一般數據結構與格式part 2

報告者：盧森 & 阿舍

2017.06.26

Factor 與 Levels

```
factor( v )  # v 必須是vector或是integer
wdays <- c("Wed","Fri","Wed","Tue","Tue","Fri","Wed","Thu")
f.wdays <- factor(wdays)
f.wdays
[1] Wed Fri Wed Tue Tue Fri Wed Thu
Levels: Fri Thu Tue Wed

#levels會告訴你有哪些內容（一種分類的感覺）
> levels(f.wdays)
[1] "Fri" "Thu" "Tue" "Wed"

#nlevels會告訴你這個factor中levels的數量
> nlevels(f.wdays)
[1] 4
```

```
#用tapply搭配factor格式來分類資料 有個matrix格式的Data如下：
Data
        [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]
score.A "20"  "40"  "24"  "56"  "32"  "53"  "42"  "43"
score.B "82"  "59"  "19"  "74"  "93"  "52"  "84"  "74"
wdays   "Wed" "Fri" "Wed" "Tue" "Tue" "Fri" "Wed" "Thu"

f.wdays <- factor(wdays)
tapply(as.numeric(Data[1,]),f.wdays,summary)
$Fri
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  40.00   43.25   46.50   46.50   49.75   53.00

$Thu
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
     43      43      43      43      43      43

$Tue
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
     32      38      44      44      50      56

$Wed
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  20.00   22.00   24.00   28.67   33.00   42.00 
```

一些簡單好用的函數（vector）

```
#產生序列的函數 seq()
seq(1:6)
[1] 1 2 3 4 5 6

#產生重複字串的函數 rep()
rep("Hello",3)
[1] "Hello" "Hello" "Hello"

#字串黏合
paste(2016,month.abb[5],20)  #未指定分隔符號
[1] "2016 May 20"
paste(2016,month.abb[5],20,sep = '-') #指定分隔符號
[1] "2016-May-20"

```

vector 與 matrix

```
#從vector變成matrix的方法
seq(1:6)   #產生一個vector，內容是序列 1到6
[1] 1 2 3 4 5 6

#方法一(上次有講過) matrix()
matrix(seq(1:6),nrow=6,ncol=1)
     [,1]
[1,]    1
[2,]    2
[3,]    3
[4,]    4
[5,]    5
[6,]    6

matrix(seq(1:6),nrow = 1,ncol = 6)
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    2    3    4    5    6

#方法二 對 vector 使用cbind() 或 rbind()
cbind(seq(1:6))
     [,1]
[1,]    1
[2,]    2
[3,]    3
[4,]    4
```

```
[5,]    5
[6,]    6

rbind(seq(1:6))
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    2    3    4    5    6
```

單行或單列的matrix是vector

```
matrix.example
     [,1] [,2] [,3]
[1,]    2    1    5
[2,]    4    2   10
[3,]    6    3   15
[4,]    8    4   20
[5,]   10    5   25
[6,]   12    6   30

matrix.example[,1]
[1]  2  4  6  8 10 12

matrix.example[3,]
[1]  6  3 15

#因為是vector格式所以沒有維度(dimension)
dim(matrix.example[3,])
NULL

#有些函數只能用在matrix,所以當對vector格式使用該函數就會出現警告或是錯誤
#對matrix加上drop=F 可以讓單行或單列的matrix保留matrix的格式
matrix.example[3,,drop=F]
     [,1] [,2] [,3]
[1,]    6    3   15

#加上drop=F後，因為格式仍然是matrix所以仍有維度資訊
 dim(matrix.example[3,,drop=F])
[1] 1 3
```

do.call()

```
#當想對vector格式的資料作資料合併時
A <- 1:10
B <- 100:91
C <- 78:87

#cbind()可以輕鬆幫你把他們合併在一起 （格式變成matrix）
cbind(A,B,C)
```

```
      A   B  C
 [1,]  1 100 78
 [2,]  2  99 79
 [3,]  3  98 80
 [4,]  4  97 81
 [5,]  5  96 82
 [6,]  6  95 83
 [7,]  7  94 84
 [8,]  8  93 85
 [9,]  9  92 86
[10,] 10  91 87

#但如果是list格式的資料呢？
list(A,B,C)
[[1]]
 [1]  1  2  3  4  5  6  7  8  9 10

[[2]]
 [1] 100  99  98  97  96  95  94  93  92  91

[[3]]
 [1] 78 79 80 81 82 83 84 85 86 87

 #我們會發現對list使用cbind()，會得到的結果並不是你想要的
 cbind(list(A,B,C))
     [,1]
[1,] Integer,10
[2,] Integer,10
[3,] Integer,10

 #這時候 do.call()就是一個非常好用的幫手
 do.call(cbind, list(A,B,C))
      [,1] [,2] [,3]
 [1,]    1  100   78
 [2,]    2   99   79
 [3,]    3   98   80
 [4,]    4   97   81
 [5,]    5   96   82
 [6,]    6   95   83
 [7,]    7   94   84
 [8,]    8   93   85
 [9,]    9   92   86
[10,]   10   91   87
```

