatrix, 1, sum)
[1] 12 15 18
>
> apply(theMatrix, 2, sum)
[1]  6 15 24
>
> rowSums(theMatrix)
[1] 12 15 18
> colSums(theMatrix)
[1]  6 15 24
>
```

R – Apply 系列函數 – apply

將矩陣的第一行、第二列的值 塞成 NA，我們可以觀察到 因為第二列有 NA值，在列加總時結果會出現 NA，如果使用 na.rm 的引數，在其列會繞過 NA值 直接加總，這個引數在 apply()、rowSums() 皆可使用...

```
theMatrix[2, 1] = NA
apply(theMatrix, 1, sum)
apply(theMatrix, 1, sum, na.rm = TRUE)
rowSums(theMatrix)
rowSums(theMatrix, na.rm = TRUE)
```




```
> theMatrix[2, 1] = NA
> apply(theMatrix, 1, sum)
[1] 12 NA 18
> apply(theMatrix, 1, sum, na.rm = TRUE)
[1] 12 13 18
> rowSums(theMatrix)
[1] 12 NA 18
> rowSums(theMatrix, na.rm = TRUE)
[1] 12 13 18
>
>
```

R – Apply 系列函數 – lapply & sapply & vapply

lapply

From base v3.4.1
by R-core R-core@R-project.org

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`.

`sapply` is a user-friendly version and wrapper of `lapply` by default returning a vector, matrix or, if `simplify = "array"`, an array if appropriate, by applying `simplify2array()`.
`sapply(x, f, simplify = FALSE, USE.NAMES = FALSE)` is the same as `lapply(x, f)`.

`vapply` is similar to `sapply`, but has a pre-specified type of return value, so it can be safer (and sometimes faster) to use.

`replicate` is a wrapper for the common use of `sapply` for repeated evaluation of an expression (which will usually involve random number generation).

`simplify2array()` is the utility called from `sapply()` when `simplify` is not false and is similarly called from `mapply()`.

Note

`sapply(*, simplify = FALSE, USE.NAMES = FALSE)` is equivalent to `lapply(*)`.

For historical reasons, the calls created by `lapply` are unevaluated, and code has been written (e.g., `bquote`) that relies on this. This means that the recorded call is always of the form `FUN(X[[i]], ...)`, with `i` replaced by the current (integer or double) index. This is not normally a problem, but it can be if `FUN` uses `sys.call` or `match.call` or if it is a primitive function that makes use of the call. This means that it is often safer to call primitive functions with a wrapper, so that e.g. `lapply(ll, function(x) is.numeric(x))` is required to ensure that method dispatch for `is.numeric` occurs correctly.

If `expr` is a function call, be aware of assumptions about where it is evaluated, and in particular what `...` might refer to. You can pass additional named arguments to a function call as additional named arguments to `replicate`: see 'Examples'.

R – Apply 系列函數 – lapply & sapply & vapply

先產生範例資料。

```
> # lapply & sapply & vapply
> # 產生範例資料
> lsv.x = list(a = 1:10,
+ beta = exp(-3:3),
+ logic = c( TRUE, FALSE, FALSE, TRUE))
>
> lsv.x
$a
 [1]  1  2  3  4  5  6  7  8  9 10

$beta
[1]  0.04978707  0.13533528  0.36787944  1.00000000  2.71828183  7.38905610
[7] 20.08553692

$logic
[1]  TRUE FALSE FALSE  TRUE

>
```

```
# lapply & sapply & vapply
# 產生範例資料
lsv.x = list(a = 1:10,
   beta = exp(-3:3),
   logic = c( TRUE, FALSE, FALSE, TRUE))

lsv.x
```

lapply 會對 list 和 data.frame 進行處理，回傳的方式將會是 list 的型態。。

```
# lapply 處理後會回傳 List
(mly = lapply( lsv.x, mean))
class(mly)
(lapply( lsv.x, quantile, probs = 1:3/4))
```

```
> # lapply 處理後會回傳 List
> (mly = lapply( lsv.x, mean)
+ )
$a
[1] 5.5

$beta
[1] 4.535125

$logic
[1] 0.5

> class(mly)
[1] "list"
> (lapply( lsv.x, quantile, probs = 1:3/4)
+ )
$a
 25%  50%  75%
3.25 5.50 7.75

$beta
      25%       50%       75%
0.2516074 1.0000000 5.0536690

$logic
25% 50% 75%
0.0 0.5 1.0
```

R – Apply 系列函數 – lapply & sapply & vapply

sapply 與 lapply 類似，差異在於回傳的方式主要為向量，並多了 simplify 與 USE.NAMES 的兩個參數，而 simplify 與 USE.NAMES 為 FALSE，其結果與 lapply 一致。

```
> (sapply( lsv.x, quantile))
          a         beta logic
0%     1.00   0.04978707   0.0
25%    3.25   0.25160736   0.0
50%    5.50   1.00000000   0.5
75%    7.75   5.05366896   1.0
100% 10.00 20.08553692   1.0
> i39 = sapply( 3:9, seq)
>
> class( i39)
[1] "list"
>
```

```
> # sapply 類似於 lapply 但會回傳向量
> # 如參數 simplify & USE.NAMES 為 FALSE
> # 結果會跟 lapply 一致
> ( sly.5n = sapply( i39, fivenum)
+ )
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]  1.0  1.0    1  1.0  1.0  1.0    1
[2,]  1.5  1.5    2  2.0  2.5  2.5    3
[3,]  2.0  2.5    3  3.5  4.0  4.5    5
[4,]  2.5  3.5    4  5.0  5.5  6.5    7
[5,]  3.0  4.0    5  6.0  7.0  8.0    9
> class(sly.5n)
[1] "matrix"
```

```
> ( sly.5nf = sapply( i39, fivenum,
+ simplify = FALSE ,
+ USE.NAMES = FALSE )
+ )
[[1]]
[1] 1.0 1.5 2.0 2.5 3.0

[[2]]
[1] 1.0 1.5 2.5 3.5 4.0

[[3]]
[1] 1 2 3 4 5

[[4]]
[1] 1.0 2.0 3.5 5.0 6.0

[[5]]
[1] 1.0 2.5 4.0 5.5 7.0

[[6]]
[1] 1.0 2.5 4.5 6.5 8.0

[[7]]
[1] 1 3 5 7 9

> class(sly.5nf)
[1] "list"
>
```

```
(sapply( lsv.x, quantile))
i39 = sapply( 3:9, seq)
class( i39)
( sly.5n = sapply( i39, fivenum))
class(sly.5n)
( sly.5nf = sapply( i39, fivenum,
   simplify = FALSE ,
   USE.NAMES = FALSE ))
class(sly.5nf)
```

R – Apply 系列函數 – lapply & sapply & vapply

lapply([所操作的物件與資料], [要套用的函數])

** 將 list 轉為 vector 形式，可使用 sapply 函數

sapply([所操作的物件與資料], [要套用的函數])

在此建立一個名為 theList 的 list 資料，當中塞了 3*3 與 2*2 兩個矩陣、一個向量與一個值

theList = list(A = matrix(1:9, 3), B = 1:5, C = matrix(1:4, 2), D = 2)

```
> theList = list( A = matrix(1:9, 3), B = 1:5, C = matrix(1:4, 2), D = 2)
> theList
$A
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

$B
[1] 1 2 3 4 5

$C
     [,1] [,2]
[1,]    1    3
[2,]    2    4

$D
[1] 2

>
```

R – Apply 系列函數 – lapply & sapply & vapply

lapply() 可以進行 list 並使用 sum 函數，將所有的 list 進行加總，sapply() 跟 lapply() 一樣，差異在sapply() 會用 vector 回傳。

```
> theList = list(A = matrix(1:9, 3), B = 1:5, C = matrix(1:4, 2), D = 2)
> lapply(theList, sum)
$A
[1] 45

$B
[1] 15

$C
[1] 10

$D
[1] 2
```

```
> sapply(theList, sum)
 A  B  C  D
45 15 10  2
>
>
> theNames = c("Jared", "Deb", "Paul")
> lapply(theNames, nchar)
[[1]]
[1] 5

[[2]]
[1] 3

[[3]]
[1] 4
```

```
lapply(theList, sum)
sapply(theList, sum)
theNames = c("Jared", "Deb", "Paul")
lapply(theNames, nchar)
```

R – Apply 系列函數 – rapply

為 lapply 的遞迴版本，在此建立 v1、v2、v3 的範例資料。

```
> # rapply 為 lapply 的遞迴版本
>
> (v1 = list( n1 = 8, n2 = 9:29, c = c('b','a')))
$n1
[1] 8

$n2
 [1]  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29

$c
[1] "b" "a"

> (v2 = pi)
[1] 3.141593
> (v3 = data.frame(a=rnorm(10),b=1:10))
           a  b
1  -0.5849540  1
2   0.3689759  2
3  -0.6991427  3
4  -0.2108264  4
5   1.0685365  5
6   0.6618870  6
7   0.5033362  7
8   1.2411706  8
9  -0.1419069  9
10 -0.6840646 10
> rly.data = list( v1, v2, v3)
```

在此用 sort 進行處理與排序，並選擇所要處理的類型，因為選擇的數值，除了當中不會對文字進行排序。

```
> rapply( rly.data, sort,
+ classes = 'numeric',how = 'replace')
[[1]]
[[1]]$n1
[1] 8

[[1]]$n2
 [1]  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29

[[1]]$c
[1] "b" "a"


[[2]]
[1] 3.141593

[[3]]
           a  b
1  -0.6991427  1
2  -0.6840646  2
3  -0.5849540  3
4  -0.2108264  4
5  -0.1419069  5
6   0.3689759  6
7   0.5033362  7
8   0.6618870  8
9   1.0685365  9
10  1.2411706 10
```

```
(v1 = list( n1 = 8, n2 = 9:29, c = c('b','a')))
(v2 = pi)
(v3 = data.frame(a=rnorm(10),b=1:10))
rly.data = list( v1, v2, v3)
rapply( rly.data, sort, classes = 'numeric',how = 'replace')
```

R – Apply 系列函數 – rapply

用另外一種方式進行處理，處理的型態設為文字，抓到文字後
對該文字的值添加 "no.sort"。

```
> rapply( rly.data, function(x)paste( x,'no.sort'),
+ classes = "character",
+ deflt = NA, how = "list")
[[1]]
[[1]]$n1
[1] NA

[[1]]$n2
[1] NA

[[1]]$c
[1] "b no.sort" "a no.sort"


[[2]]
[1] NA

[[3]]
[[3]]$a
[1] NA

[[3]]$b
[1] NA
```

```
rapply( rly.data, function(x)paste( x,'no.sort'),
   classes = "character",
   deflt = NA, how = "list")
```

R – Apply 系列函數 – mapply

mapply 可以直接收多個向量的資料直接進行處理。

```
> # mapply 與 sapply 類似
> # 好處在於可以傳入多的向量資料直接處理
> # 回傳向量和矩陣
>
> (v1 = 5:15)
 [1]  5  6  7  8  9 10 11 12 13 14 15
> (v2 = -5:-15)
 [1]  -5  -6  -7  -8  -9 -10 -11 -12 -13 -14 -15
> (v3 = round(runif(11,-5,5)))
 [1] -1 -1 -2 -4  5  2  3  5  1 -2 -1
>
> mapply( max, v1, v2, v3)
 [1]  5  6  7  8  9 10 11 12 13 14 15
> mapply( sum, v1, v2, v3)
 [1] -1 -1 -2 -4  5  2  3  5  1 -2 -1
> v1
 [1]  5  6  7  8  9 10 11 12 13 14 15
> v2
 [1]  -5  -6  -7  -8  -9 -10 -11 -12 -13 -14 -15
> v3
 [1] -1 -1 -2 -4  5  2  3  5  1 -2 -1
>
```

(v1 = 5:15)
(v2 = -5:-15)
(v3 = round(runif(11,-5,5)))
mapply(max, v1, v2, v3)
mapply(sum, v1, v2, v3)
v1
v2
v3

R – Apply 系列函數 – mapply

mapply([對所需要執行的動作],[目標 list 1],[目標 list 2],....)

如要對好幾個 list 當中的元素進行大量套用可以使用 mapply() 函數，不一定需要利用 迴圈，在此建立兩個 list 並做比對...，觀察兩個 list 當中的 B 矩陣。

```
firstList = list(A = matrix(1:16, 4),
                 B = matrix(1:16, 2),
                 C = 1:5)
secondList = list(A = matrix(1:16, 4),
                  B = matrix(1:16, 8),
                  C = 15:1)
```

```
> firstList = list(A = matrix(1:16, 4), B = matrix(1:16, 2), C = 1:5)
> firstList
$A
     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
[3,]    3    7   11   15
[4,]    4    8   12   16

$B
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]    1    3    5    7    9   11   13   15
[2,]    2    4    6    8   10   12   14   16

$C
[1] 1 2 3 4 5
```

```
> secondList = list(A = matrix(1:16, 4), B = matrix(1:16, 8), C = 15:1)
> secondList
$A
     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
[3,]    3    7   11   15
[4,]    4    8   12   16

$B
     [,1] [,2]
[1,]    1    9
[2,]    2   10
[3,]    3   11
[4,]    4   12
[5,]    5   13
[6,]    6   14
[7,]    7   15
[8,]    8   16

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

R – Apply 系列函數 – mapply

可使用 identical 函數來做比較，看看內容是否相同，再自建一個簡易的函數 simpleFunc，將各元素的 “列” 數量相加。

A -> 4 + 4

B -> 2 + 8

C -> 5 + 15

注意一下向量那邊似乎比較特殊 !!!

```
> firstList = list(A = matrix(1:16, 4), B = matrix(1:16, 2), C = 1:5)
> secondList = list(A = matrix(1:16, 4), B = matrix(1:16, 8), C = 15:1)
>
> mapply(identical, firstList, secondList)
    A     B     C
 TRUE FALSE FALSE
>
>
> simpleFunc = function(x, y)
+ {
+     NROW(x) + NROW(y)
+ }
>
> mapply(simpleFunc, firstList, secondList)
 A  B  C
 8 10 20
> |
```

```
mapply(identical, firstList, secondList)
```

```
simpleFunc = function(x, y)
{
  NROW(x) + NROW(y)
}
```

```
mapply(simpleFunc, firstList, secondList)
```

R – Apply 系列函數 – tapply

```
> # tapply
> # 目的在於分組計算，進行類似於 group by 的處理
> # 會回傳向量資料
>
> tapply( iris[ , 2], iris[ , 4], mean)
     0.1      0.2      0.3      0.4      0.5      0.6        1      1.1
3.360000 3.379310 3.328571 3.785714 3.300000 3.500000 2.371429 2.466667
     1.2      1.3      1.4      1.5      1.6      1.7      1.8      1.9
2.740000 2.746154 2.912500 2.816667 3.100000 2.750000 2.941667 2.680000
       2      2.1      2.2      2.3      2.4      2.5
3.016667 3.033333 3.200000 3.087500 3.100000 3.400000
> as.factor(iris[ , 4])
  [1] 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 0.2 0.2 0.1 0.1 0.2 0.4 0.4 0.3
 [19] 0.3 0.3 0.2 0.4 0.2 0.5 0.2 0.2 0.4 0.2 0.2 0.2 0.2 0.4 0.1 0.2 0.2 0.2
 [37] 0.2 0.1 0.2 0.2 0.3 0.3 0.2 0.6 0.4 0.3 0.2 0.2 0.2 0.2 1.4 1.5 1.5 1.3
 [55] 1.5 1.3 1.6 1   1.3 1.4 1   1.5 1   1.4 1.3 1.4 1.5 1   1.5 1.1 1.8 1.3
 [73] 1.5 1.2 1.3 1.4 1.4 1.7 1.5 1   1.1 1   1.2 1.6 1.5 1.6 1.5 1.3 1.3 1.3
 [91] 1.2 1.4 1.2 1   1.3 1.2 1.3 1.3 1.1 1.3 2.5 1.9 2.1 1.8 2.2 2.1 1.7 1.8
[109] 1.8 2.5 2   1.9 2.1 2   2.4 2.3 1.8 2.2 2.3 1.5 2.3 2   2   1.8 2.1 1.8
[127] 1.8 1.8 2.1 1.6 1.9 2   2.2 1.5 1.4 2.3 2.4 1.8 1.8 2.1 2.4 2.3 1.9 2.3
[145] 2.5 2.3 1.9 2   2.3 1.8
22 Levels: 0.1 0.2 0.3 0.4 0.5 0.6 1 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 ... 2.5
```

```
> irm = iris[ iris$Petal.Width == 0.1, ]
>
> (t.ans1 = mean(irm[ , 2]))
[1] 3.36
> (t.ans2 = as.numeric(
+ tapply( iris[ , 2], iris[ , 4], mean)[1])
+ )
[1] 3.36
> t.ans1 == t.ans2
[1] TRUE
>
```

tapply 用於分組計算，在此用 iris 資料集，然選擇用平均數來處理，iris[, 4] 傳入 tapply 的 INDEX 的參數，當中會以 INDEX 參數做為索引對 iris[, 2] 的值進行處理。

所以 iris[, 4] 為 0.1 ~ 2.5 ，在此將 iris[, 4] 在 0.1 的資料去掉，取平均數與剛剛 tapply 結果的 0.1 相比 會是一致的。

```
tapply( iris[ , 2], iris[ , 4], mean)
as.factor(iris[ , 4])
irm = iris[ iris$Petal.Width == 0.1, ]

(t.ans1 = mean(irm[ , 2]))
(t.ans2 = as.numeric(
   tapply( iris[ , 2], iris[ , 4], mean)[1])
   )
t.ans1 == t.ans2
```

R – Apply 系列函數 – eapply

eapply 為用於 R 環境的循環處理。

所以 iris[, 4] 為 0.1 ~ 2.5 ， 在此將 iris[, 4] 在 0.1 的資料去掉， 取平均數與剛剛 tapply 結果的 0.1 相比 會是一致的。

```
> # eapply
> # 用於 R 環境的處理
> ev = new.env()
> ev$val = 1:10
> ev$beta = exp(-3:3)
> ev$logic = c( TRUE, FALSE, FALSE, TRUE)
> ev
<environment: 0x000000001793c760>
>
> # R 環境中所有變量所佔記憶體大小
> eapply(ev, mean)
$logic
[1] 0.5

$beta
[1] 4.535125

$val
[1] 5.5

>
```

```
> # 查看 R 環境中所有變量
> ls()
 [1] "ev"       "i39"      "irm"      "lsv.x"    "mly"      "rly.data"
 [7] "sly.5n"   "sly.5nf"  "t.ans1"   "t.ans2"   "v1"       "v2"
[13] "v3"       "vly.5n"   "x"
>
> # 查看 R 環境中所有變量所佔的記憶體大小
> # eapply(environment(), object.size)
> str(eapply(environment(), object.size))
List of 15
 $ vly.5n  :Class 'object_size'  num 976
 $ i39     :Class 'object_size'  num 592
 $ lsv.x   :Class 'object_size'  num 648
 $ x       :Class 'object_size'  num 328
 $ t.ans1  :Class 'object_size'  num 48
 $ t.ans2  :Class 'object_size'  num 48
 $ irm     :Class 'object_size'  num 1936
 $ rly.data:Class 'object_size'  num 1840
 $ sly.5n  :Class 'object_size'  num 480
 $ sly.5nf :Class 'object_size'  num 720
 $ ev      :Class 'object_size'  num 56
 $ v1      :Class 'object_size'  num 88
 $ v2      :Class 'object_size'  num 88
 $ v3      :Class 'object_size'  num 168
 $ mly     :Class 'object_size'  num 544
>
```

```
ev = new.env()
ev$val = 1:10
ev$beta = exp(-3:3)
ev$logic = c( TRUE, FALSE, FALSE,
TRUE)
ev
eapply(ev, mean)
ls()
str(eapply(environment(), object.size))
```

R 聚合資料 – aggregate

跟 SQL 當中的 group by 分群類似，根據某個欄位進行分群，當然 R 可以利用 aggregate 根據某個變數進行分群，在此利用 “~”，formula 進行分群。

aggregate([指定做計算的欄位] ~ [指定做依據的欄位], [資料名稱], [使用函數])
aggregate([指定做計算的欄位] ~ [指定做依據的欄位] + [添加做依據的欄位], [資料名稱], [使用函數])
aggregate(cbind([計算欄位1], [計算欄位2]) ~ [指定做依據的欄位] , [資料名稱], [使用函數])
aggregate(cbind([計算欄位1], [計算欄位2]) ~ [指定做依據的欄位] + [添加做依據的欄位], [資料名稱], [使用函數])

在此使用 ggplot2 的套件中的 diamonds 資料

```
> require(ggplot2)
Loading required package: ggplot2
> data(diamonds)
> head(diamonds)
  carat       cut color clarity depth table price    x    y    z
1  0.23     Ideal     E     SI2  61.5    55   326 3.95 3.98 2.43
2  0.21   Premium     E     SI1  59.8    61   326 3.89 3.84 2.31
3  0.23      Good     E     VS1  56.9    65   327 4.05 4.07 2.31
4  0.29   Premium     I     VS2  62.4    58   334 4.20 4.23 2.63
5  0.31      Good     J     SI2  63.3    58   335 4.34 4.35 2.75
6  0.24 Very Good     J    VVS2  62.8    57   336 3.94 3.96 2.48
> |
```

require(ggplot2)
data(diamonds)
head(diamonds)

R 聚合資料 – aggregate

使用 aggregate() 函數 對 diamond 資料
中的 cut、price 的欄位，求其平均數。

```
> aggregate(price ~ cut, diamonds, mean)
        cut    price
1      Fair 4358.758
2      Good 3928.864
3 Very Good 3981.760
4   Premium 4584.258
5     Ideal 3457.542
> |
```

aggregate(price ~ cut, diamonds, mean)
aggregate(price ~ cut + color, diamonds, mean)

如果欄位不夠可以用 "+"來增加欄位

```
> aggregate(price ~ cut + color, diamonds, mean)
         cut color    price
1       Fair     D 4291.061
2       Good     D 3405.382
3  Very Good     D 3470.467
4    Premium     D 3631.293
5      Ideal     D 2629.095
6       Fair     E 3682.312
7       Good     E 3423.644
8  Very Good     E 3214.652
9    Premium     E 3538.914
10     Ideal     E 2597.550
11      Fair     F 3827.003
12      Good     F 3495.750
13 Very Good     F 3778.820
14   Premium     F 4324.890
15     Ideal     F 3374.939
16      Fair     G 4239.255
17      Good     G 4123.482
18 Very Good     G 3872.754
19   Premium     G 4500.742
20     Ideal     G 3720.706
21      Fair     H 5135.683
22      Good     H 4276.255
23 Very Good     H 4535.390
24   Premium     H 5216.707
25     Ideal     H 3889.335
26      Fair     I 4685.446
27      Good     I 5078.533
28 Very Good     I 5255.880
29   Premium     I 5946.181
30     Ideal     I 4451.970
31      Fair     J 4975.655
32      Good     J 4574.173
33 Very Good     J 5103.513
34   Premium     J 6294.592
35     Ideal     J 4918.186
```

R 聚合資料 – aggregate

如果是根據某一個欄位，對多欄位做計算可以用 cbind() 進行合併。

```
> aggregate(cbind(price, carat) ~ cut, diamonds, mean)
        cut    price     carat
1      Fair 4358.758 1.0461366
2      Good 3928.864 0.8491847
3 Very Good 3981.760 0.8063814
4   Premium 4584.258 0.8919549
5     Ideal 3457.542 0.7028370
>
```

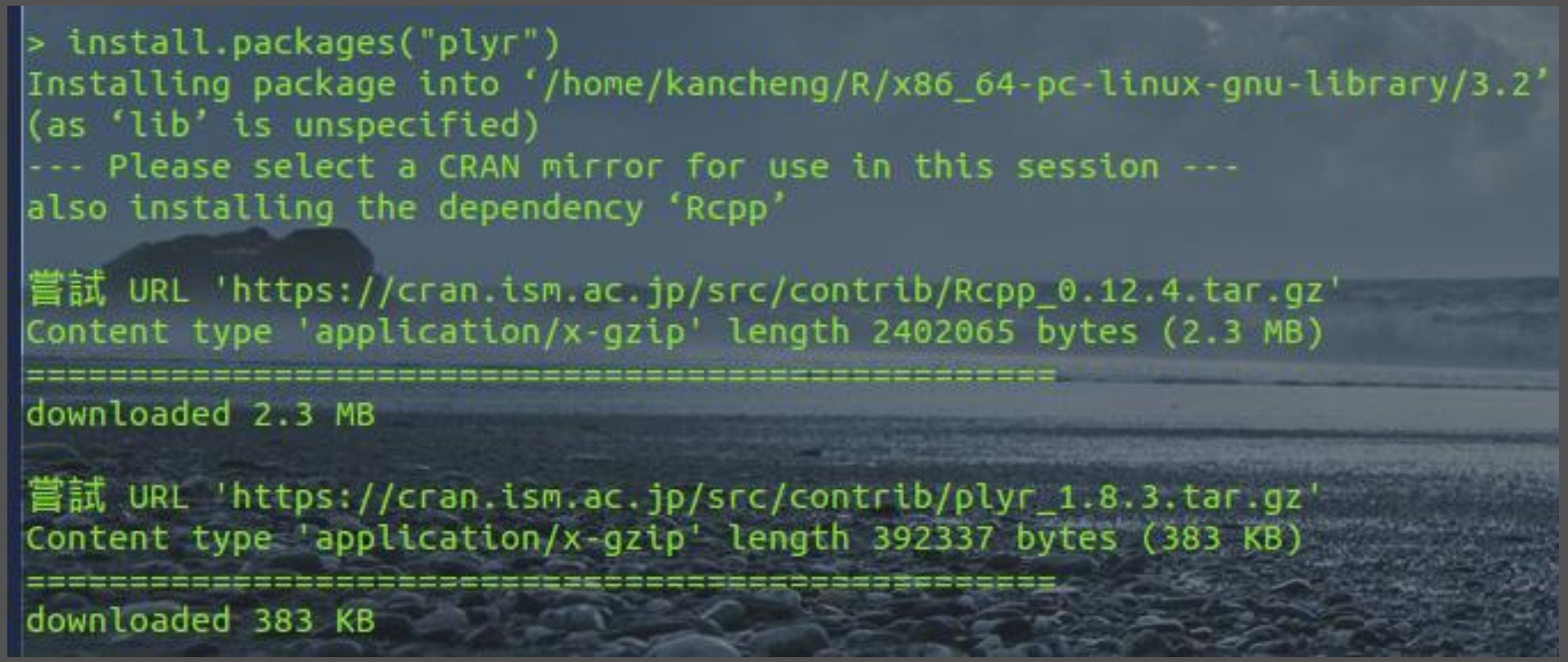
aggregate(cbind(price, carat) ~ cut, diamonds, mean)
aggregate(cbind(price, carat) ~ cut + color, diamonds, mean)

當然也可以根據多欄位對多欄位進行計算

```
> aggregate(cbind(price, carat) ~ cut + color, diamonds, mean)
        cut color    price     carat
1      Fair     D 4291.061 0.9201227
2      Good     D 3405.382 0.7445166
3 Very Good     D 3470.467 0.6964243
4   Premium     D 3631.293 0.7215471
5     Ideal     D 2629.095 0.5657657
6      Fair     E 3682.312 0.8566071
7      Good     E 3423.644 0.7451340
8 Very Good     E 3214.652 0.6763167
9   Premium     E 3538.914 0.7177450
10    Ideal     E 2597.550 0.5784012
11     Fair     F 3827.003 0.9047115
12     Good     F 3495.750 0.7759296
13 Very Good     F 3778.820 0.7409612
14  Premium     F 4324.890 0.8270356
15    Ideal     F 3374.939 0.6558285
16     Fair     G 4239.255 1.0238217
17     Good     G 4123.482 0.8508955
18 Very Good     G 3872.754 0.7667986
19  Premium     G 4500.742 0.8414877
20    Ideal     G 3720.706 0.7007146
21     Fair     H 5135.683 1.2191749
22     Good     H 4276.255 0.9147293
23 Very Good     H 4535.390 0.9159485
24  Premium     H 5216.707 1.0164492
25    Ideal     H 3889.335 0.7995249
26     Fair     I 4685.446 1.1980571
27     Good     I 5078.533 1.0572222
28 Very Good     I 5255.880 1.0469518
29  Premium     I 5946.181 1.1449370
30    Ideal     I 4451.970 0.9130291
31     Fair     J 4975.655 1.3411765
32     Good     J 4574.173 1.0995440
33 Very Good     J 5103.513 1.1332153
34  Premium     J 6294.592 1.2930941
35    Ideal     J 4918.186 1.0635937
```

R plyr

install.packages(plyr)



R plyr

require(plyr)
head(baseball)

```
> require(plyr)
Loading required package: plyr
> head(baseball)
           id year stint team lg  g  ab  r  h X2b X3b hr rbi sb cs bb so ibb hbp sh sf gidp
4   ansonca01 1871     1  RC1    25 120 29 39  11   3  0  16  6  2  2  1  NA  NA NA NA   NA
44  forceda01 1871     1  WS3    32 162 45 45   9   4  0  29  8  0  4  0  NA  NA NA NA   NA
68  mathebo01 1871     1  FW1    19  89 15 24   3   1  0  10  2  1  2  0  NA  NA NA NA   NA
99  startjo01 1871     1  NY2    33 161 35 58   5   1  1  34  4  2  3  0  NA  NA NA NA   NA
102 suttoez01 1871     1  CL1    29 128 35 45   3   7  3  23  3  1  1  0  NA  NA NA NA   NA
106 whitede01 1871     1  CL1    29 146 40 47   6   5  1  21  2  2  4  1  NA  NA NA NA   NA
> |
```

符號解釋 :

H -> 安打(Hits)
BB -> 保送("Bases on Balls" or "Walks")
HBP -> 觸身球(Times Hit by Pitch)
AB -> 打數(At Bats)
SF -> 高飛犧牲打(Sacrifice Files)

統計數據上壘率 -> On Base Percentage, OBP

$$\mathbf{OBP} = \frac{\boldsymbol{H} + \boldsymbol{BB} + \boldsymbol{HBP}}{\boldsymbol{AB} + \boldsymbol{BB} + \boldsymbol{HBP} + \boldsymbol{SF}}$$

R plyr

條件 :

注意!! 1954 年 以前 Sacrifice Files 高飛犧牲打，被視為是 犧牲觸擊的一部份，犧牲觸擊也包含短打。

1. 1954 年以前的 Sacrifice Files 高飛犧牲打 所有資料應設為 0
2. HBP (觸身球) 存有很多 NA 值，所以設為 0
3. 除掉一季當中 AB 打數少於 50 的選手資料

利用 "[ ]"，去掉 1954 年以前 SF 的資料，將 HBP的 NA換成 0，清掉 少於 50 AB 打數 的資料，最後利用 any( ) 與 is.na() 函數進行判斷

```
baseball$sf[baseball$year < 1954] = 0
any(is.na(baseball$sf))
baseball$hbp[is.na(baseball$hbp)] = 0
any(is.na(baseball$hbp))
baseball = baseball[baseball$ab >= 50, ]
```

```
> baseball$sf[baseball$year < 1954] = 0
>
> any(is.na(baseball$sf))
[1] FALSE
>
> baseball$hbp[is.na(baseball$hbp)] = 0
>
> any(is.na(baseball$hbp))
[1] FALSE
>
> baseball = baseball[baseball$ab >= 50, ]
>
```

R plyr

```
baseball$OBP = with(baseball, (h + bb + hbp)/(ab + bb + hbp + sf))
tail(baseball)
```

我們能比對一開始匯入的資料進行觀察 !!!

```
> head(baseball)
         id year stint team lg  g  ab  r  h X2b X3b hr rbi sb cs bb so ibb
4   ansonca01 1871     1  RC1    25 120 29 39  11   3  0  16  6  2  2  1  NA
44  forceda01 1871     1  WS3    32 162 45 45   9   4  0  29  8  0  4  0  NA
68  mathebo01 1871     1  FW1    19  89 15 24   3   1  0  10  2  1  2  0  NA
99  startjo01 1871     1  NY2    33 161 35 58   5   1  1  34  4  2  3  0  NA
102 suttoez01 1871     1  CL1    29 128 35 45   3   7  3  23  3  1  1  0  NA
106 whitede01 1871     1  CL1    29 146 40 47   6   5  1  21  2  2  4  1  NA
    hbp sh sf gidp
4     0 NA  0   NA
44    0 NA  0   NA
68    0 NA  0   NA
99    0 NA  0   NA
102   0 NA  0   NA
106   0 NA  0   NA
>
```

計算 上壘率 OBP

```
> baseball$OBP = with(baseball, (h + bb + hbp)/(ab + bb + hbp + sf))
> tail(baseball)
             id year stint team lg   g  ab  r   h X2b X3b hr rbi sb cs  bb
89499 claytro01 2007     1  TOR AL  69 189 23  48  14   0  1  12  2  1  14
89502 cirilje01 2007     1  MIN AL  50 153 18  40   9   2  2  21  2  0  15
89521 bondsba01 2007     1  SFN NL 126 340 75  94  14   0 28  66  5  0 132
89523 biggicr01 2007     1  HOU NL 141 517 68 130  31   3 10  50  4  3  23
89530 ausmubr01 2007     1  HOU NL 117 349 38  82  16   3  3  25  6  1  37
89533  aloumo01 2007     1  NYN NL  87 328 51 112  19   1 13  49  3  0  27
       so ibb hbp sh sf gidp       OBP
89499  50   0   1  3  3    8 0.3043478
89502  13   0   1  3  2    9 0.3274854
89521  54  43   3  0  2   13 0.4800839
89523 112   0   3  7  5    5 0.2846715
89530  74   3   6  4  1   11 0.3180662
89533  30   5   2  0  3   13 0.3916667
>
```

R plyr

在此寫一個簡單的 函數 作為 obp 的計算的處理，並利用 ddply 函數 進行運算。
ddply([資料名稱], .variables = "[指令的欄位]", .fun = [匯出物件])

```
> obp = function(data)
+     {
+         c(OBP = with(data, sum(h + bb + hbp)/sum(ab + bb + hbp + sf)))
+     }
>
>
> careerOBP = ddply(baseball, .variables = "id", .fun = obp)
>
> careerOBP = careerOBP[order(careerOBP$OBP, decreasing = TRUE), ]
>
> head(careerOBP, 10)
            id       OBP
1089 willite01 0.4816861
875   ruthba01 0.4742209
658  mcgrajo01 0.4657478
356  gehrilo01 0.4477848
85   bondsba01 0.4444622
476  hornsro01 0.4339068
184   cobbty01 0.4329655
327   foxxji01 0.4290509
953  speaktr01 0.4283386
191  collied01 0.4251246
> |
```

```
obp = function(data)
{
        c(OBP = with(data, sum(h + bb + hbp)/sum(ab + bb + hbp + sf)))
}

careerOBP = ddply(baseball, .variables = "id", .fun = obp)
careerOBP = careerOBP[order(careerOBP$OBP, decreasing = TRUE), ]
head(careerOBP, 10)
```

R llply

在前面有說過 lapply() 函數可以將 list 的物件 進行 總和，而利用 llply() 也一樣可以做到同樣的效果。在此我們可以使用 identical() 函數進行比較，希望能用 vector 的形式回傳，我們可以使用 laply() 函數進行取代前面的 sapply() 函數。

```
> theList = list(A = matrix(1:9, 3), B = 1:5, C = matrix(1:4, 2), D = 2)
> lapply(theList, sum)
$A
[1] 45

$B
[1] 15

$C
[1] 10

$D
[1] 2
```

```
> llply(theList, sum)
$A
[1] 45

$B
[1] 15

$C
[1] 10

$D
[1] 2
```

```
> identical(lapply(theList, sum), llply(theList, sum))
[1] TRUE
>
> sapply(theList, sum)
 A  B  C  D
45 15 10  2
> laply(theList, sum)
[1] 45 15 10  2
```

theList = list(A = matrix(1:9, 3), B = 1:5, C = matrix(1:4, 2), D = 2)
lapply(theList, sum)
llply(theList, sum)
identical(lapply(theList, sum), llply(theList, sum))
sapply(theList, sum)
laply(theList, sum)

R plyr輔助函數

1. each()
plyr 套件中的函數 可以與 前面的 aggreate() 函數混用...，例如 each () 函數，將多個函數 混入使用。

aggregate([指定做計算的欄位] ~ [指定做依據的欄位], [資料名稱], each([使用函數1], [使用函數2]))

2. idata.frame()
相較於面對一般的資料框能更快的抽取資料，資料量越大效果越好。

```
> aggregate(price ~ cut, diamonds, each(mean, median))
       cut price.mean price.median
1     Fair   4358.758     3282.000
2     Good   3928.864     3050.500
3 Very Good   3981.760     2648.000
4  Premium   4584.258     3185.000
5    Ideal   3457.542     1810.000
>
>
> system.time(dlply(baseball, "id", nrow))
   user  system elapsed
   0.11    0.00    0.11
> iBaseball = idata.frame(baseball)
> system.time(dlply(iBaseball, "id", nrow))
   user  system elapsed
   0.14    0.00    0.14
> |
```

aggregate(price ~ cut, diamonds, each(mean, median))
system.time(dlply(baseball, "id", nrow))
iBaseball = idata.frame(baseball)
system.time(dlply(iBaseball, "id", nrow))

R data.table

此套件是針對 data.frame 進行強化，在此同時建立 data.table 與 data.frame 並比較其差異

```
> install.packages("data.table")
Installing package into 'C:/Users/kancheng/Documents/R/win-library/3.2'
(as 'lib' is unspecified)
--- Please select a CRAN mirror for use in this session ---
嘗試 URL 'https://cran.ism.ac.jp/bin/windows/contrib/3.2/data.table_1.9.6.zip'
Content type 'application/zip' length 1883092 bytes (1.8 MB)
downloaded 1.8 MB

package 'data.table' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\kancheng\AppData\Local\Temp\RtmpchaZAh\downloaded_packages
> require(data.table)
Loading required package: data.table
data.table 1.9.6  For help type ?data.table or https://github.com/Rdatatable/data.table/wiki
The fastest way to learn (by data.table authors): https://www.datacamp.com/courses/data-analysis-the-data-table-way
>
```

```
# install.packages(data.table)
require(data.table)
```

R data.table

建立起 data.table 與 data.frame

```
> theDF = data.frame(A=1:10,
+                    B=letters[1:10],
+                    C=LETTERS[11:20],
+                    D=rep(c("One", "Two", "Three"), length.out=10))
>
>
> theDT = data.table(A=1:10,
+                    B=letters[1:10],
+                    C=LETTERS[11:20],
+                    D=rep(c("One", "Two", "Three"), length.out=10))
```

```
> theDF
    A B C     D
1   1 a K   One
2   2 b L   Two
3   3 c M Three
4   4 d N   One
5   5 e O   Two
6   6 f P Three
7   7 g Q   One
8   8 h R   Two
9   9 i S Three
10 10 j T   One
```

```
> theDT
      A B C     D
 1:   1 a K   One
 2:   2 b L   Two
 3:   3 c M Three
 4:   4 d N   One
 5:   5 e O   Two
 6:   6 f P Three
 7:   7 g Q   One
 8:   8 h R   Two
 9:   9 i S Three
10:  10 j T   One
```

```
> class(theDF$B)
[1] "factor"
> class(theDT$B)
[1] "character"
> |
```

```
theDF = data.frame(A=1:10,
                   B=letters[1:10],
                   C=LETTERS[11:20],
                   D=rep(c("One", "Two", "Three"),
length.out=10))
theDT = data.table(A=1:10,
                   B=letters[1:10],
                   C=LETTERS[11:20],
                   D=rep(c("One", "Two", "Three"),
length.out=10))
theDF
theDT
class(theDF$B)
class(theDT$B)
```

R data.table

查詢方式兩者皆一樣。

```
diamondsDT = data.table(diamonds)
diamondsDT
theDT[1:2, ]
theDT[theDT$A >= 7, ]
```

```
> diamondsDT = data.table(diamonds)
> diamondsDT
       carat       cut color clarity depth table price    x    y    z
    1:  0.23     Ideal     E     SI2  61.5    55   326 3.95 3.98 2.43
    2:  0.21   Premium     E     SI1  59.8    61   326 3.89 3.84 2.31
    3:  0.23      Good     E     VS1  56.9    65   327 4.05 4.07 2.31
    4:  0.29   Premium     I     VS2  62.4    58   334 4.20 4.23 2.63
    5:  0.31      Good     J     SI2  63.3    58   335 4.34 4.35 2.75
   ---
53936:  0.72     Ideal     D     SI1  60.8    57  2757 5.75 5.76 3.50
53937:  0.72      Good     D     SI1  63.1    55  2757 5.69 5.75 3.61
53938:  0.70 Very Good     D     SI1  62.8    60  2757 5.66 5.68 3.56
53939:  0.86   Premium     H     SI2  61.0    58  2757 6.15 6.12 3.74
53940:  0.75     Ideal     D     SI2  62.2    55  2757 5.83 5.87 3.64
>
```

```
> theDT[1:2, ]
   A B C   D
1: 1 a K One
2: 2 b L Two
> theDT[theDT$A >= 7, ]
    A B C     D
1:  7 g Q   One
2:  8 h R   Two
3:  9 i S Three
4: 10 j T   One
>
```

R data.table

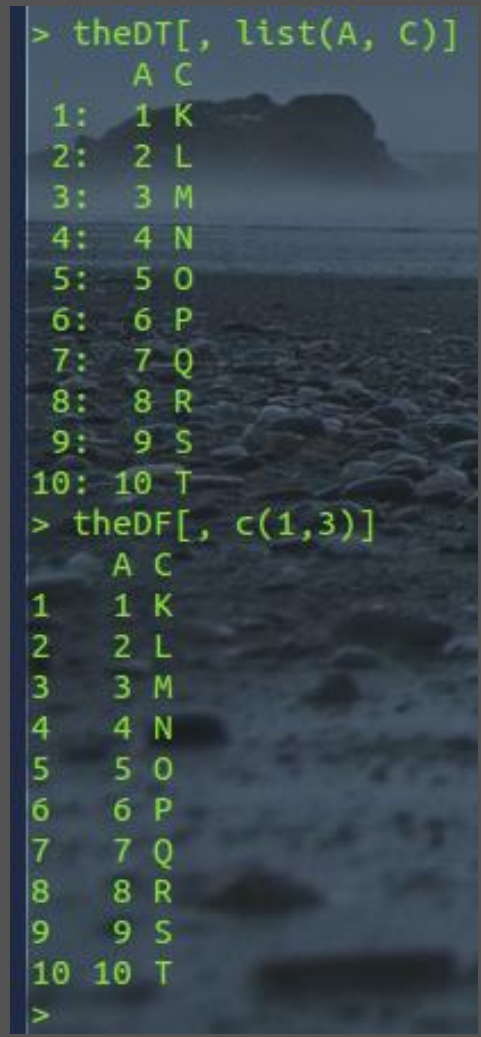
兩者的差異處在於，data.frame 與 data.table 兩者的差異在於的直行與法上的差異， data.table 使用的是 list() 。

theDT[, list(A, C)]
theDT[, B]
theDT[, list(B)]

```
> theDT[, list(A, C)]
     A C
 1:  1 K
 2:  2 L
 3:  3 M
 4:  4 N
 5:  5 O
 6:  6 P
 7:  7 Q
 8:  8 R
 9:  9 S
10: 10 T
>
> theDT[, B]
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"
>
> theDT[, list(B)]
    B
 1: a
 2: b
 3: c
 4: d
 5: e
 6: f
 7: g
 8: h
 9: i
10: j
> |
```

在此將 data.frame 與 data.table 相同的程式 打在一起對照。

theDT[, list(A, C)]
theDF[, c(1,3)]



R data.table

theDF[, 2]
theDT[, B]

```
> theDF[, 2]
 [1] a b c d e f g h i j
Levels: a b c d e f g h i j
> theDT[, B]
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"
>
```

theDF["B"]
theDT[, list(B)]

```
> theDF["B"]
   B
1  a
2  b
3  c
4  d
5  e
6  f
7  g
8  h
9  i
10 j
> theDT[, list(B)]
    B
 1: a
 2: b
 3: c
 4: d
 5: e
 6: f
 7: g
 8: h
 9: i
10: j
>
```

R data.table

theDT[, "B", with = FALSE]
theDT[, c("A", "C"), with = FALSE]

