

R for Data Science

Author: Ji-Lung Hsieh
Your City, ST 12345
(123) 456 - 7890

1-0 安裝程式開發環境	9
安裝R與RStudio	9
1.1 開始寫程式	9
匯入所需的套件	9
偵測所需套件並安裝尚未安裝的套件	9
1.2 R語言基本操作與資料型態	10
空值處理	10
NA值簡介 (NaN、Inf、-Inf、NULL)	10
NA值處理	10
1.3 字串操作	10
String concatenation: sprintf() and paste()	10
取代文章內所有的空白字元	11
1.4 R的時間物件	11
時間物件、數值表示、文字表示間的關係	11
時間物件與數值表示	11
時間物件與文字	12
設定時區	12
將字串轉為時間格式POSIXct/POSIXlt	13
字串轉為時間格式	13
浮點數或整數轉時間格式	13
改變時區	13
Accessing time objects features	14
POSIXlt	14
POSIXct	14
setlocale to	14
Measure the computer execution time	15
時間區間Interval與重疊	15
自POSIXct建立時間區間	15
合併重疊的時間區間	15
補足缺少的時間	16

重點回顧	16
Data.frame	16
Factor	16
1-A Practices & Assignments	16
2-0 何謂資料	17
2-1 讀取Excel檔：產假支薪	19
導言	19
案例說明：產假支薪	19
資料概況	20
目標	20
步驟一：用readxl套件讀取Excel檔	21
步驟二：選取所需變項	
select essential variables	23
步驟三：將NA值取代為0。	
replace NAs as 0	23
步驟四：篩出所需要的資料列	
filter rows by variable values	24
步驟五：繪圖	24
練習：完成其他的繪圖	28
2-2 讀取CSV檔：台北市竊盜案	29
步驟一：摘要與提要	29
步驟二：載入台北市竊盜案資料	30
步驟三：產生新的變項	31
步驟四：整理、清理資料	31
步驟五：計數／彙整／摘要：使用tapply()	32
步驟六：視覺化	33
綜合：資料彙整（Data Summerazation）	36
方法二：table()	36
方法三：count()	36
練習	37
2-3 讀取JSON檔	38
預覽與回顧	38
2-3-1 何謂JSON檔案？	38
(Option) 在瀏覽器上或電腦上安裝JSON Viewer	39

範例：空氣品質指標的JSON檔案	40
2-3-2 比較JSON與CSV	40
例一：紫外線指標（UVI）	40
例二：空氣品質指標（AQI）	41
例三：就診後同日於同醫院因同疾病再次就診率	42
2-3-3 JSON放在網路上的實際樣貌	44
實際上的JSON格式	45
2-3-4 自網路取得JSON並轉為R的資料型態	45
httr套件用以發送http request取得網頁檔案	46
jsonlite套件：JSON轉為R的物件	47
安裝jsonlite與httr套件	47
2-3-5 狀況一：標準JSON編排，空氣品質指標資料	47
JSON格式轉R物件	48
步驟一：取得網路檔案	48
步驟二：轉出檔案內的文字	48
步驟三：將JSON格式文字轉為R物件	49
2-3-6 狀況二：多元、彈性但階層化的JSON資料：104求職網	50
Option: 取回資料並寫在硬碟	50
2-3-7 狀況三：非典型的JSON資料結構：食品闢謠	51
處理非典型的JSON檔	52
各個JSON案例	53
2-4 讀取XML檔	54
2-X 讀取資料庫	54
SQL query list and review	56
在進行查詢時轉換資料的型態	56
2-X 讀取一般文字檔	56
2-X 寫入檔案	56
write_disk() 將GET()的檔案直接寫到Local端	56
將xml_document寫為HTML	56
write() 將GET()或POST()結果在尚未讀成xml_document前寫為檔案	57
writeLines() 將資料寫為文字檔	57
2-A Practices & Assignments	57
Practice 2-3-1 用JSON表示資料	57
Practice 2-3-2 查找並抓取網路上的JSON檔	57

