蟹老闆R語言速見表

x所指的為vector; df所指的為data.frame

A. R Basic

觀察資料

```
View(df); head(df, n=20); tail(df)
dim(df); length(df); nrow(df); nchar(v);
str(df); class(df); mode(df); summary(df);
```

字串連接與格式化

```
a.href <- paste0(pre, a.href)
paste0(dim(df), sep="\t")
sprintf("[%d]%s", pi, pid)
```

移除目前工作中的所有變數

```
rm(list=ls())
```

RDS 把某變數寫入檔案，可指定載回後指定給某變數

```
saveRDS(all.df, "data/rent5911018.rds")
rent591 <- readRDS("data/rent5911018.rds")
```

RDATA可寫入多個變數到檔案中再載回

```
save(df, df2, var2, file = "temp.rdata")
load("data/rent5911018.rdata")
```

建立新的資料

```
country <- c(“CN","US","JP","HK","KR")
b <- 1:10; b <- seq(1, 9, 2)
c <- runif(1000, 1, 5) #自1~5間取均勻分布1K個
c <- rnorm(1000, 1, 5) #自1~5間取常態分布1K個
all <- list(); df <- data.frame();#初始化變數
df$ratio <- df$sbi/df$tot #建立df的新變項
names(df) <- c('v1', 'v2') #賦予新的變數名稱
```

取用或篩選資料（透過索引或變數名稱）

```
v[3:7]; v[c(1, 3, 5)];
v[length(vec):3]
v <- v[c(T, F, T, F, T)] #利用T/F篩選資料
v <- v[v%%2==0]    #去除vec中的奇數
df[df$v1 > df$v2,] #根據col來篩選列
df[df$v1 > df$v2,c(1, 2)] #篩選列並選取變數
m55<- m5[m5$'matleave_95' == 5,]
res$data$data      #取用List內有名稱的變數
dim(safefood[[1]]) #取用List的第一個元素
```

資料的刪除

```
df$likes <- NULL   #刪除df中的likes變項
v <- v[-(3:5)];
df <- df[-4]       #刪除sf.df的第四個變項
```

資料型態的轉換

```
ubike$lng <- as.numeric(ubike$lng) #轉數字
sf.v <- unlist(sf)       #拆list為vector
sf.m <- matrix(sf.v, byrow = T, ncol = 5)
sf.df <- as.data.frame(sf.m)
options(stringsAsFactors = FALSE)
```

四則運算與邏輯運算

```
v <- v / 2         #除法
v <- v%%2; v <- v%/%2    #除法取餘數、取商
v%%2==0; v%%2!=0; #判斷a是否為偶數/奇數
any(v>11)          #是否有任意一個數大於11
all(v>11)          #是否所有數均大於11
```

排序

```
v <- sort(v); v <- v[order(v)]; #升冪排序
v <- sort(v, decreasing=T) #降冪排序
df <- df[order(-df$v1),] #根據v1降冪排序
df <- df[order(-df$v1, df$v2),]
```

A1. 時間函式

轉換字串為時間格式

```
df$poslt <- strptime(df$time, "%m %e %Y")
df$poslt$wday; df$poslt$zone;
as.Date("2017-01-01")
as.POSIXct(Sys.time(), tz="Asia/Taipei")
ctime <- as.POSIXct("2017-01-08")
months(ctime); weekdays(ctime);
```

轉小數或整數為時間

```
z <- 7.343736909722223e5
as.POSIXct((z - 719529)*86400,
       origin = "1970-01-01", tz = "UTC")
z <- 1509343484914 #瀏覽器的紀錄
as.POSIXct(z/1000, origin="1970-01-01",
       tz="Asia/Taipei")
```

將時間轉回字串

```
format(safefood.df$ctime, "%m-%d-%Y")
ctime <- format(Sys.time(), "%Y%m%d%H%M%S")
```

計算程式執行時間

```
start <- proc.time()
      # your code here
proc.time() - start
```

A2. 程式語言邏輯

函式與條件判斷式的寫法

```
assignColor <- function(ratio){
  if(ratio > 0.8){
    return("#FF0000")} # red
  else if(ratio < 0.2){
    return("#0000FF")} # blue
  else{
    return("#00FF00")} # green
}
```

三元條件判斷ifelse()

```
df$v3 <- with(df,
       ifelse(v1 == "A", -v2, v2))
msg = ifelse(!is.null(x$msg), x$msg, NA)
if(!(is.null(nexturl))){…}
```

資料清理Clean

NA值處理

```
matleave[is.na(matleave)] <- 0
sf.v <- sf.v[!is.na(sf.v)]
anyNA(safefood.v)
sum(is.na(safefood.v))
```

na.fail() returns the object if it does not contain any missing values, and signals an

```
na.fail() returns the object if it does not
contain any missing values, and signals an
error otherwise.
na.omit() returns the object with
incomplete cases removed.
na.pass() returns the object unchanged.
```