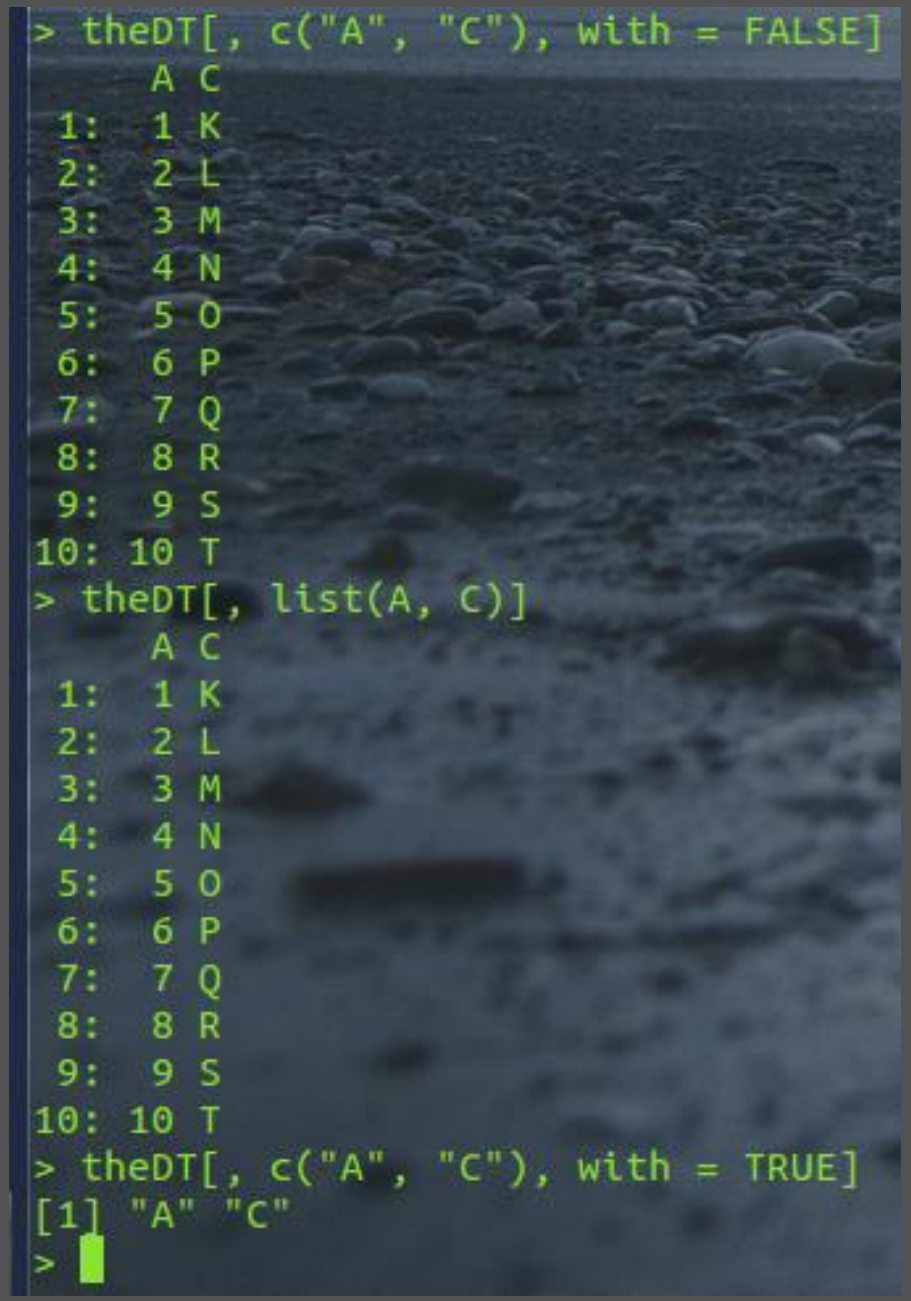
```
> theDT[, "B", with = FALSE]
    B
 1: a
 2: b
 3: c
 4: d
 5: e
 6: f
 7: g
 8: h
 9: i
10: j
>
> theDT[, c("A", "C"), with = FALSE]
     A C
 1:  1 K
 2:  2 L
 3:  3 M
 4:  4 N
 5:  5 O
 6:  6 P
 7:  7 Q
 8:  8 R
 9:  9 S
10: 10 T
>
```

theDT[, "B", with = FALSE]
theDT[, list(B)]
theDT[, "B", with = TRUE]

```
> theDT[, "B", with = FALSE]
    B
 1: a
 2: b
 3: c
 4: d
 5: e
 6: f
 7: g
 8: h
 9: i
10: j
> theDT[, list(B)]
    B
 1: a
 2: b
 3: c
 4: d
 5: e
 6: f
 7: g
 8: h
 9: i
10: j
> theDT[, "B", with = TRUE]
[1] "B"
>
```

R data.table

theDT[, c("A", "C"), with = FALSE]
theDT[, list(A, C)]
theDT[, c("A", "C"), with = TRUE]



R data.table - key

tables()
setkey(theDT, D)
theDT
key(theDT)
tables()

```
> tables()
     NAME          NROW NCOL MB COLS                                                  KEY
[1,] diamondsDT 53,940   10  4 carat,cut,color,clarity,depth,table,price,x,y,z
[2,] theDT          10    4  1 A,B,C,D
Total: 5MB
> |
```

利用 setkey() 函數，對 data.table 增加索引，被指定的欄位會進行排序。

setkey([指定的 datatable], [指定作為索引的欄位])

key() 可以檢查 data.table 的索引是哪個欄位 !!!當然 可以在 table() 當中觀察到 特定 data.table 的索引是那個欄位。

key([指定的 datatable])

```
> setkey(theDT, D)
>
> theDT
    A B C     D
 1:  1 a K   One
 2:  4 d N   One
 3:  7 g Q   One
 4: 10 j T   One
 5:  3 c M Three
 6:  6 f P Three
 7:  9 i S Three
 8:  2 b L   Two
 9:  5 e O   Two
10:  8 h R   Two
>
> key(theDT)
[1] "D"
>
```

```
> tables()
     NAME          NROW NCOL MB COLS                                                  KEY
[1,] diamondsDT 53,940   10  4 carat,cut,color,clarity,depth,table,price,x,y,z
[2,] theDT          10    4  1 A,B,C,D                                                D
Total: 5MB
> |
```

R data.table - key

theDT["One",]

theDT[c("One", "Two"),]

setkey(diamondsDT, cut, color)

```
> theDT["One", ]
     A B C   D
1:   1 a K One
2:   4 d N One
3:   7 g Q One
4:  10 j T One
> theDT[c("One", "Two"), ]
     A B C   D
1:   1 a K One
2:   4 d N One
3:   7 g Q One
4:  10 j T One
5:   2 b L Two
6:   5 e O Two
7:   8 h R Two
>
> setkey(diamondsDT, cut, color)
>
```

可以使用指定為 KEY 欄位 當中橫列的值對 data.table 進行篩選， 當然設定多個直行欄位為 KEY。

setkey([指定的 datatable], [指定作為索引的欄位 1], [指定作為索引的欄位 2])

R data.table - key

還可以使用 指定為 KEY 欄位當中多個橫列的 值 對 data.table 進行篩選

diamondsDT[J("Ideal", "E"),]
diamondsDT[J("Ideal", c("E", "D")),]

```
> diamondsDT[J("Ideal", "E"), ]
      carat   cut color clarity depth table price    x    y    z
   1:  0.23 Ideal     E     SI2  61.5    55   326 3.95 3.98 2.43
   2:  0.26 Ideal     E    VVS2  62.9    58   554 4.02 4.06 2.54
   3:  0.70 Ideal     E     SI1  62.5    57  2757 5.70 5.72 3.57
   4:  0.59 Ideal     E    VVS2  62.0    55  2761 5.38 5.43 3.35
   5:  0.74 Ideal     E     SI2  62.2    56  2761 5.80 5.84 3.62
  ---
3899:  0.70 Ideal     E     SI1  61.7    55  2745 5.71 5.74 3.53
3900:  0.51 Ideal     E    VVS1  61.9    54  2745 5.17 5.11 3.18
3901:  0.56 Ideal     E    VVS1  62.1    56  2750 5.28 5.29 3.28
3902:  0.77 Ideal     E     SI2  62.1    56  2753 5.84 5.86 3.63
3903:  0.71 Ideal     E     SI1  61.9    56  2756 5.71 5.73 3.54
> diamondsDT[J("Ideal", c("E", "D")), ]
      carat   cut color clarity depth table price    x    y    z
   1:  0.23 Ideal     E     SI2  61.5    55   326 3.95 3.98 2.43
   2:  0.26 Ideal     E    VVS2  62.9    58   554 4.02 4.06 2.54
   3:  0.70 Ideal     E     SI1  62.5    57  2757 5.70 5.72 3.57
   4:  0.59 Ideal     E    VVS2  62.0    55  2761 5.38 5.43 3.35
   5:  0.74 Ideal     E     SI2  62.2    56  2761 5.80 5.84 3.62
  ---
6733:  0.51 Ideal     D    VVS2  61.7    56  2742 5.16 5.14 3.18
6734:  0.51 Ideal     D    VVS2  61.3    57  2742 5.17 5.14 3.16
6735:  0.81 Ideal     D     SI1  61.5    57  2748 6.00 6.03 3.70
6736:  0.72 Ideal     D     SI1  60.8    57  2757 5.75 5.76 3.50
6737:  0.75 Ideal     D     SI2  62.2    55  2757 5.83 5.87 3.64
> |
```

R data.table - key

在前面我們用 aggregate() 可以進行分群，在此 data.table 也能做的到此功能。

aggregate()，利用 diamonds 資料表，根據 cut 欄位 對 price 欄位進行分析

aggregate(price ~ cut, diamonds, mean)

```
> aggregate(price ~ cut, diamonds, mean)
        cut    price
1      Fair 4358.758
2      Good 3928.864
3 Very Good 3981.760
4   Premium 4584.258
5     Ideal 3457.542
> 
```

R data.table - key

利用轉成 data.table 的 diamonds 資料表，根據 cut 欄位 對 price 欄位進行分析，但未對結果欄位命名。

diamondsDT[, mean(price), by = cut]

```
> diamondsDT[, mean(price), by = cut]
          cut       V1
1:       Fair 4358.758
2:       Good 3928.864
3: Very Good 3981.760
4:    Premium 4584.258
5:      Ideal 3457.542
> |
```

利用轉成 data.table 的 diamonds 資料表，根據 cut 欄位 對 price 欄位進行分析，並對結果欄位命名。

diamondsDT[, list(price = mean(price)), by = cut]

```
> diamondsDT[, list(price = mean(price)), by = cut]
          cut    price
1:       Fair 4358.758
2:       Good 3928.864
3: Very Good 3981.760
4:    Premium 4584.258
5:      Ideal 3457.542
> |
```

R data.table - key

前面三行程式其效果皆為一樣 !!!若想對好幾行直行欄位進行分群可以用 list() 指定。

diamondsDT[, mean(price), by = list(cut, color)]

```
> diamondsDT[, mean(price), by = list(cut, color)]
          cut color       V1
 1:       Fair     D 4291.061
 2:       Fair     E 3682.312
 3:       Fair     F 3827.003
 4:       Fair     G 4239.255
 5:       Fair     H 5135.683
 6:       Fair     I 4685.446
 7:       Fair     J 4975.655
 8:       Good     D 3405.382
 9:       Good     E 3423.644
10:       Good     F 3495.750
11:       Good     G 4123.482
12:       Good     H 4276.255
13:       Good     I 5078.533
14:       Good     J 4574.173
15: Very Good     D 3470.467
16: Very Good     E 3214.652
17: Very Good     F 3778.820
18: Very Good     G 3872.754
19: Very Good     H 4535.390
20: Very Good     I 5255.880
21: Very Good     J 5103.513
22:   Premium     D 3631.293
23:   Premium     E 3538.914
24:   Premium     F 4324.890
25:   Premium     G 4500.742
26:   Premium     H 5216.707
27:   Premium     I 5946.181
28:   Premium     J 6294.592
29:     Ideal     D 2629.095
30:     Ideal     E 2597.550
31:     Ideal     F 3374.939
32:     Ideal     G 3720.706
33:     Ideal     H 3889.335
34:     Ideal     I 4451.970
35:     Ideal     J 4918.186
          cut color       V1
```

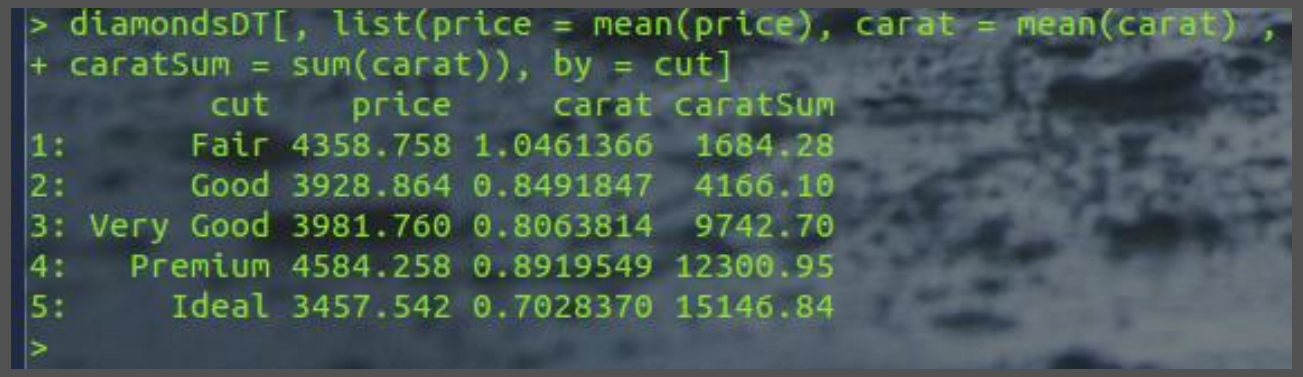
R data.table - key

對好幾行直行欄位進行 "個別" 的分群計算也可以做得到 !!!其不同的直行欄位可做出不同的計算，甚至計算出的結果做成額外的直行塞回去。

diamondsDT[, list(price = mean(price), carat = mean(carat)), by = cut]

diamondsDT[, list(price = mean(price), carat = mean(carat) ,
caratSum = sum(carat)), by = cut]

```
> diamondsDT[, list(price = mean(price), carat = mean(carat)), by = cut]
         cut    price     carat
1:       Fair 4358.758 1.0461366
2:       Good 3928.864 0.8491847
3: Very Good 3981.760 0.8063814
4:    Premium 4584.258 0.8919549
5:      Ideal 3457.542 0.7028370
>
```



R data.table - key

當然可以對直行欄位進行 "個別" 的分群，並對好幾個直行進行不同的運算

diamondsDT[, list(price = mean(price), carat = mean(carat)),
by = list(cut, color)]

```
> diamondsDT[, list(price = mean(price), carat = mean(carat)), by = list(cut, color)]
          cut color    price     carat
 1:      Fair     D 4291.061 0.9201227
 2:      Fair     E 3682.312 0.8566071
 3:      Fair     F 3827.003 0.9047115
 4:      Fair     G 4239.255 1.0238217
 5:      Fair     H 5135.683 1.2191749
 6:      Fair     I 4685.446 1.1980571
 7:      Fair     J 4975.655 1.3411765
 8:      Good     D 3405.382 0.7445166
```

R 資料整理

1. cbind 和 rbind 資料合併

2. data.frame 合併 的資料連結

3. reshape2 套件置換行列資料

R cbind & rbind

cbind()、rbind() 兩個函數在使用上，都必須擁有相同的直行、同樣的數量與名稱

在此建立三個向量 -> sport、league、trophy

此三個向量利用 cbind() 函數 合併在一起，令為 trophies1 ，利用 class() 函數進行觀察其物件為 "矩陣"(Matrix)。

cbind([向量 1], [向量 2], [向量 3],....)

**cbind() 函數 可以對其合併向量後的直行欄位指派名稱

cbind([指派名稱 1] = [向量 1], [指派名稱 2] = [向量 2], [指派名稱 3] = [向量 3], ...)

直接建立名為 trophies2 的 data.frame() ，包含 三個 "向量" (vector)

最後利用 rbind() 整合成一個 data.frame

rbind([物件 1], [物件 2], ...)

R cbind & rbind

利用 cbind() 合併 三個向量後的結果

```
> sport
[1] "Hockey"   "Baseball" "Football"
> league
[1] "NHL" "MLB" "NFL"
> trophy
[1] "Stanley Cup"           "Commissioner's Trophy" "Vince Lombardi Trophy"
> trophies1
     sport      league trophy
[1,] "Hockey"   "NHL"  "Stanley Cup"
[2,] "Baseball" "MLB"  "Commissioner's Trophy"
[3,] "Football" "NFL"  "Vince Lombardi Trophy"
>
```

trophies2 資料框 的結果

```
> trophies2
       sport league                                trophy
1 Basketball    NBA Larry O'Brien Championship Trophy
2       Golf    PGA                       Wanamaker Trophy
>
```

利用 class() 函數， 觀察 合併後的物件屬性

```
> class(trophies1)
[1] "matrix"
> class(trophies2)
[1] "data.frame"
> class(trophies)
[1] "data.frame"
>
```

R cbind & rbind

```
> sport = c("Hockey", "Baseball", "Football")
> league = c("NHL", "MLB", "NFL")
> trophy = c("Stanley Cup", "Commissioner's Trophy",
+             "Vince Lombardi Trophy")
> trophies1 = cbind(sport, league, trophy)
>
>
> trophies2 = data.frame(sport=c("Basketball", "Golf"),
+ league=c("NBA", "PGA"),
+ trophy=c("Larry O'Brien Championship Trophy", "Wanamaker Trophy"),
+ stringsAsFactors=FALSE)
>
> trophies = rbind(trophies1, trophies2)
>
> cbind(Sport = sport, Association = league, Prize = trophy)
      Sport      Association Prize
[1,] "Hockey"   "NHL"       "Stanley Cup"
[2,] "Baseball" "MLB"       "Commissioner's Trophy"
[3,] "Football" "NFL"       "Vince Lombardi Trophy"
```

```
sport = c("Hockey", "Baseball", "Football")
league = c("NHL", "MLB", "NFL")
trophy = c("Stanley Cup", "Commissioner's Trophy",
        "Vince Lombardi Trophy")
trophies1 = cbind(sport, league, trophy)
trophies2 = data.frame(sport = c("Basketball", "Golf"),
                       league = c("NBA", "PGA"),
                       trophy = c("Larry O'Brien Championship Trophy", "Wanamaker Trophy"),
                       stringsAsFactors = FALSE)
trophies = rbind(trophies1, trophies2)
cbind(Sport = sport, Association = league, Prize = trophy)
```

R data.frame 合併 的資料連結

在此利用美國 國際開發署 (USAID)的開放政府倡議，所提供的資料
網址 : http://jaredlander.com/data/US_Foreign_Aid.zip

download.file() -> 下載
unzip() -> 解壓縮 (.zip) 檔案

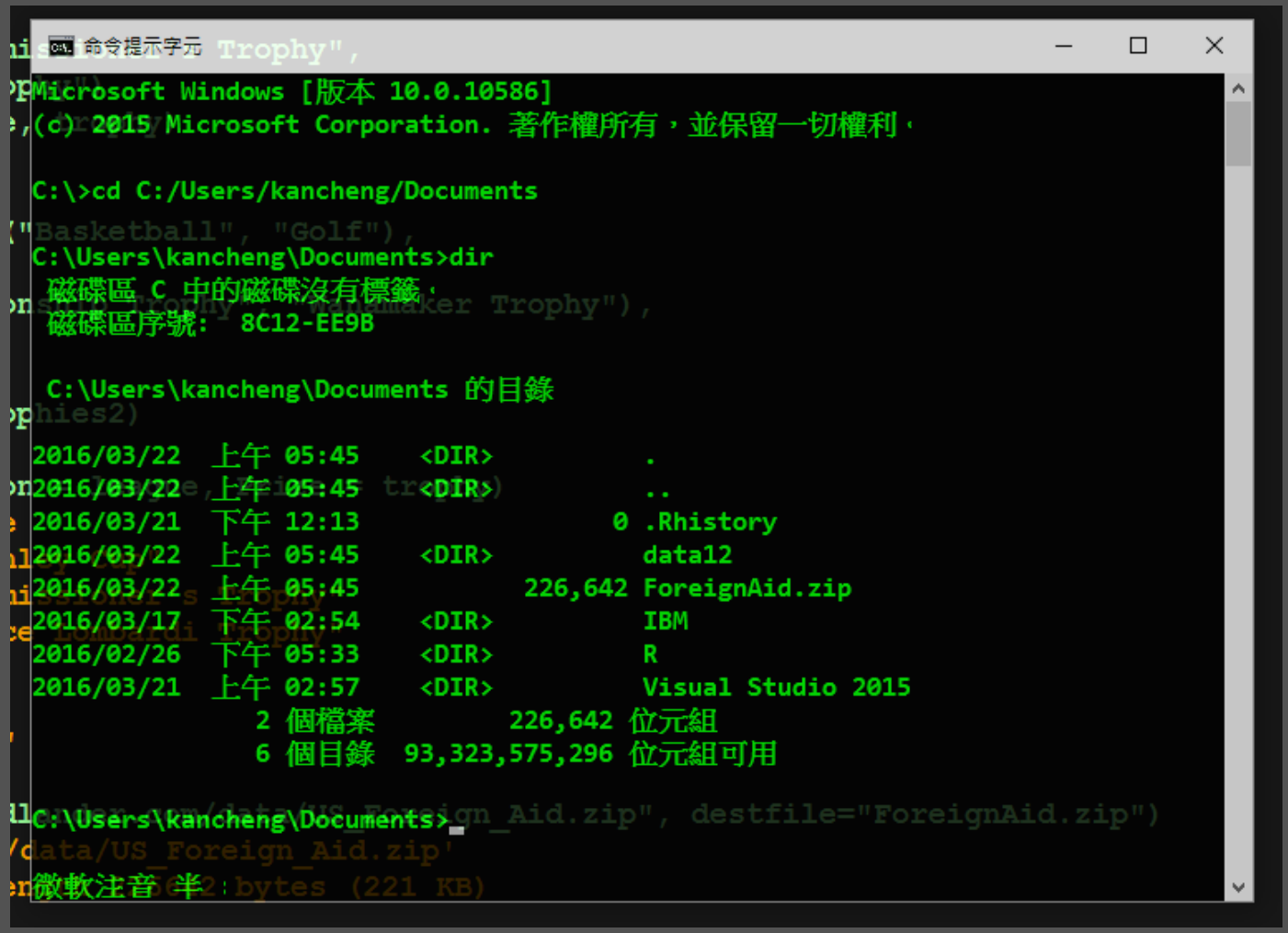
download.file(url = "[下載網址]", destfile = "[指定的檔名]")
unzip("[指定解壓縮的檔名].zip", exdir = "[指定的目錄名稱]")

```
> download.file(url="http://jaredlander.com/data/US_Foreign_Aid.zip", destfile="ForeignAid.zip")
嘗試 URL 'http://jaredlander.com/data/US_Foreign_Aid.zip'
Content type 'application/zip' length 226642 bytes (221 KB)
downloaded 221 KB

> unzip("ForeignAid.zip", exdir="data12")
>
```

```
> getwd()
[1] "C:/Users/kancheng/Documents"
>
```

R data.frame 合併 的資料連結

```
download.file( url = "https://github.com/kancheng/easyR/raw/master/C03/US_Foreign_Aid.zip",
               destfile = "ForeignAid.zip")

unzip( "ForeignAid.zip", exdir = “data12")
```



R 匯入資料

```
require(stringr)
theFiles = dir("data12/", pattern="\\.csv")
for(a in theFiles){
nameToUse = str_sub(string=a, start=12, end=18)
temp = read.table(file=file.path("data12", a), header=TRUE, sep=",", stringsAsFactors=FALSE)
assign(x=nameToUse, value=temp)
}
```

```
> require(stringr)
Loading required package: stringr
>
> theFiles = dir("data12/", pattern="\\.csv")
>
>
> for(a in theFiles)
+ {
+    nameToUse = str_sub(string=a, start=12, end=18)
+
+    temp = read.table(file=file.path("data12", a), header=TRUE, sep=",", stringsAsFactors=FALSE)
+
+    assign(x=nameToUse, value=temp)
+ }
>
```

str_sub() -> 建立起每個資料的名稱

read.table() -> 匯入資料夾和檔名

file.path() -> 指定資料夾和檔名

assign() -> 指派至工作空間

R merge

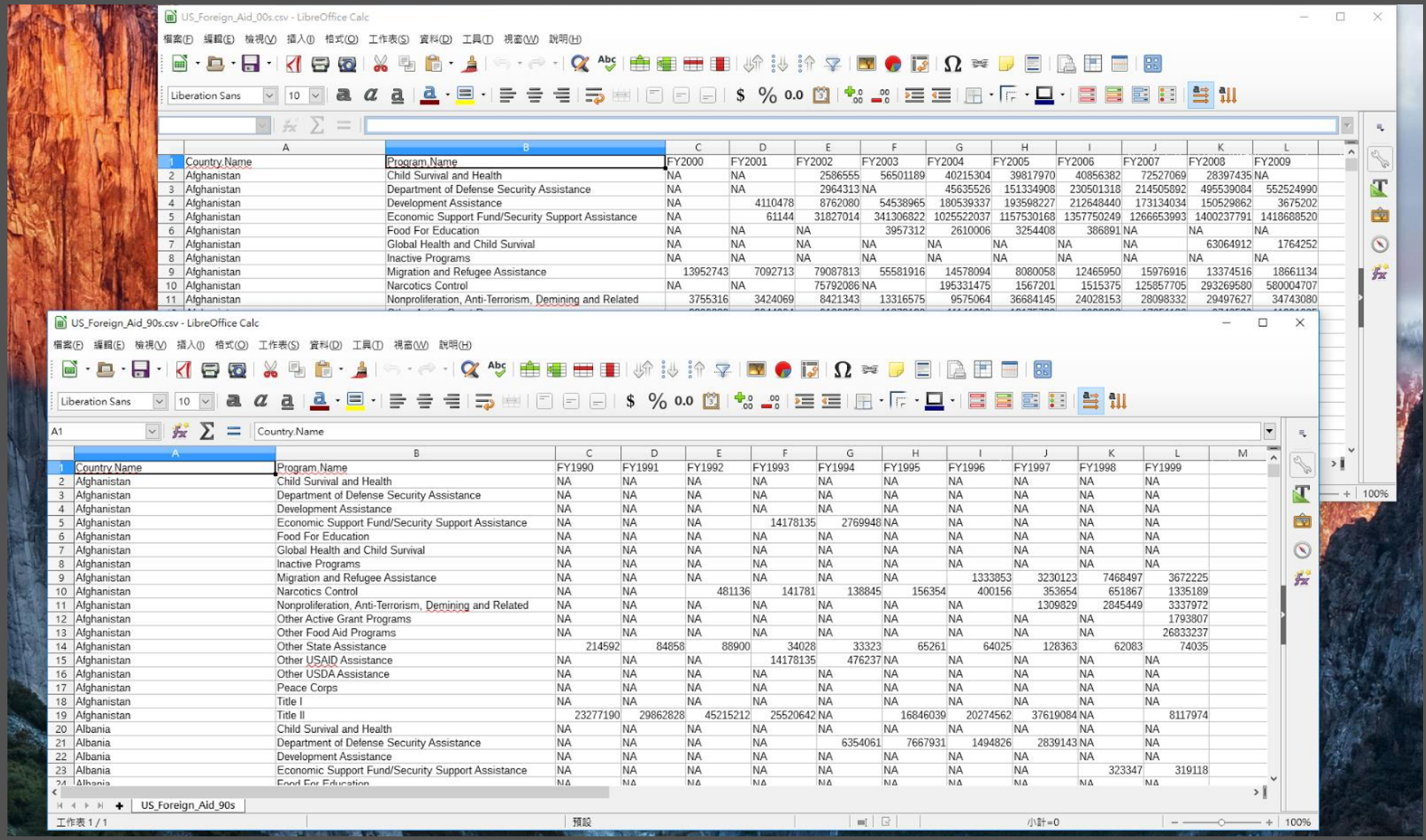
merge() 為 R 的 內建函數，書中是利用此來合併兩個 "資料框"，並利用前面下載的資料 !!!

merge([資料 1],[資料 2],
by.x = c("[關鍵詞直行欄位1]", "[關鍵詞直行欄位2]")
by.y = c("[關鍵詞直行欄位1]", "[關鍵詞直行欄位2]")
)

by.x -> 指 左邊 data.frame
by.y -> 指 右邊 data.frame

```
> Aid90s00s = merge(x=Aid_90s, y=Aid_00s,
+                   by.x=c("Country.Name", "Program.Name"),
+                   by.y=c("Country.Name", "Program.Name"))
>
> head(Aid90s00s)
  Country.Name                                Program.Name FY1990 FY1991 FY1992   FY1993  FY1994 FY1995 FY1996 FY1997 FY1998 FY1999 FY2000  FY2001   FY2002
1  Afghanistan                      Child Survival and Health     NA     NA     NA       NA      NA     NA     NA     NA     NA     NA     NA      NA  2586555
2  Afghanistan        Department of Defense Security Assistance     NA     NA     NA       NA      NA     NA     NA     NA     NA     NA     NA      NA  2964313
3  Afghanistan                          Development Assistance     NA     NA     NA       NA      NA     NA     NA     NA     NA     NA     NA 4110478  8762080
4  Afghanistan Economic Support Fund/Security Support Assistance     NA     NA     NA 14178135 2769948     NA     NA     NA     NA     NA     NA   61144 31827014
5  Afghanistan                               Food For Education     NA     NA     NA       NA      NA     NA     NA     NA     NA     NA     NA      NA       NA
6  Afghanistan                    Global Health and Child Survival     NA     NA     NA       NA      NA     NA     NA     NA     NA     NA     NA      NA       NA
     FY2003     FY2004     FY2005     FY2006     FY2007     FY2008     FY2009
1  56501189   40215304   39817970   40856382   72527069   28397435         NA
2        NA   45635526  151334908  230501318  214505892  495539084  552524990
3  54538965  180539337  193598227  212648440  173134034  150529862    3675202
4 341306822 1025522037 1157530168 1357750249 1266653993 1400237791 1418688520
5   3957312    2610006    3254408     386891         NA         NA         NA
6        NA         NA         NA         NA         NA   63064912    1764252
> |
```

R merge




```
Aid90s00s = merge(x=Aid_90s, y=Aid_00s,
      by.x=c("Country.Name", "Program.Name"),
      by.y=c("Country.Name", "Program.Name"))
head(Aid90s00s)
```

R plyr 套件 join() 合併 data.frame

join(x = [左邊表], y = [右邊表], by = c("[指定欄位 1]", "[指定欄位 2]"))

```
> require(plyr)
Loading required package: plyr
> Aid90s00sJoin = join(x = Aid_90s, y = Aid_00s, by = c("Country.Name", "Program.Name"))
> head(Aid90s00sJoin)
  Country.Name                                Program.Name FY1990 FY1991 FY1992   FY1993  FY1994 FY1995 FY1996 FY1997 FY1998 FY1999 FY2000  FY2001   FY2002
1  Afghanistan                      Child Survival and Health     NA     NA     NA       NA      NA     NA     NA     NA     NA     NA     NA      NA  2586555
2  Afghanistan        Department of Defense Security Assistance     NA     NA     NA       NA      NA     NA     NA     NA     NA     NA     NA      NA  2964313
3  Afghanistan                          Development Assistance     NA     NA     NA       NA      NA     NA     NA     NA     NA     NA     NA 4110478  8762080
4  Afghanistan Economic Support Fund/Security Support Assistance     NA     NA     NA 14178135 2769948     NA     NA     NA     NA     NA     NA   61144 31827014
5  Afghanistan                               Food For Education     NA     NA     NA       NA      NA     NA     NA     NA     NA     NA     NA      NA       NA
6  Afghanistan                    Global Health and Child Survival     NA     NA     NA       NA      NA     NA     NA     NA     NA     NA     NA      NA       NA
     FY2003     FY2004     FY2005     FY2006     FY2007     FY2008     FY2009
1  56501189   40215304   39817970   40856382   72527069   28397435         NA
2        NA   45635526  151334908  230501318  214505892  495539084  552524990
3  54538965  180539337  193598227  212648440  173134034  150529862    3675202
4 341306822 1025522037 1157530168 1357750249 1266653993 1400237791 1418688520
5   3957312    2610006    3254408     386891         NA         NA         NA
6        NA         NA         NA         NA         NA   63064912    1764252
> |
```

```
require(plyr)
Aid90s00sJoin = join(x = Aid_90s, y = Aid_00s, by = c("Country.Name", "Program.Name"))
head(Aid90s00sJoin)
```

R plyr 套件 join() 合併 data.frame

在此利用 str_sub() 函數 抓出所有資料框的名稱，而後建立一個
空 list 並利用 for 迴圈將每個 資料況塞入 list。

```
frameNames = str_sub(string = theFiles, start = 12, end = 18)
frameList = vector("list", length(frameNames))
names(frameList) = frameNames
for (a in frameNames)
{
  frameList[[a]] = eval(parse(text = a))
}
```

```
> frameNames = str_sub(string = theFiles, start = 12, end = 18)
>
>
> frameList = vector("list", length(frameNames))
> names(frameList) = frameNames
>
>
> for (a in frameNames)
+ {
+     frameList[[a]] <- eval(parse(text = a))
+ }
>
>
```

R plyr 套件 join() 合併 data.frame

frameNames 物件是用 str_sub() 函數 抓出所有資料框的名稱 。

```
> frameList = vector("list", length(frameNames))
> class(frameList)
[1] "list"
> frameList
[[1]]
NULL

[[2]]
NULL

[[3]]
NULL

[[4]]
NULL

[[5]]
NULL

[[6]]
NULL

[[7]]
NULL

[[8]]
NULL
```

用 head(, n = 2) or head(, n = 3)。

```
> frameNames = str_sub(string = theFiles, start = 12, end = 18)
> class(frameNames)
[1] "character"
> frameNames
[1] "Aid_00s" "Aid_10s" "Aid_40s" "Aid_50s" "Aid_60s" "Aid_70s" "Aid_80s"
[8] "Aid_90s"
>
```

R plyr 套件 join() 合併 data.frame

frameList 物件是用 vector() 根據 frameNames 物件 內容長度來建立起空的 list 位置

```
> names(frameList) = frameNames
> class(frameNames)
[1] "character"
> frameNames
[1] "Aid_00s" "Aid_10s" "Aid_40s" "Aid_50s" "Aid_60s" "Aid_70s" "Aid_80s"
[8] "Aid_90s"
> frameList
$Aid_00s
NULL

$Aid_10s
NULL

$Aid_40s
NULL

$Aid_50s
NULL

$Aid_60s
NULL

$Aid_70s
NULL

$Aid_80s
NULL

$Aid_90s
NULL

>
```

R plyr 套件 join() 合併 data.frame

names(frameList) = frameNames，這段是利用 names() 將 frameNames 物件 所存的所有資料框 名稱，放進 frameList 物件 作為 List 的名稱

```
> for (a in frameNames)
+ {
+     frameList[[a]] = eval(parse(text = a))
+ }
```

```
> head(frameList[[1]], n = 3)
  Country.Name                                  Program.Name FY2000  FY2001  FY2002
1  Afghanistan                    Child Survival and Health     NA      NA 2586555
2  Afghanistan Department of Defense Security Assistance     NA      NA 2964313
3  Afghanistan                       Development Assistance     NA 4110478 8762080
    FY2003    FY2004    FY2005    FY2006    FY2007    FY2008    FY2009
1 56501189  40215304  39817970  40856382  72527069  28397435        NA
2       NA  45635526 151334908 230501318 214505892 495539084 552524990
3 54538965 180539337 193598227 212648440 173134034 150529862   3675202
> head(frameList[[2]], n = 3)
  Country.Name                                  Program.Name    FY2010
1  Afghanistan                    Child Survival and Health        NA
2  Afghanistan Department of Defense Security Assistance 316514796
3  Afghanistan                       Development Assistance        NA
> head(frameList[[3]], n = 3)
  Country.Name                                  Program.Name FY1946 FY1947 FY1948
1  Afghanistan                    Child Survival and Health     NA      NA      NA
2  Afghanistan Department of Defense Security Assistance     NA      NA      NA
3  Afghanistan                       Development Assistance     NA      NA      NA
  FY1949
1     NA
2     NA
3     NA
> head(frameList[[4]], n = 3)
  Country.Name                                  Program.Name FY1950 FY1951 FY1952
1  Afghanistan                    Child Survival and Health     NA      NA      NA
2  Afghanistan Department of Defense Security Assistance     NA      NA      NA
3  Afghanistan                       Development Assistance     NA      NA      NA
  FY1953 FY1954 FY1955 FY1956 FY1957 FY1958 FY1959
1     NA     NA     NA     NA     NA     NA     NA
2     NA     NA     NA     NA     NA     NA     NA
3     NA     NA     NA     NA     NA     NA     NA
> head(frameList[[5]], n = 3)
  Country.Name                                  Program.Name FY1960 FY1961 FY1962
1  Afghanistan                    Child Survival and Health     NA      NA      NA
2  Afghanistan Department of Defense Security Assistance     NA      NA      NA
3  Afghanistan                       Development Assistance     NA      NA      NA
  FY1963 FY1964 FY1965 FY1966 FY1967 FY1968 FY1969
1     NA     NA     NA     NA     NA     NA     NA
2     NA     NA     NA     NA     NA     NA     NA
3     NA     NA     NA     NA     NA     NA     NA
>
```

R plyr 套件 join() 合併 data.frame

這邊檢視剛剛匯入到 frameList 物件中，每個資料的狀況..，可以用 List 的編號，也可以用 List 的名稱 !!!

```
head(frameList[[1]])
head(frameList[["Aid_00s"]])
head(frameList[[5]])
head(frameList[["Aid_60s"]])
```

```
> head(frameList[[1]])
  Country.Name                                Program.Name FY2000  FY2001   FY2002   FY2003   FY2004   FY2005   FY2006   FY2007   FY2008   FY2009
1  Afghanistan                      Child Survival and Health     NA      NA  2586555 56501189 40215304 39817970 40856382 72527069 28397435       NA
2  Afghanistan         Department of Defense Security Assistance     NA      NA  2964313       NA 45635526 151334908 230501318 214505892 495539084 552524990
3  Afghanistan                           Development Assistance     NA 4110478  8762080 54538965 180539337 193598227 212648440 173134034 150529862  3675202
4  Afghanistan Economic Support Fund/Security Support Assistance     NA   61144 31827014 341306822 1025522037 1157530168 1357750249 1266653993 1400237791 1418688520
5  Afghanistan                             Food For Education     NA      NA       NA  3957312  2610006  3254408   386891       NA       NA       NA
6  Afghanistan                  Global Health and Child Survival     NA      NA       NA       NA       NA       NA       NA       NA 63064912  1764252
> head(frameList[["Aid_00s"]])
  Country.Name                                Program.Name FY2000  FY2001   FY2002   FY2003   FY2004   FY2005   FY2006   FY2007   FY2008   FY2009
1  Afghanistan                      Child Survival and Health     NA      NA  2586555 56501189 40215304 39817970 40856382 72527069 28397435       NA
2  Afghanistan         Department of Defense Security Assistance     NA      NA  2964313       NA 45635526 151334908 230501318 214505892 495539084 552524990
3  Afghanistan                           Development Assistance     NA 4110478  8762080 54538965 180539337 193598227 212648440 173134034 150529862  3675202
4  Afghanistan Economic Support Fund/Security Support Assistance     NA   61144 31827014 341306822 1025522037 1157530168 1357750249 1266653993 1400237791 1418688520
5  Afghanistan                             Food For Education     NA      NA       NA  3957312  2610006  3254408   386891       NA       NA       NA
6  Afghanistan                  Global Health and Child Survival     NA      NA       NA       NA       NA       NA       NA       NA 63064912  1764252
> head(frameList[[5]])
  Country.Name                                Program.Name FY1960 FY1961    FY1962 FY1963 FY1964 FY1965 FY1966 FY1967 FY1968 FY1969
1  Afghanistan                      Child Survival and Health     NA     NA       NA     NA     NA     NA     NA     NA     NA     NA
2  Afghanistan         Department of Defense Security Assistance     NA     NA       NA     NA     NA     NA     NA     NA     NA     NA
3  Afghanistan                           Development Assistance     NA     NA       NA     NA     NA     NA     NA     NA     NA     NA
4  Afghanistan Economic Support Fund/Security Support Assistance     NA     NA 181177853     NA     NA     NA     NA     NA     NA     NA
5  Afghanistan                             Food For Education     NA     NA       NA     NA     NA     NA     NA     NA     NA     NA
6  Afghanistan                  Global Health and Child Survival     NA     NA       NA     NA     NA     NA     NA     NA     NA     NA
> head(frameList[["Aid_60s"]])
  Country.Name                                Program.Name FY1960 FY1961    FY1962 FY1963 FY1964 FY1965 FY1966 FY1967 FY1968 FY1969
1  Afghanistan                      Child Survival and Health     NA     NA       NA     NA     NA     NA     NA     NA     NA     NA
2  Afghanistan         Department of Defense Security Assistance     NA     NA       NA     NA     NA     NA     NA     NA     NA     NA
3  Afghanistan                           Development Assistance     NA     NA       NA     NA     NA     NA     NA     NA     NA     NA
4  Afghanistan Economic Support Fund/Security Support Assistance     NA     NA 181177853     NA     NA     NA     NA     NA     NA     NA
5  Afghanistan                             Food For Education     NA     NA       NA     NA     NA     NA     NA     NA     NA     NA
6  Afghanistan                  Global Health and Child Survival     NA     NA       NA     NA     NA     NA     NA     NA     NA     NA
```

R plyr 套件 join() 合併 data.frame

當然如果資料筆數太多且不易觀察，而目標只是想觀察每個資料的欄位，建議用 head() 函數時後面加行號。

```
> head(frameList[[1]], n = 2)
  Country.Name                                  Program.Name FY2000 FY2001  FY2002
1  Afghanistan                        Child Survival and Health     NA     NA 2586555
2  Afghanistan Department of Defense Security Assistance     NA     NA 2964313
    FY2003   FY2004    FY2005    FY2006    FY2007    FY2008    FY2009
1 56501189 40215304  39817970  40856382  72527069  28397435        NA
2       NA 45635526 151334908 230501318 214505892 495539084 552524990
> head(frameList[["Aid_00s"]], n = 2)
  Country.Name                                  Program.Name FY2000 FY2001  FY2002
1  Afghanistan                        Child Survival and Health     NA     NA 2586555
2  Afghanistan Department of Defense Security Assistance     NA     NA 2964313
    FY2003   FY2004    FY2005    FY2006    FY2007    FY2008    FY2009
1 56501189 40215304  39817970  40856382  72527069  28397435        NA
2       NA 45635526 151334908 230501318 214505892 495539084 552524990
> head(frameList[[5]], n = 2)
  Country.Name                                  Program.Name FY1960 FY1961 FY1962
1  Afghanistan                        Child Survival and Health     NA     NA     NA
2  Afghanistan Department of Defense Security Assistance     NA     NA     NA
  FY1963 FY1964 FY1965 FY1966 FY1967 FY1968 FY1969
1     NA     NA     NA     NA     NA     NA     NA
2     NA     NA     NA     NA     NA     NA     NA
> head(frameList[["Aid_60s"]], n = 2)
  Country.Name                                  Program.Name FY1960 FY1961 FY1962
1  Afghanistan                        Child Survival and Health     NA     NA     NA
2  Afghanistan Department of Defense Security Assistance     NA     NA     NA
  FY1963 FY1964 FY1965 FY1966 FY1967 FY1968 FY1969
1     NA     NA     NA     NA     NA     NA     NA
2     NA     NA     NA     NA     NA     NA     NA
>
```

R plyr 套件 join() 合併 data.frame


```
allAid = Reduce(function(...)
{
join(..., by = c("Country.Name", "Program.Name"))
}, frameList)
dim(allAid)
require(useful)
corner(allAid, c = 15)
bottomleft(allAid, c = 15)
```

```
> allAid = Reduce(function(...)
+ {
+    join(..., by = c("Country.Name", "Program.Name"))
+ }, frameList)
> dim(allAid)
[1] 2453   67
>
> require(useful)
Loading required package: useful
Loading required package: ggplot2
> corner(allAid, c = 15)
  Country.Name                                Program.Name FY2000  FY2001   FY2002    FY2003     FY2004     FY2005     FY2006     FY2007     FY2008     FY2009
1  Afghanistan                      Child Survival and Health     NA      NA  2586555  56501189   40215304   39817970   40856382   72527069   28397435         NA
2  Afghanistan        Department of Defense Security Assistance     NA      NA  2964313        NA   45635526  151334908  230501318  214505892  495539084  552524990
3  Afghanistan                           Development Assistance     NA 4110478  8762080  54538965  180539337  193598227  212648440  173134034  150529862    3675202
4  Afghanistan Economic Support Fund/Security Support Assistance     NA   61144 31827014 341306822 1025522037 1157530168 1357750249 1266653993 1400237791 1418688520
5  Afghanistan                                 Food For Education     NA      NA       NA   3957312    2610006    3254408     386891         NA         NA         NA
      FY2010 FY1946 FY1947
1         NA     NA     NA
2  316514796     NA     NA
3         NA     NA     NA
4 2797488331     NA     NA
5         NA     NA     NA
> bottomleft(allAid, c = 15)
     Country.Name           Program.Name  FY2000  FY2001   FY2002   FY2003   FY2004   FY2005  FY2006    FY2007    FY2008    FY2009   FY2010 FY1946 FY1947
2449     Zimbabwe Other State Assistance 1341952  322842       NA       NA   318655    44553  883546   1164632   2455592   2193057  1605765     NA     NA
2450     Zimbabwe Other USAID Assistance 3033599 8464897  6624408 11580999 12805688 10091759 4567577  10627613  11466426  41940500 30011970     NA     NA
2451     Zimbabwe            Peace Corps 2140530 1150732   407834       NA       NA       NA      NA        NA        NA        NA       NA     NA     NA
2452     Zimbabwe                Title I      NA      NA       NA       NA       NA       NA      NA        NA        NA        NA       NA     NA     NA
2453     Zimbabwe               Title II      NA      NA 31019776       NA       NA       NA  277468 100053600 180000717 174572685 79545100     NA     NA
```