Practice 2-3-3 抓取並轉譯JSON文件為data.frame	58
2-B 本章節有用的系統指令	58
3-1 使用Chrome DevTools檢查網頁頁面操作	60
Chrome DevTools	60
步驟一：選檢查開啟Chrome DevTools	61
步驟二：選Network Panel進行前置步驟	61
步驟三：載入頁面並觀察	62
步驟四：找尋所需的JSON檔案	63
步驟五：複製JSON檔案連結。	64
3-3 逐頁爬取資料-104人力銀行	65
3-3-1 載入讀取JSON所需函式庫	66
3-3-2 取得第一個頁面的JSON資料	66
3-3-3 合併兩個頁面的資料	67
3-3-4 「扁平化」	67
3-3-5 找到最後一頁的頁碼	68
3-3-6 黏貼頁碼組成新的url並測試	68
3-3-7 用for迴圈抓回所有資料	69
3-3-8 合併抓回來的資料	69
3-4 剖析HTML檔案	69
3-4-1 HTML的結構	71
元素與屬性 Elements and Attributes	73
範例 Hyperlink <a>	73
HTML中的資料在哪裡？	74
Id and class	74
Using DevTools to get data path	75
3-4-2 XPath and CSS selector	76
使用Chrome DevTools找尋CSS Selector與XPath	76
一般化路徑與簡化路徑	77
3-5 爬取ibon資料	77
3-5-1 使用GET()爬取縣市	78
不使用pipeline的寫法	78
使用pipeline的寫法	78
3-5-2 用POST()撈取一個縣市的ibon地址	79
Options. 使用html_table()抓回整個表格	80

3-5-3 用for-loop撈回所有縣市的ibon地址	80
3-5-4 清理資料並取出鄉鎮市區	80
3-6 爬取PTT討論版	82
Step 1. 載入所需套件	83
Step 2. 取回並剖析HTML檔案	84
Step 2-1. read_html() 將網頁取回並轉為xml_document	85
Step 2-2 以html_nodes() 以選擇所需的資料節點	85
Step 2-2 補充說明與XPath、CSS Selector的最佳化	86
Step 2-3 html_text()或html_attr()轉出所要的資料	87
Step 3. 用for迴圈打撈多頁的連結	89
Step 4. 根據連結取回所有貼文	90
補充(1) 較好的寫法	92
補充(2) 最佳的寫法	93
3-7 rtweet 使用API獲取資料	94
Step 1. 獲取資料權限	95
Step 2. Environment settings	95
Setting-up tokens	96
Step 3. Getting tweets	96
Step 4. Getting user timeline	96
補充：使用youtube API	96
3-8 定時爬取即時資料	96
Step 0. Loading packages	97
Step 1. Getting data twelve times per 5 mins	97
Step 2. Renaming files by timestamp	97
Step 3. Full code	98
3-X Miscellaneous	98
3-A Practices & Assignments	99
Practice 3-1-1：找到這些網站背後的JSON檔	99
Practice 3-1-2	99
Practice 3-3-1	100
Practice 3-4-1 查找XPath與CSS Selector	100
Practice 3-5-1 爬取新聞搜尋結果	101
Practice 3-5-2 爬取並比較新聞媒體	101

4-0 tidy型態的資料	102
4-1 台灣2018年公投案例分析簡介	104
4-1-1 下載資料	106
4-1-2 讀取資料並觀察資料	107
Tidy型態的資料	107
4-1-3 分析企劃	108
4-2 用base清理與彙整資料	109
Step 1. 去除第一行CSV的標題列，另外保留前四個文字變項	109
Step 2. 轉變多個變項的資料欄位 <- Very important	109
Step 3. 計算各村里的人口數	110
Step 4. 計算各村里曾結婚過的人口總數	110
Step 5. 計算超過六十五歲的高齡人口數	111
Step 6. 計算比例	111
Step 7. (cont.)以下我們將改用dplyr來處理資料	111
4-3 用dplyr清理並產生tidy form資料	111
Step 1. 產生村里的全名	111
Step 2. 用gather()收合變數產生tidy form資料	112
Step 3. 切割key欄位為數個變數	113
Step 4. 轉換資料為數值型態並建立65歲以上的變項指標	113
4-4 建立鄉鎮市區與村里指標	114
4-4-1 使用group_by()建立村里指標	114
4-4-2 建立鄉鎮市區指標	116
4-4-3 計算組距資料的中位數	117
Step 1. 建立鄉鎮市區之於年齡的二階層tidy型態資料	117
Step 2. 計算組距固定下的年齡中位數	118
Step 3. 合併新產生的年齡中位數	119
4-5 讀取公投資料	120
4-5-1 讀取資料並重新命名	120
4-5-2 產生同意票的比例	120
鄉鎮市區資料和公投資料結合後再繪圖	120
4-5-3 視覺化：加入人口數，並描繪為點的大小	121
4-5-4 讀取所有公投資料並存檔	122
儲存為rda檔	122