data.frame的小應用

```
#這是上次R basic我們分享的 data.frame 例子
x1 <- c("father","mother","brother","sister")
```

```r
x2 <- c("Canon","Pentax","Olympus","Nikon")
x3 <- c("gold","red","green","blue")
x4 <- c(2,1,1,2)
camera <- data.frame(member = x1, brand = x2, color = x3, amount = x4)
camera
   member   brand color amount
1  father   Canon  gold      2
2  mother  Pentax   red      1
3 brother Olympus green      1
4  sister   Nikon  blue      2

#只取用部分資料 subset()
#從camera這個資料框架中 挑選amount>1的資料 只顯示member欄位
subset(camera, select=member,subset=(amount>1))
  member
1 father
4 sister

# select也可以用c()選擇顯示的多項欄位資料
# subset可以有很多條件，以&符號分隔選擇條件
```

修改資料（不建議使用）

```r
# fix() 會顯示函數內完整資料，並可點選指定欄位作修改
fix(camera)

# 這個方法非常快就可以更改資料的內容
# 但一般都會建議不要修改原始資料（非必要少用）
```

從data.frame中拿掉遺失值

```r
# 假設有一個名稱為dfm的data.frame格式資料
dfm
   V1  V2
1 0.9 0.8
2  NA 2.7
3 0.3 0.1
4 2.4  NA
5 5.2 2.0

#有遺失值的資料會難以做計算 （範例已累積加總為例）
cumsum(dfm)
   V1  V2
1 0.9 0.8
2  NA 3.5
3  NA 3.6
4  NA  NA
5  NA  NA

#加上na.omit() 會拿掉含遺失值的整列資料唷！！
 cumsum(na.omit(dfm))
```

```
   V1  V2
1 0.9 0.8
3 1.2 0.9
5 6.4 2.9

#如果只想不要計算有遺失值的欄，也可以分開計算
cumsum(na.omit(dfm[,1]))
[1] 0.9 1.2 3.6 8.8
cumsum(na.omit(dfm[,2]))
[1] 0.8 3.5 3.6 5.6
```

字串處理

substr　擷取部分字串

```
x<-'My name is Emma.'
substr(x,3,10)  #取第3~10之字元
[1] " name is"
```

substring 擷取部分字串

```
x.strings <- c(
  "abcdefghij",
  "ABCDEFGHIJ",
  "1234567890"
)
substr(x.strings, 1:4, 8)
[1] "abcdefgh" "BCDEFGH"  "345678"
substring(x.strings, 1:4, 8)
[1] "abcdefgh" "BCDEFGH"  "345678"   "defgh"
```

stringr 套件

可搭配正規表達法使用

str_split　資料剖析

```
fruits <- c( "apples and oranges and pears and bananas", "pineapples and mangos and guavas" )

str_split(fruits, " and ") #可搭配正規表示法
[[1]]
[1] "apples"  "oranges" "pears"   "bananas"

```

```
[[2]]
[1] "pineapples" "mangos"     "guavas"
```

str_mach &str_match_all 查找匹配字符

```
Name <- c("王小名","李美美","董小丁和方小姬","王阿枝","陳小雲")
str_match(Name,".小.") #傳回第一個匹配值  #傳回matrix
     [,1]
[1,] "王小名"
[2,] NA
[3,] "董小丁"
[4,] NA
[5,] "陳小雲"

str_match_all(Name,".小.") #傳回所有匹配值  #傳回list
[[1]]
     [,1]
[1,] "王小名"

[[2]]
     [,1]

[[3]]
     [,1]
[1,] "董小丁"
[2,] "方小姬"

[[4]]
     [,1]

[[5]]
     [,1]
[1,] "陳小雲"
```

str_extract & str_extract_all 提取匹配字符

```
str_extract(Name,".小.") #返回第一個匹配值    #返回向量
[1] "王小名" NA        "董小丁" NA        "陳小雲"

str_extract_all(Name,".小.") #返回所有匹配值  #返回list
[[1]]
```

```
[1] "王小名"

[[2]]
character(0)

[[3]]
[1] "董小丁" "方小姬"

[[4]]
character(0)

[[5]]
[1] "陳小雲"
```

str_replace & str_replace_all 取代

```
> str_replace(Name, "小", "阿")
[1] "王阿名"         "李美美"         "董阿丁和方小姬" "王阿枝"         "陳阿雲"

> str_replace_all(Name, "小", "阿")
[1] "王阿名"         "李美美"         "董阿丁和方阿姬" "王阿枝"         "陳阿雲"
```

Thank you!