R plyr 套件 join()
合併 data.frame

head() 顯示，我們可以看到所有表皆合併在一起

```
> head(allAid, n = 2)
  Country.Name                          Program.Name FY2000 FY2001  FY2002   FY2003   FY2004    FY2005    FY2006    FY2007    FY2008    FY2009    FY2010 FY1946 FY1947
1  Afghanistan                Child Survival and Health     NA     NA 2586555 56501189 40215304  39817970  40856382  72527069  28397435        NA        NA     NA     NA
2  Afghanistan Department of Defense Security Assistance     NA     NA 2964313       NA 45635526 151334908 230501318 214505892 495539084 552524990 316514796     NA     NA
  FY1948 FY1949 FY1950 FY1951 FY1952 FY1953 FY1954 FY1955 FY1956 FY1957 FY1958 FY1959 FY1960 FY1961 FY1962 FY1963 FY1964 FY1965 FY1966 FY1967 FY1968 FY1969 FY1970 FY1971
1     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
2     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
  FY1972 FY1973 FY1974 FY1975 FY1976 FY1977 FY1978 FY1979 FY1980 FY1981 FY1982 FY1983 FY1984 FY1985 FY1986 FY1987 FY1988 FY1989 FY1990 FY1991 FY1992 FY1993 FY1994 FY1995
1     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
2     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
  FY1996 FY1997 FY1998 FY1999
1     NA     NA     NA     NA
2     NA     NA     NA     NA
> dim(allAid)
[1] 2453   67
> |
```

R data.table 中的資料合併

這裡是利用將資料框轉成 data.table，並利用索引的方式進行合併

dt90 -> 左邊
dt00 -> 右邊
dt90[dt00] 用 左連結的方式並指向 dt0090

```
require(data.table)
dt90 = data.table(Aid_90s, key = c("Country.Name", "Program.Name"))
dt00 = data.table(Aid_00s, key = c("Country.Name", "Program.Name"))
dt0090 = dt90[dt00]
```

```
> require(data.table)
Loading required package: data.table
data.table 1.9.6  For help type ?data.table or https://github.com/Rdata
The fastest way to learn (by data.table authors): https://www.datacamp.
> dt90 = data.table(Aid_90s, key = c("Country.Name", "Program.Name"))
> dt00 = data.table(Aid_00s, key = c("Country.Name", "Program.Name"))
>
> dt0090 = dt90[dt00]
>
```

R data.table 中的資料合併


```
> class(dt90)
[1] "data.table" "data.frame"
> class(dt00)
[1] "data.table" "data.frame"
> head(dt00, n = 2)
   Country.Name                          Program.Name FY2000 FY2001  FY2002   FY2003   FY2004   FY2005   FY2006   FY2007
1:  Afghanistan                Child Survival and Health     NA     NA 2586555 56501189 40215304 39817970 40856382 72527069
2:  Afghanistan Department of Defense Security Assistance     NA     NA 2964313       NA 45635526 151334908 230501318 214505892
> head(dt90, n = 2)
   Country.Name                          Program.Name FY1990 FY1991 FY1992 FY1993 FY1994 FY1995 FY1996 FY1997 FY1998 FY1999
1:  Afghanistan                Child Survival and Health     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
2:  Afghanistan Department of Defense Security Assistance     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
> head(dt0090, n = 2)
   Country.Name                          Program.Name FY1990 FY1991 FY1992 FY1993 FY1994 FY1995 FY1996 FY1997 FY1998 FY1999
1:  Afghanistan                Child Survival and Health     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
2:  Afghanistan Department of Defense Security Assistance     NA     NA     NA     NA     NA     NA     NA     NA     NA     NA
      FY2005    FY2006    FY2007    FY2008    FY2009
1:  39817970  40856382  72527069  28397435        NA
2: 151334908 230501318 214505892 495539084 552524990
>
```

R reshape2 套件置換行列資料 - melt

1. melt()

在這裡要將國家援助計畫的金額欄位資料給抽出來，並建立成 "國家-援助計畫-年份" 的表，並做出一個 Dollars 的直欄。在此檢視 Aid_00s !!!

head(Aid_00s)

```
> head(Aid_00s)
  Country.Name                                Program.Name FY2000  FY2001   FY2002    FY2003    FY2004    FY2005    FY2006    FY2007    FY2008    FY2009
1  Afghanistan                      Child Survival and Health     NA      NA  2586555  56501189  40215304  39817970  40856382  72527069  28397435        NA
2  Afghanistan         Department of Defense Security Assistance     NA      NA  2964313        NA  45635526 151334908 230501318 214505892 495539084 552524990
3  Afghanistan                           Development Assistance     NA 4110478  8762080  54538965 180539337 193598227 212648440 173134034 150529862   3675202
4  Afghanistan Economic Support Fund/Security Support Assistance     NA   61144 31827014 341306822 1025522037 1157530168 1357750249 1266653993 1400237791 1418688520
5  Afghanistan                             Food For Education     NA      NA       NA   3957312   2610006   3254408    386891        NA        NA        NA
6  Afghanistan                   Global Health and Child Survival     NA      NA       NA        NA        NA        NA        NA        NA  63064912   1764252
> |
```

R reshape2 套件置換行列資料 - melt

在此利用 reshape2 套件中的 melt() 函數，當中的 id.vars 引數 是用於辨識，最後用 tail (, 10) 檢視 melt 物件的 最後幾行。

```
> require(reshape2)
Loading required package: reshape2

Attaching package: 'reshape2'

The following objects are masked from 'package:data.table':

    dcast, melt

> melt00 = melt(Aid_00s, id.vars=c("Country.Name", "Program.Name"), variable.name="Year", value.name="Dollars")
> tail(melt00, 10)
      Country.Name                                          Program.Name   Year   Dollars
24521     Zimbabwe                       Migration and Refugee Assistance FY2009   3627384
24522     Zimbabwe                                     Narcotics Control FY2009        NA
24523     Zimbabwe Nonproliferation, Anti-Terrorism, Demining and Related FY2009        NA
24524     Zimbabwe                             Other Active Grant Programs FY2009   7951032
24525     Zimbabwe                                 Other Food Aid Programs FY2009        NA
24526     Zimbabwe                                  Other State Assistance FY2009   2193057
24527     Zimbabwe                                  Other USAID Assistance FY2009  41940500
24528     Zimbabwe                                          Peace Corps FY2009        NA
24529     Zimbabwe                                             Title I FY2009        NA
24530     Zimbabwe                                            Title II FY2009 174572685
> |
```

R reshape2 套件置換行列資料 - melt


```
require(reshape2)
melt00 = melt(Aid_00s, id.vars=c("Country.Name", "Program.Name"), variable.name="Year",
value.name="Dollars")
tail(melt00, 10)
```

R reshape2 套件置換行列資料 - melt

利用 之前的 aggregate() 抽出欄位進行分析， 並利用 ggplot() 套件進行繪圖

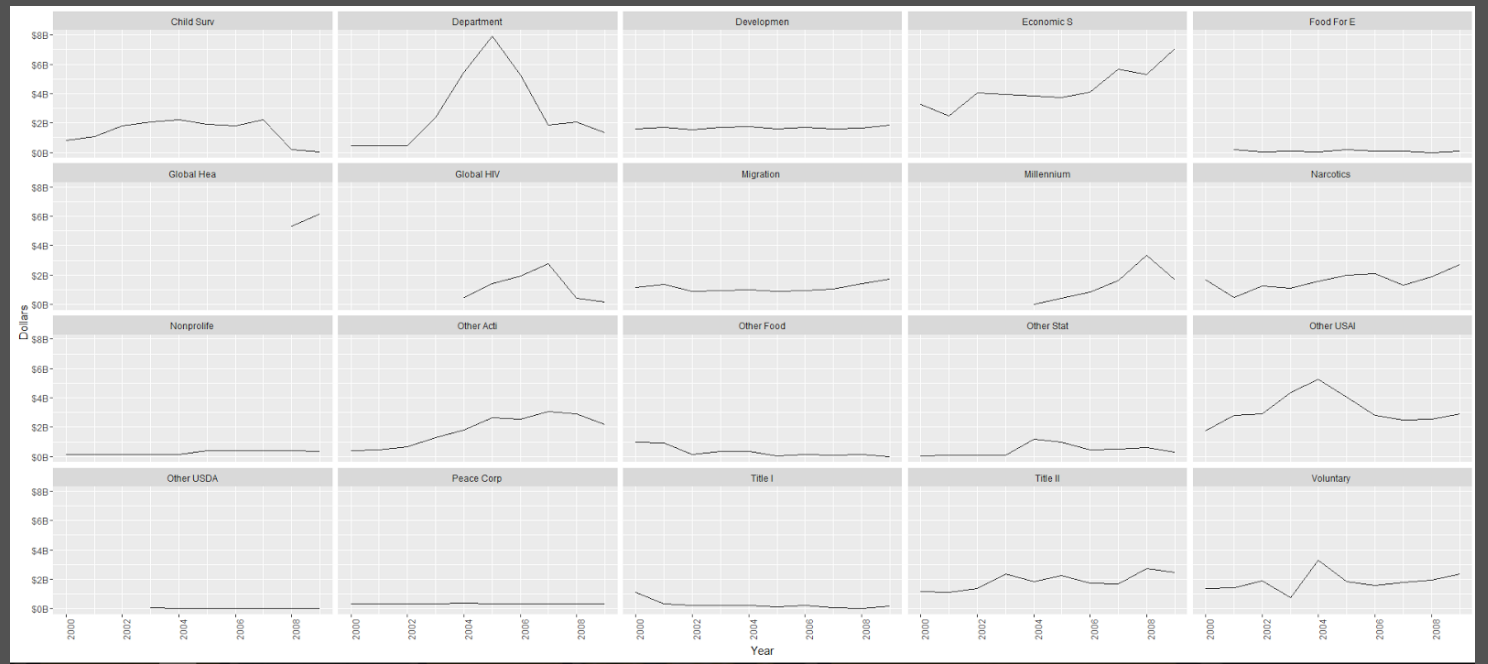
```
> require(scales)
Loading required package: scales
>
> melt00$Year = as.numeric(str_sub(melt00$Year, start=3, 6))
>
> meltAgg = aggregate(Dollars ~ Program.Name + Year, data=melt00, sum, na.rm=TRUE)
>
> meltAgg$Program.Name <- str_sub(meltAgg$Program.Name, start=1, end=10)
>
> ggplot(meltAgg, aes(x=Year, y=Dollars)) +
+       geom_line(aes(group=Program.Name)) +
+       facet_wrap(~ Program.Name) +
+       scale_x_continuous(breaks=seq(from=2000, to=2009, by=2)) +
+       theme(axis.text.x=element_text(angle=90, vjust=1, hjust=0)) +
+       scale_y_continuous(labels=multiple_format(extra=dollar, multiple="B"))
>
```

```
> class(melt00$Year)
[1] "numeric"
> class(melt00)
[1] "data.frame"
>
```

```
> class(meltAgg$Program.Name)
[1] "character"
> class(meltAgg)
[1] "data.frame"
> head(meltAgg, n = 10)
   Program.Name Year    Dollars
1    Child Surv    0  796270850
2    Department    0  459055045
3    Developmen    0 1575525119
4    Economic S    0 3247215048
5    Migration     0 1136208206
6    Narcotics     0 1655269245
7    Nonprolife    0  133733048
8    Other Acti    0  410886318
9    Other Food    0  987565339
10   Other Stat    0   37664755
>
```

R reshape2 套件置換行列資料 - melt

```
require(scales)
melt00$Year = as.numeric(str_sub(melt00$Year, start=3, 6))
meltAgg = aggregate(Dollars ~ Program.Name + Year, data=melt00, sum, na.rm=TRUE)
meltAgg$Program.Name = str_sub(meltAgg$Program.Name, start=1, end=10)
ggplot(meltAgg, aes(x=Year, y=Dollars)) +
    geom_line(aes(group=Program.Name)) +
    facet_wrap(~ Program.Name) +
    scale_x_continuous(breaks=seq(from=2000, to=2009, by=2)) +
    theme(axis.text.x=element_text(angle=90, vjust=1, hjust=0)) +
    scale_y_continuous(labels=multiple_format(extra=dollar, multiple="B"))
```



R reshape2 套件置換行列資料 - dcast

2. dcast()

可以還原其 melt() 函數的狀態

cast00 = dcast(melt00, Country.Name + Program.Name ~ Year, value.var = "Dollars")
head(cast00)

```
> cast00 = dcast(melt00, Country.Name + Program.Name ~ Year, value.var = "Dollars")
> head(cast00)
  Country.Name                                    Program.Name 2000    2001     2002      2003
1  Afghanistan                      Child Survival and Health   NA      NA  2586555  56501189
2  Afghanistan         Department of Defense Security Assistance   NA      NA  2964313        NA
3  Afghanistan                           Development Assistance   NA 4110478  8762080  54538965
4  Afghanistan Economic Support Fund/Security Support Assistance   NA   61144 31827014 341306822
5  Afghanistan                                Food For Education   NA      NA       NA   3957312
6  Afghanistan                  Global Health and Child Survival   NA      NA       NA        NA
> |
```

R 字串處理

1. 利用 paste() 函數建立字串
2. 利用 sprintf() 函數建立字串
3. 抽取文字
4. 正規表式法

R 字串處理 - paste

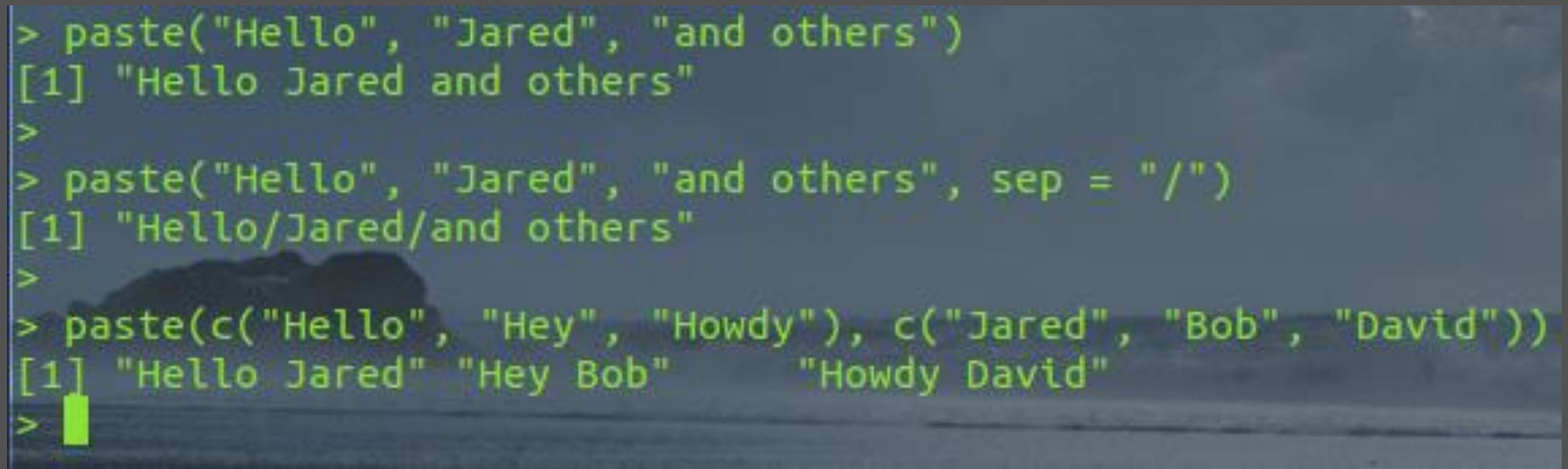
paste("[字串 1]", "[字串 2]", "[字串 3]", ...)

paste("[字串 1]", "[字串 2]", "[字串 3]", sep = "[字串間的分隔符號]")

利用向量的方式進行合併

paste("[字串 a1]", "[字串 b1]", "[字串 c1]", c("[字串 a2]", "[字串 b2]", "[字串 c2]"))

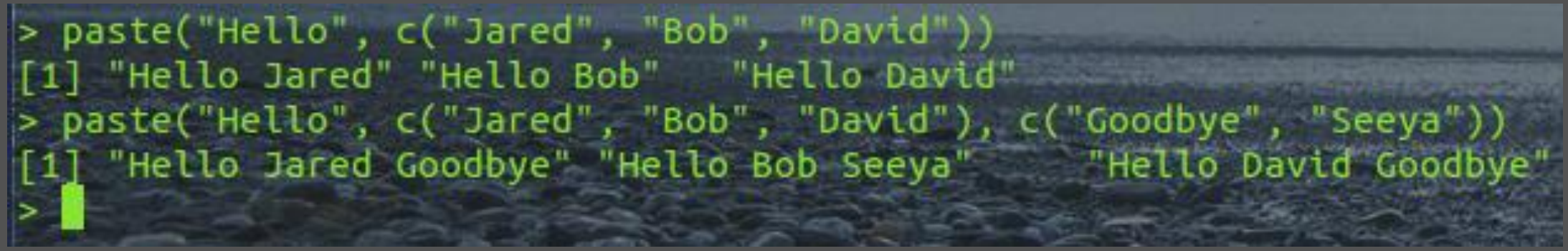
paste("Hello", "Jared", "and others")
paste("Hello", "Jared", "and others", sep = "/")
paste(c("Hello", "Hey", "Howdy"), c("Jared", "Bob", "David"))



R 字串處理 - paste

若向量與對應的字串長度不一，則長度短的會被重複使用

paste("Hello", c("Jared", "Bob", "David"))
paste("Hello", c("Jared", "Bob", "David"), c("Goodbye", "Seeya"))



R 字串處理 - paste

vectorOfText = c("Hello", "Everyone", "out there", ".")
paste(vectorOfText, collapse = " ")
paste(vectorOfText, collapse = "*")

paste() 的 collapse 引數可將多個字串塞在一起

```
> vectorOfText = c("Hello", "Everyone", "out there", ".")
> paste(vectorOfText, collapse = " ")
[1] "Hello Everyone out there ."
> paste(vectorOfText, collapse = "*")
[1] "Hello*Everyone*out there*."
>
```

在此將 paste(vectorOfText, collapse = "*") 指向另一個名為 kanchar 物件

```
> vectorOfText = c("Hello", "Everyone", "out there", ".")
> paste(vectorOfText, collapse = " ")
[1] "Hello Everyone out there ."
> paste(vectorOfText, collapse = "*")
[1] "Hello*Everyone*out there*."
> kanchar = paste(vectorOfText, collapse = "*")
> vectorOfText2 = c("Hello", "Everyone", "out there", ".", kanchar)
> vectorOfText2
[1] "Hello"                    "Everyone"                 "out there"
>
```

R 字串處理 - sprintf

簡單的用 -> paste() 函數
複雜的用 -> sprintf() 函數

sprintf() 函數相對於paste() 函數，其前者在處理向量化運算要注意"長度"是否為倍數，後者僅為長度較短的進行重複使用

person = "Jared"
partySize = "eight"
waitTime = 25

paste("Hello ", person, ", your party of ", partySize,
" will be seated in ", waitTime, " minutes.", sep="")

```
> person = "Jared"
> partySize = "eight"
> waitTime = 25
>
```

```
> paste("Hello ", person, ", your party of ", partySize,
+ " will be seated in ", waitTime, " minutes.", sep="")
[1] "Hello Jared, your party of eight will be seated in 25 minutes."
>
```

R 字串處理 - sprintf

我們可以看相對應的字串匯入至指定的地方，但資料規模與複雜度都變大，則改用 sprintf() 函數。

sprintf("Hello %s, your party of %s will be seated in %s minutes", person, partySize, waitTime)

```
> sprintf("Hello %s, your party of %s will be seated in %s minutes",
+ person, partySize, waitTime)
[1] "Hello Jared, your party of eight will be seated in 25 minutes"
>
```

R 字串處理 - sprintf

注意 sprintf()的向量化處理字串時的長度倍數關係!!!

當中有 3 個 %s，相對匯入得向量字串為

第一個位置 -> c("Jared", "Bob")
第二個位置 -> c("eight", 16, "four", 10)
第三的位置 -> waitTime

waitTime 為已開始建立的物件，其值只有一個 為 25，所以該 sprintf()
的向量化處理的長度互為倍數關係，所以可以成功

sprintf("Hello %s, your party of %s will be seated in %s minutes",
c("Jared", "Bob"), c("eight", 16, "four", 10), waitTime)

```
> sprintf("Hello %s, your party of %s will be seated in %s minutes",
+ c("Jared", "Bob"), c("eight", 16, "four", 10), waitTime)
[1] "Hello Jared, your party of eight will be seated in 25 minutes"
[2] "Hello Bob, your party of 16 will be seated in 25 minutes"
[3] "Hello Jared, your party of four will be seated in 25 minutes"
[4] "Hello Bob, your party of 10 will be seated in 25 minutes"
>
```

R 字串處理 -抽取文字

在此是利用 R 的 XML 套件與 readHTMLTable() 將其網路上的歷屆美國正副總統的資料進行匯入與分析!!!!

網址 :
http://www.loc.gov/rr/print/list/057_chron.html

若 找不到套件可試著用 install.packages("[套件名稱]")，安裝後再執行看看 !!!

```
> install.packages("XML")
Installing package into ‘/home/kancheng/R/x86_64-pc-linux-gnu-library/3.2’
(as ‘lib’ is unspecified)
--- Please select a CRAN mirror for use in this session ---
嘗試 URL 'https://cran.ism.ac.jp/src/contrib/XML_3.98-1.4.tar.gz'
Content type 'application/x-gzip' length 1599214 bytes (1.5 MB)
==================================================
downloaded 1.5 MB

* installing *source* package ‘XML’ ...
** package ‘XML’ successfully unpacked and MD5 sums checked
checking for gcc... gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking how to run the C preprocessor... gcc -E
checking for sed... /bin/sed
checking for pkg-config... /usr/bin/pkg-config
checking for xml2-config... no
Cannot find xml2-config
ERROR: configuration failed for package ‘XML’
* removing ‘/home/kancheng/R/x86_64-pc-linux-gnu-library/3.2/XML’

The downloaded source packages are in
        ‘/tmp/RtmpmKFeRx/downloaded_packages’
Warning message:
In install.packages("XML") :
  installation of package ‘XML’ had non-zero exit status
>
```

R 字串處理 -抽取文字

require(XML)

theURL = "http://www.loc.gov/rr/print/list/057_chron.html"

presidents = readHTMLTable(theURL, which=3, as.data.frame=TRUE,
skip.rows=1, header=TRUE,
stringsAsFactors=FALSE)

```
> require(XML)
Loading required package: XML
>
> theURL = "http://www.loc.gov/rr/print/list/057_chron.html"
>
> presidents = readHTMLTable(theURL, which=3, as.data.frame=TRUE,
+ skip.rows=1, header=TRUE,
+ stringsAsFactors=FALSE)
>
```

R 字串處理 -抽取文字

head()

```
> head(presidents)
       YEAR          PRESIDENT                                        FIRST LADY   VICE PRESIDENT
1 1789-1797 George Washington                                  Martha Washington       John Adams
2 1797-1801        John Adams                                      Abigail Adams Thomas Jefferson
3 1801-1805  Thomas Jefferson Martha Wayles Skelton Jefferson\n   (no image)       Aaron Burr
4 1805-1809  Thomas Jefferson Martha Wayles Skelton Jefferson\n   (no image)   George Clinton
5 1809-1812     James Madison                                    Dolley Madison   George Clinton
6 1812-1813     James Madison                                    Dolley Madison    office vacant
> |
```

tail()

```
> tail(presidents$YEAR)
[1] "2001-2009"
[2] "2009-"
[3] "Presidents: Introduction (Rights/Ordering\n        Info.) | Adams\n      - Clev
[4] "First Ladies: Introduction\n                    (Rights/Ordering Info.) | Adams\n
[5] "Vice Presidents: Introduction (Rights/Ordering Info.) | Adams - Coolidge | Curt
[6] "Top\n                of Page"
> presidents = presidents[1:64, ]
> |
```

```
$
$
$    - Jefferson | Johnson - McKinley | Monroe\n                    - Roosevelt | Taft - Truman | Tyler\n                    - WilsonList of names, Alphabetically"
$wer - HooverJackson\n                 - Pierce  | \n                          Polk - Wilson | List\n             of names, Alphabetically"
$oosevelt - WilsonList of names, Alphabetically"
$
```

R 字串處理 -抽取文字

可以看到 presidents的 資料在 64 之後基本上是多餘的...，再來利用 names()、class()、head()、tail() 等函數，放在一起觀察。

```
> names(presidents)
[1] "YEAR"           "PRESIDENT"      "FIRST LADY"     "VICE PRESIDENT"
> class(presidents)
[1] "data.frame"
> head(presidents, n = 3)
       YEAR         PRESIDENT                                    FIRST LADY   VICE PRESIDENT
1 1789-1797 George Washington                              Martha Washington       John Adams
2 1797-1801        John Adams                                  Abigail Adams Thomas Jefferson
3 1801-1805  Thomas Jefferson Martha Wayles Skelton Jefferson\n    (no image)       Aaron Burr
> tail(presidents, n = 10)

59
60
61
62
63
64
65 Presidents: Introduction (Rights/Ordering\n        Info.) | Adams\n      - Cleveland | Clin
66                       First Ladies: Introduction\n                    (Rights/Ordering Info.)
67
68

       PRESIDENT              FIRST LADY    VICE PRESIDENT
59   Jimmy Carter         Rosalynn Carter Walter F. Mondale
60  Ronald Reagan            Nancy Reagan       George Bush
61    George Bush            Barbara Bush        Dan Quayle
62   Bill Clinton Hillary Rodham Clinton       Albert Gore
63 George W. Bush              Laura Bush     Richard Cheney
64   Barack Obama          Michelle Obama    Joseph R. Biden
65           <NA>                    <NA>              <NA>
66           <NA>                    <NA>              <NA>
67           <NA>                    <NA>              <NA>
68           <NA>                    <NA>              <NA>
>
```

R 字串處理 -抽取文字

利用 stringr 的套件將 年份 拆開，對其建立兩個直行，一個是上任年份、一個是卸任年份，而他們之間 是用 " - " 作為分隔。

```
> require(stringr)
Loading required package: stringr
>
> yearList = str_split(string = presidents$YEAR, pattern = "-")
> head(yearList)
[[1]]
[1] "1789" "1797"

[[2]]
[1] "1797" "1801"

[[3]]
[1] "1801" "1805"

[[4]]
[1] "1805" "1809"

[[5]]
[1] "1809" "1812"

[[6]]
[1] "1812" "1813"
```

str_split(string = [資料表]$[字串欄位], pattern = "[判別字串的分隔符號]")

```
require(stringr)
yearList = str_split(string = presidents$YEAR, pattern = "-")
head(yearList)
```

R 字串處理 -抽取文字

利用 data.frame()、 Reduce()、 rbind() 等函數，將剛剛處理的資料合成矩陣(matrix)

```
> yearMatrix = data.frame(Reduce(rbind, yearList))
Warning message:
In data.row.names(row.names, rowsi, i) :
  some row.names duplicated: 3,4,5,6,7,8,9,10,11,12
> head(yearMatrix)
    X1   X2
1 1789 1797
2 1797 1801
3 1801 1805
4 1805 1809
5 1809 1812
6 1812 1813
>
```

```
yearMatrix = data.frame(Reduce(rbind, yearList))
head(yearMatrix)
```

R 字串處理 -抽取文字

names() -> 對直行命名
cbind() -> 合併資料表
as.numeric() -> 將資料轉成 numeric

as.numeric(as.character()) -> 將字串資料轉成 numeric
as.numeric(as.character([資料名稱]$[直行欄位]))

names(yearMatrix) = c("Start", "Stop")
presidents = cbind(presidents, yearMatrix)
presidents$Start = as.numeric(as.character(presidents$Start))
presidents$Stop = as.numeric(as.character(presidents$Stop))

```
> names(yearMatrix) = c("Start", "Stop")
>
> presidents = cbind(presidents, yearMatrix)
>
>
> presidents$Start = as.numeric(as.character(presidents$Start))
> presidents$Stop = as.numeric(as.character(presidents$Stop))
>
```

R 字串處理 -抽取文字

觀察處理過後的資料
head(presidents)
tail(presidents)

```
> head(presidents)
       YEAR         PRESIDENT                                    FIRST LADY   VICE PRESIDENT Start Stop
1 1789-1797 George Washington                              Martha Washington       John Adams  1789 1797
2 1797-1801        John Adams                                  Abigail Adams Thomas Jefferson  1797 1801
3 1801-1805  Thomas Jefferson Martha Wayles Skelton Jefferson\n   (no image)       Aaron Burr  1801 1805
4 1805-1809  Thomas Jefferson Martha Wayles Skelton Jefferson\n   (no image)   George Clinton  1805 1809
5 1809-1812     James Madison                                 Dolley Madison   George Clinton  1809 1812
6 1812-1813     James Madison                                 Dolley Madison    office vacant  1812 1813
> |
```

```
> tail(presidents)
        YEAR      PRESIDENT              FIRST LADY    VICE PRESIDENT Start Stop
59 1977-1981   Jimmy Carter        Rosalynn Carter Walter F. Mondale  1977 1981
60 1981-1989  Ronald Reagan           Nancy Reagan       George Bush  1981 1989
61 1989-1993    George Bush           Barbara Bush        Dan Quayle  1989 1993
62 1993-2001   Bill Clinton Hillary Rodham Clinton       Albert Gore  1993 2001
63 2001-2009 George W. Bush             Laura Bush     Richard Cheney  2001 2009
64     2009-   Barack Obama         Michelle Obama    Joseph R. Biden  2009   NA
> |
```

R 字串處理 -抽取文字

基本上利用 str_sub() 進行抽取資料....，針對表中的總統名字的欄位，進行抽取

str_sub(string = [資料名稱]$[指定直行], start = [起始字元順序], end = [結束字元順序])

在此抽取 第一個 至 第三個字元的資料
例如 抽取資料欄位的第一筆資料為 -> George Washington
因為抽出 第一個 至 第三個字元，其結果為 -> "Geo"

```
> str_sub(string = presidents$PRESIDENT, start = 1, end = 3)
 [1] "Geo" "Joh" "Tho" "Tho" "Jam" "Jam" "Jam" "Jam" "Jam" "Joh" "And" "And" "Mar
[29] "Jam" "Che" "Gro" "Gro" "Ben" "Gro" "Wil" "Wil" "Wil" "The" "The" "Wil" "Wil
[57] "Ric" "Ger" "Jim" "Ron" "Geo" "Bil" "Geo" "Bar"
> |
```

如上類推 第四至第八的字串

```
> str_sub(string = presidents$PRESIDENT, start = 4, end = 8)
 [1] "rge W" "n Ada" "mas J" "mas J" "es Ma" "es Ma" "es Ma" "es Ma" "es Mo" "n Qui" "rew J" "
[22] "aham " "aham " "rew J" "sses " "sses " "sses " "herfo" "es A." "ster " "ver C" "ver C" "
[43] "ren G" "vin C" "vin C" "bert " "nklin" "nklin" "nklin" "ry S." "ry S." "ght D" "n F. " "
[64] "ack O"
> |
```

R 字串處理 -抽取文字

可以找到上任年份結尾為 "1 "的美國總統

```
> presidents[str_sub(string = presidents$Start, start = 4, end = 4) == 1, c("YEAR", "PRESIDENT", "Start", "Stop")]
        YEAR              PRESIDENT Start Stop
3  1801-1805       Thomas Jefferson  1801 1805
14      1841 William Henry Harrison  1841 1841
15 1841-1845             John Tyler  1841 1845
22 1861-1865       Abraham Lincoln  1861 1865
29      1881      James A. Garfield  1881 1881
30 1881-1885      Chester A. Arthur  1881 1885
37      1901       William McKinley  1901 1901
38 1901-1905     Theodore Roosevelt  1901 1905
43 1921-1923      Warren G. Harding  1921 1923
48 1941-1945  Franklin D. Roosevelt  1941 1945
53 1961-1963        John F. Kennedy  1961 1963
60 1981-1989          Ronald Reagan  1981 1989
63 2001-2009         George W. Bush  2001 2009
> |
```

str_sub(string = presidents$PRESIDENT, start = 1, end = 3)
str_sub(string = presidents$PRESIDENT, start = 4, end = 8)
presidents[str_sub(string = presidents$Start, start = 4, end = 4) == 1, c("YEAR", "PRESIDENT", "Start", "Stop")]

R 字串處理 - 正規表式法

Regular Expression 詳見 :

https://zh.wikipedia.org/zh-tw/%E6%AD%A3%E5%88%99%E8%A1%A8%E8%BE%BE%E5%BC%8F

http://linux.vbird.org/linux_basic/0330regularex.php

https://developer.mozilla.org/zh-TW/docs/Web/JavaScript/Obsolete_Pages/Obsolete_Pages/Obsolete_Pages/%E6%AD%A3%E8%A6%8F%E8%A1%A8%E9%81%94%E5%BC%8F%E6%A8%A1%E5%BC%8F%E7%9A%84%E7%B7%A8%E5%AF%AB

R 字串處理 - 正規表式法

在此搜尋指定欄位的特定 "字串" 值 -> str_detect() 函數，總統名稱的直行上搜尋 "John" 這個值。

johnPos = str_detect(string = presidents$PRESIDENT, pattern = "John")
presidents[johnPos, c("YEAR", "PRESIDENT", "Start", "Stop")]

```
> johnPos = str_detect(string = presidents$PRESIDENT, pattern = "John")
> presidents[johnPos, c("YEAR", "PRESIDENT", "Start", "Stop")]
        YEAR          PRESIDENT Start Stop
2  1797-1801         John Adams  1797 1801
10 1825-1829 John Quincy Adams  1825 1829
15 1841-1845         John Tyler  1841 1845
24 1865-1869     Andrew Johnson  1865 1869
53 1961-1963    John F. Kennedy  1961 1963
54 1963-1965 Lyndon B. Johnson  1963 1965
55 1965-1969 Lyndon B. Johnson  1965 1969
>
```

R 字串處理 - 正規表式法

若要忽略大小寫問題，請將條件設在 ignore.case() 當中

```
badSearch = str_detect(presidents$PRESIDENT, "john")
goodSearch = str_detect(presidents$PRESIDENT,
                        ignore.case("John"))
sum(badSearch)
sum(goodSearch)
```

```
> badSearch = str_detect(presidents$PRESIDENT, "john")
> goodSearch = str_detect(presidents$PRESIDENT, ignore.case("John"))
Please use (fixed|coll|regexp)(x, ignore_case = TRUE) instead of ignore.case(x)
> sum(badSearch)
[1] 0
> sum(goodSearch)
[1] 7
> |
```

```
> class(goodSearch)
[1] "logical"
> class(badSearch)
[1] "logical"
> goodSearch
 [1] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
[29] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[57] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
> badSearch
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[29] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[57] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
> presidents[badSearch, c("YEAR", "PRESIDENT", "Start", "Stop")]
[1] YEAR      PRESIDENT Start     Stop
<0 rows> (or 0-length row.names)
> presidents[goodSearch, c("YEAR", "PRESIDENT", "Start", "Stop")]
        YEAR          PRESIDENT Start Stop
2  1797-1801         John Adams  1797 1801
10 1825-1829 John Quincy Adams  1825 1829
15 1841-1845         John Tyler  1841 1845
24 1865-1869     Andrew Johnson  1865 1869
53 1961-1963    John F. Kennedy  1961 1963
54 1963-1965 Lyndon B. Johnson  1963 1965
55 1965-1969 Lyndon B. Johnson  1965 1969
```

R 字串處理 - 正規表式法

範例當中將美國戰爭的表格進行示範，並利用 url() 網址匯入 (.rdata) 的檔案。

資料 : http://www.jaredlander.com/data/warTimes.rdata

url("[網址]")

```
con = url("http://www.jaredlander.com/data/warTimes.rdata")
load(con)
close(con)
```

```
AA = url("https://github.com/kancheng/easyR/blob/master/C03/warTimes.rdata?raw=true")
load(AA)
close()
```

```
> con = url("http://www.jaredlander.com/data/warTimes.rdata")
> load(con)
> close(con)
>
```

R 字串處理 - 正規表式法

warTimes 屬性為 "字串"

```
> head(warTimes, 10)
 [1] "September 1, 1774 ACAEA September 3, 1783" "September 1, 1774 ACAEA March 17, 1776"
 [4] "June 1775 ACAEA October 1776"              "July 1776 ACAEA March 1777"
 [7] "1777ACAEA1778"                             "1775ACAEA1782"
[10] "1778ACAEA1782"
>
```

head(warTimes, 10)

R 字串處理 - 正規表式法

目標是要將不規則的字串，整理成有規則且能夠分析的直行欄位，因為資料為 維基百科編碼 分隔使用的是 "ACAEA" 、 "-"，觀察 warTimes 部分資料，我們可以看到作為分隔的字串。

warTimes[str_detect(string = warTimes, pattern = "-")]

```
> warTimes[str_detect(string = warTimes, pattern = "-")]
[1] "6 June 1944 ACAEA mid-July 1944" "25 August-17 December 1944"
```

```
> warTimes
 [1] "September 1, 1774 ACAEA September 3, 1783" "September 1, 1774 ACAEA March 17, 1776"
 [4] "June 1775 ACAEA October 1776"              "July 1776 ACAEA March 1777"
 [7] "1777ACAEA1778"                             "1775ACAEA1782"
[10] "1778ACAEA1782"                             "1775ACAEA1782"
[13] "January ACAEA October, 1781"               "1785ACAEA1795"
[16] "1801ACAEA1805"                             "August ACAEA November 1811"
[19] "1812ACAEA1815"                             "1813ACAEA1814"
[22] "1812ACAEA1813"                             "1813"
[25] "1813ACAEA1814"                             "1814"
[28] "1814ACAEA1815"                             "1813ACAEA1814"
[31] "November 22, 1817 ACAEA April 12, 1818"    "1817ACAEA1825"
[34] "November 5ACAEA6, 1820"                    "1823"
[37] "1827"                                      "MayACAEAAugust 1832"
[40] "1838ACAEA1842"                             "December 23, 1835 ACAEA August 14, 1842"
```

R 字串處理 - 正規表式法

```
> theTimes = str_split(string = warTimes, pattern = "(ACAEA)|-", n = 2)
> head(theTimes)
[[1]]
[1] "September 1, 1774 " " September 3, 1783"

[[2]]
[1] "September 1, 1774 " " March 17, 1776"

[[3]]
[1] "1775" "1783"

[[4]]
[1] "June 1775 "    " October 1776"

[[5]]
[1] "July 1776 "   " March 1777"

[[6]]
[1] "June 14, 1777 "    " October 17, 1777"

> |
```

利用 str_split() 函數進行實作
在此 n = 2，書中的用意是最多將文字拆散成兩個部分或者不拆散。

若要加入括號的條件 -> (\)

若直接用 ( )，可能會被視為程式的一部分

再來資料中的目的是要將日、月、年 拆開，但是當中有 "mid-July" (七月中)，時間並不明確的資料，"mid-July" 中的 "-" 則避免抽取!!!

str_split(string = [資料物件名稱], pattern = "(ACAEA)|-", n = [回傳值的數量])

theTimes = str_split(string = warTimes, pattern = "(ACAEA)|-", n = 2)
head(theTimes)

R 字串處理 - 正規表式法

在此可以看到 "mid-July" 並沒有被拆開!!!

而當中的 theTimes 用 class() 觀察為 list 物件

```
> which(str_detect(string = warTimes, pattern = "-"))
[1] 147 150
> theTimes[[147]]
[1] "6 June 1944 "    " mid-July 1944"
> theTimes[[150]]
[1] "25 August"          "17 December 1944"
> 
```

which(str_detect(string = warTimes, pattern = "-"))
theTimes[[147]]
theTimes[[150]]

R 字串處理 - 正規表式法

在此將戰爭開始時間抽出，在此利用 sapply() 與自行建立的函數 function(x) 抽取資料，並命名為 theStart

```
> theStart = sapply(theTimes, FUN = function(x) x[1])
> head(theStart)
[1] "September 1, 1774 " "September 1, 1774 " "1775"
> 
```

theStart = sapply(theTimes, FUN = function(x) x[1])
head(theStart)

R 字串處理 - 正規表式法

若要移除資料中的多餘空格 -> str_trim() 函數

```
> theStart = str_trim(theStart)
> head(theStart)
[1] "September 1, 1774" "September 1, 1774" "1775"
> 
```

theStart = str_trim(theStart)
head(theStart)

R 字串處理 - 正規表式法

若要抽出資料中的特定字串 -> str_extract() 函數

str_extract(string = [資料的物件名稱], pattern = "[特定字串]")

str_extract(string = theStart, pattern = "January")

```
> str_extract(string = theStart, pattern = "January")
  [1] NA        NA        NA        NA        NA        NA
 [17] NA        NA        NA        NA        NA        NA
 [33] NA        NA        NA        NA        NA        NA
 [49] NA        NA        NA        NA        NA        NA
 [65] NA        NA        NA        NA        NA        NA
 [81] NA        NA        NA        NA        NA        NA
 [97] NA        NA        "January" NA        NA        NA
[113] NA        NA        NA        NA        NA        NA
[129] NA        NA        "January" NA        NA        NA
[145] "January" "January" NA        NA        NA        NA
[161] NA        NA        NA        NA        NA        NA
[177] NA        NA        NA        NA        "January" NA
> |
```

R 字串處理 - 正規表式法

str_detect() 將所要特定的的字串回傳!!!