4-5-5 視覺化各案公投比例	123
4-6 ggplot2整理	124
單一連續變數	124
X和Y均為連續變數	125
X為離散、Y為連續變項。	125
4-7 應用 - 網民行為分析	125
分析策略	127
第四章練習	129
練習4-1-1 base and dplyr	129
練習4-1-2 讀取村里的教育水平	134
練習4-1-3 彙整鄉鎮市區的教育水平	134
練習4-1-4 將ibon所在地資料轉為鄉鎮市區的發展指標	134
練習4-1-5 用視覺化方法找出公投案和各個鄉鎮市區指標間的關係。	134
4-Truncated-1-4	134
Step 3. 用字串處理函式string::str_sub()取出性別	134
Step 4. 用tidyr::separate()切割字串（未婚、結婚、離婚、喪偶）	134
Step 5. 用RE偵測年齡群組的邊界	135
5-1 Trump's tweets 字串操作與視覺化	136
5-1-1 載入資料與套件	136
4-1-2 字串抽取與正規表示式	137
4-1-3 用dplyr來改寫	139
4-1-4 用dplyr + magrittr的pipeline風格來改寫	139
5-2 視覺化	140
4-2-1 視覺化：比較發文時間分佈	140
4-2-2 視覺化：比較有無插圖	142
5-3 文字探勘	144
4-3-1 用字比較	144
4-3-2 熱門用字的視覺化	144
4-3-3 不同載具的用字差異	145
用字情緒分析	146
5-A Practices & Assignments	146
6-1 ggplot	146

6-1-X Mac上的中文字體修正	146
6-Map 用rworldmap繪製產假支薪的世界地圖	148
載入所需套件與資料	148
結合資料與地圖檔	149
繪製地圖	150
Practice 5-Map-1	152
Practice 5-Map-2	152
輸出資料至HTML	153
6-A Practices & Assignments	154
Appendix - Rmarkdown	154
R Markdown	154
R presentation - xaringan	155

第一章、R基礎

在本章，你會學到vector、factor、matrix、list與data.frame這些「資料型態」

- vector就它當是google或Excel試算表的一欄，但一列也可以抽成一個vector。這一天到晚用，不會也難。
- data.frame要特別留心
- factor和matrix你先看看
- data.frame其實相當於是「list of vector」的組成。

有些資料型態非常雷同（例如matrix或data.frame都是將資料儲存為欄列型態的二維資料），但學習過程你要特別注意，不同的資料型態之間有什麼差異？

- matrix和data.frame有什麼差？
- factor和vector有什麼差？
- data.frame和list有什麼差？
- 每一個資料型態有什麼操作

- 產生：怎麼新建一個這個資料型態？
- 選取：我可不可以只要這幾項？
- 合併：兩個這種資料型態可不可以疊在一起？
- 刪除：怎麼把某一個或某幾個刪掉（要先選）？
- 轉換：可不可以轉成另一種型態？

1-0 安裝程式開發環境

安裝R與RStudio

1. 至<https://cran.r-project.org> 下載R的最新版本並安裝之。此為R的主程式。
2. 至RStudio (<https://www.rstudio.com/>) 下載最新的RStudio Desktop Open Source License 並安裝之。此為R的編輯器，提供相當多好用的功能讓使用者更容易撰寫R。下載連結為<https://www.rstudio.com/products/rstudio/download/>