僅保留每個欄位都有資料的資料列

```
tdf <- table.df[complete.cases(table.df),]
```

重複資料清理

```
df <- df[!duplicated(df), ] #刪除df中的重複列
df <- dplyr::distinct(df, x, y)
anyDuplicated(df)         #偵測是否還有重複
```

僅列出非重複值unique()

```
unique(post_reactions$type)
```

文字資料處理

依照字元位置取出子字串

```
df$time <- substr(df$發生時段, 1, 2)
```

利用正規表示式取代字串內指定的pattern

測試網站：`https://regexr.com/`

`gsub`除用來取代外，也可以指定所需資料取代雜亂的資料

`grelp`是用以偵測字串中是否有該`pattern`，有則TRUE

將逗號取代為空字串

```
df$text <- gsub(",", "", df$text)
df$text <- sub(",", "", df$text)#僅取代第一個
```

將逗號或者句號取代為空字串

```
v <- gsub("[,.。]", "", v)
v <- gsub("[,.。]", NA, v)#偵測後整個字串轉NA
```

去除文字變項首尾的空白

```
content <- trimws(content)
```

清除變數中所有空字元包含換行符號、tab或者全半形空白

```
v <- gsub("\\s", "", v)
```

清除變數中所有的HTML標籤

```
v <- gsub("<.*?>", "", v)
```

清除所有空白行

```
v <- gsub("\n\n", "\n", v)
```

清除字串前後的空白，字串中的空白不去除

```
v <- gsub("^\\s+|\\s+$", "", v)
```

取出index到.html間的數字（第二種版本才是對的）

```
s <- "http://123.index123.html"
gsub("index(\\d+).html", "\\1", s)
gsub(".*index(\\d+).html", "\\1", s)
```

偵測出哪些column名稱內有matleave

```
cols <- grepl("matleave", names(ldata))
```

超鏈結的pattern

```
url.pattern <- "^((https?|ftp)://|(www|ftp)
\\.)?[a-z0-9-]+(\\.[a-z0-9-]+)+([/?].*)?$"
```

資料匯總 Summarization

密度函數

```
plot(density(c))
```

tapply(var1, var2, func) # table()和count()都只能計算次數

```
df1 <- tapply(df$v1, df$v2, length)
tapply(df$v1, df$v2, mean)
tapply(df$v1, df$v2, sum)
tapply(df$v1, list(df$v2, df$v3), length)
```

table()

```
res2 <- with(df, table(time, region))
```

aggregate(df, by=list(var1, var2), func)

#aggregate()和count()都先轉long-form再用spread()轉table

#只有aggregate()和tapply()可以apply某函式例如sum,mean

```
res3 <- aggregate(df,
        by=list(df$time, df$region), length)
res4 <- spread(df2, Group.2, x, fill = 0)
```

count(df, var1, var2) #僅能計算次數

```
res5 <- dplyr::count(df, time, region)
res6 <- spread(res5, region, n, fill = 0)
```

Long-form轉table

```
df2 <- aggregate(df, by=list(df$v1), mean)
```

Table轉long-form

```
df3 <- gather(df, "year", "degree", 2:20)
```

讀檔

```
df <- read_excel(path, sheet = NULL,
        range = NULL, col_names = TRUE)
        mutate(year3 = strftime(year2, "%Y"))
df1 <- read.csv(url,
        fileEncoding = "big5",
        stringsAsFactors = F)
df2 <- readr::read_csv(res)
df <- fromJSON(content(GET(url), "text"))
```

讀取url後將結果儲存到電腦而不直接處理

```
GET(url, write_disk(fname, overwrite=T))
```

爬蟲相關套件

GET() and POST()

```
df <- fromJSON(content(GET(url), "text"))
res <- POST(url,
        body = list(searchMode = "Adv",
        searchType = "text",
        querystrA = "年金"))
res <- GET(url, #設定cookie
        config = set_cookies("over18" = "1"))
```

解析HTML檔案

1. 將某個url取回讀成xml_document、xml_node

```
doc   <- read_html(url, encoding="UTF-8")
```

2.1 rvest::以xpath找到所有符合的節點

```
tbs <- html_nodes(doc, xpath = "//table")
```

2.2 rvest::以CSS Selector找到所有符合的節點

```
tbs <- html_nodes(doc, "table")
tb <- html_node(doc, "#Main table")
```

3. rvest::取出HTML內的值

```
html_tag()  #get tag name
html_text() #get tag content
html_attr() #get attribute value of a tag
html_attrs() #get all attributes
```

4. 將某個html table節點轉成data.frame

```
tb.df <- html_table(tbs[[2]], fill = T)
```

4.1 將所有tables轉成list of data.frame

```
tb.list <- html_table(tbs, fill = TRUE)
```