在此抽取 "January" -> 一月

theStart[str_detect(string = theStart, pattern = "January")]

```
> theStart[str_detect(string = theStart, pattern = "January")]
[1] "January"           "January 21"        "January 1942"      "January"
[9] "January 14, 2010"
> |
```

R 字串處理 - 正規表式法

利用 str_extract () 與 表示式 將 四個並排的 0~9 的數字 找出來 [0-9][0-9][0-9][0-9]

這三個用途是一樣的
[0-9][0-9][0-9][0-9] <-> [0-9]{4} <-> \\d{4}

```
head(str_extract(string = theStart, "[0-9][0-9][0-9][0-9]"), 20)
head(str_extract(string = theStart, "[0-9]{4}"), 20)
head(str_extract(string = theStart, "\\d{4}"), 20)
```

```
> head(str_extract(string = theStart, "[0-9][0-9][0-9][0-9]"), 20)
 [1] "1774" "1774" "1775" "1775" "1776" "1777" "1777" "1775" "1776"
>
```

```
> head(str_extract(string = theStart, "[0-9]{4}"), 20)
 [1] "1774" "1774" "1775" "1775" "1776" "1777" "1777" "1775" "1776"
>
```

```
> head(str_extract(string = theStart, "\\d{4}"), 20)
 [1] "1774" "1774" "1775" "1775" "1776" "1777" "1777" "1775" "1776"
>
```

R 字串處理 - 正規表式法

str_extract ()函數與表達式，也可以找出重複出現過的次數

str_extract(string = [資料名稱], "\\d{[開始次數],[結束次數]}")

目標是找出重複 1次、2次、3次 的結果

str_extract(string = theStart, "\\d{1,3}")

```
> str_extract(string = theStart, "\\d{1,3}")
  [1] "1"   "1"   "177" "177" "177" "14"  "177" "177" "177" "177"
 [28] "181" "181" "181" "22"  "181" "181" "5"   "182" "182" "182"
 [55] "13"  "4"   "185" "185" "185" "185" "185" "185" "6"   "185"
 [82] "17"  "1"   "6"   "12"  "27"  "187" "187" "187" "187" "187"
[109] "189" "28"  "191" "21"  "28"  "191" "191" "191" "191" "191"
[136] "1"   "16"  "194" "8"   "194" "17"  "9"   "194" "3"   "22"
[163] "15"  "24"  "19"  "198" "15"  "198" "4"   "20"  "2"   "199"
>
```

R 字串處理 - 正規表式法

找出 文字前面 與 文字後面 的特定文字!!!
"^" -> 搜尋前面
"$" -> 搜尋後面

```
head(str_extract(string = theStart, pattern = "^\\d{4}"), 30)
head(str_extract(string = theStart, pattern = "\\d{4}$"), 30)
head(str_extract(string = theStart, pattern = "^\\d{4}$"), 30)
```

搜尋前面

```
> head(str_extract(string = theStart, pattern = "^\\d{4}"), 30)
 [1] NA     NA     "1775" NA     NA     NA     "1777" "1775" "1776"
[25] "1813" "1814" "1813" "1814" "1813" "1815"
```

搜尋後面

```
> head(str_extract(string = theStart, pattern = "\\d{4}$"), 30)
 [1] "1774" "1774" "1775" "1775" "1776" "1777" "1777" "1775" "1776"
[25] "1813" "1814" "1813" "1814" "1813" "1815"
```

搜尋前面與搜尋後面

```
> head(str_extract(string = theStart, pattern = "^\\d{4}$"), 30)
 [1] NA     NA     "1775" NA     NA     NA     "1777" "1775" "1776"
[25] "1813" "1814" "1813" "1814" "1813" "1815"
> |
```

R 字串處理 - 正規表式法

利用 str_replace( )將值取代

str_replace(string=[物件名稱], pattern="[範圍]", replacement="[進行去取代的值]")

所有值取代 -> str_replace_all( )

str_replace_all(string=[物件名稱], pattern="[範圍]", replacement="[進行去取代的值]")

將第一個數字取代成 x

```
> head(str_replace(string=theStart, pattern="\\d", replacement="x"), 30)
 [1] "September x, 1774" "September x, 1774" "x775"              "June x775"
 [9] "x776"              "x778"              "x775"              "x779"
[17] "August"            "June x8, 1812"     "x812"              "x813"
[25] "x813"              "x814"              "x813"              "x814"
> |
```

R 字串處理 - 正規表式法

將所有數字取代成 x

```
> head(str_replace_all(string=theStart, pattern="\\d", replacement="x"), 30)
 [1] "September x, xxxx" "September x, xxxx" "xxxx"                "June xxxx"
 [9] "xxxx"              "xxxx"              "xxxx"               "xxxx"
[17] "August"            "June xx, xxxx"     "xxxx"               "xxxx"
[25] "xxxx"              "xxxx"              "xxxx"               "xxxx"
> |
```

將任何 1位數 、 4位數 取代成 x

```
> head(str_replace_all(string=theStart, pattern="\\d{1,4}", replacement="x"), 30)
 [1] "September x, x" "September x, x" "x"              "June x"          "July x"
[10] "x"              "x"              "x"              "January"         "x"
[19] "x"              "x"              "x"              "x"               "x"
[28] "x"              "x"              "x"
> |
```

head(str_replace(string=theStart, pattern="\\d", replacement="x"), 30)
head(str_replace_all(string=theStart, pattern="\\d", replacement="x"), 30)
head(str_replace_all(string=theStart, pattern="\\d{1,4}", replacement="x"), 30)

R 字串處理 - 正規表式法

在此可以處理分析網頁上的資料!!!

```
> commands = c("<a href=index.html>The Link is here</a>", "<b>This is bold text</b>")
>
> str_replace(string=commands, pattern="<.+?>(.+?)<.+>", replacement="\\1")
[1] "The Link is here"  "This is bold text"
>
```

commands = c("The Link is here", "This is bold text")
str_replace(string=commands, pattern="<.+?>(.+?)<.+>", replacement="\\1")

R 矩陣條件式篩選記錄 - which

在此可以處理分析網頁上的資料!!!
將 三 直行 同樣皆為 0 的資料筆數刪除

47	SD1011173	63	29	86	0.00
48	SD1011174	97	60	83	0.00
49	SD1011176	0	92	0	0.00
50	SD1011177	60	0	0	0.00
51	SD1011178	0	0	0	0.00
52	SD1011179	68	73	65	0.00
53	SD1011181	0	0	0	45206.00
54	SD1011182	0	68	0	0.00
		0	0	0	0.00
		76	78	65	0.00
		69	88	73	0.00
		79	84	84	0.00
		71	78	89	27412.25
		84	75	84	0.00

```
> ( im12an$imca12 > 0 )& (im12an$imcpg12 > 0 )& (im12an$imec12 > 0 )
  [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [15]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [29]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [43]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE  TRUE FALSE
 [57]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [71]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [85]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [99]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE
[113] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[127]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[141]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[155]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE
> ( im12an$imca12 > 0 )&& (im12an$imcpg12 > 0 )&& (im12an$imec12 > 0 )
[1] TRUE
>
```

R 矩陣條件式篩選記錄 - which

在此可以處理分析網頁上的資料!!!
將 三 直行 同樣皆為 0 的資料筆數刪除

```
> im12an[( im12an$imca12 > 0 )& (im12an$imcpg12 > 0 )& (im12an$imec12 > 0 ),]
          sid imca12 imcpg12 imec12     loam
1   SD1011122     79      60     90     0.00
2   SD1011123     97      71     96 45206.00
3   SD1011124     67      68     86     0.00
4   SD1011125     92      81     88     0.00
5   SD1011127     92      77     80     0.00
6   SD1011129     91      75     84     0.00
7   SD1011130     60      70     78 45206.00
8   SD1011131    100      85     72     0.00
9   SD1011132     93      80     96 57106.00
10  SD1011133     63      67     80     0.00
11  SD1011134     73      60     88     0.00
12  SD1011135     89      80     77     0.00
13  SD1011136     80      64     86 27677.88
```

R 矩陣條件式篩選記錄 - which


```
# set.seed 設為 168
set.seed(168)
# 建立範例資料
mydata <- data.frame(x1=47:56,
                     x2=paste0("SD", 1011173:1011182),
                     x3=sample(0:6, 10, replace=TRUE),
                     x4=sample(0:6, 10, replace=TRUE),
                     x5=sample(0:6, 10, replace=TRUE),
                     x6=sample(3, 10, replace=TRUE))
# 在範例資料用出 3, 4, 5 列有同時為 0 的 狀況
# Dataframea名稱[ 橫列 , 直行]
mydata[c(3,6,7), c(3:5)] <- c(0, 0, 0)
# 檢視範例資料
mydata
```

```
> set.seed(168)
> mydata <- data.frame(x1=47:56,
+                      x2=paste0("SD", 1011173:1011182),
+                      x3=sample(0:6, 10, replace=TRUE),
+                      x4=sample(0:6, 10, replace=TRUE),
+                      x5=sample(0:6, 10, replace=TRUE),
+                      x6=sample(3, 10, replace=TRUE))
> mydata[c(3,6,7), c(3:5)] <- c(0, 0, 0)
> mydata
   x1         x2 x3 x4 x5 x6
1  47 SD1011173  2  6  4  2
2  48 SD1011174  6  5  4  2
3  49 SD1011175  0  0  0  2
4  50 SD1011176  6  6  4  3
5  51 SD1011177  5  2  0  1
6  52 SD1011178  0  0  0  3
7  53 SD1011179  0  0  0  3
8  54 SD1011180  5  1  2  1
9  55 SD1011181  5  3  4  1
10 56 SD1011182  1  3  4  3
>
```

R 矩陣條件式篩選記錄 - which


```
> ind <- which(mydata$x3 == 0 & mydata$x4 == 0, mydata$x5 == 0)
> ind
[1] 3 6 7
> mydata[ind, ]
  x1         x2 x3 x4 x5 x6
3 49 SD1011175  0  0  0  2
6 52 SD1011178  0  0  0  3
7 53 SD1011179  0  0  0  3
> mydata[-ind, ]
   x1         x2 x3 x4 x5 x6
1  47 SD1011173  2  6  4  2
2  48 SD1011174  6  5  4  2
4  50 SD1011176  6  6  4  3
5  51 SD1011177  5  2  0  1
8  54 SD1011180  5  1  2  1
9  55 SD1011181  5  3  4  1
10 56 SD1011182  1  3  4  3
```

```
# 使用 which 抓值
# which 當中的 mydata$x3 == 0 & mydata$x4 == 0, mydata$x5 == 0 為 x3、x4、x5 同時為 0 的條件
ind <- which(mydata$x3 == 0 & mydata$x4 == 0, mydata$x5 == 0)
# 檢視 符合條件的橫列編號
ind
# 檢視 範例資料中符合條件的橫列
mydata[ind, ]
# 刪除範例資料中符合條件的橫列
mydata[-ind, ]
```

R 矩陣條件式篩選記錄 - which

如果是資料三個欄位皆無0的資料，直接 which() 執行會有問題!!!所以必須再加個判斷。

```
> mydata <- data.frame(x1=47:56,
+                      x2=paste0("SD", 1011173:1011182),
+                      x3=sample(0:6, 10, replace=TRUE),
+                      x4=sample(0:6, 10, replace=TRUE),
+                      x5=sample(0:6, 10, replace=TRUE),
+                      x6=sample(3, 10, replace=TRUE))
>
> mydata
   x1         x2 x3 x4 x5 x6
1  47 SD1011173  6  5  2  3
2  48 SD1011174  2  3  5  1
3  49 SD1011175  3  1  1  2
4  50 SD1011176  6  6  2  2
5  51 SD1011177  1  2  3  3
6  52 SD1011178  0  2  6  2
7  53 SD1011179  5  1  5  3
8  54 SD1011180  5  0  1  1
9  55 SD1011181  3  1  4  2
10 56 SD1011182  2  1  4  2
>
```

```
> ind <- which(mydata$x3 == 0 & mydata$x4 == 0, mydata$x5 == 0)
> ind
integer(0)
> mydata[ind, ]
[1] x1 x2 x3 x4 x5 x6
<0 rows> (or 0-length row.names)
> mydata[-ind, ]
[1] x1 x2 x3 x4 x5 x6
<0 rows> (or 0-length row.names)
>
```

R 矩陣條件式篩選記錄 - which

所以用 if 與 any() 函數解決。

```
> if ( any(mydata$x3 == 0 & mydata$x4 == 0 & mydata$x5 == 0 ) ){
+ ind = which(mydata$x3 == 0 & mydata$x4 == 0 & mydata$x5 == 0 )
+ mydata = mydata[ -ind, ]
+ }else{
+ cat( "OwO\n")
+ }
OwO
>
```

```
if ( any(mydata$x3 == 0 & mydata$x4 == 0 & mydata$x5 == 0 ) ){
ind = which(mydata$x3 == 0 & mydata$x4 == 0 & mydata$x5 == 0 )
mydata = mydata[ -ind, ]
}else{
cat( "OwO\n")
}
```

R data.frame 橫列資料的 NA值數量判斷與清洗

目的在於刪去橫列比數中 NA值 數量過多的資料

黃框 是 學號

藍框 是 該屆歷年所有學生各科成績

紅框 是 單筆學生歷年成績

利用自訂函式去計算跟清洗

訂為

(有成績的科目數量)/(所有科目數量)

```
> # weg -> 0.70
>
> nac = function( data, weg){
+ ## data.frame row 轉換成向量
+ dfvtr = function( data, dfrow){
+ vtcda = as.numeric(data[ dfrow, 2:NCOL(data)])
+ vtcda
+ }
+
+ ## 求算 NA 比率
+ nacllog = function( data, dcol){
+ # 權重 DATA
+ na2dt = dfvtr( data, dcol)
+
+ # 權重分子 molecular
+ n2dmcr = length(na.omit(na2dt))
+
+ # 權重分母 Denominator
+ n2dtmr = length(na2dt)
+
+ nall = n2dmcr/n2dtmr
+ nall
+ }
+
+ ## NA 基準合併成向量
+ nabc = function(data){
+ d = numeric()
+ for( ncg in 1:NROW(data)){
+ k = nacllog( data, ncg)
+ d = c( d, k)
+ }
+ d
+ }
+
+ ## 篩選
+ data[!(nabc(data) < weg),]
+ }
>
```

R data.frame 橫列資料的 NA值數量判斷與清洗


```
nac = function( data, weg){
  ## data.frame row 轉換成向量
  dfvtr = function( data, dfrow){
    vtcda = as.numeric(data[ dfrow, 2:NCOL(data)])
    vtcda
  }
  ## 求算 NA 比率
  nacllog = function( data, dcol){
    # 權重 DATA
    na2dt = dfvtr( data, dcol)
    # 權重分子 molecular
    n2dmcr = length(na.omit(na2dt))
    # 權重分母 Denominator
    n2dtmr = length(na2dt)
    nall = n2dmcr/n2dtmr
    nall
  }
  ## NA 基準合併成向量
  nabc = function(data){
    d = numeric()
    for( ncg in 1:NROW(data)){
      k = nacllog( data, ncg)
      d = c( d, k)
    }
    d
  }
  ## 篩選
  data[!(nabc(data) < weg),]
}
```

在此就暫定比率為 0.70，若都有成績的學生比率為 1。

結果

原來 122 筆，刪去不符資格的剩下 104 名 學生。

```
> dim(lhudata$im13)
[1] 122  22
> dim(nac( lhudata$im13, 0.70))
[1] 104  22
>
```

R 拆解包含多個 Data.Frame 的 List

C03 - io.R

1. rcsvdf() - 輸入多個 CSV 為多個 Data.Frame

2. rcsvlt() - 輸入多個 CSV 為單一 List

3. wrta() - 輸出 Data.Frame 檔名不重複

R 自動建立變數與物件 - assign

assign() 這個函數可以自行建立變數並給值

```
> for(i in 1:6) {
+ nam = paste("r", i, sep = ".")
+ assign(nam, 1:i)
+ }
> ls(pattern = "^r..$")
[1] "r.1" "r.2" "r.3" "r.4" "r.5" "r.6"
>
> r.1
[1] 1
> r.2
[1] 1 2
> r.3
[1] 1 2 3
> r.4
[1] 1 2 3 4
> r.5
[1] 1 2 3 4 5
> r.6
[1] 1 2 3 4 5 6
>
```

可以看到塞入的字串變成物件的名稱

```
for(i in 1:6) {
    nam = paste("r", i, sep = ".")
    assign(nam, 1:i)
}
ls(pattern = "^r..$")
```

r.1
r.2
r.3
r.4
r.5
r.6

R 自動建立變數與物件 - assign

另外用 字串向量來測

```
> kan = c( "t1", "t2", "t3", "t4")
> assign( kan, 1:4)
Warning message:
In assign(kan, 1:4) : 只用了第一個元素做為變數名
> ls()
 [1] "i"   "kan" "nam" "r.1" "r.2" "r.3" "r.4" "r.5" "r.6" "t1"
> t1
[1] 1 2 3 4
>
```

只能抓第一個字串值，所以要大量建立還是需要 for 來處理。

```
kan = c( "t1", "t2", "t3", "t4")
assign( kan, 1:4)
ls()
t1
```

R 自動建立變數與物件 eval & assign

eval 可以將所要輸入的 R 指令用字串的方式執行。

可以將字串執行成 R 的指令，這個東西可以跟 assign() 一起玩。

在此拿 iris 資料集來做實驗，用 names() 抓出直行名稱。

colna = names(iris)
colna

```
> colna = names(iris)
> colna
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"
[5] "Species"
> 
```

R 自動建立變數與物件 eval & assign

eval 可以將所要輸入的 R 指令用字串的方式執行。

再用 for 跟 paste0() 與 前面建立的 colna 資料 建立起所要執行的字串。

```
for( i in 1:ncol(iris)){
   comds = paste0( "iris$", colna[i])
   print(comds)
}
```

```
> for( i in 1:ncol(iris)){
+ comds = paste0( "iris$", colna[i])
+ print(comds)
+ }
[1] "iris$Sepal.Length"
[1] "iris$Sepal.Width"
[1] "iris$Petal.Length"
[1] "iris$Petal.Width"
[1] "iris$Species"
>
```

R 自動建立變數與物件 eval & assign

測試 eval()。

```
for( i in 1:ncol(iris)){
   comds = paste0( "iris$", colna[i])
   evalp = eval(parse(text = comds))
   print(head(evalp))
}

head(iris)
```

```
> for( i in 1:ncol(iris)){
+ comds = paste0( "iris$", colna[i])
+ evalp = eval(parse(text = comds))
+ print(head(evalp))
+ }
[1] 5.1 4.9 4.7 4.6 5.0 5.4
[1] 3.5 3.0 3.2 3.1 3.6 3.9
[1] 1.4 1.4 1.3 1.5 1.4 1.7
[1] 0.2 0.2 0.2 0.2 0.2 0.4
[1] setosa setosa setosa setosa setosa setosa
Levels: setosa versicolor virginica
>
> head(iris)
  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
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
>
```

R 自動建立變數與物件 eval & assign

再來跟 assign() 一起玩，而建立的物件名稱則是 前面抓出的直行名稱

先用 ls() 看目前建立的物件

執行完後再用 ls() 觀察

```
ls()

for( i in 1:ncol(iris)){
   comds = paste0( "iris$", colna[i])
   evalp = eval(parse(text = comds))
   objna = colna[i]
   assign( objna,
      evalp
   , env = .GlobalEnv)
}

ls()
```

```
> ls()
[1] "colna" "comds" "evalp" "i"
>
> for( i in 1:ncol(iris)){
+ comds = paste0( "iris$", colna[i])
+ evalp = eval(parse(text = comds))
+ objna = colna[i]
+ assign( objna,
+ evalp
+ , env = .GlobalEnv)
+ }
>
> ls()
 [1] "colna"        "comds"        "evalp"        "i"
 [5] "objna"        "Petal.Length" "Petal.Width"  "Sepal.Length"
 [9] "Sepal.Width"  "Species"
>
```

R 自動建立變數與物件 eval & assign

head(iris)
head(Petal.Length, 10)
head(Petal.Width, 10)
head(Sepal.Length, 10)
head(Sepal.Width, 10)
head(Species, 10)

```
> head( iris)
  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
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
> head( Petal.Length, 10)
 [1] 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5
> head( Petal.Width, 10)
 [1] 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1
> head( Sepal.Length, 10)
 [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9
> head( Sepal.Width, 10)
 [1] 3.5 3.0 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1
> head( Species, 10)
 [1] setosa setosa setosa setosa setosa setosa setosa setosa setosa setosa
[10] setosa
Levels: setosa versicolor virginica
> |
```

R 排序 - sort

處理 TEJ 資料庫的上市公司每月報酬率資料

```
# 資料來源的工作目錄
setwd("C:/rtej")
getwd()
# 匯入 CSV function
tmprt = function(rtcsv){
read.csv( rtcsv, stringsAsFactors = FALSE)
}
#輸出 CSV function
wrta = function(xdo , ycsv)
{
    write.table( xdo, file= ycsv, quote = FALSE, sep = ",",
row.names = FALSE)
}
# 觀察 工作目錄的所有 File
dir()
# 匯入檔案
tejdata = tmprt("tej.csv")
# 檢視該檔案的前三橫行
head( tejdata, 3)
```

	A	B	C	D	E	F	G	H
1	年月	2016/10/31	2016/9/30	2016/8/31	2016/7/29	2016/6/30	2016/5/31	2016/4/29
2	1101 台泥	6.1712	-1.6551	6.9321	9.8977	6.6553	-8.6626	4.4446
3	1102 亞泥	0.5477	-3.3509	0.8563	4.6594	12.2737	-13.5652	-2.2109
4	1103 嘉泥	-0.8891	-0.3323	2.3708	6.8264	-0.1196	-8.833	-2.8603
5	1104 環泥	0.0001	1.4461	12.8206	3.6336	9.2269	-2.9055	-7.6063
6	1108 幸福	-1.7315	-0.8585	-0.6398	1.1634	-0.0001	-0.4926	-0.9756
7	1109 信大	-2.4389	-4.2055	5.4186	3.5198	-0.4975	0.5	0.7049
8	1110 東泥	-2.5807	0.9773	0.3269	1.9824	-2.5642	0	2.9702
9	1201 味全	0.5617	-7.5324	-3.99	-4.9764	3.9408	1.7543	0.7577
10	1203 味王	0.2278	2.8211	2.5228	0.4609	-0.2299	1.1625	-0.2319
11	1210 大成	6.0884	0.5567	2.1757	-8.1494	14.1471	17.8083	5.2885
12	1213 大飲	5.9998	-6.2501	-6.7055	-0.9712	-10.1415	21.8391	-5.1769
13	1215 卜蜂	19.0729	-2.9563	5.1351	4.7972	16.0935	14.4903	12.7018
14	1216 統一	3.9115	-1.9998	-5.0769	2.6772	2.2541	6.7011	3.009
15	1217 愛之味	-4.1984	-0.5249	-1.7159	0.0001	-0.6091	2.2417	-0.6189
16	1218 泰山	4.9843	6.2914	11.852	2.2728	-0.7518	9.9175	-2.8113
17	1219 福壽	0.6514	0.3268	-0.0001	2.3026	0.3301	0.331	-1.3072

R 排序 - sort

```
# 將直行名稱存起來
tejcol = names(tejdata)
# 由第一個橫列是中文，自己習慣先處理成英文，所以
先將存出來的第一個值換成 year
tejcol[1] = "year"
# 將直行名稱塞回去
colnames(tejdata) = tejcol
# 將第一行去掉
tejdata = tejdata[, 2:ncol(tejdata)]
# 檢視處理後的結果(前三橫行)
head( tejdata, 3)
# 測試 sort 函數
# 小至大排列，其預設引數 decreasing = FALSE
sort( tejdata$X2016.10.31)
# 大至小排列
sort( tejdata$X2016.10.31, decreasing = TRUE)
```

	A	B	C	D	E	F	G	H
1	年月	2016/10/31	2016/9/30	2016/8/31	2016/7/29	2016/6/30	2016/5/31	2016/4/29
2	1101 台泥	6.1712	-1.6551	6.9321	9.8977	6.6553	-8.6626	4.4446
3	1102 亞泥	0.5477	-3.3509	0.8563	4.6594	12.2737	-13.5652	-2.2109
4	1103 嘉泥	-0.8891	-0.3323	2.3708	6.8264	-0.1196	-8.833	-2.8603
5	1104 環泥	0.0001	1.4461	12.8206	3.6336	9.2269	-2.9055	-7.6063
6	1108 幸福	-1.7315	-0.8585	-0.6398	1.1634	-0.0001	-0.4926	-0.9756
7	1109 信大	-2.4389	-4.2055	5.4186	3.5198	-0.4975	0.5	0.7049
8	1110 東泥	-2.5807	0.9773	0.3269	1.9824	-2.5642	0	2.9702
9	1201 味全	0.5617	-7.5324	-3.99	-4.9764	3.9408	1.7543	0.7577
10	1203 味王	0.2278	2.8211	2.5228	0.4609	-0.2299	1.1625	-0.2319
11	1210 大成	6.0884	0.5567	2.1757	-8.1494	14.1471	17.8083	5.2885
12	1213 大飲	5.9998	-6.2501	-6.7055	-0.9712	-10.1415	21.8391	-5.1769
13	1215 卜蜂	19.0729	-2.9563	5.1351	4.7972	16.0935	14.4903	12.7018
14	1216 統一	3.9115	-1.9998	-5.0769	2.6772	2.2541	6.7011	3.009
15	1217 愛之味	-4.1984	-0.5249	-1.7159	0.0001	-0.6091	2.2417	-0.6189
16	1218 泰山	4.9843	6.2914	11.852	2.2728	-0.7518	9.9175	-2.8113
17	1219 福壽	0.6514	0.3268	-0.0001	2.3026	0.3301	0.331	-1.3072

R 排序 - sort

程式 - 處理 的 自訂函數

1. dt 為 data.frame 的物件名稱
2. chrn 之後要輸出建立 DF 所要命名的字串

worksort(data.frame名稱, 命名的字串)

```
worksort = function(dt ,chrn) {
  # 根據匯入 data.frame 的建立一個空的直行，並進行序列編號
  sortdf = data.frame( nm = 1:rep(nrow(dt)))
  # 將匯入 data.frame 的 個直行名稱，存起來
  dtna = names(dt)
  # 根據 data.frame 的直行數量來進行處理
  for( sorwn in 1:ncol(dt)){
    # 之後執行的 R命令 ，用 eval() 前的字串處理
    cmds = paste0(chrn,"$", dtna[sorwn])
    # eval() 匯入所要執行的字串，作為命令執行
    evaltem = eval(parse(text = cmds))
    # 執行由小至大排列的排序，其 NA 也要進行處理，引數為 na.last
    # 執行後 數字 小至大排列 ， NA 值 置最後。
    temscl = sort( evaltem, decreasing = FALSE, na.last= TRUE)
    # 合併 處理後的結果至 sortdf
    sortdf = cbind(sortdf, temscl)
  }
  # 將匯入存起來 data.frame 的 個直行名稱，前面在塞入序列直行的名稱值
  sortdfcol = c( "nm", dtna)
  # 將處理過的直行名稱塞回去
  colnames(sortdf) = sortdfcol
  # 處理一開始匯進來所要命名的字串
  sortnm = paste0( chrn,"sortproc")
  # 使用 assign 建立物件，env = .GlobalEnv 則會將其定為全域
  assign( sortnm,
    sortdf
  , env = .GlobalEnv)
}
```

R 排序 - sort

程式 - 處理 的 自訂函數

1. dt 為 data.frame 的物件名稱
2. chrn 之後要輸出建立 DF 所要命名的字串

worksort(data.frame名稱, 命名的字串)

	A	B	C	D	E	F	G	H	I	J
1	nm	X2016.10.31	X2016.9.30	X2016.8.31	X2016.7.29	X2016.6.30	X2016.5.31	X2016.4.29	X2016.3.31	X2016.2.26
2	1	-73.494	-45.8069	-42.3934	-28.1189	-28.6021	-49.9999	-56.2257	-45.5041	-36.6667
3	2	-65.2366	-34.8403	-34.4261	-25	-28.5012	-44.6809	-40.0578	-35.3094	-30
4	3	-38.8524	-33.1325	-31.8447	-21.5684	-27.6	-35.0961	-39.7893	-32.8167	-27.0868
5	4	-30.2754	-29.7044	-29.4986	-19.6348	-26.9096	-32.4445	-33.2723	-29.6037	-25.3846
6	5	-28.983	-27.6023	-23.7027	-18.0392	-22.3433	-31.7618	-30.4393	-23.342	-24.7788
7	6	-27.9068	-27.3869	-21.9298	-16.6975	-19.6498	-28.4931	-29	-22.7501	-23.7793
8	7	-27.087	-26.384	-21.8217	-15.9794	-16.9558	-28.0001	-28.1045	-21.6981	-17.3002
9	8	-24.9999	-26.3118	-21.3286	-15.9532	-16.7591	-23.2516	-27.0269	-20.9878	-16.5451
10	9	-23.8335	-26.0811	-21.0564	-15.2123	-16.6666	-22.5923	-25.4006	-20.5357	-16.4475
11	10	-23.8271	-20.8428	-20.7595	-14.9322	-16.508	-22.171	-25.3522	-20.0218	-16.3771
12	11	-23.8	-19.6291	-19.9753	-14.7766	-16.1185	-21.4912	-24.5599	-19.6541	-16.0526
13	12	-22.7272	-19.3716	-19.9463	-14.2214	-15.8751	-21.2903	-24.5345	-19.4595	-15.7896
14	13	-22.1054	-19.0084	-19.4726	-14.0844	-15.8293	-21.282	-24.3672	-17.7992	-15.4009

R 樞紐分析 - xtab

xtab 可以產生類似於 Excel 樞紐分析功能

xtabs {stats} R Documentation

Cross Tabulation

Description

Create a contingency table (optionally a sparse matrix) from cross-classifying factors, usually contained in a data frame, using a formula interface.

Usage

```
xtabs(formula = ~., data = parent.frame(), subset, sparse = FALSE,
      na.action, addNA = FALSE, exclude = if(!addNA) c(NA, NaN),
      drop.unused.levels = FALSE)

## S3 method for class 'xtabs'
print(x, na.print = "", ...)
```

R 樞紐分析 - xtab

UCBAdmissions 該資料集為加州大學伯克利分校的學生招生狀況，為 1973 年根據入學和性別分類的伯克利研究生，在最大的六個部門中所提供的綜合數據。這個資料集常用於說明辛普森的悖論，表明入學做法中存在性別偏見。由 3 個變量對 4526 個觀測值進行交叉列表所得到的 3 維資料。

UCBAdmissions {datasets} R Documentation

Student Admissions at UC Berkeley

Description

Aggregate data on applicants to graduate school at Berkeley for the six largest departments in 1973 classified by admission and sex.

Usage

```
UCBAdmissions
```

R 樞紐分析 - xtab

DF = as.data.frame(UCBAdmissions)
head(DF)
xtabs(Freq ~ Gender + Admit, DF)
xtabs(Freq ~ ., DF)
summary(xtabs(Freq ~ ., DF))

```
> DF = as.data.frame(UCBAdmissions)
> head(DF)
     Admit Gender Dept Freq
1 Admitted   Male    A  512
2 Rejected   Male    A  313
3 Admitted Female    A   89
4 Rejected Female    A   19
5 Admitted   Male    B  353
6 Rejected   Male    B  207
```

```
> summary(xtabs(Freq ~ ., DF))
Call: xtabs(formula = Freq ~ ., data = DF)
Number of cases in table: 4526
Number of factors: 3
Test for independence of all factors:
        Chisq = 2000.3, df = 16, p-value = 0
>
```

```
> xtabs(Freq ~ ., DF)
, , Dept = A

          Gender
Admit      Male Female
  Admitted  512     89
  Rejected  313     19

, , Dept = B

          Gender
Admit      Male Female
  Admitted  353     17
  Rejected  207      8

, , Dept = C

          Gender
Admit      Male Female
  Admitted  120    202
  Rejected  205    391

, , Dept = D

          Gender
Admit      Male Female
  Admitted  138    131
  Rejected  279    244
```

R 區間與因素表 - cut & table

cut

From base v3.4.1
by R-core R-core@R-project.org

Convert Numeric To Factor

`cut` divides the range of `x` into intervals and codes the values in `x` according to which interval they fall. The leftmost interval corresponds to level one, the next leftmost to level two and so on.

Keywords category

Usage

```
cut(x, …)

# S3 method for default
cut(x, breaks, labels = NULL,
    include.lowest = FALSE, right = TRUE, dig.lab = 3,
    ordered_result = FALSE, …)
```

table {base} R Documentation

Cross Tabulation and Table Creation

Description

`table` uses the cross-classifying factors to build a contingency table of the counts at each combination of factor levels.

Usage

```
table(...,
      exclude = if (useNA == "no") c(NA, NaN),
      useNA = c("no", "ifany", "always"),
      dnn = list.names(...), deparse.level = 1)

as.table(x, ...)
is.table(x)

## S3 method for class 'table'
as.data.frame(x, row.names = NULL, ...,
              responseName = "Freq", stringsAsFactors = TRUE,
              sep = "", base = list(LETTERS))
```

R 區間與因素表 - cut & table

建立常態的 50 值資料，並用 cut 這個函數進行切割，從 class 函數可以知道該物件的類別。

```
> (x = rnorm(50))
 [1]  0.25783390  0.50859737  0.74929317 -0.21216633  0.63626661  0.81522757
 [7] -0.25970231  0.48666863  0.66095111 -0.42823595 -1.36847530  0.95710717
[13]  0.73058852 -0.26119485  0.75676985 -0.63206127 -1.88625642  0.65528790
[19]  1.34165224  1.07576504  0.46967153 -0.79234343  0.99210584 -0.81089793
[25]  1.66694447 -1.04313482  0.07823729 -0.61247239  0.13621472 -0.22929644
[31]  0.57409594  1.17243935  0.83961342  1.20419538  0.75066388  1.39201052
[37]  0.02034245 -0.20049307 -0.49139039  0.18725477  1.61524164  0.06449300
[43] -0.17375994 -0.74878524  0.23645221 -0.84587919 -0.11358689 -2.43564240
[49]  1.00720792 -0.13474380
> (x.cut1 = cut( x, breaks = -5:5))
 [1] (0,1]   (0,1]   (0,1]   (-1,0]  (0,1]   (0,1]   (-1,0]  (0,1]   (0,1]
[10] (-1,0]  (-2,-1] (0,1]   (0,1]   (-1,0]  (0,1]   (-1,0]  (-2,-1] (0,1]
[19] (1,2]   (1,2]   (0,1]   (-1,0]  (0,1]   (-1,0]  (1,2]   (-2,-1] (0,1]
[28] (-1,0]  (0,1]   (-1,0]  (0,1]   (1,2]   (0,1]   (1,2]   (0,1]   (1,2]
[37] (0,1]   (-1,0]  (-1,0]  (0,1]   (1,2]   (0,1]   (-1,0]  (-1,0]  (0,1]
[46] (-1,0]  (-1,0]  (-3,-2] (1,2]   (-1,0]
10 Levels: (-5,-4] (-4,-3] (-3,-2] (-2,-1] (-1,0] (0,1] (1,2] ... (4,5]
> class(x)
[1] "numeric"
> class(x.cut1)
[1] "factor"
>
```

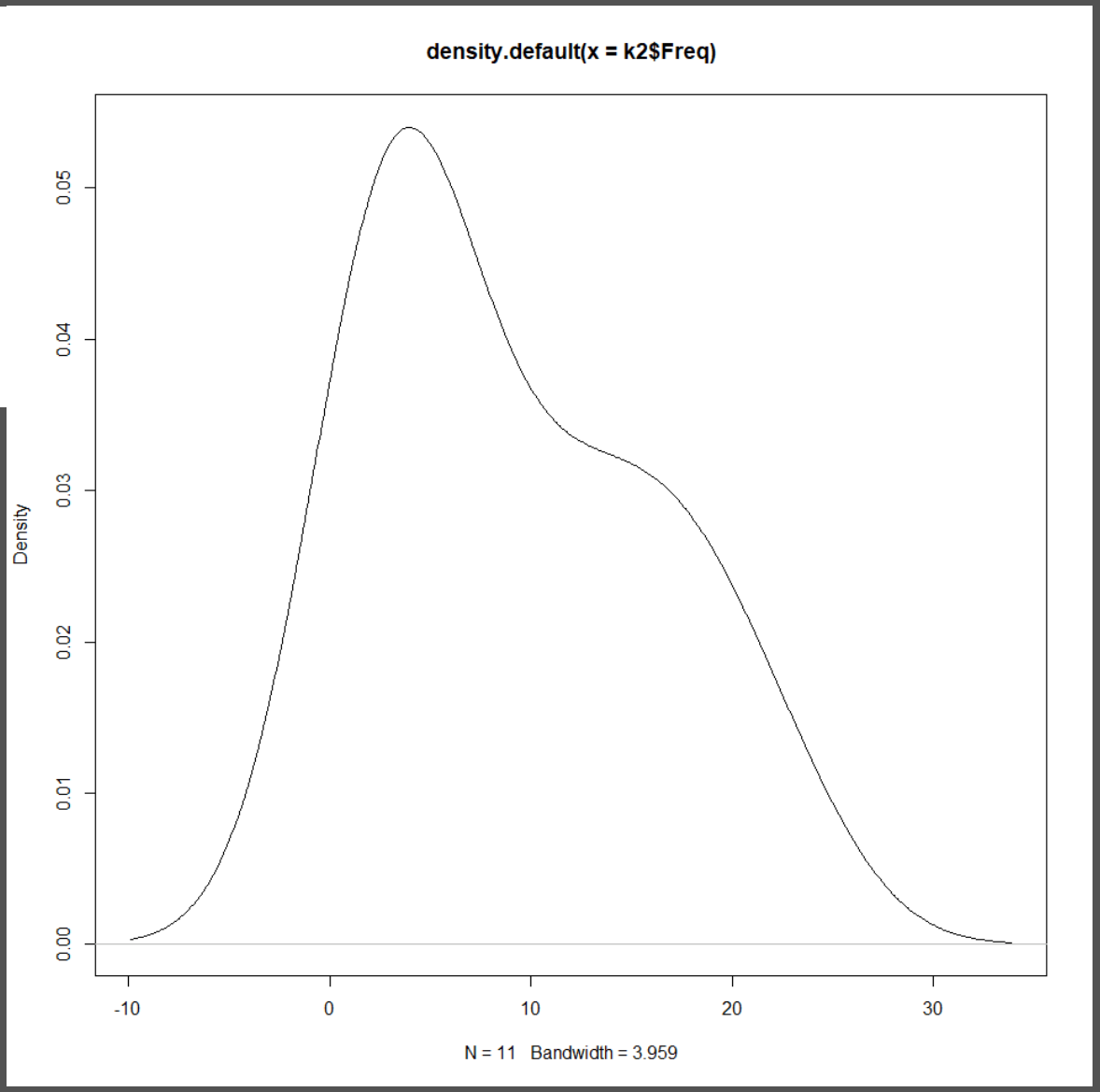
R 區間與因素表 - cut & table

在此建立一筆 10000 值的隨機常態資料，然後用 table 函數進行觀察。

```
> Z = stats::rnorm(10000)
> head(Z)
[1]  0.5549906 -1.1641565  1.2202559  0.8979036 -0.1663917 -1.8936824
> tail(Z)
[1] -0.2383977  0.7651919 -0.2826311 -0.4379089 -0.3635988  2.4769229
> (zt = table(cut(Z, breaks = -6:6)))

(-6,-5] (-5,-4] (-4,-3] (-3,-2] (-2,-1]  (-1,0]   (0,1]   (1,2]   (2,3]
      0       1      15     238    1329    3387    3449    1343     222
  (3,4]   (4,5]   (5,6]
     16       0       0
>
```

單純用 table 來做範例，在此用 rpois 建立 100 個隨機值，根據 lamdba = 5 的 Poisson 分佈，並產生密度的繪圖，圖整體趨勢為鐘形。

```
> class(rpois(100, 5))
[1] "integer"
> k1 = table(rpois(100, 5))
> k2 = as.data.frame(table(rpois(100, 5)))
> plot(density(k2$Freq))
>
```



R 區間與因素表 - cut & table

```
(x = rnorm(50))
(x.cut1 = cut( x, breaks = -5:5))

class(x)
class(x.cut1)

Z = stats::rnorm(10000)
head(Z)
tail(Z)
(zt = table(cut(Z, breaks = -6:6)))

class(rpois(100, 5))
k1 = table(rpois(100, 5))
k2 = as.data.frame(table(rpois(100, 5)))
plot(density(k2$Freq))
```

R 自動產生資料集 - expand.table & epitools

根據 Array 來產生 data.frame 的資料集，屬於名為 epitools 套件下的函數

RDocumentation | Search for packages, func | R package | Leaderboard | Sign in

epitools v0.5-9

Other versions | 1,696 Monthly downloads | 79th Percentile

by Adam Omidpanah | https://www.rdocumentation.org/packages/epitools | Copy

Epidemiology Tools

Tools for training and practicing epidemiologists including methods for two-way and multi-way contingency tables.

ageadjust.indirect	Age standardization by indirect method, with exact confidence intervals
expand.table	Expand contingency table into individual-level data set
expected	Expected values in a table
pois.conf.int	Confidence intervals for Poisson counts or rates

Expand contingency table into individual-level data set

Description

Expands contingency table or array into individual-level data set.

Usage

```
expand.table(x)
```

Arguments

x
table or array with `dimnames(x)` and `names(dimnames(x))`

R 自動產生資料集 - expand.table & epitools


```
> install.packages("epitools")
Installing package into 'D:/USERDATA/Documents/R/win-library/3.4'
(as 'lib' is unspecified)
嘗試 URL 'https://cloud.r-project.org/bin/windows/contrib/3.4/epitools_0.5-9$
Content type 'application/zip' length 233317 bytes (227 KB)
downloaded 227 KB

package 'epitools' successfully unpacked and MD5 sums checke

The downloaded binary packages are in
        C:\Users\USER\AppData\Local\Temp\RtmpU3QDfx\downloade
>
```

```
> # 範例資料
> survey = array(0, dim=c(3, 2, 1))
> survey[,1,1] = c(2, 0, 1)
> survey[,2,1] = c(3, 2, 4)
> Satisfactory = c("Good", "Fair", "Bad")
> Sex = c("Female", "Male")
> Times = c("First")
> dimnames(survey) = list(Satisfactory, Sex, Times)
> names(dimnames(survey)) = c("Satisfactory", "Sex", "Times")
> survey
, , Times = First

             Sex
Satisfactory Female Male
        Good      2    3
        Fair      0    2
        Bad       1    4

>
```

```
# install.pckages("epitools")
# 範例資料
survey = array(0, dim=c(3, 2, 1))
survey[,1,1] = c(2, 0, 1)
survey[,2,1] = c(3, 2, 4)
Satisfactory = c("Good", "Fair", "Bad")
Sex = c("Female", "Male")
Times = c("First")
dimnames(survey) = list(Satisfactory, Sex, Times)
names(dimnames(survey)) = c("Satisfactory", "Sex", "Times")
survey
```

R 自動產生資料集 - expand.table & epitools


```
> library("epitools")
> (survey.ex = expand.table(survey))
   Satisfactory    Sex Times
1          Good Female First
2          Good Female First
3          Good   Male First
4          Good   Male First
5          Good   Male First
6          Fair   Male First
7          Fair   Male First
8           Bad Female First
9           Bad   Male First
10          Bad   Male First
11          Bad   Male First
12          Bad   Male First
> 
```

```
> class(survey)
[1] "array"
> class(survey.ex)
[1] "data.frame"
> 
```

```
# install.packages("epitools")
library("epitools")
(survey.ex = expand.table(survey))

class(survey)
class(survey.ex)
```

R 多向量轉單一向量進行處理 - stack & unstack

stack

From utils v3.4.1
by R-core R-core@R-project.org

Stack Or Unstack Vectors From A Data Frame Or List

Stacking vectors concatenates multiple vectors into a single vector along with a factor indicating where each observation originated. Unstacking reverses this operation.

Keywords manip

Usage

```
stack(x, ...)
# S3 method for default
stack(x, drop=FALSE, ...)
# S3 method for data.frame
stack(x, select, drop=FALSE, ...)

unstack(x, ...)
# S3 method for default
unstack(x, form, ...)
# S3 method for data.frame
unstack(x, form, ...)
```

在此先建立一組測試的範例資料集

```
> test.data = data.frame(
+ v1 = c( 12, 15, 19, 22, 15),
+ v2 = c( 18, 12, 42,29, 44),
+ v3 = c( 8, 17, 22, 19, 31))
> dim(test.data)
[1] 5 3
> test.data
  v1 v2 v3
1 12 18  8
2 15 12 17
3 19 42 22
4 22 29 19
5 15 44 31
>
```

R 多向量轉單一向量進行處理 - stack & unstack

stack 函數可以將原先多個直行向量，轉換成單一向量進行標示。

當然也可以根據 select 參數，來指定所要的向量。

```
> sktd1 = stack(test.data)
> dim(sktd1)
[1] 15  2
> sktd1
   values ind
1      12  v1
2      15  v1
3      19  v1
4      22  v1
5      15  v1
6      18  v2
7      12  v2
8      42  v2
9      29  v2
10     44  v2
11      8  v3
12     17  v3
13     22  v3
14     19  v3
15     31  v3
>
```

```
> sktd2 = stack(test.data, select = c( v2, v3))
> dim(sktd2)
[1] 10  2
> sktd2
   values ind
1      18  v2
2      12  v2
3      42  v2
4      29  v2
5      44  v2
6       8  v3
7      17  v3
8      22  v3
9      19  v3
10     31  v3
>
```

R 多向量轉單一向量進行處理 - stack & unstack

在此確定沒有任何 v1 的標記。

```
> sktd2$ind == "v1"
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
>
```

unstack 可以反轉 stack 的處理。

```
> usktd = unstack( sktd2, values ~ ind)
> dim(usktd)
[1] 5 2
> class(usktd)
[1] "data.frame"
> usktd
  v2 v3
1 18  8
2 12 17
3 42 22
4 29 19
5 44 31
>
```

R 多向量轉單一向量進行處理 - stack & unstack


```
test.data = data.frame(
      v1 = c( 12, 15, 19, 22, 15),
      v2 = c( 18, 12, 42,29, 44),
      v3 = c( 8, 17, 22, 19, 31))
dim(test.data)
test.data
sktd1 = stack(test.data)
dim(sktd1)
sktd1
sktd2 = stack(test.data, select = c( v2, v3))
dim(sktd2)
sktd2
sktd2$ind == "v1"
usktd = unstack( sktd2, values ~ ind)
dim(usktd)
class(usktd)
usktd
```

R 分割與合併資料 - split & unsplit

base v3.4.1 Other versions

by R-core R-core@R-project.org https://www.rdocumentation.org/packages/base Copy

The R Base Package

Base R functions.

From base v3.4.1
by R-core R-core@R-project.org

split

Divide Into Groups And Reassemble

`split` divides the data in the vector `x` into the groups defined by `f`. The replacement forms replace values corresponding to such a division. `unsplit` reverses the effect of `split`.