1.1 開始寫程式

匯入所需的套件

偵測所需套件並安裝尚未安裝的套件

```
pkgs <- c("tidyverse", "jiebaR", "data.table", "text2vec", "fmsb",
         "rjson", "rpart", "randomForest", "ggmap",
         "zoo", "twitteR", "tidytext", "tm")
pkgs <- pkgs[!(pkgs %in% installed.packages()[, "Package"])]
if(length(pkgs)) install.packages(pkgs)
```

1.2 R語言基本操作與資料型態

一開始用Excel去想像抽象的語言。一欄（一直行、Column、一個變數）就是一個vector，所以可想像一個vector裡面應該資料型態要是一致的。常用基本資料型態包含 numeric（數值型態）、character（文字型態）。單一個Excel試算表就像一個list或data.frame。把好幾欄湊在一起，不同欄包含的資料形態就可能不同（有些欄是人名，有些欄是數字），湊在一起變成一個list，而data.frame就是一種特殊list的型態。

空值處理

NA值簡介（NaN、Inf、-Inf、NULL）

在讀取資料時或在處理程式的過程，經常會遇到部分資料是空的，在R語言中稱為空值，有特定的代號NA。但實際上空值的情形非常多種，包含NA、NaN、Inf、-Inf、NULL都是。NA為缺失值（Missing value），為「Not Available」縮寫。Inf和-Inf指正負無窮，通常在除以0時會出現。NaN即為「not a number」。NULL經常用於某個函式的參數沒有被賦予任何值，或者某些函式沒有傳回值，若把NULL指派給某一個變數，相當於把該變數刪除。

NA值處理

na.omit(), na.rm=T若要加總的項目有NA的話，那就無法加總應該要先將NA轉為0，但其實R有更方便的工具，使用na.rm=TRUE自動忽略NA值。

```
df$sum <- rowSum(df, na.rm=T)
```

1.3 字串操作

String concatenation: sprintf() and paste()

```
token <- ""
id <- "DoctorKoWJ"
```

```
query <-
"?fields=posts.limit(100){id,link,type,created_time,updated_time,description,message,caption,shares,name,source,parent_id}"
fburl <- paste0("https://graph.facebook.com/v2.10/", id, query, "&access_token=", token)
fburl <- sprintf("https://graph.facebook.com/v2.10/%s%s&access_token=%s", id, query, token)
```

取代文章內所有的空白字元

注意，應該要使用`str_replace_all()`而不是`str_replace()`。

```
content <- str_replace_all(content, "\\s+", "")
```

1.4 R的時間物件

時間物件、數值表示、文字表示間的關係

在程式語言中，時間通常有三種格式，一種是該程式語言用來做時間運算的時間物件，Python有Python的時間物件、R有R的時間物件；另一種是該時間物件可以被展示為字串；最後一種是該時間物件實際上被儲存的方式多半為一個整數，之後再把該整數用數學公式轉為時間（例如用自1900年1月1日0分0秒至今日的總秒數的整數來作為時間的儲存單位）。

一般來說，系統在輸出資料時所自動吐出來的時間普遍是用整數來記錄，所以如果你處理的是瀏覽器所吐出來的時間，那大概是整數。但用API可以存取的時間，或者說資料紀錄的時間欄位，多半是文字。

不過文字狀態的時間沒辦法被運算（就文字而已，要怎麼做運算？），所以通常程式語言必須要將其轉換為該語言特有的時間物件。但就google sheet或EXCEL這種使用者介面而言，他不會讓你知道有時間物件這個東西，但會允許你在數值型態或文字顯示的型態間作切換。