將某個list中的data.frame轉出來

```
tb.df <- as.data.frame(tb.list[[1]])
```

將某個節點寫為html檔

```
write_html(all[[13]], "test.html")
```

Problem Shooting

改變RStudio目前預設的Encoding

```
Sys.setlocale("LC_ALL", "cht") #看得見中文
Sys.setlocale("LC_ALL", "C") #轉C語言base
Sys.setlocale("LC_ALL", "cht") #看得見中文
```

在View中顯示test.html

```
file.show("test.html")
```

執行Terminal命令

```
system("open test.html")
```

dplyr

```
iris <- as_tibble(iris)
```

General purpose的語法

```
models <- by_cyl %>% do(mod = lm(mpg ~
disp, data = .))
```

選取 Select Columns

按index選取

```
mtcars %>% pull(-1) # 選取最右邊一個col
mtcars %>% pull(cyl)
```

按變數名稱選取

```
select(iris, starts_with("Petal"))
select(iris, -starts_with("Petal"))
select(iris, contains("Petal"))
select(iris, ends_with("Width"))
select(iris, matches(…)) # match re
select(df, V4:V6)
select(df, num_range("V", 4:6))
```

將x搬到最前面

```
select(df, x, everything())
```

選取順便改名

```
select(iris, petal_length = Petal.Length)
```

篩選 Filter rows

```
slice(mtcars, n()) # 篩選出最後一筆資料
slice(mtcars, 5:n()) # 篩選出範圍內的資料
filter(mtcars, row_number() == n())
filter(mtcars, between(row_number(), 5,
n()))
distinct(x, y) # 篩除重複的(x, y)欄紀錄
top_n() # 篩選出最大或最小的值
```

Data binding

```
bind_rows(list(one, two)) # 其實不用do.call
bind_rows(list(one, two), list(two, one))
bind_rows("group 1" = one, "group 2" = two,
.id = "groups") # 順便加變數
bind_rows(list(a = one, b = two), .id =
"id")
```

Summarization

tally和count同目的，但不會做group()

```
tally(x, wt, sort = FALSE)
count(x, ..., wt = NULL, sort = FALSE)
```

若僅是要計算一個值的總數，用add_tally()和add_count()

就好，可以直接在整個df新增一個n，為每個數值的總數

```
add_tally(x, wt, sort = FALSE)
add_count(x, ..., wt = NULL, sort = FALSE)
```

Vector

```
n_distinct(x) # 有幾個唯一值
recode(x, `2` = "b", `4` = "d") # 轉類別變數
recode_factor(x, `1` = "z", `2` =
"y", .default = "D")
```

Others

```
lead(x, n = 1L, default = NA)
lag(x, n = 1L, default = NA)
na_if(x, y) # 把某些你不要的值轉為NA
cumsum(), cummean(), cummin(), cummax(),
cumany(), cumall()
```

蟹老闆R語言速見表 II

資料視覺化 Visualization

RColorBrewer (http://colorbrewer2.org/)

```
pcolor <- brewer.pal(12, "Paired")
```

更改base套件的標籤語言為中文

```
par(family=("Heiti TC Light")); par(family=("STKaiti"))
```

樞紐分析資料的繪製

```
assocplot(res)
mosaicplot(res, color=colors, border=0, off = 3,
      main="Theft rate of Taipei city (region by hour)")
```

繪製密度函數曲線圖

```
plot(density(c)); plot(1:10, 1:10);
text(v1, v2, labels=v3, cex= 0.5, pos=3)
title(m55[i,1], line = -4, cex.main=3)
lines(1:25, 1:25, col='red')
par(mfrow=c(4,6), mai= c(0.2, 0.2, 0.2, 0.2))
barplot(v, border=NA, space=0,xaxt="n", yaxt="n", ylim = c(0,5))
```

世界地圖Area Map

```
myMap <- rworldmap::joinCountryData2Map(mdata, joinCode = "ISO3",
      nameJoinColumn = "iso3")
rworldmap::mapCountryData(myMap, nameColumnToPlot="matleave_13",
      catMethod = "categorical",
      colourPalette = colors, addLegend="FALSE")
```

ggplot2

```
matleave %>%
      ggplot() +
      aes(year3, degree) +
      facet_grid(iso3~.) +
      geom_bar(stat = "identity", fill = "blue")
ggmap(get_googlemap(center=c(121.516898,25.055536),
      zoom=12, maptype='terrain')) +
  geom_point(data=ubike.df, aes(x=lng, y=lat),
      colour=ubike.df$color, size=ubike.df$tot/10, alpha=0.4)
```

dplyr

```
matleave <- ldata %>%
      select(iso3, contains("matleave"), -contains("wrr")) %>%
      filter(matleave_13==5, matleave_95==5) %>%
      gather("year", "degree", 2:20) %>%
      replace_na(list(degree=0)) %>%
      mutate(y2=as.POSIXct(strptime(year, "matleave_%y"))) %>%
      mutate(y3 = strftime(year2, "%Y"))
```