Keywords category

Usage

```
split(x, f, drop = FALSE, ...)
# S3 method for default
split(x, f, drop = FALSE, sep = ".", lex.order = FALSE, ...)

split(x, f, drop = FALSE, ...) <- value
unsplit(value, f, drop = FALSE)
```

R 分割與合併資料 - split & unsplit

在此建立 data.table 資料做為範例，而 split 則是根據 x1 與 x2 拆成 list 的資料。

```
> # data.table
> set.seed(929)
> dt = data.table(x1 = rep(letters[1:2], 6),
+                 x2 = rep(letters[3:5], 4),
+                 x3 = rep(letters[5:8], 3),
+                 y = rnorm(12))
> (split(dt, by=c("x1", "x2")))
$a.c
   x1 x2 x3         y
1:  a  c  e 0.1019624
2:  a  c  g 0.8063666

$b.d
   x1 x2 x3          y
1:  b  d  f -0.9435405
2:  b  d  h -1.0287596

$a.e
   x1 x2 x3          y
1:  a  e  g -3.1990147
2:  a  e  e  0.3332565
```

flatten 的這個參數則是會將資料做更獨立的細分。

```
> (split(dt, by=c("x1", "x2"), flatten=FALSE))
$a
$a$c
   x1 x2 x3         y
1:  a  c  e 0.1019624
2:  a  c  g 0.8063666

$a$e
   x1 x2 x3          y
1:  a  e  g -3.1990147
2:  a  e  e  0.3332565

$a$d
   x1 x2 x3          y
1:  a  d  e  0.3988211
2:  a  d  g -0.5583995
```

R 分割與合併資料 - split & unsplit

將前面 data.table 的資料轉成 data.frame 的型態。

```
> # data.frame
> (df = as.data.frame(dt))
   x1 x2 x3           y
1   a  c  e  0.10196237
2   b  d  f -0.94354049
3   a  e  g -3.19901472
4   b  c  h  1.15272531
5   a  d  e  0.39882112
6   b  e  f -0.72804678
7   a  c  g  0.80636658
8   b  d  h -1.02875961
9   a  e  e  0.33325647
10  b  c  f -0.06946328
11  a  d  g -0.55839954
12  b  e  h  0.24296892
```

如果用 data.frame 進行 split，其 R 的寫法如下。

```
> (sdf = split(df, list(df$x1, df$x2)))
$a.c
  x1 x2 x3          y
1  a  c  e 0.1019624
7  a  c  g 0.8063666

$b.c
   x1 x2 x3           y
4   b  c  h  1.15272531
10  b  c  f -0.06946328

$a.d
   x1 x2 x3           y
5   a  d  e  0.3988211
11  a  d  g -0.5583995

$b.d
  x1 x2 x3           y
2  b  d  f -0.9435405
8  b  d  h -1.0287596
```

R 分割與合併資料 - split & unsplit

可以使用 unsplit 反轉 split 的處理。

```
> # split & unsplit
> n = 10
> (edu = factor(sample(1:4, n, replace=T)))
 [1] 4 1 1 2 3 3 4 3 4 1
Levels: 1 2 3 4
> (score = sample( 0:100, n))
 [1] 48 65 13 62 67 96 42  2  6 59
> usdf = cbind(edu, score)
> score.edu = split(score, edu)
> score.edu
$`1`
[1] 65 13 59

$`2`
[1] 62

$`3`
[1] 67 96  2

$`4`
[1] 48 42  6

> class(score.edu)
[1] "list"
>
```

```
> unsplit(score.edu, edu)
 [1] 48 65 13 62 67 96 42  2  6 59
> sort(edu)
 [1] 1 1 1 2 3 3 3 4 4 4
Levels: 1 2 3 4
> (us.data = unsplit(score.edu, sort(edu)))
 [1] 65 13 59 62 67 96  2 48 42  6
> class(us.data)
[1] "integer"
>
```

```
set.seed(929)
dt = data.table(x1 = rep(letters[1:2], 6),
                x2 = rep(letters[3:5], 4),
                x3 = rep(letters[5:8], 3),
                y = rnorm(12))
(split(dt, by=c("x1", "x2")))
(split(dt, by=c("x1", "x2"), flatten=FALSE))
(df = as.data.frame(dt))
(sdf = split(df, list(df$x1, df$x2)))
n = 10
(edu = factor(sample(1:4, n, replace=T)))
(score = sample( 0:100, n))
usdf = cbind(edu, score)
score.edu = split(score, edu)
score.edu
class(score.edu)
unsplit(score.edu, edu)
sort(edu)
(us.data = unsplit(score.edu, sort(edu)))
class(us.data)
```

R 快速處理 data.frame 資料 - by & with

colSums()、rowSums()、colMeans()、rowMeans()
colSums()、rowSums()、colMeans()、rowMeans() 可以針對 data.frame 進行快速的處理，在此使用 iris 的資料當作範例。

```
> head(iris)
  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
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
> head(iris[ , 1:4])
  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
3          4.7         3.2          1.3         0.2
4          4.6         3.1          1.5         0.2
5          5.0         3.6          1.4         0.2
6          5.4         3.9          1.7         0.4
```

R 快速處理 data.frame 資料 - by & with

由於第五列為文字的向量，所以要處理掉 iris[, 1:4]

```
> colSums (iris[ , 1:4])
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
       876.5         458.6         563.7         179.9
> rowSums (iris[ , 1:4])
  [1] 10.2  9.5  9.4  9.4 10.2 11.4  9.7 10.1  8.9  9.6 10.8 10.0  9.3  8.5 11.2
 [16] 12.0 11.0 10.3 11.5 10.7 10.7 10.7  9.4 10.6 10.3  9.8 10.4 10.4 10.2  9.7
 [31]  9.7 10.7 10.9 11.3  9.7  9.6 10.5 10.0  8.9 10.2 10.1  8.4  9.1 10.7 11.2
 [46]  9.5 10.7  9.4 10.7  9.9 16.3 15.6 16.4 13.1 15.4 14.3 15.9 11.6 15.4 13.2
 [61] 11.5 14.6 13.2 15.1 13.4 15.6 14.6 13.6 14.4 13.1 15.7 14.2 15.2 14.8 14.9
 [76] 15.4 15.8 16.4 14.9 12.8 12.8 12.6 13.6 15.4 14.4 15.5 16.0 14.3 14.0 13.3
 [91] 13.7 15.1 13.6 11.6 13.8 14.1 14.1 14.7 11.7 13.9 18.1 15.5 18.1 16.6 17.5
[106] 19.3 13.6 18.3 16.8 19.4 16.8 16.3 17.4 15.2 16.1 17.2 16.8 20.4 19.5 14.7
[121] 18.1 15.3 19.2 15.7 17.8 18.2 15.6 15.8 16.9 17.6 18.2 20.1 17.0 15.7 15.7
[136] 19.1 17.7 16.8 15.6 17.5 17.8 17.4 15.5 18.2 18.2 17.2 15.7 16.7 17.3 15.8
> colMeans(iris[ , 1:4])
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
    5.843333     3.057333     3.758000     1.199333
> rowMeans(iris[ , 1:4])
  [1] 2.550 2.375 2.350 2.350 2.550 2.850 2.425 2.525 2.225 2.400 2.700 2.500 2.325
 [14] 2.125 2.800 3.000 2.750 2.575 2.875 2.675 2.675 2.675 2.350 2.650 2.575 2.450
 [27] 2.600 2.600 2.550 2.425 2.425 2.675 2.725 2.825 2.425 2.400 2.625 2.500 2.225
 [40] 2.550 2.525 2.100 2.275 2.675 2.800 2.375 2.675 2.350 2.675 2.475 4.075 3.900
 [53] 4.100 3.275 3.850 3.575 3.975 2.900 3.850 3.300 2.875 3.650 3.300 3.775 3.350
 [66] 3.900 3.650 3.400 3.600 3.275 3.925 3.550 3.800 3.700 3.725 3.850 3.950 4.100
 [79] 3.725 3.200 3.200 3.150 3.400 3.850 3.600 3.875 4.000 3.575 3.500 3.325 3.425
 [92] 3.775 3.400 2.900 3.450 3.525 3.525 3.675 2.925 3.475 4.525 3.875 4.525 4.150
[105] 4.375 4.825 3.400 4.575 4.200 4.850 4.200 4.075 4.350 3.800 4.025 4.300 4.200
[118] 5.100 4.875 3.675 4.525 3.825 4.800 3.925 4.450 4.550 3.900 3.950 4.225 4.400
[131] 4.550 5.025 4.250 3.925 3.925 4.775 4.425 4.200 3.900 4.375 4.450 4.350 3.875
[144] 4.550 4.550 4.300 3.925 4.175 4.325 3.950
>
```

R 快速處理 data.frame 資料 - by & with

在此作為基準的欄位 iris$Species，為 factor 的屬性。

```
> head(iris$Species)
[1] setosa setosa setosa setosa setosa setosa
Levels: setosa versicolor virginica
> class(iris$Species)
[1] "factor"
>
```

所以可以根據 factor 的直欄屬性，進行 sd、mode、sum、mean 的求算。

```
> by(iris[,1] , iris$Species , sd)
iris$Species: setosa
[1] 0.3524897
------------------------------------------
iris$Species: versicolor
[1] 0.5161711
------------------------------------------
iris$Species: virginica
[1] 0.6358796
> by(iris[,1] , iris$Species , mode)
iris$Species: setosa
[1] "numeric"
------------------------------------------
iris$Species: versicolor
[1] "numeric"
------------------------------------------
iris$Species: virginica
[1] "numeric"
```

```
> by(iris[,1] , iris$Species , sum)
iris$Species: setosa
[1] 250.3
------------------------------------------
iris$Species: versicolor
[1] 296.8
------------------------------------------
iris$Species: virginica
[1] 329.4
> by(iris[,1] , iris$Species , mean)
iris$Species: setosa
[1] 5.006
------------------------------------------
iris$Species: versicolor
[1] 5.936
------------------------------------------
iris$Species: virginica
[1] 6.588
```

R 快速處理 data.frame 資料 - by & with

在此作為基準的欄位 iris$Species，為 factor 的屬性。

```
> by(iris[,1:4] , iris$Species , rowSums)
iris$Species: setosa
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
10.2  9.5  9.4  9.4 10.2 11.4  9.7 10.1  8.9  9.6 10.8 10.0  9.3  8.5 11.2 12.0
  17   18   19   20   21   22   23   24   25   26   27   28   29   30   31   32
11.0 10.3 11.5 10.7 10.7 10.7  9.4 10.6 10.3  9.8 10.4 10.4 10.2  9.7  9.7 10.7
  33   34   35   36   37   38   39   40   41   42   43   44   45   46   47   48
10.9 11.3  9.7  9.6 10.5 10.0  8.9 10.2 10.1  8.4  9.1 10.7 11.2  9.5 10.7  9.4
  49   50
10.7  9.9
------------------------------------------------------------
iris$Species: versicolor
  51   52   53   54   55   56   57   58   59   60   61   62   63   64   65   66
16.3 15.6 16.4 13.1 15.4 14.3 15.9 11.6 15.4 13.2 11.5 14.6 13.2 15.1 13.4 15.6
  67   68   69   70   71   72   73   74   75   76   77   78   79   80   81   82
14.6 13.6 14.4 13.1 15.7 14.2 15.2 14.8 14.9 15.4 15.8 16.4 14.9 12.8 12.8 12.6
  83   84   85   86   87   88   89   90   91   92   93   94   95   96   97   98
13.6 15.4 14.4 15.5 16.0 14.3 14.0 13.3 13.7 15.1 13.6 11.6 13.8 14.1 14.1 14.7
  99  100
11.7 13.9
------------------------------------------------------------
iris$Species: virginica
 101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116
18.1 15.5 18.1 16.6 17.5 19.3 13.6 18.3 16.8 19.4 16.8 16.3 17.4 15.2 16.1 17.2
 117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132
16.8 20.4 19.5 14.7 18.1 15.3 19.2 15.7 17.8 18.2 15.6 15.8 16.9 17.6 18.2 20.1
 133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148
17.0 15.7 15.7 19.1 17.7 16.8 15.6 17.5 17.8 17.4 15.5 18.2 18.2 17.2 15.7 16.7
 149  150
17.3 15.8
```

R 快速處理 data.frame 資料 - by & with


```
> by(iris[,1:4] , iris$Species , rowMeans)
iris$Species: setosa
    1     2     3     4     5     6     7     8     9    10
2.550 2.375 2.350 2.350 2.550 2.850 2.425 2.525 2.225 2.400
   14    15    16    17    18    19    20    21    22    23
2.125 2.800 3.000 2.750 2.575 2.875 2.675 2.675 2.675 2.350
   27    28    29    30    31    32    33    34    35    36
2.600 2.600 2.550 2.425 2.425 2.675 2.725 2.825 2.425 2.400
   40    41    42    43    44    45    46    47    48    49
2.550 2.525 2.100 2.275 2.675 2.800 2.375 2.675 2.350 2.675
------------------------------------------------------------
iris$Species: versicolor
   51    52    53    54    55    56    57    58    59    60
4.075 3.900 4.100 3.275 3.850 3.575 3.975 2.900 3.850 3.300
   64    65    66    67    68    69    70    71    72    73
3.775 3.350 3.900 3.650 3.400 3.600 3.275 3.925 3.550 3.800
   77    78    79    80    81    82    83    84    85    86
3.950 4.100 3.725 3.200 3.200 3.150 3.400 3.850 3.600 3.875
   90    91    92    93    94    95    96    97    98    99
3.325 3.425 3.775 3.400 2.900 3.450 3.525 3.525 3.675 2.925
------------------------------------------------------------
iris$Species: virginica
  101   102   103   104   105   106   107   108   109   110
4.525 3.875 4.525 4.150 4.375 4.825 3.400 4.575 4.200 4.850
  114   115   116   117   118   119   120   121   122   123
3.800 4.025 4.300 4.200 5.100 4.875 3.675 4.525 3.825 4.800
  127   128   129   130   131   132   133   134   135   136
3.900 3.950 4.225 4.400 4.550 5.025 4.250 3.925 3.925 4.775
  140   141   142   143   144   145   146   147   148   149
4.375 4.450 4.350 3.875 4.550 4.550 4.300 3.925 4.175 4.325
```

```
> by(iris[,1:4] , iris$Species , colSums)
iris$Species: setosa
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
       250.3        171.4         73.1         12.3
------------------------------------------------------------
iris$Species: versicolor
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
       296.8        138.5         213.0         66.3
------------------------------------------------------------
iris$Species: virginica
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
       329.4        148.7         277.6         101.3
```

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

R 快速處理 data.frame 資料 - by & with


```
head(iris)
head(iris[ , 1:4])
colSums (iris[ , 1:4])
rowSums (iris[ , 1:4])
colMeans(iris[ , 1:4])
rowMeans(iris[ , 1:4])

head(iris$Species)
class(iris$Species)

by(iris[,1] , iris$Species , sd)
by(iris[,1] , iris$Species , mode)
by(iris[,1] , iris$Species , sum)
by(iris[,1] , iris$Species , mean)

by(iris[,1:4] , iris$Species , rowSums)
by(iris[,1:4] , iris$Species , colSums)
by(iris[,1:4] , iris$Species , rowMeans)
by(iris[,1:4] , iris$Species , colMeans)
```

R 快速處理 data.frame 資料 - by & with

這個函數可以將所要的資料塞到指定的函數中進行處理。

```
> head(iris)
  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
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
> iris.lm = lm(Sepal.Length ~ Petal.Length, iris)
> summary(iris.lm)

Call:
lm(formula = Sepal.Length ~ Petal.Length, data = iris)

Residuals:
     Min       1Q   Median       3Q      Max
-1.24675 -0.29657 -0.01515  0.27676  1.00269

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   4.30660    0.07839   54.94   <2e-16 ***
Petal.Length  0.40892    0.01889   21.65   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4071 on 148 degrees of freedom
Multiple R-squared:   0.76,     Adjusted R-squared:  0.7583
F-statistic: 468.6 on 1 and 148 DF,  p-value: < 2.2e-16
```

R 快速處理 data.frame 資料 - by & with

用 with 的執行方式

```
> with(iris, {
+ iris.lm = lm(Sepal.Length ~ Petal.Length)
+ summary(iris.lm)})

Call:
lm(formula = Sepal.Length ~ Petal.Length)

Residuals:
     Min       1Q   Median       3Q      Max
-1.24675 -0.29657 -0.01515  0.27676  1.00269

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   4.30660    0.07839   54.94   <2e-16 ***
Petal.Length  0.40892    0.01889   21.65   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4071 on 148 degrees of freedom
Multiple R-squared:   0.76,     Adjusted R-squared:  0.7583
F-statistic: 468.6 on 1 and 148 DF,  p-value: < 2.2e-16
```

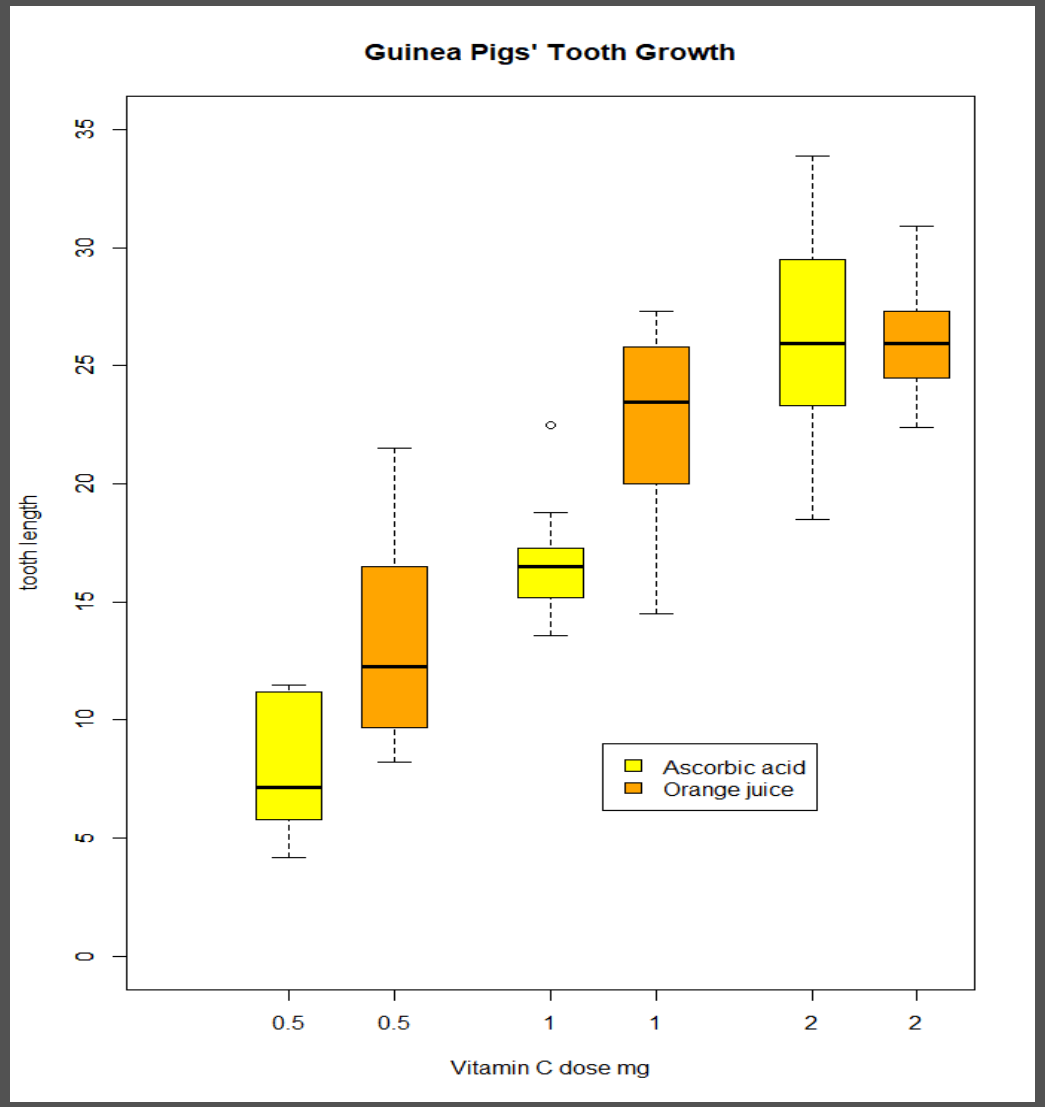
R 快速處理 data.frame 資料 - by & with

這裡執行文件中的範例，匯入 ToothGrowth 的資料，隨後執行函數中的 boxplot() 進行繪圖。

```
> head(ToothGrowth)
   len supp dose
1  4.2   VC  0.5
2 11.5   VC  0.5
3  7.3   VC  0.5
4  5.8   VC  0.5
5  6.4   VC  0.5
6 10.0   VC  0.5
> with(ToothGrowth, {
+     boxplot(len ~ dose, boxwex = 0.25, at = 1:3 - 0.2,
+             subset = (supp == "VC"), col = "yellow",
+             main = "Guinea Pigs' Tooth Growth",
+             xlab = "Vitamin C dose mg",
+             ylab = "tooth length", ylim = c(0, 35))
+     boxplot(len ~ dose, add = TRUE, boxwex = 0.25, at = 1:3 + 0.2,
+             subset = supp == "OJ", col = "orange")
+     legend(2, 9, c("Ascorbic acid", "Orange juice"),
+            fill = c("yellow", "orange"))
+ })
>
```

R 快速處理 data.frame 資料 - by & with


```
head(iris)
iris.lm = lm(Sepal.Length ~ Petal.Length, iris)
summary(iris.lm)

with(iris, {
iris.lm = lm(Sepal.Length ~ Petal.Length)
summary(iris.lm)})

head(ToothGrowth)
with(ToothGrowth, {
    boxplot(len ~ dose, boxwex = 0.25, at = 1:3 - 0.2,
            subset = (supp == "VC"), col = "yellow",
            main = "Guinea Pigs' Tooth Growth",
            xlab = "Vitamin C dose mg",
            ylab = "tooth length", ylim = c(0, 35))
    boxplot(len ~ dose, add = TRUE, boxwex = 0.25, at = 1:3 + 0.2,
            subset = supp == "OJ", col = "orange")
    legend(2, 9, c("Ascorbic acid", "Orange juice"),
           fill = c("yellow", "orange"))
})
```



R 機率分佈

1. 常態分佈
2. 二項分佈
3. Poisson 分佈

R 常態分佈 - Normal Distribution

公式

$$f(x;\ \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

"μ" -> 期望值
"σ" -> 標準差

1. 隨機變數

抽隨機變數 -> rnorm()

rnorm(n=[所需要的數值])

範圍為由 0 至 1，抽取十個值則為 n = 10，引數可設定額外的條件。

期望值 -> mean
標準差 -> sd

rnorm(n = 10)
rnorm(n = 10, mean = 100, sd = 20)

```
> rnorm(n = 10)
 [1] -0.89425240  0.10380779 -0.49412657 -0.32734290 -1.12901543  0.18039339  1.00151944  0.04965701  0.38603977  1.01110425
>
> rnorm(n = 10, mean = 100, sd = 20)
 [1] 115.81463  85.86599 106.45553 108.23275 132.59479 103.49363  82.95909 123.37284 108.50389  94.19420
>
```

R 常態分佈 - Normal Distribution

randNorm10 = rnorm(10)
randNorm10
dnorm(randNorm10)
dnorm(c(-1, 0, 1))

2. 機率密度

-> dnorm()

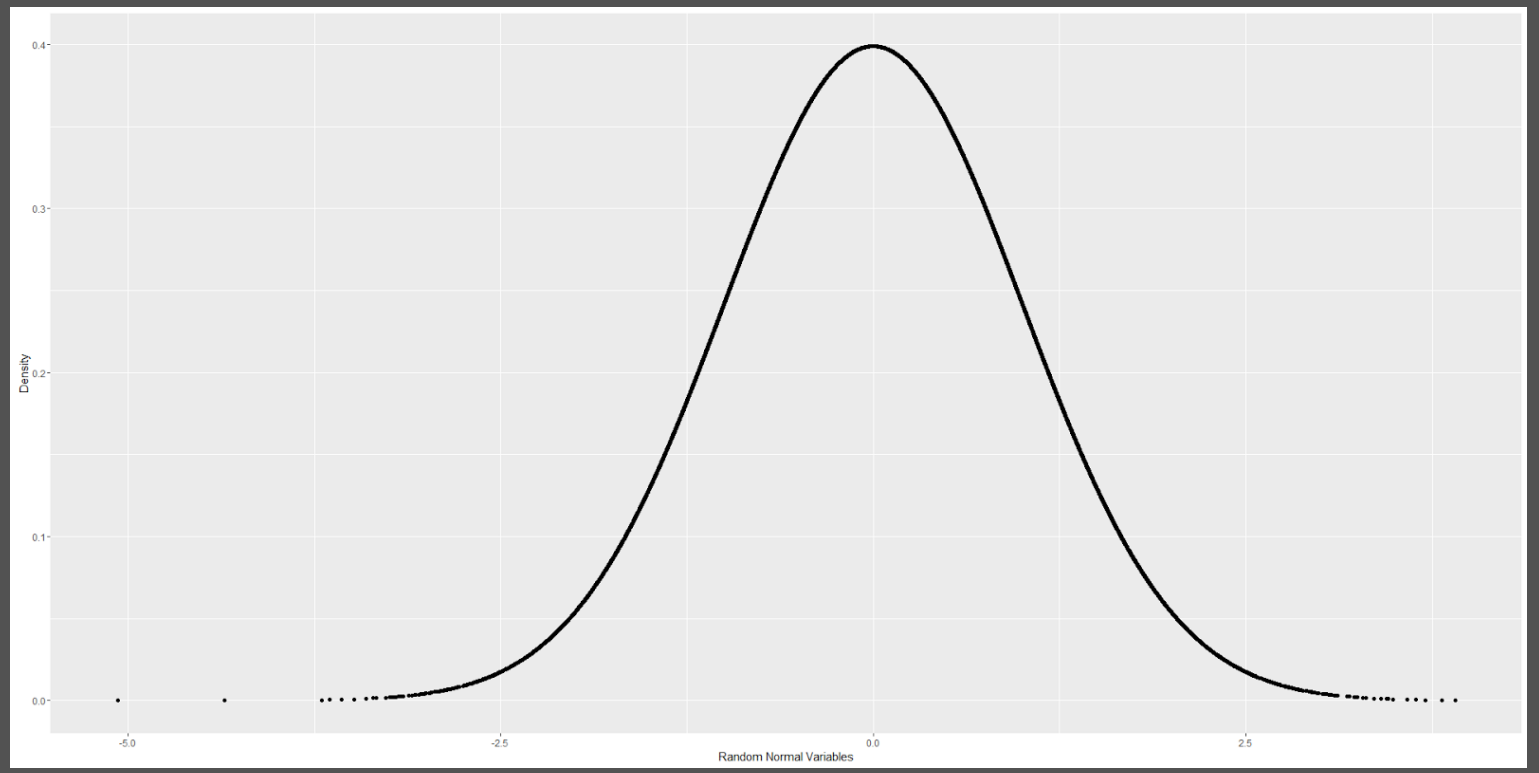
求某值機率，也可用向量的方式找相對的值
dnorm(c(,,,....))

```
> randNorm10 = rnorm(10)
> randNorm10
 [1]  0.48266870 -0.84887339  0.20676393  0.38754635 -1.41716699  0.07714806  0.40399902  0.69741284  1.22780992  0.47022069
> dnorm(randNorm10)
 [1] 0.3550761 0.2782510 0.3905051 0.3700805 0.1461503 0.3977568 0.3676786 0.3128189 0.1877400 0.3571883
> dnorm(c(-1, 0, 1))
[1] 0.2419707 0.3989423 0.2419707
> 
```

R 常態分佈 - Normal Distribution

rnorm() 生成 30000 個隨機變數，dnorm() 計算分佈，利用 ggplot 套件 進行繪圖。

```
randNorm = rnorm(30000)
randDensity = dnorm(randNorm)
require(ggplot2)
ggplot(data.frame(x = randNorm, y = randDensity)) +
aes(x = x, y = y) + geom_point() + labs(x = "Random Normal Variables", y = "Density")
```



R 常態分佈 - Normal Distribution

3. 累積分佈

-> pnorm()

求算 之前建立的 randNorm 物件 -> rnorm(10)

也可用向量的方式找相對的值 pnorm(c(,,,....))

pnorm(randNorm10)
pnorm(c(-3, 0, 3))
pnorm(-1)

```
> pnorm(randNorm10)
 [1] 0.68533450 0.19797587 0.58190289 0.65082411 0.07821706 0.53074712 0.65689328 0.75722777 0.89024083 0.68090132
> pnorm(c(-3, 0, 3))
[1] 0.001349898 0.500000000 0.998650102
> pnorm(-1)
[1] 0.1586553
>
```

R 常態分佈 - Normal Distribution

求算左尾機率，先算兩點機率，再取差 !!

pnorm(1) - pnorm(0)
pnorm(1) - pnorm(-1)

```
> pnorm(1) - pnorm(0)
[1] 0.3413447
> pnorm(1) - pnorm(-1)
[1] 0.6826895
```

繪製 "分佈圖"，最外面是 ggplot() ，而所使用的資料為自行建立的 資料框架，randNorm、 randDensity 存入資料框，資料框中有兩個欄位 x,y，x 指向 randNorm、y 指向 randDensity

x -> 為序列建立
y -> 為序列分佈
geom_line() -> 線條
labs() -> 標籤、命名
最後指向 物件 p

```
> p = ggplot(data.frame(x=randNorm, y=randDensity)) + aes(x=x, y=y) + geom_line() + labs(x="x", y="Density")
>
> neg1Seq = seq(from=min(randNorm), to=-1, by=.1)
>
> lessThanNeg1 = data.frame(x=neg1Seq, y=dnorm(neg1Seq))
> head(lessThanNeg1)
          x           y
1 -5.069501 1.047758e-06
2 -4.969501 1.730833e-06
3 -4.869501 2.830782e-06
4 -4.769501 4.583684e-06
5 -4.669501 7.348183e-06
6 -4.569501 1.166279e-05
> |
```

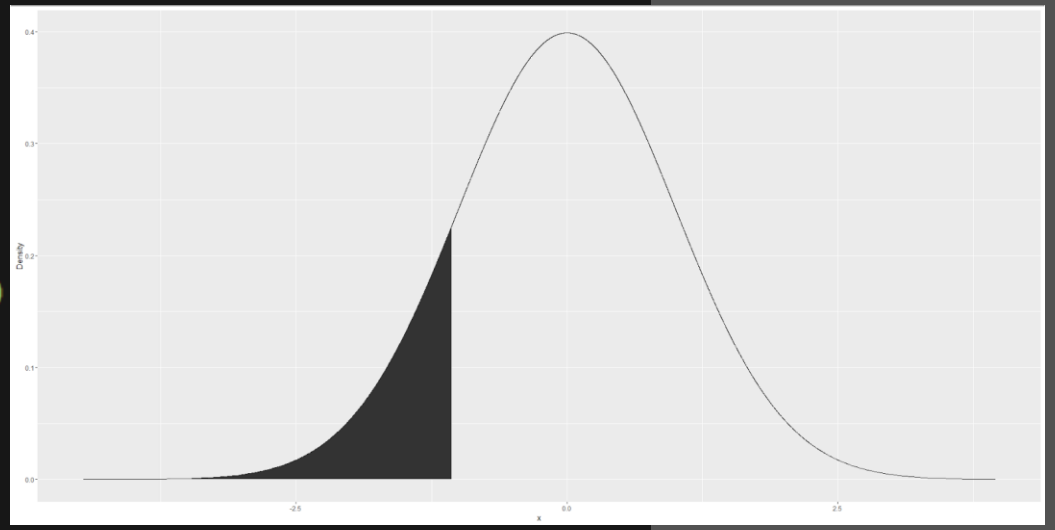
```
p = ggplot(data.frame(x=randNorm, y=randDensity)) + aes(x=x, y=y) + geom_line() +
 labs(x="x", y="Density")
neg1Seq = seq(from=min(randNorm), to=-1, by=.1)
lessThanNeg1 = data.frame(x=neg1Seq, y=dnorm(neg1Seq))
head(lessThanNeg1)
```

R 常態分佈 - Normal Distribution

利用 rbind() 將左右資料進行合併!!!

p = ggplot(data.frame(x=randNorm, y=randDensity)) + aes(x=x, y=y) + geom_line() + labs(x="x", y="Density")
neg1Seq = seq(from=min(randNorm), to=-1, by=.1)
lessThanNeg1 = data.frame(x=neg1Seq, y=dnorm(neg1Seq))
head(lessThanNeg1)

```
> lessThanNeg1 = rbind(c(min(randNorm), 0), lessThanNeg1, c(max(lessThanNeg1$x), 0))
>
> p + geom_polygon(data=lessThanNeg1, aes(x=x, y=y))
>
> neg1Pos1Seq = seq(from=-1, to=1, by=.1)
>
>
> neg1To1 = data.frame(x=neg1Pos1Seq, y=dnorm(neg1Pos1Seq))
> head(neg1To1)
     x         y
1 -1.0 0.2419707
2 -0.9 0.2660852
3 -0.8 0.2896916
4 -0.7 0.3122539
5 -0.6 0.3332246
6 -0.5 0.3520653
>
```

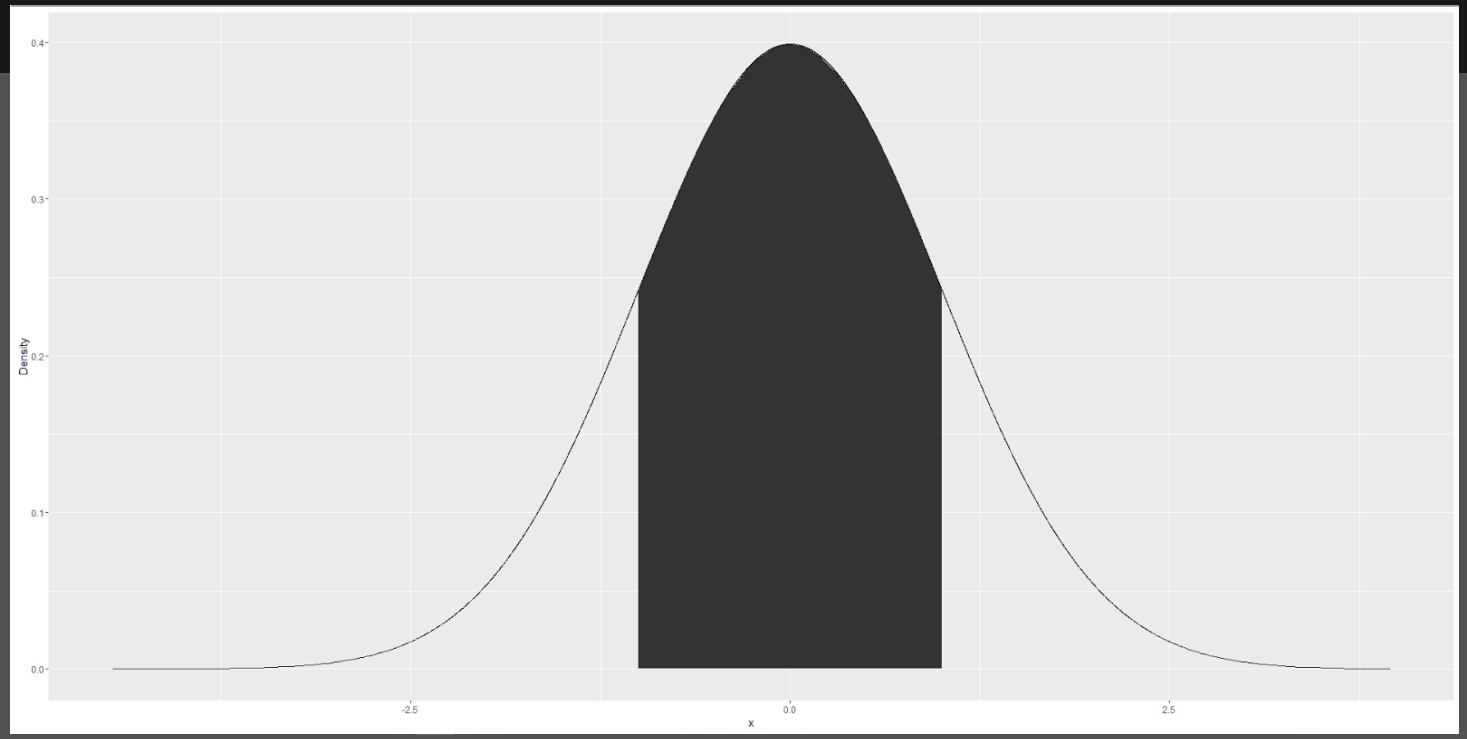


R 常態分佈 - Normal Distribution

將左右兩邊的終點資料 利用 rbind() 進行合併，利用 geom_polygon() 定義陰影部分。

neg1To1 = rbind(c(min(neg1To1$x), 0), neg1To1, c(max(neg1To1$x), 0))
p + geom_polygon(data=neg1To1, aes(x=x, y=y))

```
> neg1To1 = rbind(c(min(neg1To1$x), 0), neg1To1, c(max(neg1To1$x), 0))
>
> p + geom_polygon(data=neg1To1, aes(x=x, y=y))
>
```

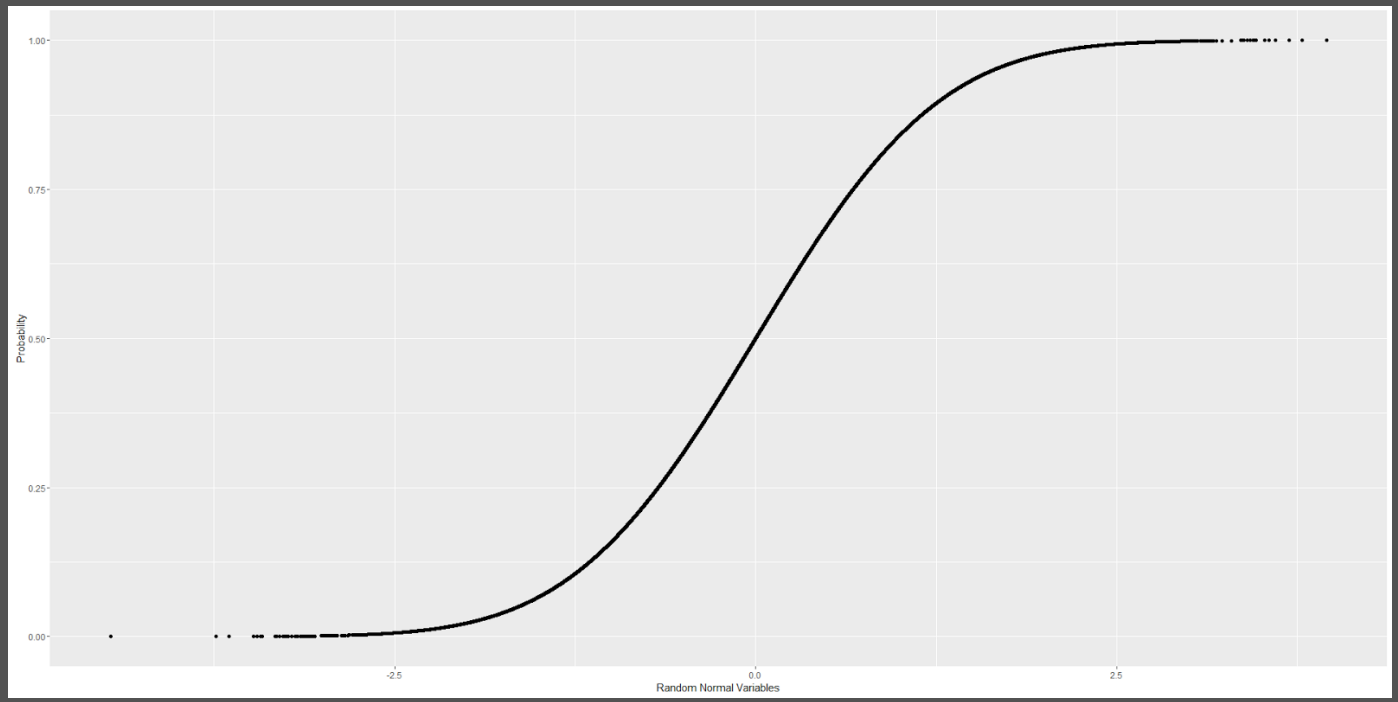


R 常態分佈 - Normal Distribution

x 為 randNorm -> rnorm(30000)
y 為 randProb -> pnorm(rnorm(30000))

利用 ggplot() 進行繪圖 !!!

```
randProb = pnorm(randNorm)
ggplot(data.frame(x=randNorm, y=randProb)) + aes(x=x, y=y) + geom_point() +
labs(x="Random Normal Variables", y="Probability")
```



R 常態分佈 - Normal Distribution

4. 分位數

qnorm() 其使用方式與 前面的 pnorm() 相反，給定累積機率則會回傳該機率的分位數。

```
randNorm10
qnorm(pnorm(randNorm10))
all.equal(randNorm10, qnorm(pnorm(randNorm10)))
```

```
> randNorm10
 [1]  1.2956858  0.4816070  0.4534113 -1.1923348  0.6597784  0.1778639 -2.1017596  0.5129748  0.3982614  2.1986222
> qnorm(pnorm(randNorm10))
 [1]  1.2956858  0.4816070  0.4534113 -1.1923348  0.6597784  0.1778639 -2.1017596  0.5129748  0.3982614  2.1986222
> all.equal(randNorm10, qnorm(pnorm(randNorm10)))
[1] TRUE
>
```

R 二項分佈 - Binomial Distribution

公式 : 當中的 C 為 n! / x!(n-x)!

$$f(x;\ n, p) = C_x^n\ p^x\ (1-p)^{n-x}$$

rbinom(n = 1, size = 10, prob = 0.4)
rbinom(n = 1, size = 10, prob = 0.4)
rbinom(n = 5, size = 10, prob = 0.4)
rbinom(n = 10, size = 10, prob = 0.4)
rbinom(n = 1, size = 1, prob = 0.4)
rbinom(n = 5, size = 1, prob = 0.4)
rbinom(n = 10, size = 1, prob = 0.4)

```
> rbinom(n = 1, size = 10, prob = 0.4)
[1] 8
>
> rbinom(n = 1, size = 10, prob = 0.4)
[1] 1
> rbinom(n = 5, size = 10, prob = 0.4)
[1] 4 7 2 1 3
> rbinom(n = 10, size = 10, prob = 0.4)
 [1] 6 3 6 3 6 5 2 4 5 5
>
> rbinom(n = 1, size = 1, prob = 0.4)
[1] 0
> rbinom(n = 5, size = 1, prob = 0.4)
[1] 1 0 0 1 0
> rbinom(n = 10, size = 1, prob = 0.4)
 [1] 0 0 0 1 1 0 0 0 0 0
>
```

1. 隨機變數-> rbinom()

引數 :
n -> 試驗數量
size -> 試驗次數
prob -> 成功機率

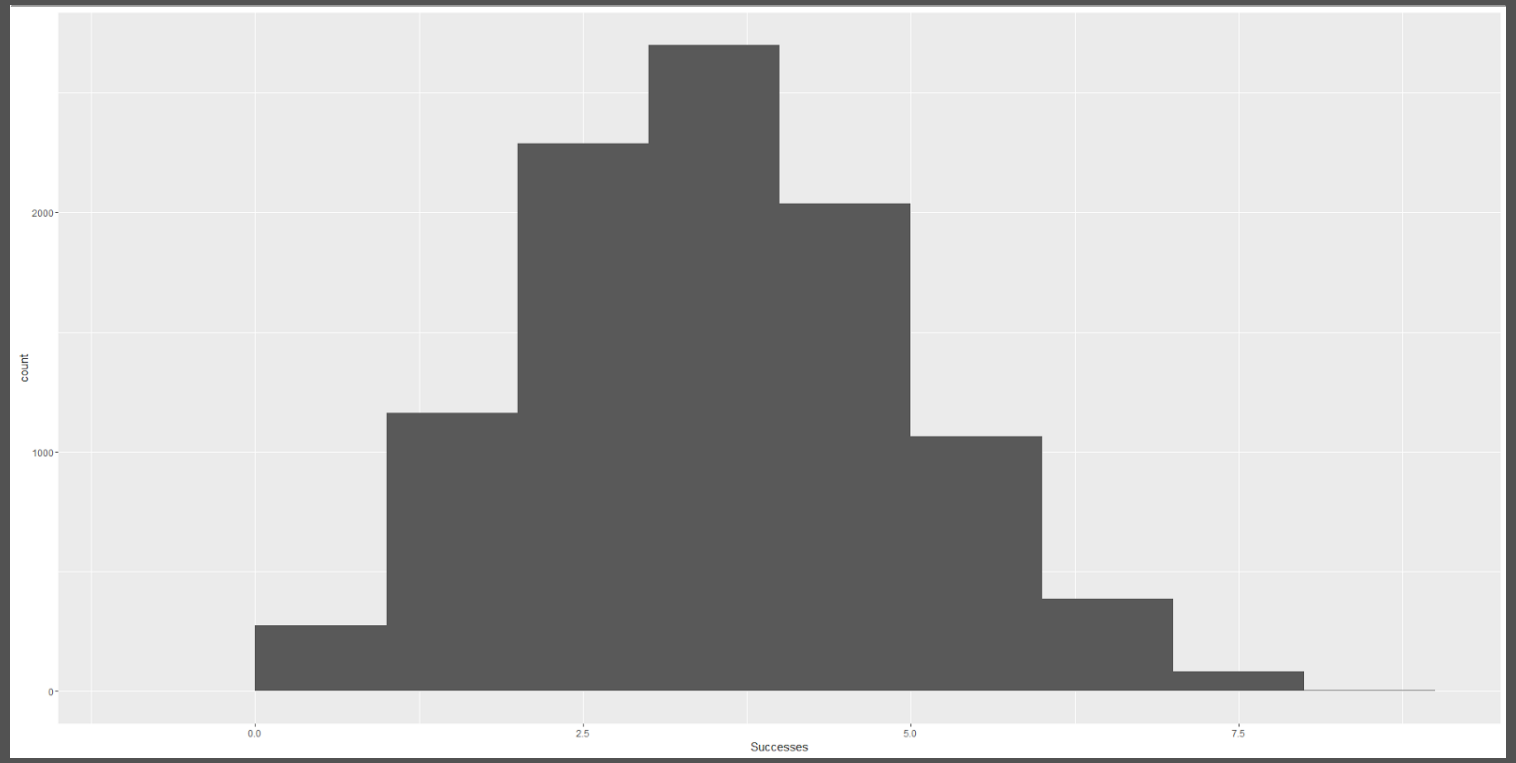
在這裡要注意，若設 n = 1 -> 二項分佈會簡化為白努利分佈
size = 1，則會變為 白努利隨機變數 : 1 -> 成功 & 0 -> 失敗

R 二項分佈 - Binomial Distribution

隨機生成 10000 個試驗，每組 10 個試驗，成功機率為 0.3

n -> 10000
size -> 10
prob -> 0.3

利用 ggplot()、 geom_histogtam()，繪製長方圖!!!



binomData = data.frame(Successes = rbinom(n = 10000, size = 10, prob = 0.3))
ggplot(binomData, aes(x = Successes)) + geom_histogram(binwidth = 1)

```
> binomData = data.frame(Successes = rbinom(n = 10000, size = 10, prob = 0.3))
> ggplot(binomData, aes(x = Successes)) + geom_histogram(binwidth = 1)
>
>
```