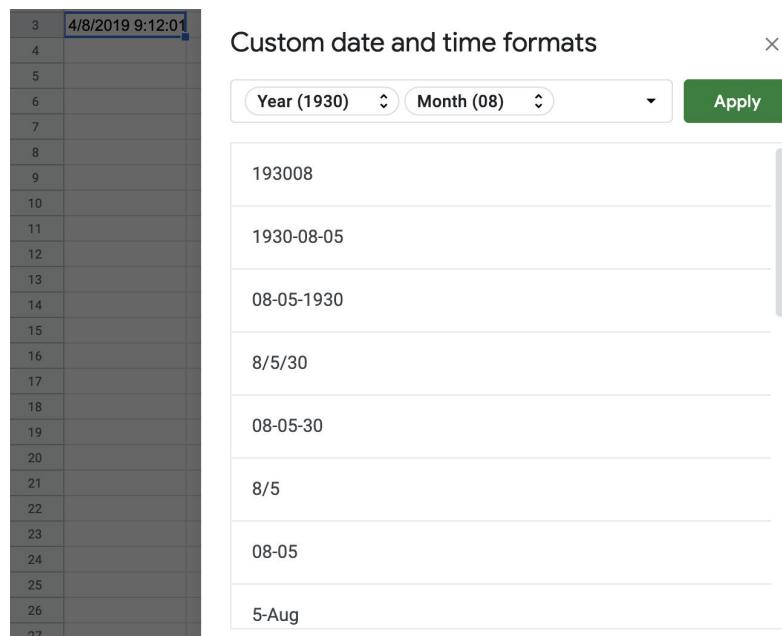
時間物件與數值表示

以Google sheet來說，其基準點是1900年1月1日0分0秒，如果是Date，就是這個基準加上一串整數。如果是時間，就是這個基準加上一串帶小數點的整數，而儲存在電腦裡，就是上述的整數，或者是帶小數點的整數（Numeric一欄）

Timestamp	Numeric	Baseline	Baseline+43563	Can be stored as
4/8/2019	43563	1/1/1900	4/10/2019	43563
4/8/2019 9:06:54	43563.37979		9:06:54 AM 43563.37979 + 1/1/1900	3763876014 43563.37979*86400

時間物件與文字

總之在資料交換間是不會用各種語言獨有的時間型態，要不是文字型態，不然就是數值型態。因此，處理時間的資料項第一件事情就是要知道他的時間是什麼格式。再來，通常該語言的某一個函式庫會提供一系列的轉換方式，讓寫程式的人將文字型態或數值型態的時間轉換為該語言所獨有的時間物件，或將時間物件轉為文字或數值型態。



設定時區

設定時區（如果沒時區的話），當時區設定後，`as.POSIXct`會自動調整時區。

```
> Sys.setenv(TZ="Asia/Taipei")
> Sys.timezone()
[1] "Asia/Taipei"
```

```
> Sys.time()
[1] "2017-11-27 00:32:10 CST"
```

將字串轉為時間格式POSIXct/POSIXlt

而用文字表達時間的方法很多樣，因此就會有控制符號可以進行控制。

<https://www.rdocumentation.org/packages/base/versions/3.5.3/topics/strptime>

%Y Year as 2017

%m Month as decimal number (01–12).

%d Day of the month as decimal number (01–31).

%H Hours

%M Minute as 00–59.

%S Second as 00-59

%w Weekday as decimal number (0–6, Sunday is 0).

%a Abbreviated weekday name in the current locale

%b Abbreviated month name in the current locale on this platform

字串轉為時間格式

```
> safefood.df$ltime <- strptime(safefood.df$timestamp, "%m %e %Y")
> class(safefood.df$ltime)
[1] "POSIXlt" "POSIXt"
> class(as.Date("2017-01-01"))
[1] "Date"
> class(as.POSIXct("2017-01-08"))
[1] "POSIXct" "POSIXt"
```

浮點數或整數轉時間格式

```
> z <- 7.343736909722223e5
> as.POSIXct((z - 719529)*86400, origin = "1970-01-01", tz = "UTC")
[1] "2010-08-23 16:35:00 UTC"
> as.POSIXlt((z - 719529)*86400, origin = "1970-01-01", tz = "Asia/Taipei")
[1] "2010-08-24 00:35:00 CST"
```

改變時區

```
> as.POSIXct(Sys.time(), tz="CST")
[1] "2017-10-26 08:26:18 CST"
> as.POSIXct(Sys.time(), tz="Asia/Taipei")
[1] "2017-10-26 08:26:19 CST"
> as.POSIXlt(Sys.time(), tz="CST")
```

```
[1] "2017-10-26 00:26:19 GMT"
> as.POSIXlt(Sys.time(), tz="America/New_York")
[1] "2017-10-25 20:26:19 EDT"
> as.POSIXlt(Sys.time(), tz="Asia/Taipei")
[1] "2017-10-26 08:26:20 CST"
```