R 二項分佈 - Binomial Distribution


```
> binom5 = data.frame(Successes=rbinom(n=10000, size=5, prob=.3), Size=5)
> dim(binom5)
[1] 10000     2
> head(binom5)
  Successes Size
1         4    5
2         0    5
3         3    5
4         2    5
5         2    5
6         0    5
> |
```

當實驗次數越高，二項分佈會近似於常態分佈，在此用 R 證明，rbinom()。

n -> 10000
size -> 5
porb -> 0.3

在此建立 一個資料框，兩個直行欄位、10000個橫列。一欄為 successes 為 rbinom() 的結果，另一欄 存放 size，最後用 head() 檢視。

binom5 = data.frame(Successes=rbinom(n=10000, size=5, prob=.3), Size=5)
dim(binom5)
head(binom5)

R 二項分佈 - Binomial Distribution


```
> binom10 = data.frame(Successes=rbinom(n=10000, size=10, prob=.3), Size=10)
> dim(binom10)
[1] 10000     2
> head(binom10)
  Successes Size
1         4   10
2         5   10
3         5   10
4         4   10
5         4   10
6         4   10
> binom100 = data.frame(Successes=rbinom(n=10000, size=100, prob=.3), Size=100)
> binom1000 = data.frame(Successes=rbinom(n=10000, size=1000, prob=.3), Size=1000)
> 
```

再建一個 rbinom() 資料框，不過 size 為 10、100、1000

n -> 10000
size -> 10, 100, 1000
porb -> 0.3

```
binom10 = data.frame(Successes=rbinom(n=10000, size=10, prob=.3), Size=10)
dim(binom10)
head(binom10)
binom100 = data.frame(Successes=rbinom(n=10000, size=100, prob=.3), Size=100)
binom1000 = data.frame(Successes=rbinom(n=10000, size=1000, prob=.3), Size=1000)
```

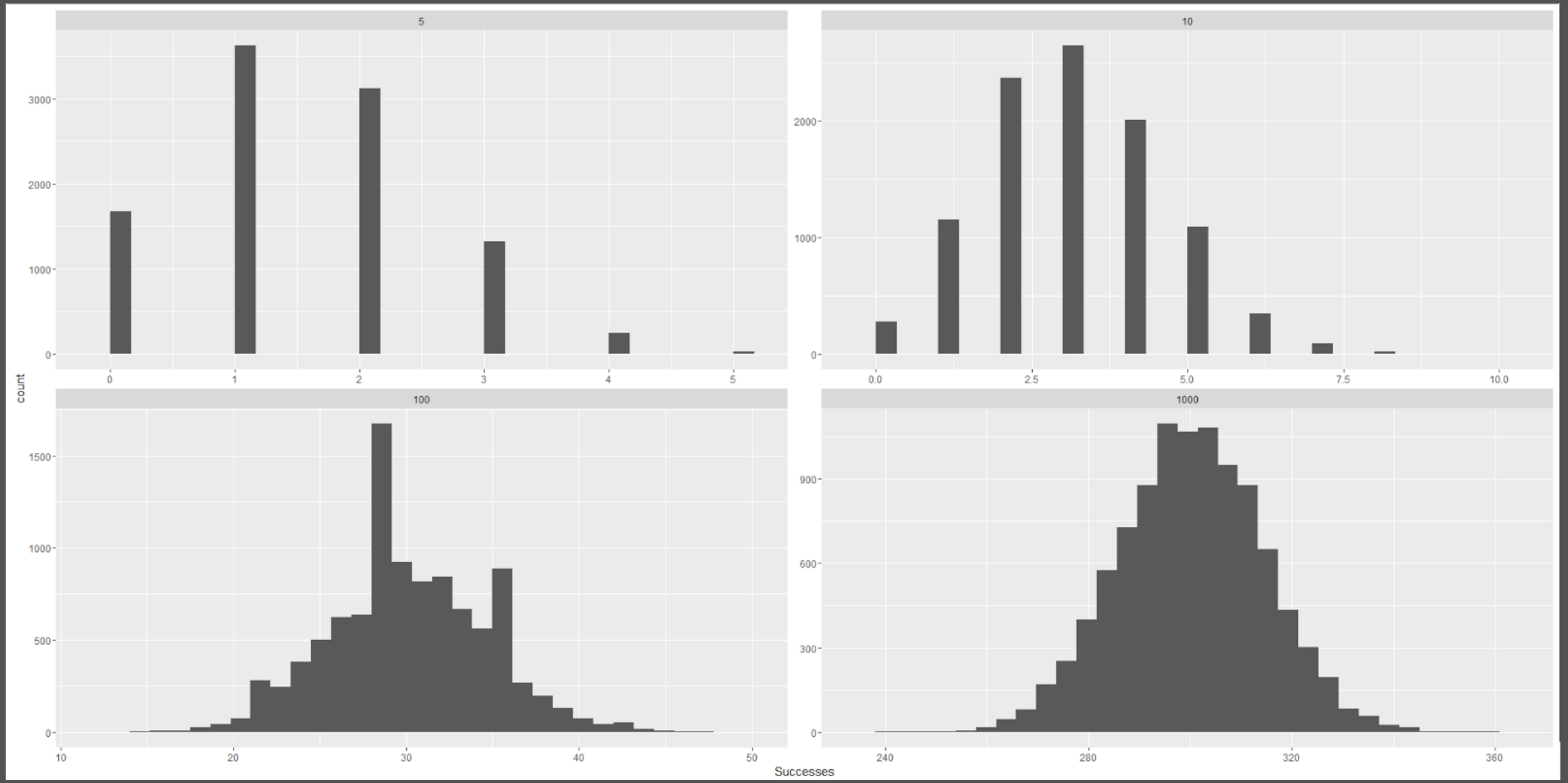
R 二項分佈 - Binomial Distribution


```
> binomAll = rbind(binom5, binom10, binom100, binom1000)
> dim(binomAll)
[1] 40000     2
> head(binomAll, 10)
   Successes Size
1          4    5
2          0    5
3          3    5
4          2    5
5          2    5
6          0    5
7          2    5
8          0    5
9          3    5
10         2    5
> tail(binomAll, 10)
      Successes Size
39991       295 1000
39992       296 1000
39993       297 1000
39994       308 1000
39995       325 1000
39996       308 1000
39997       301 1000
39998       297 1000
39999       293 1000
40000       306 1000
>
> ggplot(binomAll, aes(x=Successes)) + geom_histogram() + facet_wrap(~ Size, scales="free")
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

最後用 rbind() 將 四個 二項的物件合併在一起!!! 並命名物件為 binomAll
利用 ggplot()、geom_historgan() 繪製 "長方圖"

binomAll = rbind(binom5, binom10, binom100, binom1000)
dim(binomAll)
head(binomAll, 10)
tail(binomAll, 10)
ggplot(binomAll, aes(x=Successes)) + geom_histogram() +
facet_wrap(~ Size, scales="free")

R 二項分佈 - Binomial Distribution



R 二項分佈 - Binomial Distribution

2. 累積機率與機率密度

```
> dbinom(x = 3, size = 10, prob = 0.3)
[1] 0.2668279
>
> pbinom(q = 3, size = 10, prob = 0.3)
[1] 0.6496107
>
> dbinom(x = 1:10, size = 10, prob = 0.3)
 [1] 0.1210608210 0.2334744405 0.2668279320 0.2001209490 0.1029193452 0.0367569090 0.0090016920 0.0014467005 0.0001377810 0.0000059049
> pbinom(q = 1:10, size = 10, prob = 0.3)
 [1] 0.1493083 0.3827828 0.6496107 0.8497317 0.9526510 0.9894079 0.9984096 0.9998563 0.9999941 1.0000000
>
```

機率密度 -> dbinom()
累積機率 -> pbinom()

注意!!! 這兩個函數可以使用向量化運算 dbinom()、pbinom()

dbinom(x = 3, size = 10, prob = 0.3)
pbinom(q = 3, size = 10, prob = 0.3)
dbinom(x = 1:10, size = 10, prob = 0.3)
pbinom(q = 1:10, size = 10, prob = 0.3)

R 二項分佈 - Binomial Distribution

3. 分位數 -> qbinom()

```
> qbinom(p = 0.3, size = 10, prob = 0.3)
[1] 2
> qbinom(p = c(0.3, 0.35, 0.4, 0.5, 0.6), size = 10, prob = 0.3)
[1] 2 2 3 3 3
> 
```

在此 二項 回傳的對應值會是 該分位數的成功次數!!!

qbinom(p = 0.3, size = 10, prob = 0.3)
qbinom(p = c(0.3, 0.35, 0.4, 0.5, 0.6), size = 10, prob = 0.3)

R Poisson 分佈 - Poisson Distribution

公式 $$p(x;\ \lambda) = \frac{\lambda^x e^{-\lambda}}{x!}$$

其期望值與變異數 -> λ
在此建立 樣本 n =10000 、 λ 為 1, 2, 5, 10, 20 的資料，並塞入
pois 的資料框架中...

```
pois1 = rpois(n=10000, lambda=1)
pois2 = rpois(n=10000, lambda=2)
pois5 = rpois(n=10000, lambda=5)
pois10 = rpois(n=10000, lambda=10)
pois20 = rpois(n=10000, lambda=20)
pois = data.frame(Lambda.1=pois1, Lambda.2=pois2, Lambda.5=pois5, Lambda.10=pois10, Lambda.20=pois20)
```

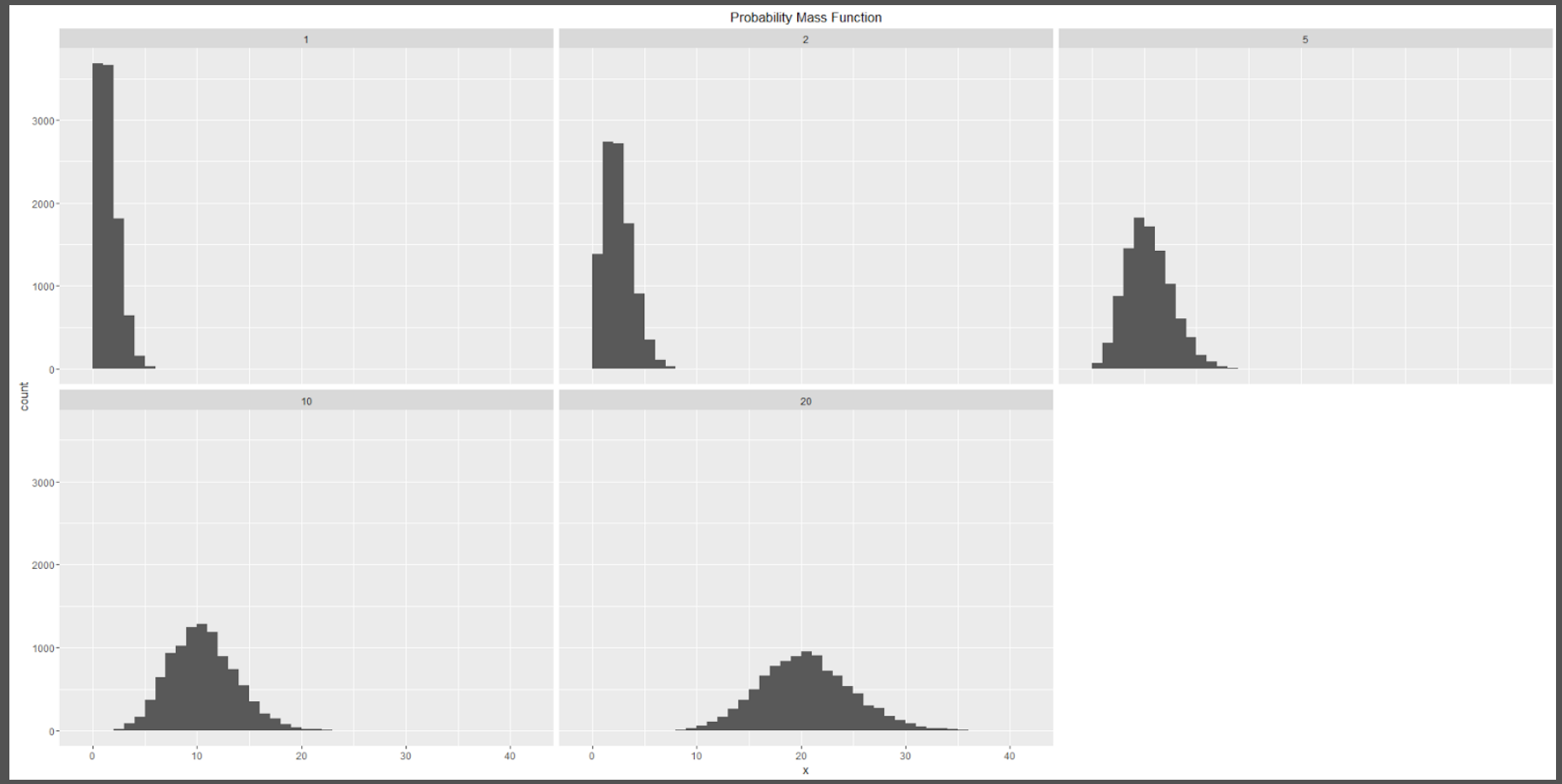
R Poisson 分佈 - Poisson Distribution

利用 reshape2 的套件，整理資料，還要利用 stringr 套件 處理直行名稱 !!!!

```
> require(reshape2)
Loading required package: reshape2
>
> pois = melt(data=pois, variable.name="Lambda", value.name="x")
No id variables; using all as measure variables
>
> require(stringr)
Loading required package: stringr
>
> pois$Lambda = as.factor(as.numeric(str_extract(string=pois$Lambda, pattern="\\d+")))
>
> head(pois)
  Lambda x
1      1 0
2      1 1
3      1 2
4      1 3
5      1 1
6      1 3
> tail(pois)
      Lambda  x
49995     20 18
49996     20 23
49997     20 26
49998     20 21
49999     20 21
50000     20 22
```

```
require(reshape2)
pois = melt(data=pois, variable.name="Lambda", value.name="x")
require(stringr)
pois$Lambda = as.factor(as.numeric(str_extract(string=pois$Lambda, pattern="\\d+")))
head(pois)
tail(pois)
```

R Poisson 分佈 - Poisson Distribution



利用 ggplot2 套件進行繪圖

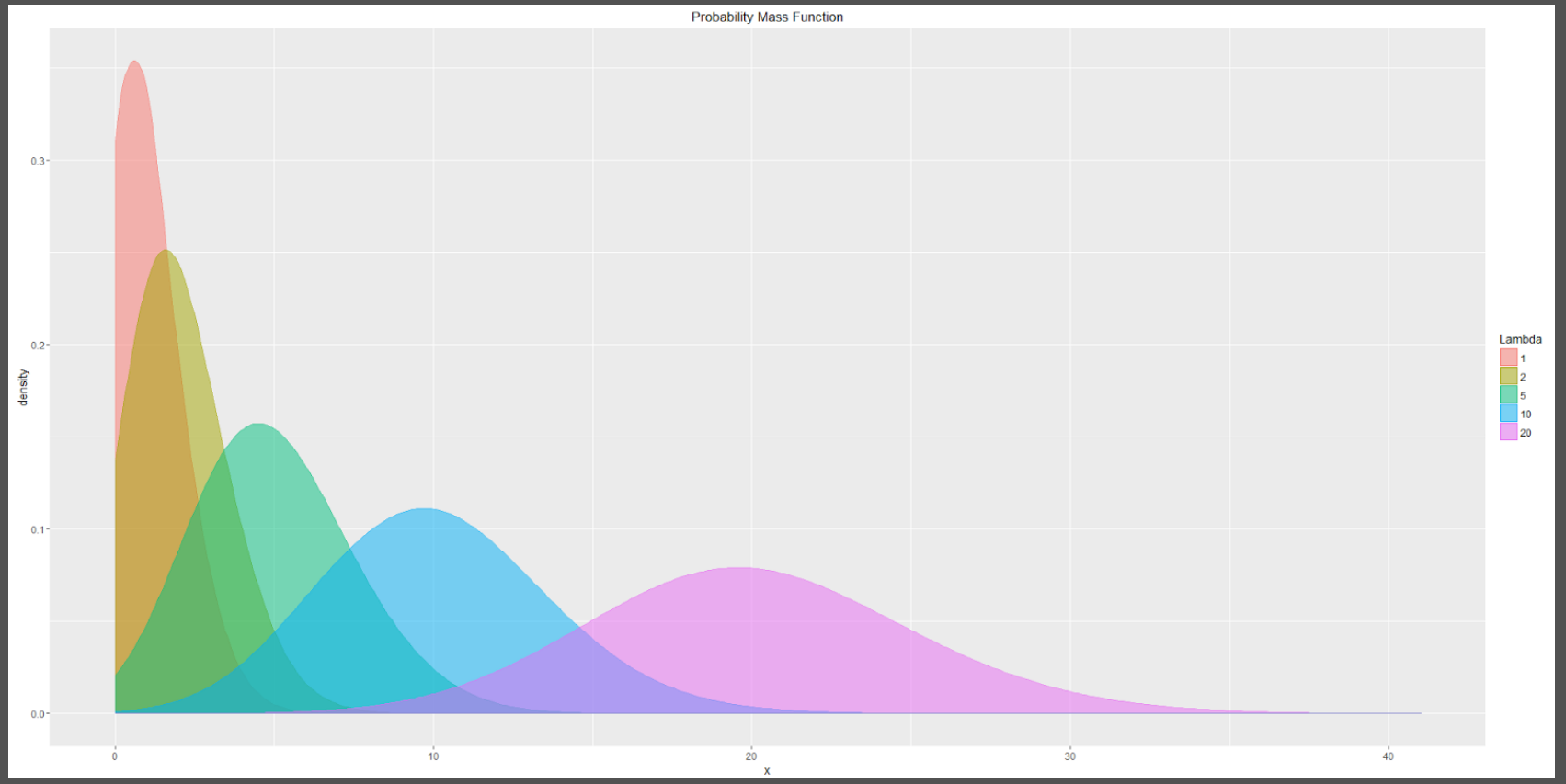
geom_histogram() -> 直方圖
facet_warp() -> 根據的欄位
ggtitle() ->圖名

```
require(ggplot2)
ggplot(pois, aes(x=x)) + geom_histogram(binwidth=1) +
facet_wrap(~ Lambda) + ggtitle("Probability Mass Function")
```

R Poisson 分佈 - Poisson Distribution

利用 ggplot2 套件進行繪圖

```
ggplot(pois, aes(x=x)) + geom_density(aes(group=Lambda,
color=Lambda, fill=Lambda), adjust=4, alpha=1/2) +
scale_color_discrete() + scale_fill_discrete() +
ggtitle("Probability Mass Function")
```



R 摘要統計

1. 算數平均數、期望值 - arithmetic mean

$$\mathrm{E}[\mathrm{x}] = \frac{\sum_{i=1}^{N} x_i}{N}$$

先講講 sample() 函數
x -> 值範圍
size -> 隨機生成的數字
下面為 在 1:100 間隨機生成100個數字
mean() -> 為算數平均數、期望值

x = sample(x = 1:100, size = 100, replace = TRUE)
x
mean(x)

```
> x = sample(x = 1:100, size = 100, replace = TRUE)
> x
  [1]  32  64  13  57  44   7  19  69   7   3  75  87  26  33  11  40  94  94  12  93
 [42]  47  21  42  23  51  21   1   5  21  98  36   2  25  52  30  53  22  16  37  97
 [83]  88  41  63  68  89  65  97  42  66  49  79  77  74  38  84  96  98  53
>
> mean(x)
[1] 47.4
>
```

R 摘要統計

2. 算數平均數、期望值的 NA 問題

算數平均數、期望值的 NA 問題，再來則是處理在 mean() 過程中的 NA 問題，複製x，塞入名為 y 的物件，利用 sample() 隨機將20個值設為 NA，若單純計算 mean() 會直接回傳 NA 值，必須加入 na.rm 的引數(預設為 FALSE)，mean() 就會省略 NA，回傳平均數

```
y = x
y[sample(x = 1:100, size = 20, replace = FALSE)] = NA
y
mean(y)
mean(y, na.rm = TRUE)
```

```
> y = x
>
> y[sample(x = 1:100, size = 20, replace = FALSE)] = NA
> y
  [1] 32 64 13 NA 44  7 19 69  7  3 75 87 26 NA 11 40 94 94 12 NA
 [56] NA NA 22 16 37 97  6 10 93 32 14 NA 60 85 NA 50 11 18 85 65
>
> mean(y)
[1] NA
> mean(y, na.rm = TRUE)
[1] 46.75
>
```

R 摘要統計


```
> grades = c(95, 72, 87, 66)
> weights = c(1/2, 1/4, 1/8, 1/8)
> mean(grades)
[1] 80
> weighted.mean(x = grades, w = weights)
[1] 84.625
> 
```

3. 加權平均數

weighted.mean()

weighted.mean(x = [處理值], w = [權重])

grades = c(95, 72, 87, 66)
weights = c(1/2, 1/4, 1/8, 1/8)
mean(grades)
weighted.mean(x = grades, w = weights)

$$\mathrm{E}[\mathrm{x}] = \frac{\sum_{i=1}^{N} w_i x_i}{\sum_{i=1}^{N} w_i}$$

$$= \sum_{i=1}^{N} p_i \, x_i$$

R 摘要統計

$$\mathrm{Var}(x) = \frac{\sum_{i=1}^{N}(x_i - \overline{x})^2}{N-1}$$

4. 變異數 - variance

使用函數為 -> var()

當然了我們可以根據 變異數的公式 進行求值，這個跟用 var() 會得到一樣的結果!!!

根據公式轉換

sum(((x - mean(x)) ^ 2)/(length(x) - 1))

var(x)
var(x)
sum((x - mean(x))^2)/(length(x) - 1)

```
> var(x)
[1] 896.4242
>
```

```
> var(x)
[1] 896.4242
> sum((x - mean(x))^2)/(length(x) - 1)
[1] 896.4242
>
```

R 摘要統計

5. 標準差 - standard deviation

其實就是 對變異數開根號

使用函數為 -> sd()

sd(x)

我們也可以用 sqrt() 根號這個函數對 var() 變異數函數進行使用

sqrt(var(x))

類推上面

sqrt(sum((x - mean(x))^2)/(length(x) - 1))

上述三個都是一樣的東西

再來就是處理 遺失值 NA，記得要加 引數 na.rm = TRUE

```
> sqrt(var(x))
[1] 29.94034
> sd(x)
[1] 29.94034
> sd(y)
[1] NA
> sd(y, na.rm = TRUE)
[1] 29.79657
>
```

sqrt(var(x))
sd(x)
sd(y)
sd(y, na.rm = TRUE)

R 摘要統計

6. 敘述統計

極大值 max()

極小值 min()

中位數 median()

Q1 - 第一四分位數、Q3 - 第三四分位數，當然不只可以單求某分位數，只要給定 百分比就夠了，如果有 NA 值 一定要加 na.rm = TRUE。

quantile([資料], probs = c(0.25, 0.75))

同時計算 極大值、極小值、中位數、Q1、Q3，可以使用 summary() 函數。

summary()

```
> summary(x)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    1.0    21.0    43.0    47.4    75.0   100.0
> summary(y)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
   1.00   20.50   42.50   46.75   75.00   98.00      20
> |
```

```
> min(x)
[1] 1
> max(x)
[1] 100
> median(x)
[1] 43
> min(y)
[1] NA
> min(y, na.rm = TRUE)
[1] 1
> |
```

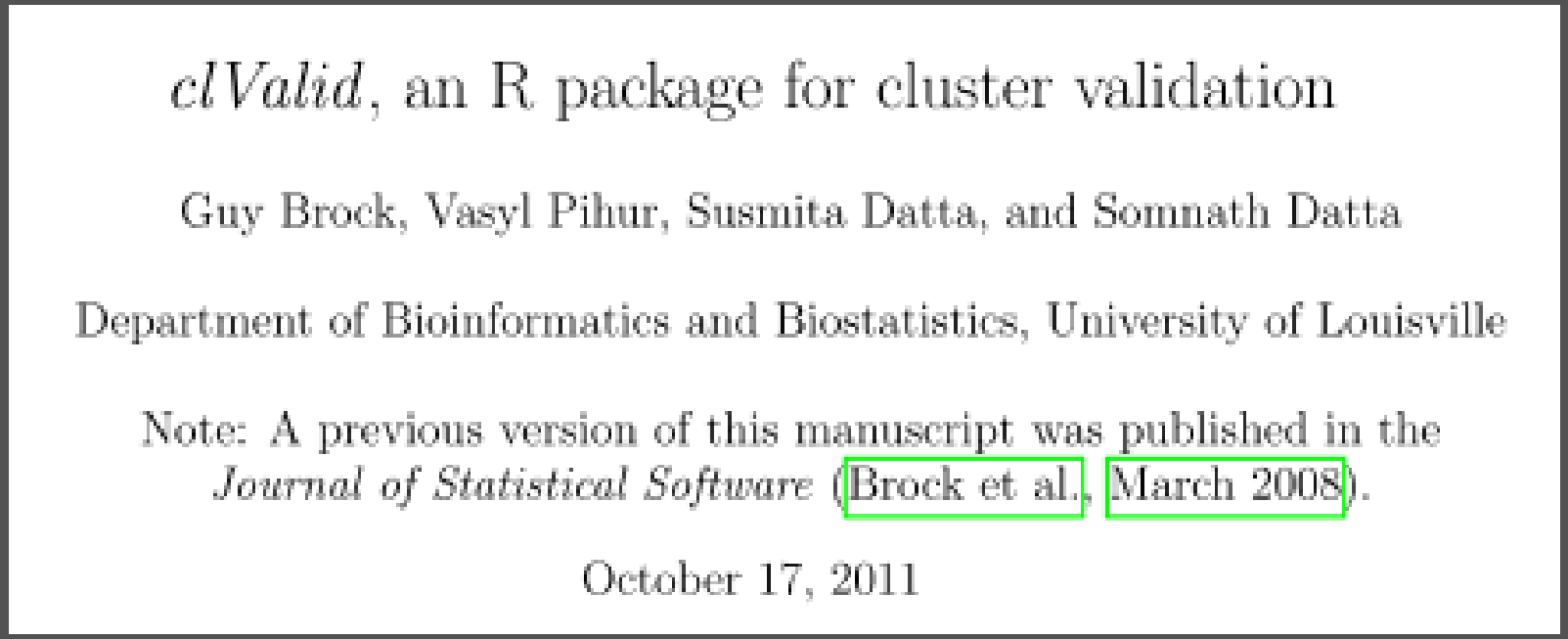
```
> quantile(x, probs = c(0.25, 0.75))
25% 75%
 21  75
>
> quantile(y, probs = c(0.25, 0.75))
Error in quantile.default(y, probs = c(0.25, 0.75)) :
  missing values and NaN's not allowed if 'na.rm' is FALSE
>
> quantile(y, probs = c(0.25, 0.75), na.rm = TRUE)
 25%  75%
20.5 75.0
>
> quantile(x, probs = c(0.1, 0.25, 0.5, 0.75, 0.99))
  10%   25%   50%   75%   99%
 9.70 21.00 43.00 75.00 98.02
> |
```

min(x)
max(x)
median(x)
min(y)
min(y, na.rm = TRUE)
summary(x)
summary(y)
quantile(x, probs = c(0.25, 0.75))
quantile(y, probs = c(0.25, 0.75))
quantile(y, probs = c(0.25, 0.75), na.rm = TRUE)
quantile(x, probs = c(0.1, 0.25, 0.5, 0.75, 0.99))

R 集群分群與衡量 - clValid

Guy Brock, Vasyl Pihur, Susmita Datta, and Somnath Datta, 2008, "clValid , an R package for cluster validation", Journal of Statistical Software, DOI : 10.18637/jss.v025.i04

clValid 的穩定性衡量 (Guy Brock, Vasyl Pihur, Susmita Datta & Somnath Datta, 2008)，是將完整資料的集群結果與逐一刪除的每一列進行比較。其所包含的措施共分為四種，Average Proportion of Non-overlap 、 Average Distance 、 Average Distance between Means 、 Figure of Merit。



clValid, an R package for cluster validation

Guy Brock, Vasyl Pihur, Susmita Datta, and Somnath Datta

Department of Bioinformatics and Biostatistics, University of Louisville

Note: A previous version of this manuscript was published in the *Journal of Statistical Software* (Brock et al., March 2008).

October 17, 2011

R 集群分群與衡量 - clValid

衡量方法

1. APN
APN(Average Proportion of Non-overlap) 是通過基於逐一刪除單行的資料與完整的資料進行集群的方式，來測量未放置在同一集群中的平均觀察比例。

$$APN(K) = \frac{1}{MN}\sum_{i=1}^{N}\sum_{l=1}^{M}\left(1 - \frac{n(C^{i,l} \cap C^{i,0})}{n(C^{i,0})}\right)$$

$C^{i,l}$ 為原始的集群(包含所有可用的資料) 來觀察 i 集群，其 i 集群為基於刪除 l 的直行資料集所建立。然後將集群的總數設置為 K，不重疊的平均比例被定義為範圍在區間 $[0, 1]$，其數值越接近 0 會有應於高度一致的集群結果。

R 集群分群與衡量 - clValid

衡量方法

2. AD
AD(Average Distance) 的計算方式是根據完整資料的集群與逐一刪除單行資料的集群來計算放置在同一群中觀察值的平均距離。其定義為 :

$$AD(K) = \frac{1}{MN}\sum_{i=1}^{N}\sum_{l=1}^{M}\frac{1}{n(C^{i,0})n(C^{i,l})}\left[\sum_{i\in C^{i,0},j\in C^{i.l}} dist(i,j)\right]$$

平均距離為 0 至 ∞ 範圍間的值，其數值越小越好。

R 集群分群與衡量 - clValid

衡量方法

3. ADM
ADM(Average Distance between Means)，此計算方式是利用集群方法計算同一集群中所觀察的集群中心的距離，所使用的基礎為完整的資料和逐一刪除直行的資料，其定義為：

$$ADM(K) = \frac{1}{MN}\sum_{i=1}^{N}\sum_{l=1}^{M} dist(\bar{x}_{C^{i,l}}, \bar{x}_{C^{i,0}})$$

其 $C^{i,0}$ 是集群中包含了觀察 i 的平均值，當集群基於完整的資料時會類似於 $C^{i,l}$ 的定義。目前平均距離之間的平均值只有歐氏距離。它具有 0 至 ∞ 之間的值，且較小的值優先。

R 集群分群與衡量 - clValid

衡量方法

4. FOM
Figure of Merit(FOM) 為計算刪除列的平均群變異數，其集群則基於剩下未刪除的部分作為樣本。當中使用集群平均值來預測平均誤差。l 為特定的剩餘列， FOM 為 :

$$FOM(l,K) = \sqrt{\frac{1}{N}\sum_{k=1}^{K}\sum_{i\in C_k(l)} dist(x_{i,l}, \bar{x}_{C_k(l)})}$$

其值 $x_{i,1}$ 為 $C_k(l)$ 集群下的 l 直行觀察值，而 $C_k(l)$ 為 $\bar{x}_{C_k(l)}$ 集群下的平均值。目前唯一可用於最優數值的距離為 「歐幾里得距離」。最優數值乘上調整因子 $\sqrt{\frac{N}{N-K}}$ ，來緩解隨著集群數量增加所減少的趨勢。最終的分數為在所有已移除的直行上進行平均，具有 0 至 ∞ 之間的值，其值越小越好。

R 集群分群與衡量 - clValid

下面整理該篇第四章研究所提到的兩個驗證 Internal Validation 跟 Stability Validation (4.1 & 4.2)。

1. Internal Validation 包含 connectivity、 Silhouette Width、 Dunn Index。

2. Stability Validation 包括 APN、 AD、 ADM 和 FOM。

```
> install.packages("clValid")
Installing package into ‘D:/USERDATA/Documents/R/win-library/3.4’
(as ‘lib’ is unspecified)
trying URL 'https://cloud.r-project.org/bin/windows/contrib/3.4/clValid_0.6-6.zip'
Content type 'application/zip' length 493390 bytes (481 KB)
downloaded 481 KB

package ‘clValid’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\USER\AppData\Local\Temp\Rtmpm2akpY\downloaded_packages
> library("clValid")
Loading required package: cluster
>
```

R 集群分群與衡量 - clValid

在此利用他本身的小鼠資料集

```
> data(mouse)
> head(mouse)
            ID       M1       M2       M3      NC1      NC2      NC3                         FC
1   1448995_at 4.706812 4.528291 4.325836 5.568435 6.915079 7.353144 Growth/Differentiation
2 1436392_s_at 3.867962 4.052354 3.474651 4.995836 5.056199 5.183585   Transcription factor
3 1437434_a_at 2.875112 3.379619 3.239800 3.877053 4.459629 4.850978          Miscellaneous
4   1428922_at 5.326943 5.498930 5.629814 6.795194 6.535522 6.622577          Miscellaneous
5 1452671_s_at 5.370125 4.546810 5.704810 6.407555 6.310487 6.195847          ECM/Receptors
6   1448147_at 3.471347 4.129992 3.964431 4.474737 5.185631 5.177967 Growth/Differentiation
> NROW(mouse)
[1] 147
```

在此將文字的資料移至rowname

```
> express = mouse[,c("M1","M2","M3","NC1","NC2","NC3")]
>
> rownames(express) = mouse$ID
> head(express)
                 M1       M2       M3      NC1      NC2      NC3
1448995_at   4.706812 4.528291 4.325836 5.568435 6.915079 7.353144
1436392_s_at 3.867962 4.052354 3.474651 4.995836 5.056199 5.183585
1437434_a_at 2.875112 3.379619 3.239800 3.877053 4.459629 4.850978
1428922_at   5.326943 5.498930 5.629814 6.795194 6.535522 6.622577
1452671_s_at 5.370125 4.546810 5.704810 6.407555 6.310487 6.195847
1448147_at   3.471347 4.129992 3.964431 4.474737 5.185631 5.177967
```

R 集群分群與衡量 - clValid

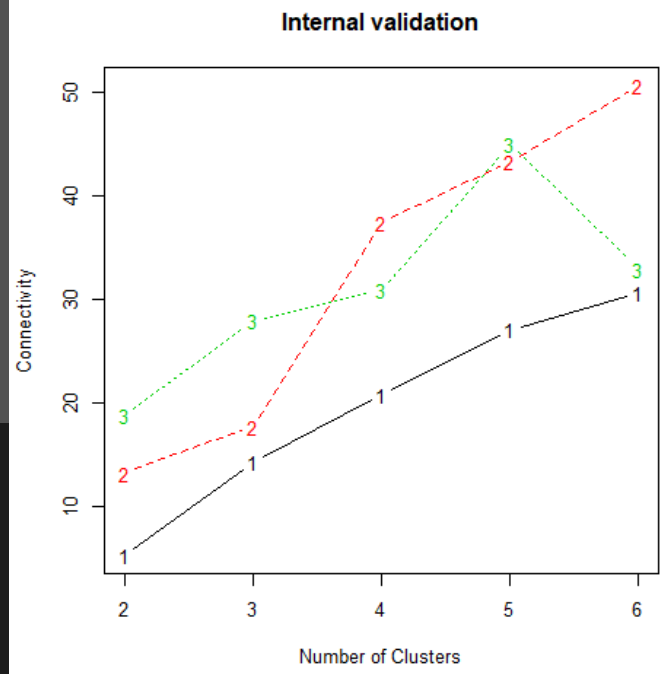
下面為 Internal Validation 的衡量分數。

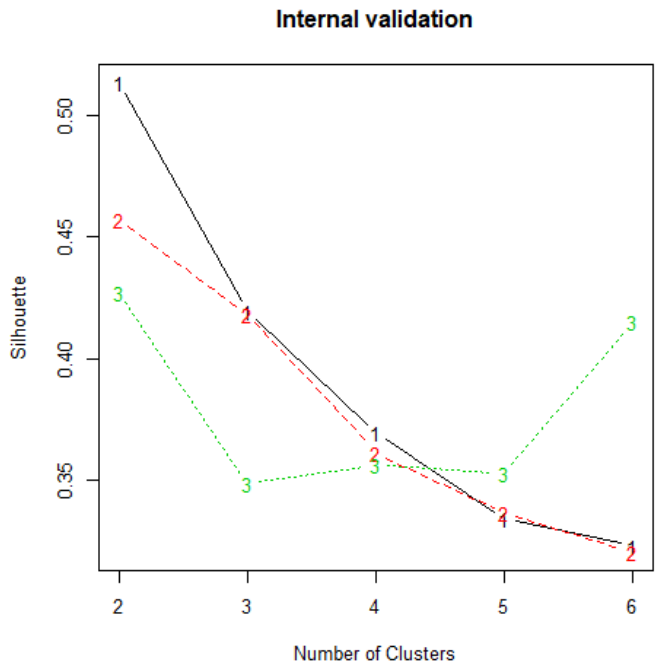
第一個參數為資料，第二個參數為要測試的分群數量，第三個名為 clMethods 為分群的方法，第四個名為 valiation 參數為所要選擇的衡量方式。

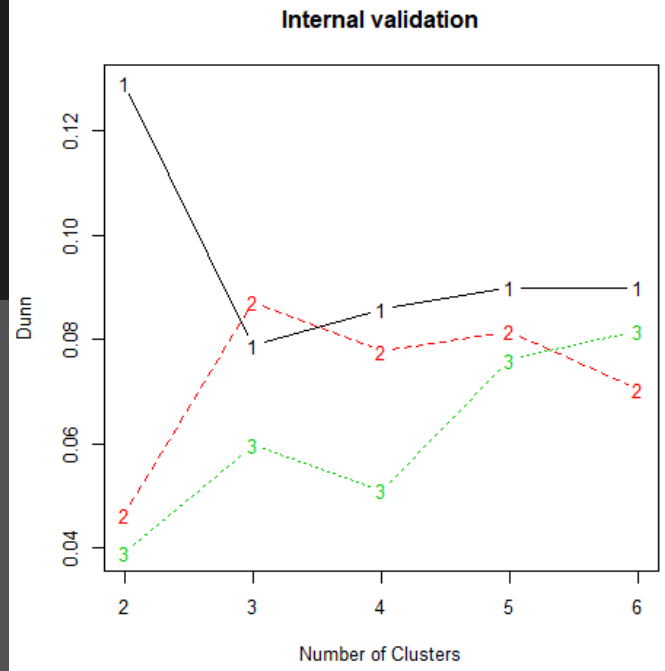
```
> # Internal Validation
> intern.it = clValid(express, 2:6,
+                     clMethods=c("hierarchical","kmeans","pam"),
+                     validation= "internal")
> summary(intern.it)

Clustering Methods:
 hierarchical kmeans pam

Cluster sizes:
 2 3 4 5 6

Validation Measures:
                                 2       3       4       5       6

hierarchical Connectivity   5.3270 14.2528 20.7520 27.0726 30.6194
             Dunn           0.1291  0.0788  0.0857  0.0899  0.0899
             Silhouette     0.5133  0.4195  0.3700  0.3343  0.3233
kmeans       Connectivity  13.2548 17.6651 37.3980 43.2655 50.6095
             Dunn           0.0464  0.0873  0.0777  0.0815  0.0703
             Silhouette     0.4571  0.4182  0.3615  0.3367  0.3207
pam          Connectivity  18.7917 27.9651 30.9302 44.9671 32.9667
             Dunn           0.0391  0.0597  0.0510  0.0761  0.0816
             Silhouette     0.4271  0.3489  0.3563  0.3530  0.4152

Optimal Scores:

             Score  Method       Clusters
Connectivity 5.3270 hierarchical 2
Dunn         0.1291 hierarchical 2
Silhouette   0.5133 hierarchical 2
```

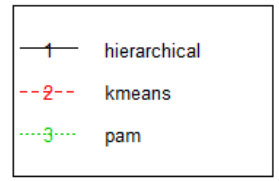
R 集群分群與衡量 - clValid


```
> cvpit = par( mfrow = c( 2, 2))
> (cvmit = matrix(c( 1, 2, 3, 4), 2, 2))
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> layout(cvmit)
> plot( intern.it, legend = FALSE)
> plot(nClusters(intern.it),
+      measures( intern.it, "Dunn")[,,1],
+      type = "n", axes = F, xlab = "", ylab = "")
> legend( "center", clusterMethods(intern.it),
+         col = 1:9, lty = 1:9, pch = paste(1:9))
> par(cvpit)
>
```









R 集群分群與衡量 - clValid

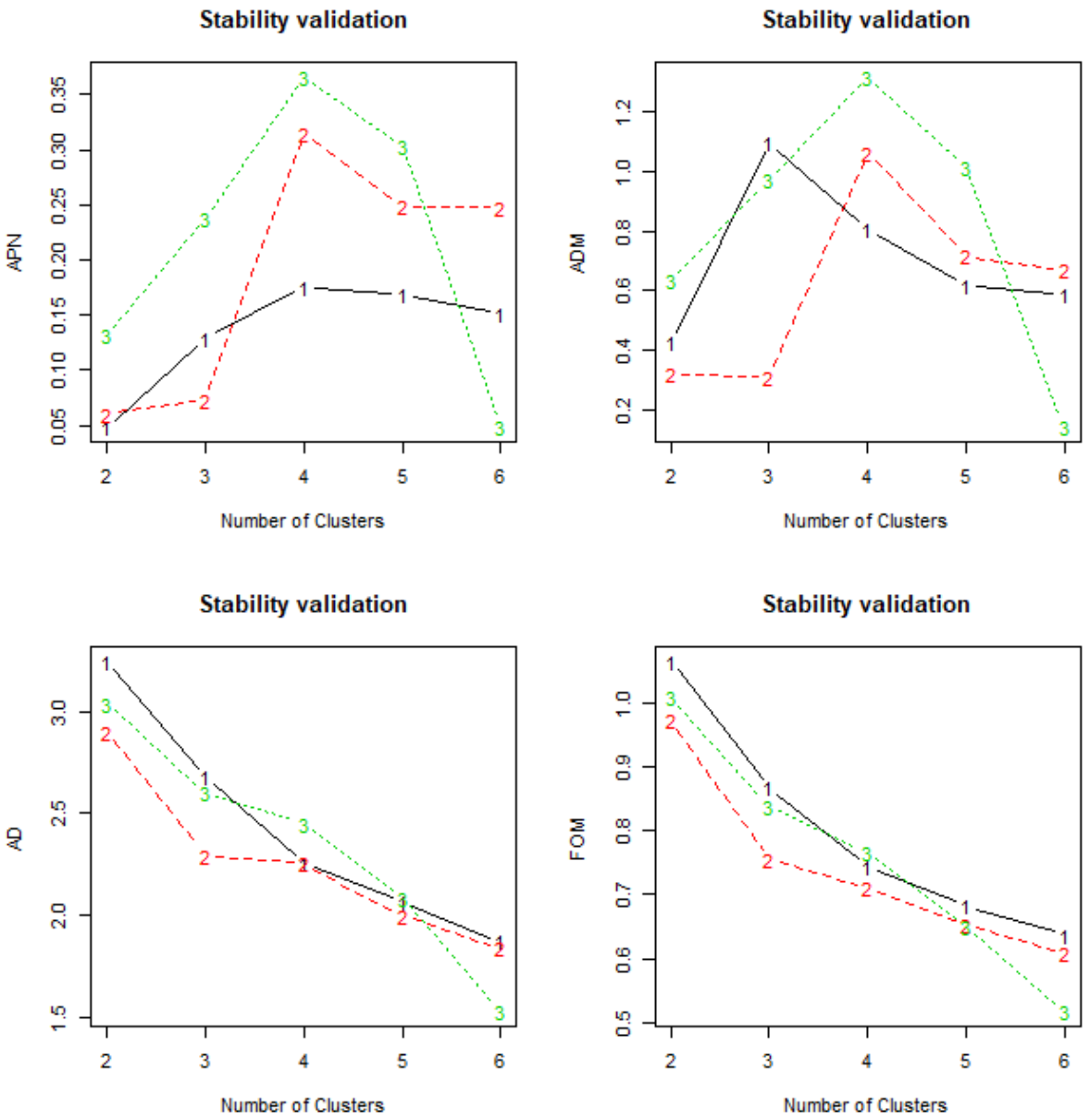
下面為 Stability Validation 的衡量分數。

在此將 valiation 參數設為 Stability。

```
> # Stability Validation
> intern.sb = clValid(express, 2:6,
+                     clMethods=c("hierarchical","kmeans","pam"),
+                     validation= "stability")
> summary(intern.sb)

Clustering Methods:
 hierarchical kmeans pam

Cluster sizes:
 2 3 4 5 6

Validation Measures:
                      2      3      4      5      6

hierarchical APN  0.0478 0.1288 0.1755 0.1689 0.1516
              AD  3.2430 2.6814 2.2571 2.0642 1.8732
              ADM 0.4283 1.0953 0.8070 0.6196 0.5867
              FOM 1.0658 0.8678 0.7451 0.6823 0.6371
kmeans        APN 0.0603 0.0726 0.3146 0.2485 0.2470
              AD  2.9001 2.2923 2.2529 1.9978 1.8389
              ADM 0.3196 0.3101 1.0621 0.7151 0.6700
              FOM 0.9745 0.7548 0.7114 0.6528 0.6074
pam           APN 0.1318 0.2376 0.3658 0.3029 0.0486
              AD  3.0382 2.5993 2.4492 2.0840 1.5272
              ADM 0.6372 0.9733 1.3172 1.0164 0.1401
              FOM 1.0092 0.8391 0.7663 0.6490 0.5158

Optimal Scores:

    Score  Method       Clusters
APN 0.0478 hierarchical 2
AD  1.5272 pam          6
ADM 0.1401 pam          6
FOM 0.5158 pam          6
```

R 集群分群與衡量 - clValid


```
> cvpsb = par( mfrow = c( 2, 3))
> (cvmsb = matrix(c( 1, 2, 3, 4, 5, 5), 2, 3))
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    5
> layout(cvmsb)
> plot( intern.sb, legend = FALSE)
> plot(nClusters(intern.sb),
+      measures( intern.sb, "AD")[,,1],
+      type = "n", axes = F, xlab = "", ylab = "")
> legend( "center", clusterMethods(intern.sb),
+         col = 1:9, lty = 1:9, pch = paste(1:9))
> par(cvpsb)
>
```



R 集群分群與衡量 - clValid

其實可以多個衡量方式放在一起。
P.S : validation 其實分為三個 "internal" 、 "stability" 、 "biological"。

```
> intern = clValid(express, 2:6,
+                  clMethods=c("hierarchical","kmeans","pam"),
+                  validation= c("internal", "stability"))
> intern

Call:
clValid(obj = express, nClust = 2:6, clMethods = c("hierarchical",
    "kmeans", "pam"), validation = c("internal", "stability"))

Clustering Methods:
 hierarchical kmeans pam

Cluster sizes:
 2 3 4 5 6

Validation measures:
 APN AD ADM FOM Connectivity Dunn Silhouette
```

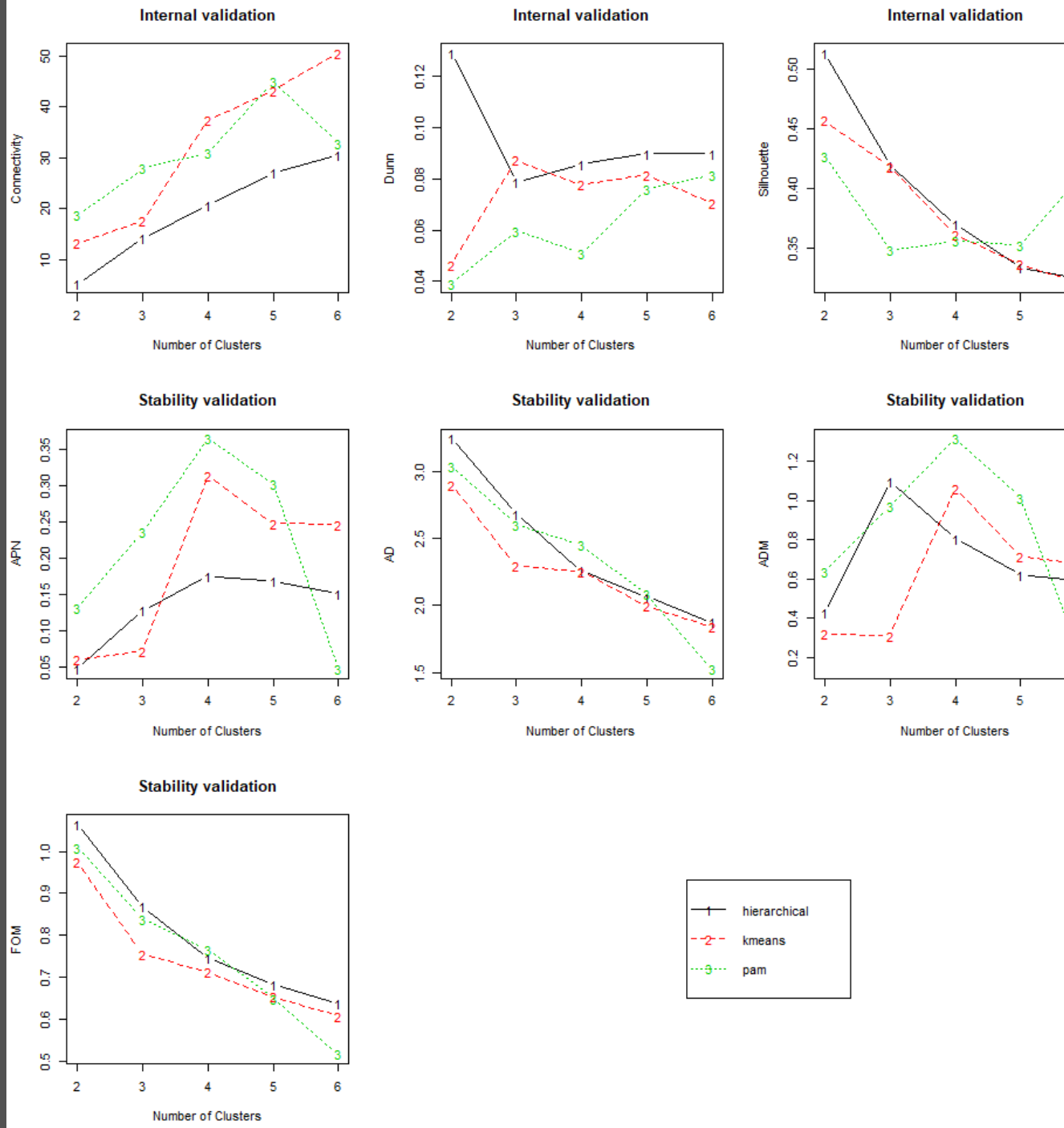
R 集群分群與衡量 - clValid

其實可以多個衡量方式放在一起。
P.S : validation 其實分為三個 "internal" 、 "stability" 、 "biological"。


```
> summary(intern)

Clustering Methods:
 hierarchical kmeans pam

Cluster sizes:
 2 3 4 5 6

Validation Measures:
                                 2       3       4       5       6

hierarchical APN            0.0478  0.1288  0.1755  0.1689  0.1516
             AD             3.2430  2.6814  2.2571  2.0642  1.8732
             ADM            0.4283  1.0953  0.8070  0.6196  0.5867
             FOM            1.0658  0.8678  0.7451  0.6823  0.6371
             Connectivity   5.3270 14.2528 20.7520 27.0726 30.6194
             Dunn           0.1291  0.0788  0.0857  0.0899  0.0899
             Silhouette     0.5133  0.4195  0.3700  0.3343  0.3233
kmeans       APN            0.0603  0.0726  0.3146  0.2485  0.2470
             AD             2.9001  2.2923  2.2529  1.9978  1.8389
             ADM            0.3196  0.3101  1.0621  0.7151  0.6700
             FOM            0.9745  0.7548  0.7114  0.6528  0.6074
             Connectivity  13.2548 17.6651 37.3980 43.2655 50.6095
             Dunn           0.0464  0.0873  0.0777  0.0815  0.0703
             Silhouette     0.4571  0.4182  0.3615  0.3367  0.3207
pam          APN            0.1318  0.2376  0.3658  0.3029  0.0486
             AD             3.0382  2.5993  2.4492  2.0840  1.5272
             ADM            0.6372  0.9733  1.3172  1.0164  0.1401
             FOM            1.0092  0.8391  0.7663  0.6490  0.5158
             Connectivity  18.7917 27.9651 30.9302 44.9671 32.9667
             Dunn           0.0391  0.0597  0.0510  0.0761  0.0816
             Silhouette     0.4271  0.3489  0.3563  0.3530  0.4152

Optimal Scores:

             Score  Method       Clusters
APN          0.0478 hierarchical 2
AD           1.5272 pam          6
ADM          0.1401 pam          6
FOM          0.5158 pam          6
Connectivity 5.3270 hierarchical 2
Dunn         0.1291 hierarchical 2
Silhouette   0.5133 hierarchical 2
```

R 集群分群與衡量 - clValid

其實可以多個衡量方式放在一起。
P.S : validation 其實分為三個 "internal"、"stability"、"biological"。



R 集群分群與衡量 - clValid


```
# install.packages("clValid")
library("clValid")
data(mouse)
head(mouse)
NROW(mouse)
express = mouse[,c("M1","M2","M3","NC1","NC2","NC3")]
rownames(express) = mouse$ID
head(express)
# Internal Validation
intern.it = clValid(express, 2:6,
   clMethods=c("hierarchical","kmeans","pam"),
   validation= "internal")
summary(intern.it)
cvpit = par( mfrow = c( 2, 2))
(cvmit = matrix(c( 1, 2, 3, 4), 2, 2))
layout(cvmit)
plot( intern.it, legend = FALSE)
plot(nClusters(intern.it),
   measures( intern.it, "Dunn")[,,1],
   type = "n", axes = F, xlab = "", ylab = "")
legend( "center", clusterMethods(intern.it),
col = 1:9, lty = 1:9, pch = paste(1:9))
par(cvpit)
```

```
# Stability Validation
intern.sb = clValid(express, 2:6,
   clMethods=c("hierarchical","kmeans","pam"),
   validation= "stability")
summary(intern.sb)
cvpsb = par( mfrow = c( 2, 3))
(cvmsb = matrix(c( 1, 2, 3, 4, 5, 5), 2, 3))
layout(cvmsb)
plot( intern.sb, legend = FALSE)
plot(nClusters(intern.sb),
   measures( intern.sb, "AD")[,,1],
   type = "n", axes = F, xlab = "", ylab = "")
legend( "center", clusterMethods(intern.sb),
col = 1:9, lty = 1:9, pch = paste(1:9))
par(cvpsb)
intern = clValid(express, 2:6,
   clMethods=c("hierarchical","kmeans","pam"),
   validation= c("internal", "stability"))
intern
summary(intern)
cvplot = par( mfrow = c( 3, 3))
(cvmat = matrix(c(1,4,7,2,5,8,3,6,8), 3, 3))
layout(cvmat)
plot( intern.it, legend = FALSE)
plot( intern.sb, legend = FALSE)
plot(nClusters(intern.sb),
   measures( intern.sb, "AD")[,,1],
   type = "n", axes = F, xlab = "", ylab = "")
legend( "center", clusterMethods(intern.sb),
col = 1:9, lty = 1:9, pch = paste(1:9))
par(cvplot)
```

R 資料分群 kmeans 與 cluster

下面群集分析的方法分類

1. 階層式群集分析

2. 分割式群集分析

3. 密度為基礎

此章分為三部分

1. K-means 分群

2. PAM 分割環繞物件 - K-medoids 分析

3. 階層分群法 by cluster

R 資料分群 kmeans 與 cluster

常用階層式集群分析的方法如下

1. 單一連結法 - single linkage method
定義 : 兩群集間資料點中之最小距離來表示。

2. 完全連結法 - complete linkage method
定義 : 是以兩群集間資料點之最大距離來表示兩群集的距離及兩群資料的鄰近程度。

3. 平均連結法 - average linkage method
定義 : 衡量群集內所有點至另一群集內所有點的距離平均來表示兩群集的鄰近程度，以避免群集間的距離衡量受雜訊影響。

4. 中心點連結法 - centroid linked method
定義 : 以兩群群中心點距離作為衡量兩群集的距離，以表示其鄰近程度。
可解決距離衡量對異常值過於敏感的問題，好進一步處理類別型的資料

5. 華德法 - Ward's method
定義 : 是衡量組內 within-cluster 變異做為衡量群集相似度的分群方法。

依序將所有群集合併，反覆計算與合併每階段最小群集的組內變異，直至所有資料均合併為一群為止。

R 資料分群 kmeans 與 cluster

1. K-means 分群

利用加州大學的機器學習數據集中的葡萄酒資料作為範例，以此進行 K-means 分群

```
> wine = read.table("http://jaredlander.com/data/wine.csv", header = TRUE, sep = ",")
> head(wine)
  Cultivar Alcohol Malic.acid  Ash Alcalinity.of.ash Magnesium Total.phenols Flavanoids
1        1   14.23       1.71 2.43               15.6       127          2.80       3.06
2        1   13.20       1.78 2.14               11.2       100          2.65       2.76
3        1   13.16       2.36 2.67               18.6       101          2.80       3.24
4        1   14.37       1.95 2.50               16.8       113          3.85       3.49
5        1   13.24       2.59 2.87               21.0       118          2.80       2.69
6        1   14.20       1.76 2.45               15.2       112          3.27       3.39
  OD280.OD315.of.diluted.wines Proline
1                         3.92    1065
2                         3.40    1050
3                         3.17    1185
4                         3.45    1480
5                         2.93     735
6                         2.85    1450
>
> wineTrain = wine[, which(names(wine) != "Cultivar")]
> set.seed(278613)
> wineK3 = kmeans(x = wineTrain, centers = 3)
>
```

R 資料分群 kmeans 與 cluster

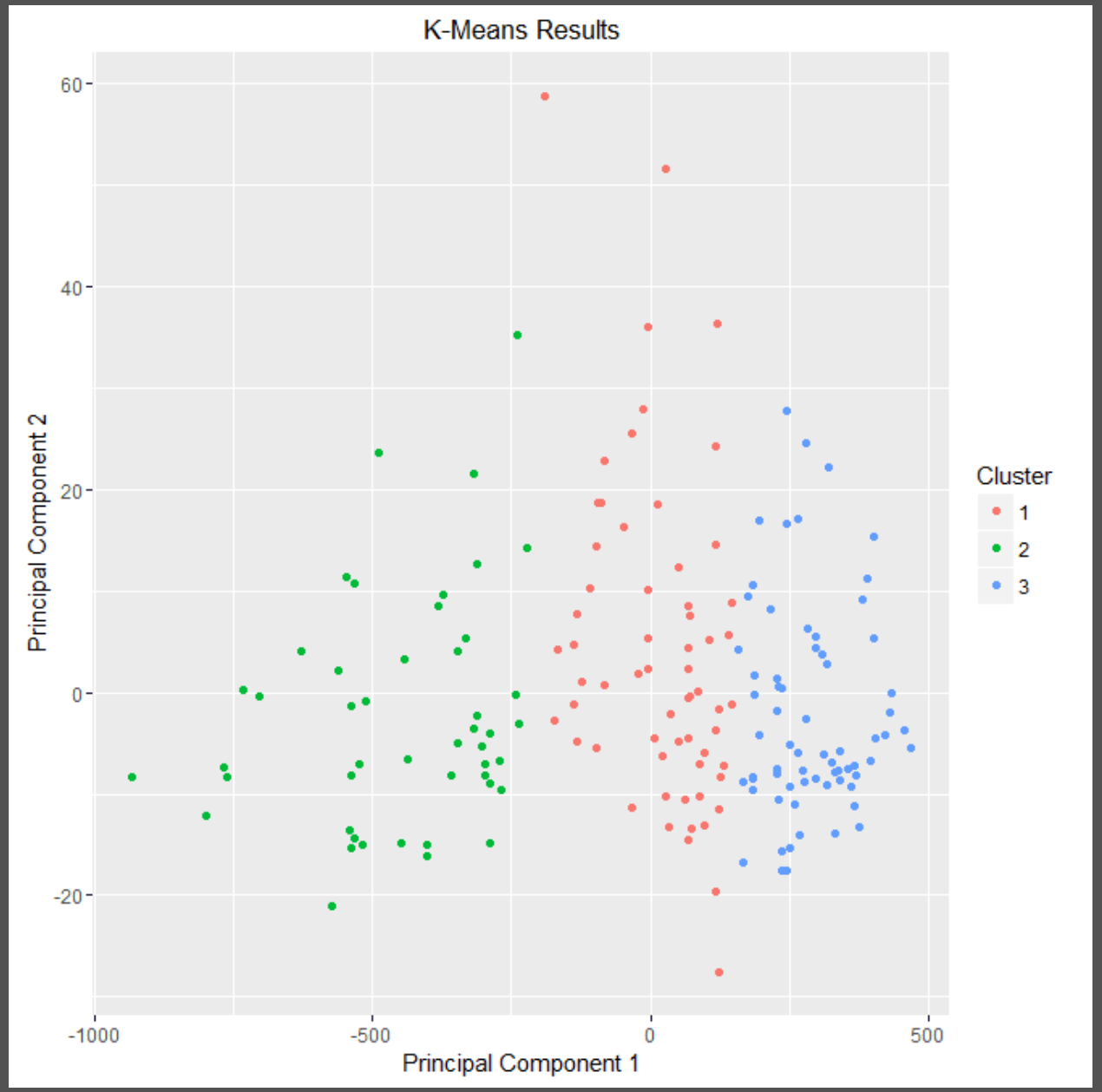

```
> wineK3
K-means clustering with 3 clusters of sizes 62, 47, 69

Cluster means:
   Alcohol Malic.acid      Ash Alcalinity.of.ash Magnesium Total.phenols
1 12.92984   2.504032 2.408065          19.89032 103.59677      2.111129
2 13.80447   1.883404 2.426170          17.02340 105.51064      2.867234
3 12.51667   2.494203 2.288551          20.82319  92.34783      2.070725
  Flavanoids Nonflavanoid.phenols Proanthocyanins Color.intensity
1   1.584032            0.3883871        1.503387        5.650323
2   3.014255            0.2853191        1.910426        5.702553
3   1.758406            0.3901449        1.451884        4.086957
        Hue OD280.OD315.of.diluted.wines   Proline
1 0.8839677                     2.365484  728.3387
2 1.0782979                     3.114043 1195.1489
3 0.9411594                     2.490725  458.2319

Clustering vector:
  [1] 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 2 2 1 1 2 2 1 2 2 2 2 2 2
 [36] 1 1 2 2 1 1 2 2 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 1 3 1 3 3 1 3 3 1 1
 [71] 1 3 3 2 1 3 3 3 1 3 3 1 1 3 3 3 3 3 1 1 3 3 3 3 3 1 1 3 1 3 1 3 3 3 1
[106] 3 3 3 3 1 3 3 1 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 1 3 3 1 1 1 1 3 3 3
[141] 1 1 3 3 1 1 3 1 1 3 3 3 3 1 1 1 3 1 1 1 3 1 3 1 1 3 1 1 1 1 3 3 1 1 1
[176] 1 1 3

Within cluster sum of squares by cluster:
[1]  566572.5 1360950.5  443166.7
 (between_SS / total_SS =  86.5 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"
[5] "tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"
> |
```

```
wine = read.table("http://jaredlander.com/data/wine.csv", header = TRUE, sep = ",")
head(wine)
wineTrain = wine[, which(names(wine) != "Cultivar")]
set.seed(278613)
wineK3 = kmeans(x = wineTrain, centers = 3)
wineK3
```

R 資料分群 kmeans 與 cluster

在此使用 useful 套件，將處理好的資料(wineTrain)進行繪圖。

```
> require(useful)
Loading required package: useful
Loading required package: ggplot2
> plot(wineK3, data = wineTrain)
>
```



R 資料分群
kmeans 與 cluster

```
require(useful)
plot(wineK3, data = wineTrain)
plot(wineK3, data = wine, class = "Cultivar")
```


R 資料分群 kmeans 與 cluster

在這裡要注意的是 nstart 引數，可以用不同的初始條件進行分群。

```
> set.seed(278613)
> wineK3N25 = kmeans(wineTrain, centers = 3, nstart = 25)
> wineK3$size
[1] 62 47 69
> wineK3N25$size
[1] 62 47 69
>
```

set.seed(278613)
wineK3N25 = kmeans(wineTrain, centers = 3, nstart = 25)
wineK3$size
wineK3N25$size

R 資料分群 kmeans 與 cluster

這裡是利用 useful 套件中的 FitMeans 進行 k個分群群內平方和與 k+1 個分群群內平方和的比例。

```
> wineBest = FitKMeans(wineTrain, max.clusters=20, nstart=25, seed=278613)
> wineBest
   Clusters   Hartigan AddCluster
1         2 505.429310       TRUE
2         3 160.411331       TRUE
3         4 135.707228       TRUE
4         5  78.445289       TRUE
5         6  71.489710       TRUE
6         7  97.582072       TRUE
7         8  46.772501       TRUE
8         9  33.198650       TRUE
9        10  33.277952       TRUE
10       11  33.465424       TRUE
11       12  17.940296       TRUE
12       13  33.268151       TRUE
13       14   6.434996      FALSE
14       15   7.833562      FALSE
15       16  46.783444       TRUE
16       17  12.229408       TRUE
17       18  10.261821       TRUE
18       19 -13.576343      FALSE
19       20  56.373939       TRUE
> |
```

R 資料分群 kmeans 與 cluster

階層式集群

hcad() 是輸出產生的分群結果， hcadpic() 是輸出分群的樹狀圖，預設分群數(hck)為 5，預設方法(hcm)為華德法，預設距離(dism)為 歐式距離。

預設分群方法如下 :

"ward.D", "single", "complete", "average", "mcquitty", "median", "centroid", "ward.D2"

距離求算方法如下 :

"euclidean", "maximum", "manhattan", "canberra", "binary" , "minkowski"

R 資料分群 kmeans 與 cluster

```
# HCA
hcad = function( hcdata, hck = 5, hcm = "ward.D", dism = "euclidean", ...){
    hcdif = dist( hcdata, method = dism)
    hctf = hclust( hcdif, method = hcm)
    hctrf = cutree( hctf, k = hck)
    hctrf
}

# HCA PIC
hcadpic = function( hcdata, hck = 5, hcm = "ward.D", dism = "euclidean", ...){
    hcdif = dist( hcdata, method = dism)
    hctf = hclust( hcdif, method = hcm)
    plot(hctf)
    rect.hclust( hctf, k = hck, border = "red")
}
```

R 資料分群 kmeans 與 cluster

將 iris 資料集下的第 5 直行刪去，因為資料類別為字串不利於分析。

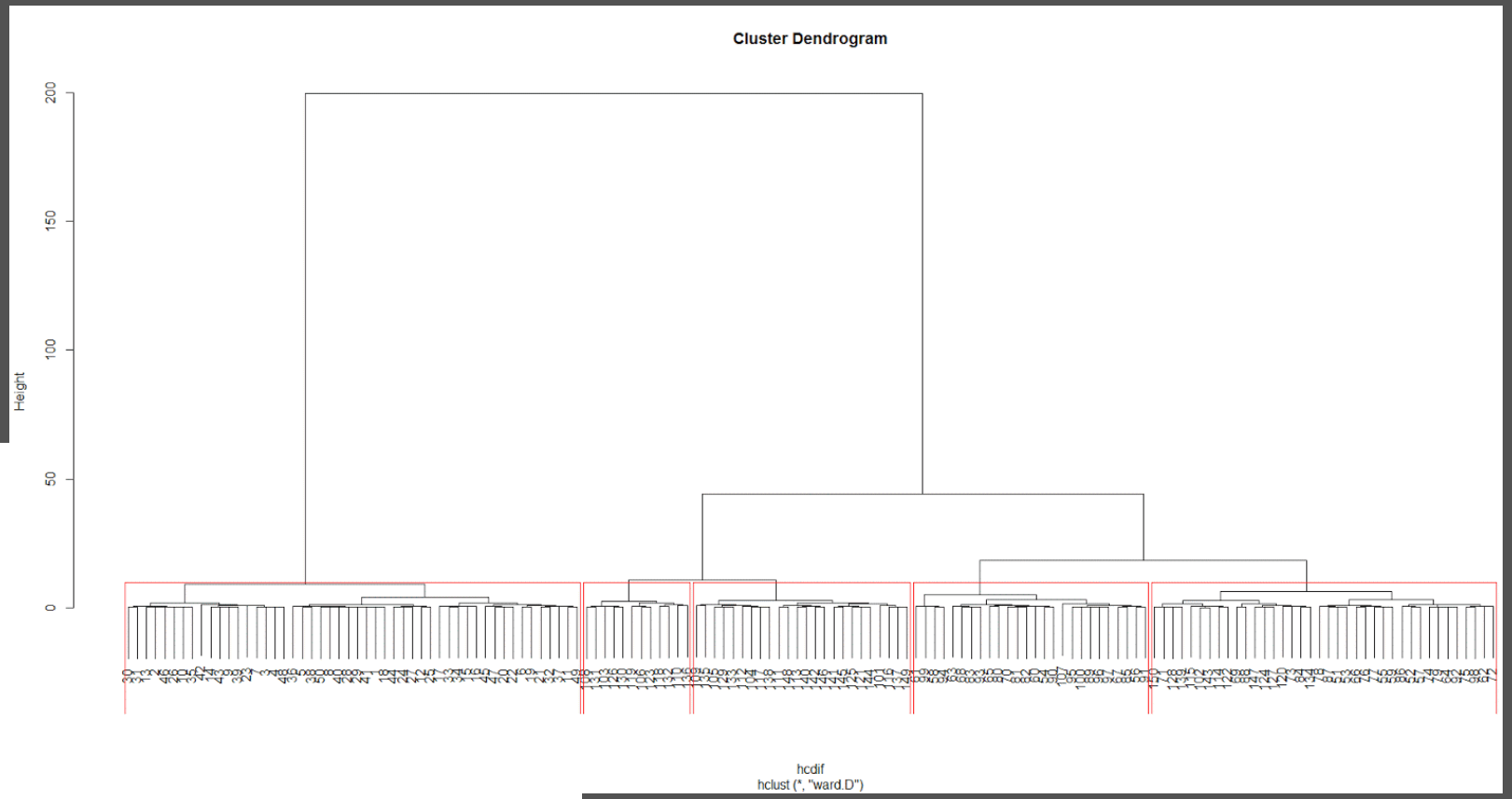
```
Irist = iris[,-c(5)]
head(irist)
hcad(irist)
hcad(irist, hck = 6)
```

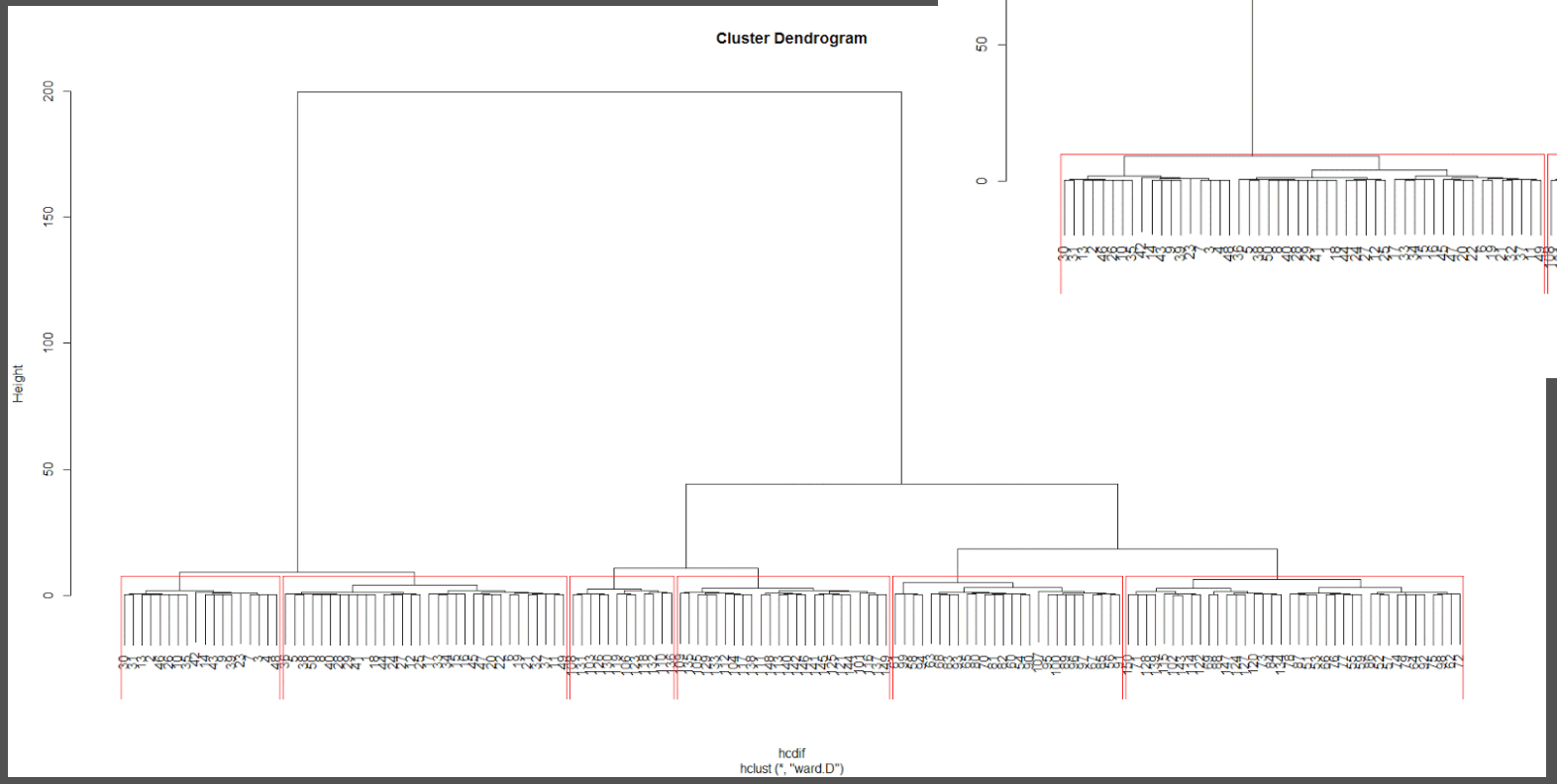
```
> irist = iris[,-c(5)]
> head(irist)
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1          5.1         3.5          1.4         0.2
2          4.9         3.0          1.4         0.2
3          4.7         3.2          1.3         0.2
4          4.6         3.1          1.5         0.2
5          5.0         3.6          1.4         0.2
6          5.4         3.9          1.7         0.4
>
```

```
> hcad(irist)
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [39] 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 3 2 3 2 3 2 3 3 2 3 2 3 2 3 3 2 3 2 2 2 2 2 2
 [77] 2 2 2 3 3 3 3 2 3 2 2 2 3 3 3 2 3 3 3 3 3 2 3 3 4 2 5 4 4 5 3 5 4 5 4 4 4 2
[115] 2 4 4 5 5 2 4 2 5 2 4 5 2 2 4 5 5 5 4 2 4 5 4 4 2 4 4 4 2 4 4 4 2 4 4 2
> hcad(irist, hck = 6)
  [1] 1 2 2 2 1 1 2 1 2 2 1 1 2 2 1 1 1 1 1 1 1 1 2 1 1 2 1 1 1 2 2 1 1 1 2 1 1 1
 [39] 2 1 1 2 2 1 1 2 1 2 1 1 3 3 3 4 3 4 3 4 3 4 4 3 4 3 4 3 4 4 3 4 3 3 3 3 3 3
 [77] 3 3 3 4 4 4 4 3 4 3 3 3 4 4 4 3 4 4 4 4 4 3 4 4 5 3 6 5 5 6 4 6 5 6 5 5 5 3
[115] 3 5 5 6 6 3 5 3 6 3 5 6 3 3 5 6 6 6 5 3 5 6 5 5 3 5 5 5 3 5 5 5 3 5 5 3
>
```

R 資料分群 kmeans 與 cluster

hcadpic(irist)
hcadpic(irist, hck = 6)





R DBSCAN 集群方法

DBSCAN，英文全寫為Density-based spatial clustering of applications with noise，是在 1996 年由 Martin Ester, Hans-Peter Kriegel, Jörg Sander 及 Xiaowei Xu 提出的聚類分析算法， 這個算法是以密度為本的：給定某空間裡的一個點集合，這算法能把附近的點分成一組（有很多相鄰點的點），並標記出位於低密度區域的局外點（最接近它的點也十分遠），DBSCAN 是其中一個最常用的聚類分析算法，也是其中一個科學文章中最常引用的。在 2014 年，這個算法在領頭數據挖掘會議 KDD 上獲頒發了 Test of Time award，該獎項是頒發給一些於理論及實際層面均獲得持續性的關注的算法。

https://en.wikipedia.org/wiki/DBSCAN

相較於階層式集群分析、KMean、PAM 不同，不用設定一開始所需要的集群分群數量 !!!

安裝 factoextra 套件

```
install.packages("factoextra")
library("factoextra")
```

```
> install.packages("factoextra")
also installing the dependency 'ggpubr'

嘗試 URL 'https://cloud.r-project.org/bin/windows/contrib/3.4/ggpubr_0.1.4.zip'
Content type 'application/zip' length 1473576 bytes (1.4 MB)
downloaded 1.4 MB

嘗試 URL 'https://cloud.r-project.org/bin/windows/contrib/3.4/factoextra_1.0.4.zip'
Content type 'application/zip' length 254773 bytes (248 KB)
downloaded 248 KB

package 'ggpubr' successfully unpacked and MD5 sums checked
package 'factoextra' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\USER\AppData\Local\Temp\RtmpGK95MF\downloaded_packages
> library("factoextra")
Warning message:
package 'factoextra' was built under R version 3.4.1
>
```

R DBSCAN 集群方法

在此利用 multishapes 資料集

```
data(multishapes)
head(multishapes)
df = multishapes[, 1:2]
head(df)
```

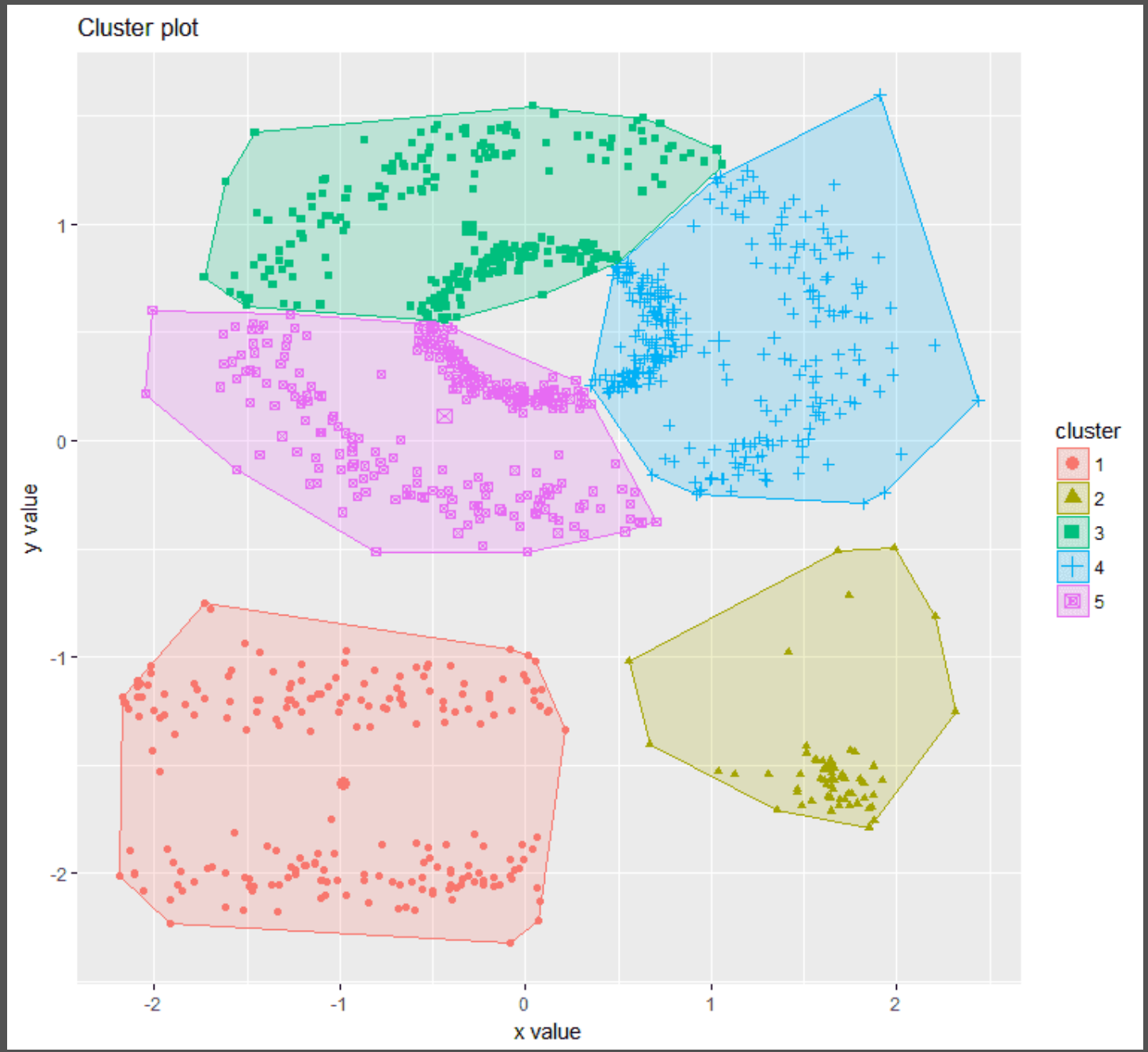
這邊會先用 KMeans 進行分群與資料的呈現

```
set.seed(929)
km.res = kmeans(df, 5, nstart = 25)
fviz_cluster(km.res, df, frame = FALSE, geom = "point")
```

```
> data(multishapes)
> head(multishapes)
           x          y shape
1 -0.8037393 -0.8530526     1
2  0.8528507  0.3676184     1
3  0.9271795 -0.2749024     1
4 -0.7526261 -0.5115652     1
5  0.7068462  0.8106792     1
6  1.0346985  0.3946550     1
>
> df = multishapes[, 1:2]
> head(df)
           x          y
1 -0.8037393 -0.8530526
2  0.8528507  0.3676184
3  0.9271795 -0.2749024
4 -0.7526261 -0.5115652
5  0.7068462  0.8106792
6  1.0346985  0.3946550
>
```

```
> set.seed(929)
> km.res = kmeans(df, 5, nstart = 25)
> fviz_cluster(km.res, df, frame = FALSE, geom = "point")
Warning message:
argument frame is deprecated; please use ellipse instead.
>
>
```

R DBSCAN 集群方法



R DBSCAN 集群方法

安裝 fcp & dbscan 套件

install.packages("fpc")
install.packages("dbscan")

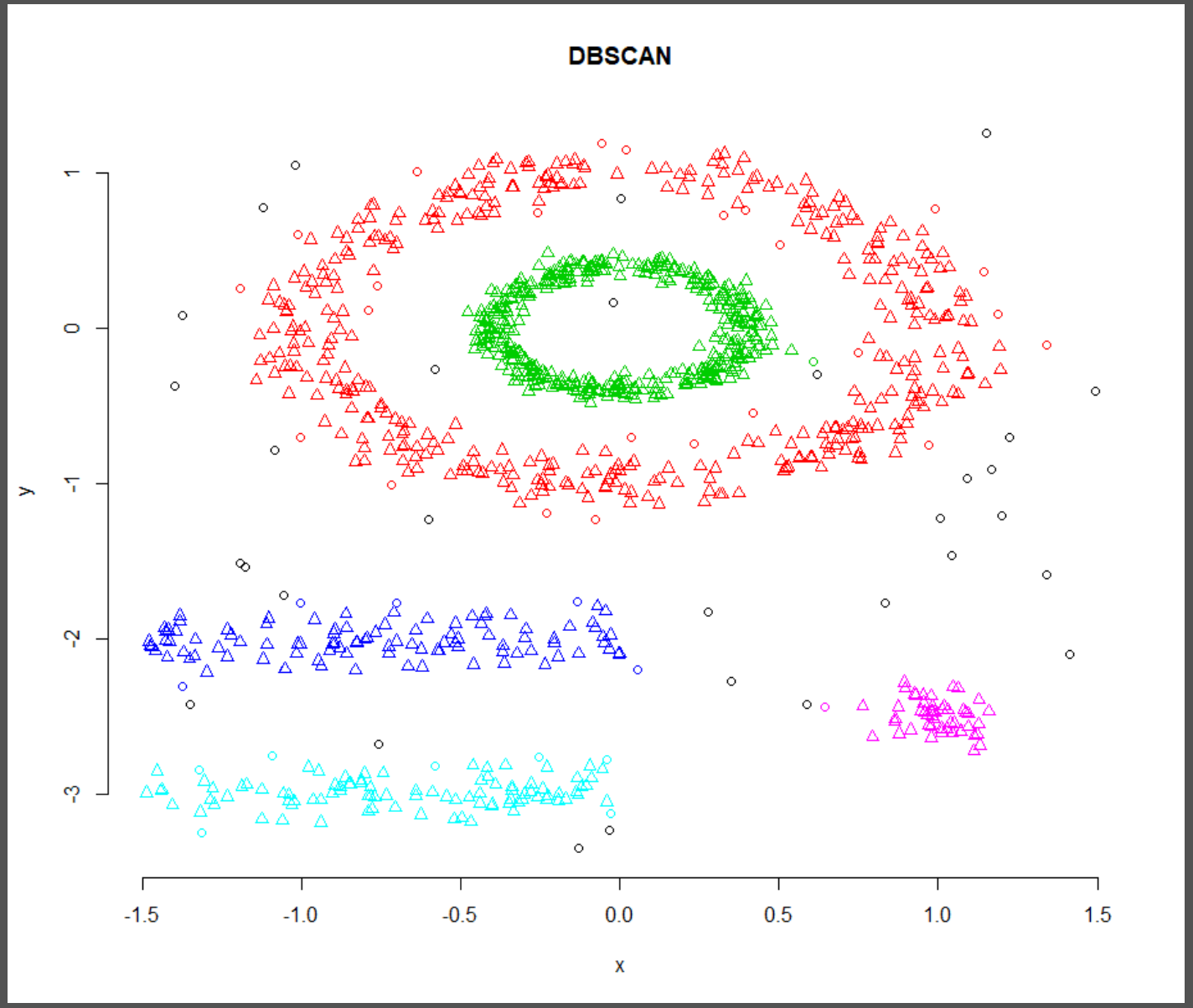
```
> install.packages("fpc")
嘗試 URL 'https://cloud.r-project.org/bin/windows/contrib/3.4/fpc_2.1-10.zip'
Content type 'application/zip' length 437188 bytes (426 KB)
downloaded 426 KB

package ‘fpc’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\USER\AppData\Local\Temp\Rtmp8uC5yj\downloaded_packages
> install.packages("dbscan")
嘗試 URL 'https://cloud.r-project.org/bin/windows/contrib/3.4/dbscan_1.1-1.zip'
Content type 'application/zip' length 2648928 bytes (2.5 MB)
downloaded 2.5 MB

package ‘dbscan’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\USER\AppData\Local\Temp\Rtmp8uC5yj\downloaded_packages
>
```

R DBSCAN 集群方法

在此利用 fpc 套件，可以看到 DBSCAN 集群的呈現

```
library("fpc")
db = fpc::dbscan(df, eps = 0.15, MinPts = 5)
plot(db, df, main = "DBSCAN", frame = FALSE)
```

```
> library("fpc")
> db = fpc::dbscan(df, eps = 0.15, MinPts = 5)
> plot(db, df, main = "DBSCAN", frame = FALSE)
>
```

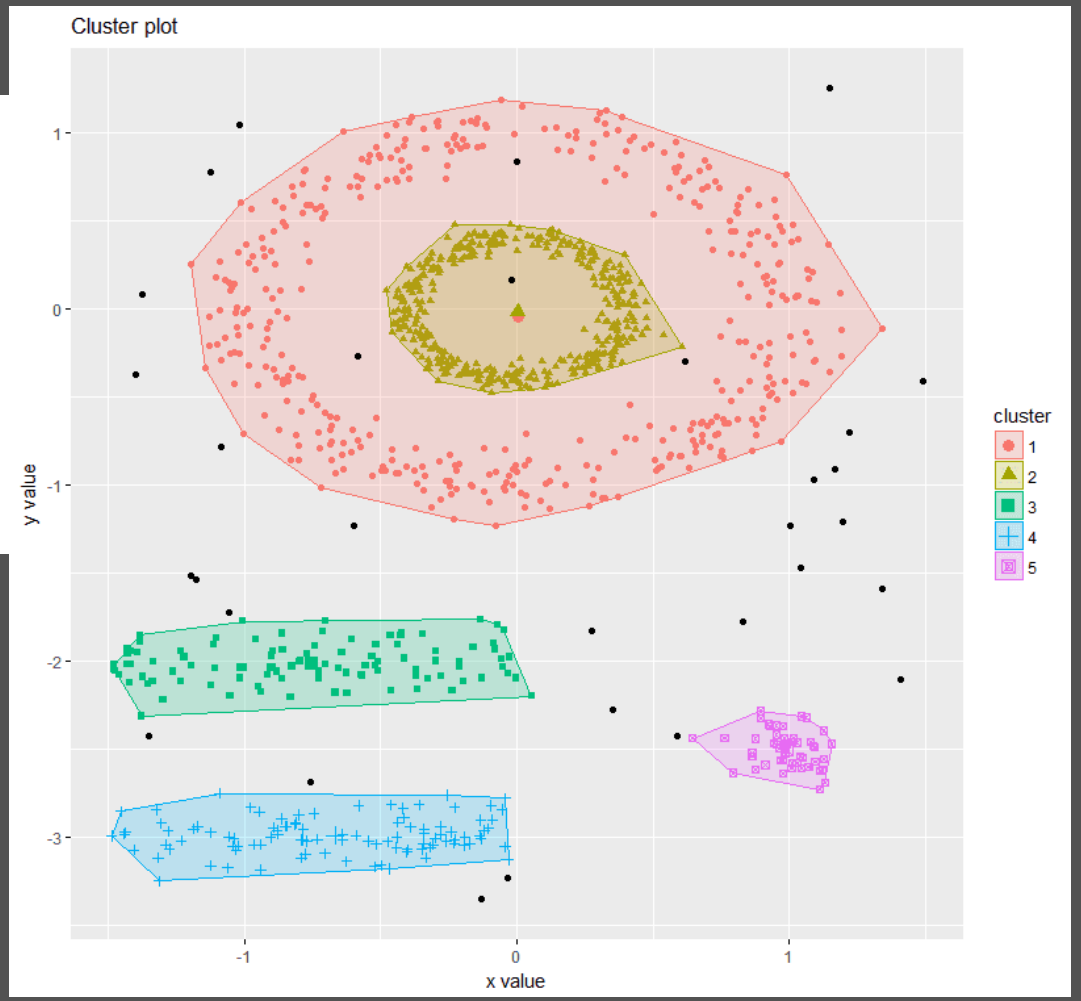


R DBSCAN 集群方法

factoextra 對 DBSCAN 套件的呈現，從這裡可以看出與 KMeans 等其他集群差異。