Accessing time objects features

POSIXlt

```
> safefood.df$ltime$mday
[1] 11 16 19 ... 1
...
> safefood.df$ltime$year # year since 1900
[1] 117 117 117 115 117 116 116 115 115 115 115 115 115
...
> safefood.df$ltime$wday # 0~6 day of the week
[1] 4 2 5 4 5 4 1 1 1 1 4 4 3 3 3 3 3 2 2 2 5 5 5 1 3 4 2 5
...
> safefood.df$ltime$yday # 0~365 day of the year
[1] 130 135 138 343 26 321 318 116 116 116 116 126 126 132
...
> safefood.df$ltime$zone # ChungYuan Standard Time
[1] "CST" "CST" "CST" "CST" "CST" "CST" "CST" "CST" "CST"
...
...
```

POSIXct

```
> safefood.df$ctime <- as.POSIXct(safefood.df$ltime) # POSIXct
> months(safefood.df$ctime)
[1] "May"      "May"      "May"      "December"
...
> sort(safefood.df$ctime)
[1] "2015-04-27 CST" "2015-04-27 CST" "2015-04-27 CST"
...
> sort(safefood.df$ltime)
[1] "2015-04-27 CST" "2015-04-27 CST" "2015-04-27 CST"
...
```

setlocale to

電腦的locale如果是在cht的話，那月份的顯示就會變成三月、四月等等國字，為了避免這樣的情形，可以將locate設定為C，C所代表的是C語言，此舉會讓時間的輸出變為英文的格式。

```
> Sys.setlocale(category = "LC_ALL", locale = "C")
[1] "C/C/C/C/zh_TW.UTF-8"
> format(safefood.df$ctime, "%m-%d-%Y")
[1] "05-11-2017" "05-16-2017" "05-19-2017" "12-10-2015"
...
> Sys.time()
[1] "2017-10-26 08:26:28 CST"
> Sys.Date()
[1] "2017-10-26"
```

Measure the computer execution time

```
> start <- proc.time()
> proc.time() - start
    user   system elapsed
 0.005   0.001   0.785
```

時間區間Interval與重疊

在計算Chrome History時，通常使用者在開Chrome瀏覽器時都會開多個分頁，此時，如果你將使用者每一個分頁的時間加起來，會使得總時間大於一天24小時。因此，必須要想辦法偵測那些分頁的使用時間是重疊的，才可以精準計算出使用facebook的時間佔瀏覽器總使用時間的比例。[lubridate](#)這個套件提供了interval物件來建立時間區間，並可用來判斷兩個時間區間是否重疊。

自POSIXct建立時間區間

```
library(lubridate)
sdata <- data %>%
  mutate(visit_duration = as.numeric(visit_duration), visit_time =
as.numeric(visit_time)) %>%
  filter(visit_duration > 1000000) %>%
  mutate(sec=visit_duration/1000000) %>%
  filter(!str_detect(url, "account|ftp|itc")) %>%
  filter(str_detect(url, "^http")) %>%
  mutate(fb = str_detect(url, "facebook")) %>%
  mutate(vtime = as.POSIXct(as.numeric(visit_time)/1000000, origin = "1601-01-01")) %>%
  mutate(date = as.character(format(vtime, format = "%m%d")),
         hour = as.numeric(format(vtime, format = "%H"))) %>%
  mutate(time_interval = lubridate::interval(vtime, vtime+sec))
```

合併重疊的時間區間

非常奇怪的是，如果把interval物件加到一個list中會使得其只剩下時間區間的秒數，原本的時間區間就消失了。

```
func_overlapping <- function(my){
```

```

merged <- c()
old <- my[1]

for(i in 2:length(my)){
  # print(i)
  # message("old:", old)
  # message("new:", my[i])
  if(int_overlaps(old, my[i])){
    start <- if(int_start(old)<int_start(my[i]))int_start(old) else
int_start(my[i])
    end <- if(int_end(old)>int_end(my[i]))int_end(old) else int_end(my[i])
    old <- interval(start, end, tzone = attr(old, "tzone"))
    # message("OVERLAP:", old)
  }
  else{
    merged <- c(merged, old)
    # message("NO-OVERLAP:", int_length(old))
    # message("LastMerged:", merged[length(merged)])
    old <- my[i]
  }
}
merged
}