```
library("factoextra")
fviz_cluster(db, df, stand = FALSE, frame = FALSE, geom = "point")
print(db)
```

```
> library("factoextra")
> fviz_cluster(db, df, stand = FALSE, frame = FALSE, geom = "point")
Warning message:
argument frame is deprecated; please use ellipse instead.
> print(db)
dbscan Pts=1100 MinPts=5 eps=0.15
         0   1   2   3  4  5
border 31  24   1   5  7  1
seed    0 386 404  99 92 50
total  31 410 405 104 99 51
>
```



R DBSCAN 集群方法

db$cluster 可以觀察集群分群的資料

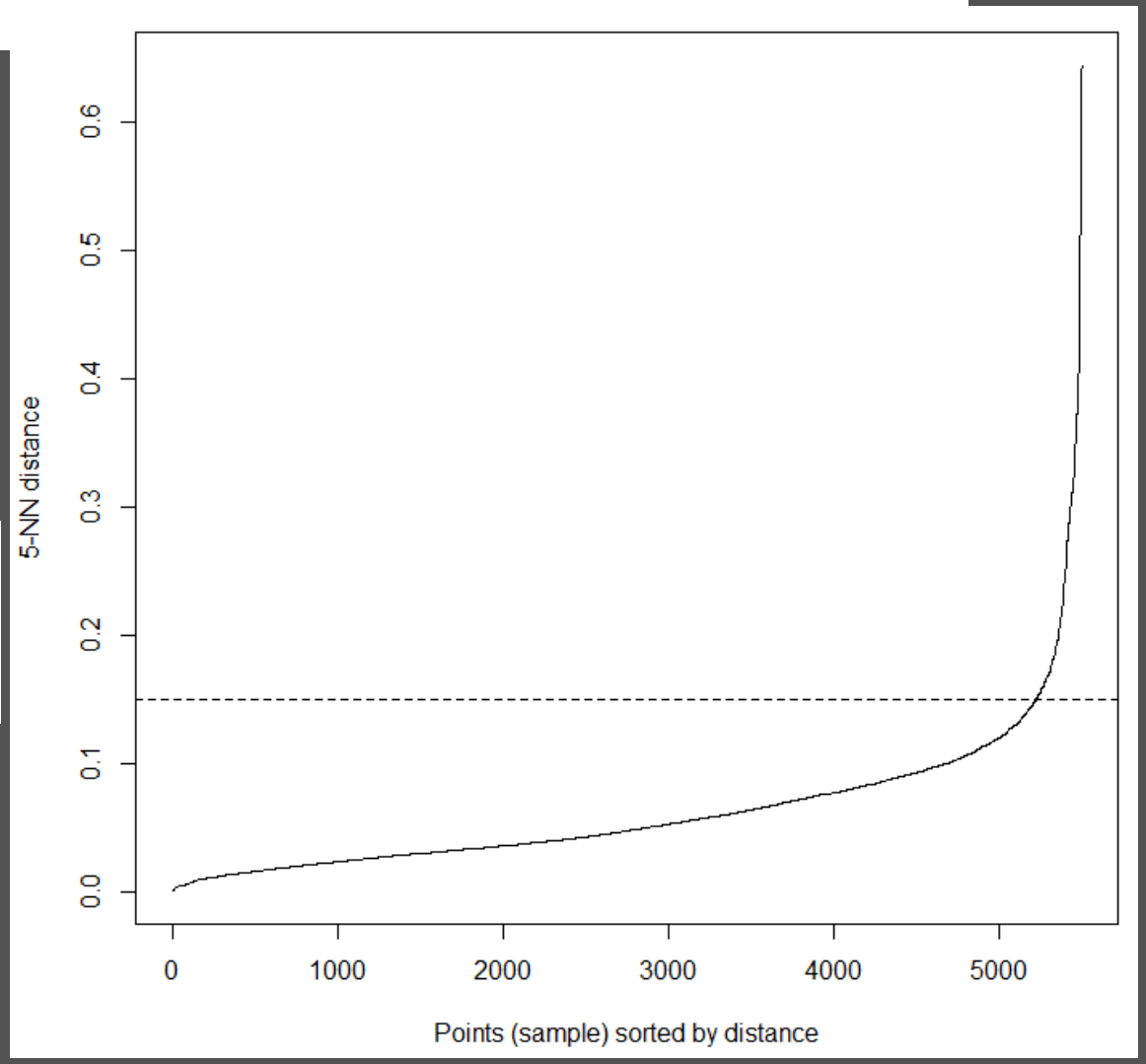
NROW(db$cluster)
db$cluster[sample(1:1100, 50)]

```
> NROW(db$cluster)
[1] 1100
> db$cluster[sample(1:1100, 50)]
 [1] 2 2 3 1 2 2 2 4 2 1 2 0 1 1 2 1 2 1 1 5 2 3 1 5
>
```

判斷最佳距離

dbscan::kNNdistplot(df, k = 5)
abline(h = 0.15, lty = 2)

```
> dbscan::kNNdistplot(df, k =  5)
> abline(h = 0.15, lty = 2)
>
```



R DBSCAN 集群方法

在此使用 iris 資料集做示範

```
data("iris")
head(iris)
iris2 = as.matrix(iris[, 1:4])
head(iris2)
```

```
> data("iris")
> head(iris)
  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
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

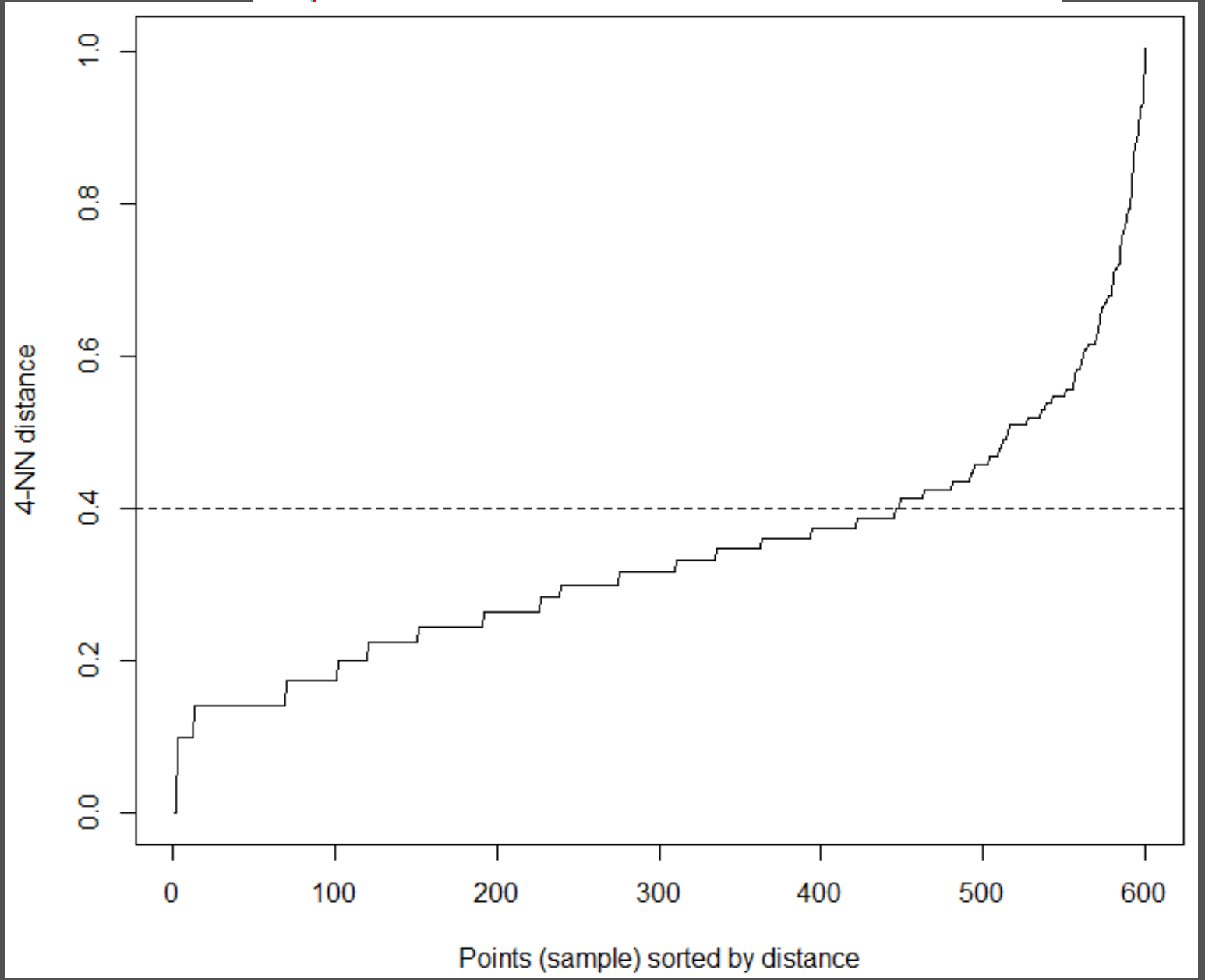
```
> iris2 = as.matrix(iris[, 1:4])
> head(iris2)
     Sepal.Length Sepal.Width Petal.Length Petal.Width
[1,]          5.1         3.5          1.4         0.2
[2,]          4.9         3.0          1.4         0.2
[3,]          4.7         3.2          1.3         0.2
[4,]          4.6         3.1          1.5         0.2
[5,]          5.0         3.6          1.4         0.2
[6,]          5.4         3.9          1.7         0.4
>
```

R DBSCAN 集群方法

判斷合適的距離

dbscan::kNNdistplot(iris2, k = 4)
abline(h = 0.4, lty = 2)

```
> dbscan::kNNdistplot(iris2, k =  4)
> abline(h = 0.4, lty = 2)
>
```



R DBSCAN 集群方法

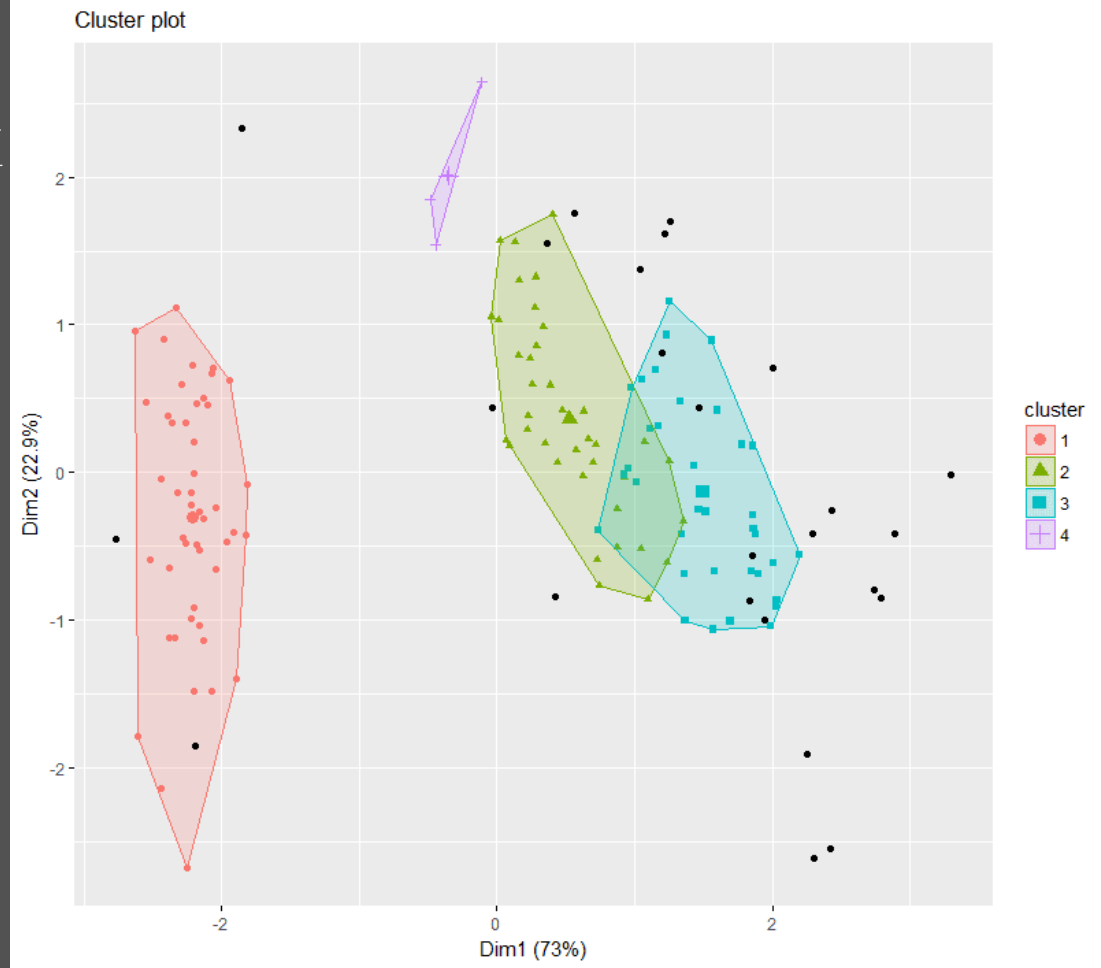
在此判斷 fpc 與 dbscan 套件函數在 dbscan() 兩者集群分群的結果。

```
res.fpc = fpc::dbscan(iris2, eps = 0.4, MinPts = 4)
res.db = dbscan::dbscan(iris2, 0.4, 4)
all(res.fpc$cluster == res.db$cluster)
```

可以看到繪圖中的黑色圓點為並未分群的異常值

```
fviz_cluster(res.fpc, iris2, geom = "point")
```

```
> res.fpc = fpc::dbscan(iris2, eps = 0.4, MinPts = 4)
> res.db = dbscan::dbscan(iris2, 0.4, 4)
> all(res.fpc$cluster == res.db$cluster)
[1] TRUE
```



R FactoMineR 主成分分析與階層式集群

1. 主成分分析

在多元統計分析中，主成分分析（英語：Principal components analysis，PCA）是一種分析、簡化數據集的技術。主成分分析經常用於減少數據集的維數，同時保持數據集中的對變異數貢獻最大的特徵。這是通過保留低階主成分，忽略高階主成分做到的。這樣低階成分往往能夠保留住數據的最重要方面。但這也不是一定的，要視具體應用而定。由於主成分分析依賴所給數據，所以數據的準確性對分析結果影響很大。

https://en.wikipedia.org/wiki/Principal_component_analysis

2.聚類分析 (集群分析)
聚類分析（英語：Cluster analysis，亦稱為群集分析）是對於統計數據分析的一門技術，在許多領域受到廣泛應用，包括機器學習，數據挖掘，模式識別，圖像分析以及生物信息。聚類是把相似的對象通過靜態分類的方法分成不同的組別或者更多的子集（subset），這樣讓在同一個子集中的成員對象都有相似的一些屬性，常見的包括在坐標系中更加短的空間距離等。一般把數據聚類歸納為一種非監督式學習。

https://en.wikipedia.org/wiki/Cluster_analysis

R FactoMineR 主成分分析與階層式集群

install.packages("FactoMineR")

```
> install.packages("FactoMineR")
also installing the dependencies 'minqa', 'nloptr', 'RcppEigen', 'lme4', 'SparseM', 'MatrixModels', 'pbkrtest', $

嘗試 URL 'https://cloud.r-project.org/bin/windows/contrib/3.4/minqa_1.2.4.zip'
Content type 'application/zip' length 667937 bytes (652 KB)
downloaded 652 KB

嘗試 URL 'https://cloud.r-project.org/bin/windows/contrib/3.4/nloptr_1.0.4.zip'
Content type 'application/zip' length 1173094 bytes (1.1 MB)
```

在此匯入套件，並選擇使用 iris 作為範例資料。

library("FactoMineR")
data(iris)
head(iris)

```
> library("FactoMineR")
Warning message:
package 'FactoMineR' was built under R version 3.4.1
>
> data(iris)
> head(iris)
  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
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
>
```

R FactoMineR 主成分分析與階層式集群

在此使用 PCA 進行分析，並利用 summary 進行觀察。

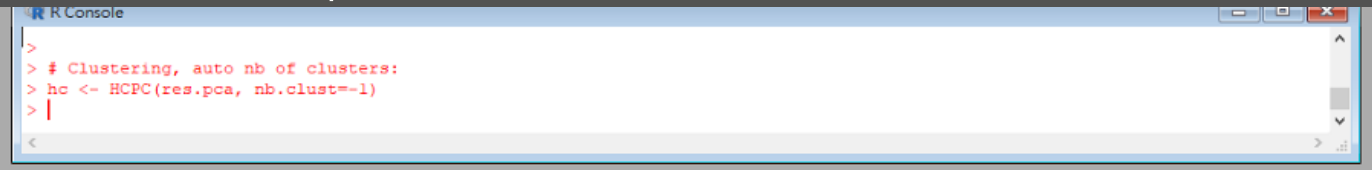
# Principal Component Analysis:
res.pca <- PCA(iris[,1:4], graph=FALSE)
summary(res.pca)

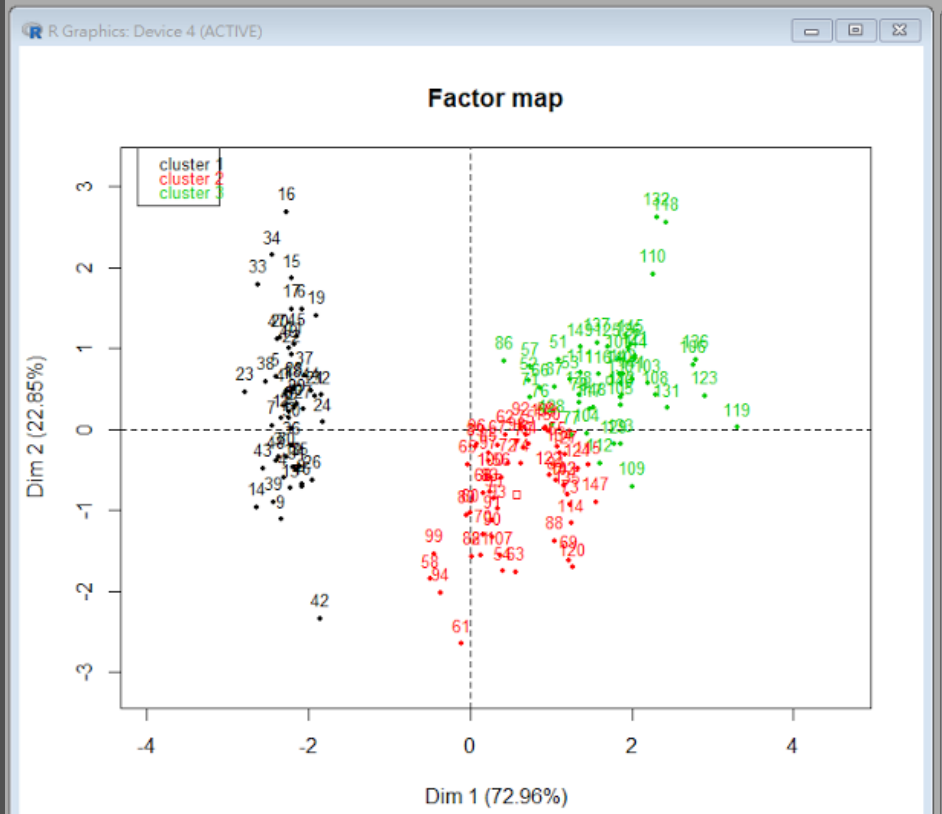
```
> # Principal Component Analysis:
> res.pca <- PCA(iris[,1:4], graph=FALSE)
> summary(res.pca)

Call:
PCA(X = iris[, 1:4], graph = FALSE)


Eigenvalues
                       Dim.1   Dim.2   Dim.3   Dim.4
Variance               2.918   0.914   0.147   0.021
% of var.             72.962  22.851   3.669   0.518
Cumulative % of var.  72.962  95.813  99.482 100.000

Individuals (the 10 first)
                  Dist    Dim.1    ctr   cos2    Dim.2    ctr   cos2    Dim.3    ctr   cos2
1              |  2.319 | -2.265  1.172  0.954 |  0.480  0.168  0.043 | -0.128  0.074  0.003 |
2              |  2.202 | -2.081  0.989  0.893 | -0.674  0.331  0.094 | -0.235  0.250  0.011 |
3              |  2.389 | -2.364  1.277  0.979 | -0.342  0.085  0.020 |  0.044  0.009  0.000 |
4              |  2.378 | -2.299  1.208  0.935 | -0.597  0.260  0.063 |  0.091  0.038  0.001 |
5              |  2.476 | -2.390  1.305  0.932 |  0.647  0.305  0.068 |  0.016  0.001  0.000 |
6              |  2.555 | -2.076  0.984  0.660 |  1.489  1.617  0.340 |  0.027  0.003  0.000 |
7              |  2.468 | -2.444  1.364  0.981 |  0.048  0.002  0.000 |  0.335  0.511  0.018 |
8              |  2.246 | -2.233  1.139  0.988 |  0.223  0.036  0.010 | -0.089  0.036  0.002 |
9              |  2.592 | -2.335  1.245  0.812 | -1.115  0.907  0.185 |  0.145  0.096  0.003 |
10             |  2.249 | -2.184  1.090  0.943 | -0.469  0.160  0.043 | -0.254  0.293  0.013 |

Variables
                Dim.1    ctr   cos2    Dim.2    ctr   cos2    Dim.3    ctr   cos2
Sepal.Length |  0.890 27.151  0.792 |  0.361 14.244  0.130 | -0.276 51.778  0.076 |
Sepal.Width  | -0.460  7.255  0.212 |  0.883 85.247  0.779 |  0.094  5.972  0.009 |
Petal.Length |  0.992 33.688  0.983 |  0.023  0.060  0.001 |  0.054  2.020  0.003 |
Petal.Width  |  0.965 31.906  0.931 |  0.064  0.448  0.004 |  0.243 40.230  0.059 |
>
```

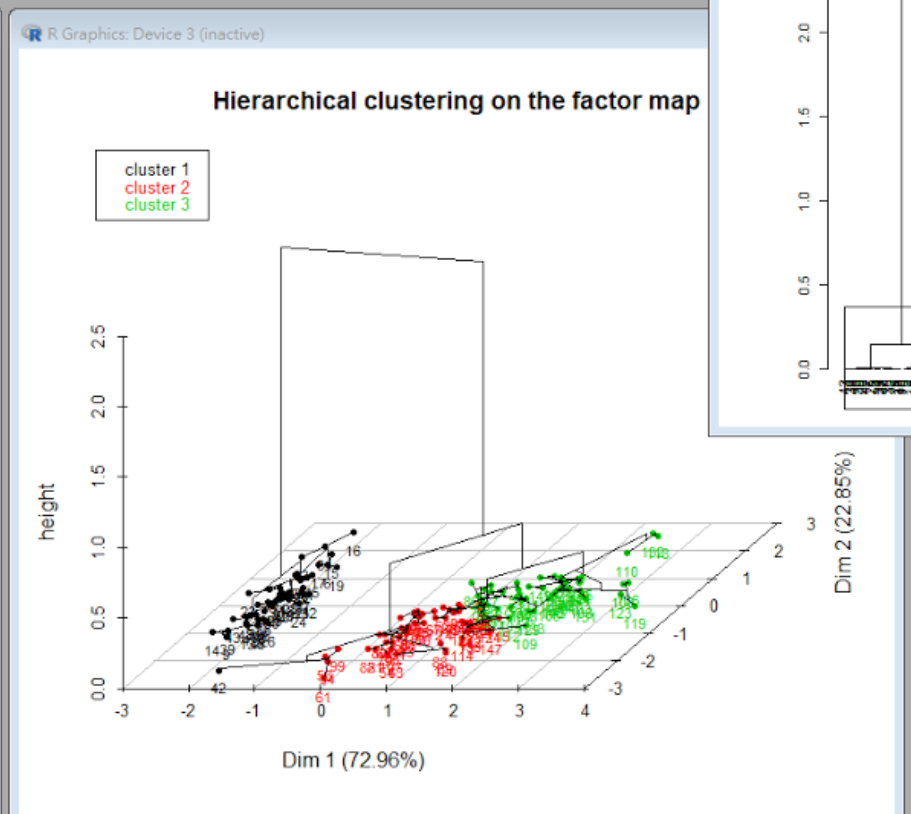
R FactoMineR 主成分分析與階層式集群

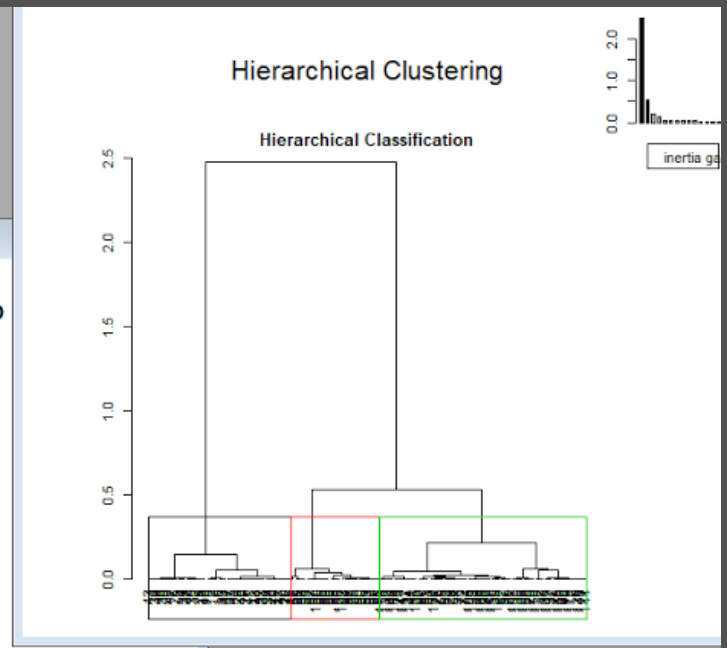
再將先前建立的 res.pca 物件運用 HCPC() 進行集群，即可呈現 !!!

# Clustering, auto nb of clusters:
hc <- HCPC(res.pca, nb.clust=-1)



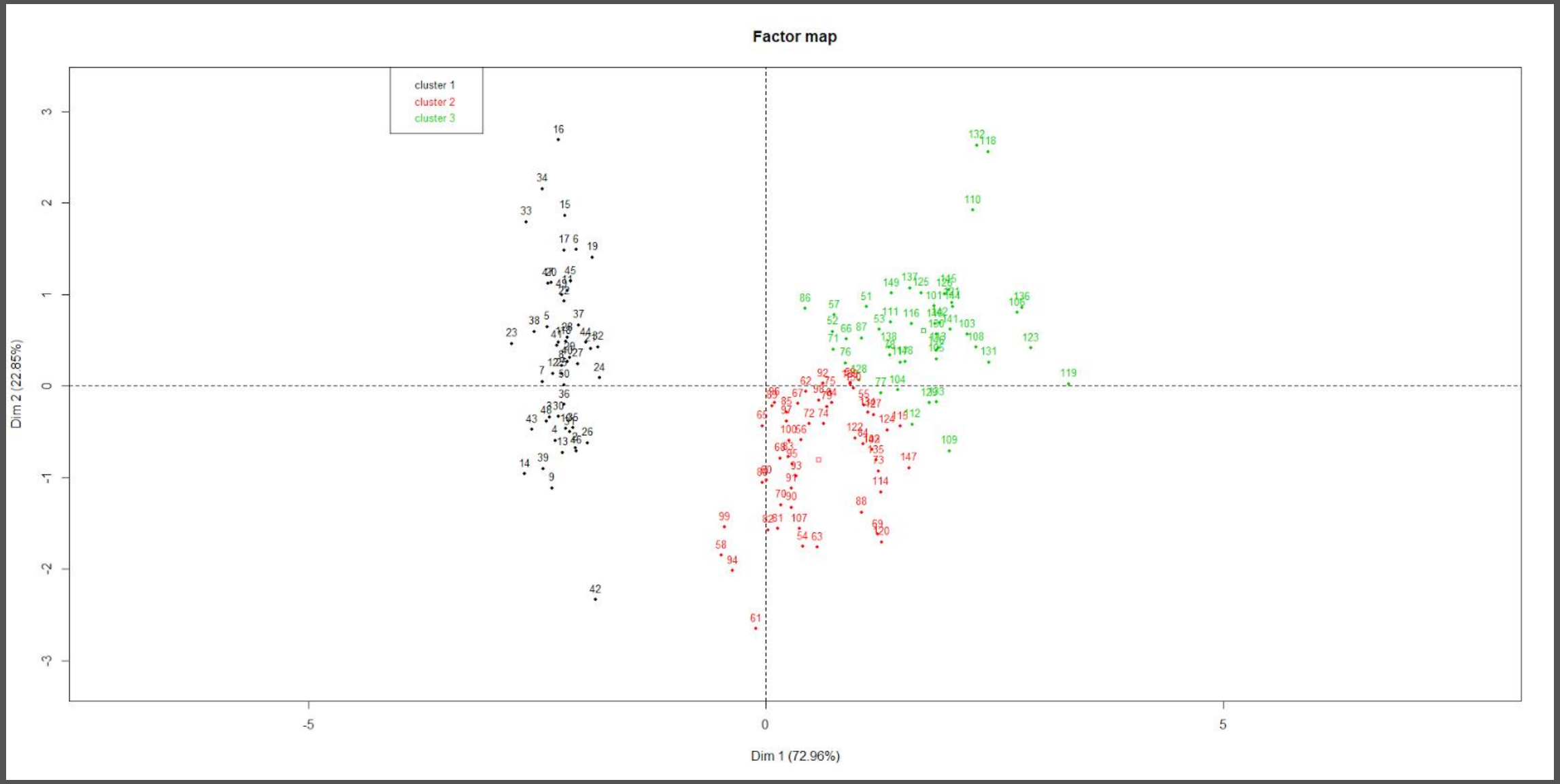






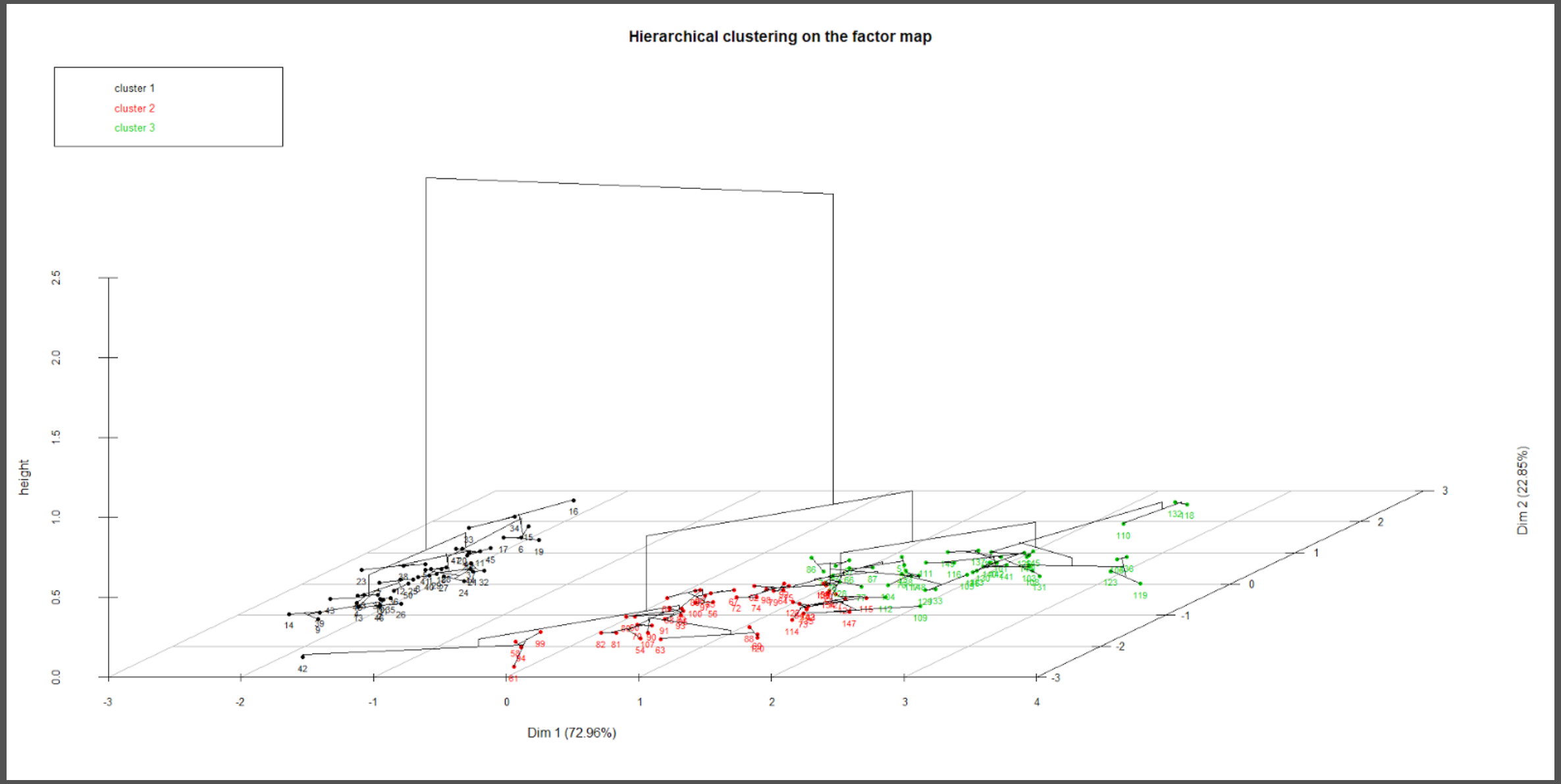
R FactoMineR 主成分分析與階層式集群

集群的散佈圖



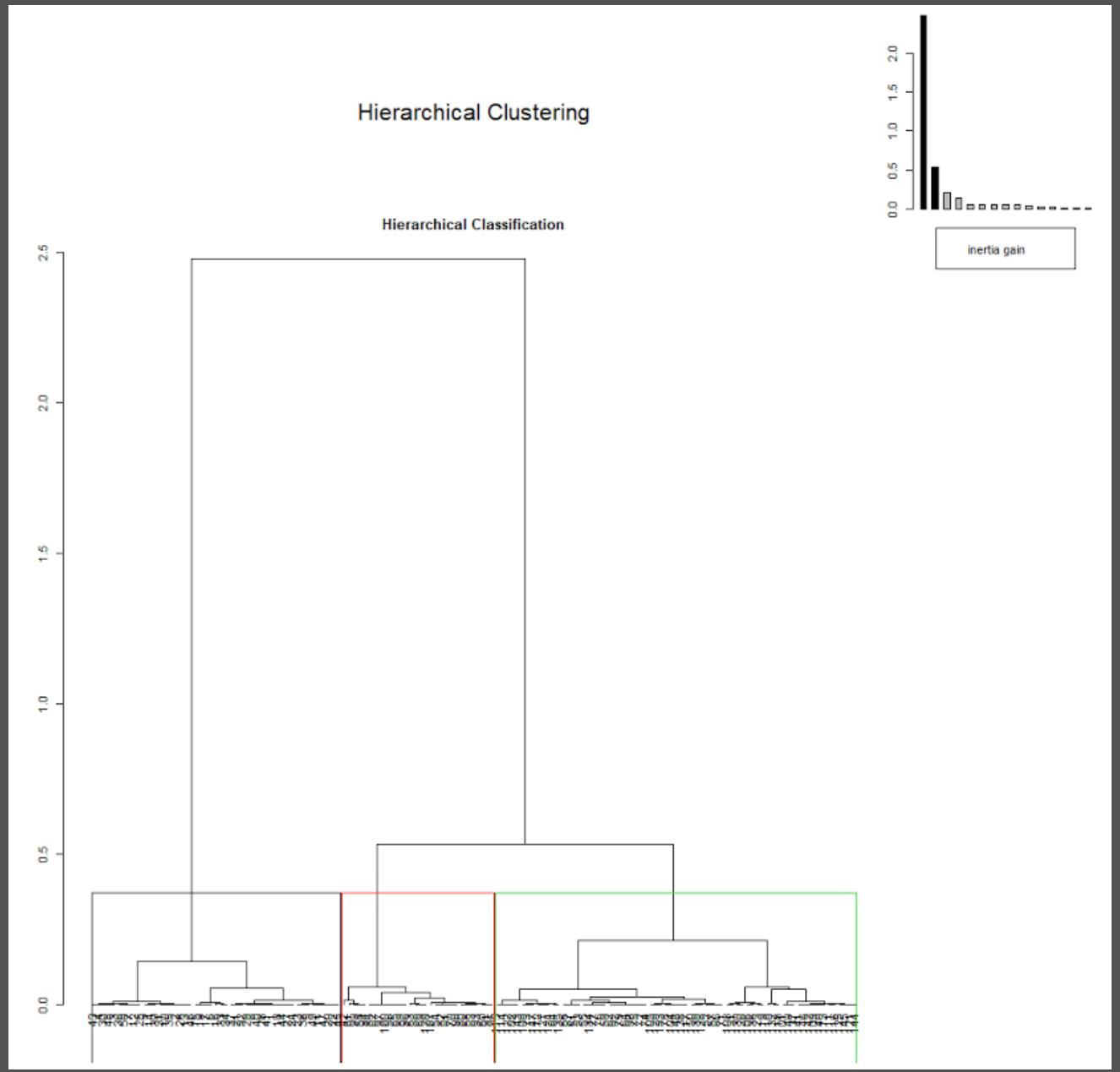
R FactoMineR 主成分分析與階層式集群

階層式集群視覺化圖形



R FactoMineR 主成分分析與階層式集群

集群的樹狀結構



R TukeyHSD

杜凱確實差異檢定檢驗 (Tukey HSD Test)，是多因子變異數分析後進行的驗證，通常有兩個以上的實驗處理，而每個實驗處理都有幾個比較，以各組人數相等時用來比較 K 個平均數之間的差距實驗，如經過變異數分析的處理過後，發現結果 F 值為顯著時，可用 Tukey HSD 法 (Tukey J. W., 1949)。隨後檢定配對平均數間的差異。此時各組的個數必須一致，如果兩個平均數的差異大於 HSD 值，則差異為顯著。

Tukey J. W., 1949, "Comparing individual means in the analysis of variance" Biometrics, 5, 99-114.

$$HSD = q\sqrt{\frac{MS_w}{n}}$$

其值 MS_w 為組內均方，q 為 $\frac{\bar{x}_i - \bar{x}_j}{\sqrt{\frac{MS_w}{n}}}$，當中的 $\bar{x}_i$ 與 $\bar{x}_j$ 代表該組資料的平均數，n 為組資料的數量。

R TukeyHSD

TukeyHSD {stats} R Documentation

Compute Tukey Honest Significant Differences

Description

Create a set of confidence intervals on the differences between the means of the levels of a factor with the specified family-wise probability of coverage. The intervals are based on the Studentized range statistic, Tukey's 'Honest Significant Difference' method.

Usage

```
TukeyHSD(x, which, ordered = FALSE, conf.level = 0.95, ...)
```

在此用文件上的範例進行修改，匯入 graphics 並使用 warpbreaks 資料集。

```
> require(graphics)
>
> head(warpbreaks,10)
   breaks wool tension
1      26    A       L
2      30    A       L
3      54    A       L
4      25    A       L
5      70    A       L
6      52    A       L
7      51    A       L
8      26    A       L
9      67    A       L
10     18    A       M
> NROW(warpbreaks)
[1] 54
>
```

R TukeyHSD

進行 ANOVA。

```
> fm1 = aov(breaks ~  tension, data = warpbreaks)
> fm1
Call:
   aov(formula = breaks ~ tension, data = warpbreaks)

Terms:
                  tension Residuals
Sum of Squares  2034.259  7198.556
Deg. of Freedom         2        51

Residual standard error: 11.88058
Estimated effects may be unbalanced
> summary(fm1)
             Df Sum Sq Mean Sq F value  Pr(>F)
tension       2   2034  1017.1   7.206 0.00175 **
Residuals    51   7199   141.1
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

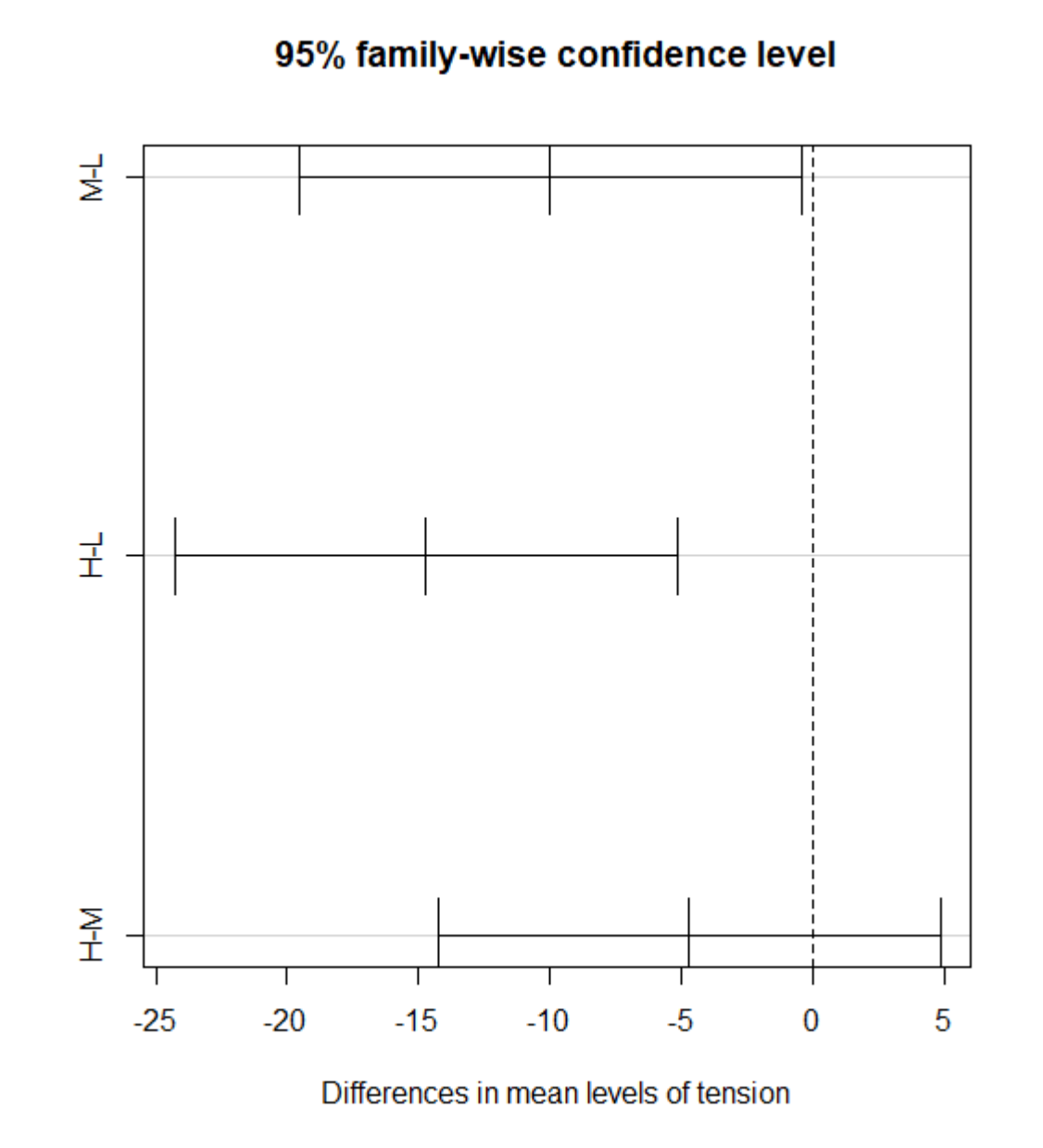
R TukeyHSD

TukeyHSD

```
> TukeyHSD(fm1, "tension", ordered = TRUE)
  Tukey multiple comparisons of means
    95% family-wise confidence level
    factor levels have been ordered

Fit: aov(formula = breaks ~ tension, data = warpbreaks)

$tension
          diff         lwr      upr     p adj
M-H  4.722222 -4.8376022 14.28205 0.4630831
L-H 14.722222  5.1623978 24.28205 0.0014315
L-M 10.000000  0.4401756 19.55982 0.0384598

>
```

R TukeyHSD

進行視覺化




```
require(graphics)

head(warpbreaks,10)
NROW(warpbreaks)

fm1 = aov(breaks ~  tension, data = warpbreaks
fm1
summary(fm1)
TukeyHSD(fm1, "tension", ordered = TRUE)
plot(TukeyHSD(fm1, "tension"))
```