```

補足缺少的時間

```

t <- post2016 %>%
  count(hour, wday) %>%
  spread(wday, n, fill=0) %>%
  right_join(data.frame(hour=0:23)) %>%
  mutate_all(funs(ifelse(is.na(.), 0, .)))

```

重點回顧

Data.frame

Factor

- 在Excel中的Nominal類別欄位，R會用factor來處理。並有助於對這些類別進行統計（例如在資料中有幾個男的幾個女的）

第二章、資料篩選與讀檔

2-0 何謂資料V

根據資料抓取的情形與，大致上可以把資料切分為以下2x2四種：

- 統計資料如果是歷史紀錄，多半抓下來就可以直接繪圖完工了。但如果該統計資料的單位不是你要的單位，例如台灣的氣候歷史資料是以「月」來統計，如果我們想要拿氣候資料和傳染病的資料進行歸因的話，傳染病卻是用「週」作為統計單位，就無法進行歸因。
- 紀錄資料通常要進一步進行彙整才能夠繪圖或者做分析。
- 即時更新的資料的話，則需要撰寫爬蟲幫研究者抓取多天的資料。即使做不出來時序上的差異，也可就時間點（例如每小時或週間等）進行平均

	觀察紀錄	統計資料
歷史紀錄	<ul style="list-style-type: none"> 登革熱近12個月每日確定病例統計 空品歷史紀錄資料 	<ul style="list-style-type: none"> 疾管局傳染病統計資料查詢系統
即時更新資料	<ul style="list-style-type: none"> 台北市即時交通資訊（沒有最小統計單位，只有車速等等所以並非統計資料） 	<ul style="list-style-type: none"> YouBike臺北市公共自行車即時資訊（以各站的腳踏車樹作為統計） 交通部 ETC 即時資料庫

登革熱近12個月每日確定病例統計

- <http://data.gov.tw/node/21026>

- 實際上並非「病例統計」，而是紀錄。所以資料的標題應該是「每日確診病例紀錄」

發病日	個案研判日	通報日	性別	年齡層	居住縣市	居住鄉鎮	居住村里	最小統計區	最小統計區中心點X	最小統計區中心點Y	一級統計區
2016/08/01	2016/08/09	2016/08/08	女	45-49	臺南市	新市區	社內里	A6720-0155-00	120.28718	23.07684	A6720-10-003
2016/08/01	2016/08/19	2016/08/05	女	15-19	台北市	內湖區	寶湖里	A6310-0825-00	121.59389	25.07177	A6310-60-002
2016/08/01	2016/08/07	2016/08/05	男	15-19	台北市	內湖區	寶湖里	A6310-0825-00	121.59389	25.07177	A6310-60-002
2016/08/01	2016/08/07	2016/08/05	男	20-24	台北市	內湖區	寶湖里	A6310-0825-00	121.59389	25.07177	A6310-60-002
2016/08/02	2016/08/07	2016/08/04	男	65-69	台中市	太平區	新福里	A6627-0363-00	120.7487	24.14718	A6627-29-001
2016/08/03	2016/08/06	2016/08/06	女	20-24	南投縣	埔里鎮	泰安里	A0802-0358-00	120.97576	23.96764	A0802-15-004
2016/08/03	2016/08/05	2016/08/04	女	50-54	新竹縣	竹北市	鹿場里	A0401-0885-00	121.0288	24.814	A0401-49-001
2016/08/03	2016/08/07	2016/08/04	男	25-29	新竹縣	竹北市	中興里	A0401-0885-00	121.0288	24.814	A0401-49-001
2016/08/04	2016/08/08	2016/08/07	女	10-14	桃園市	桃園區	泰山里	A0301-1610-00	121.31193	24.98763	A0301-A6-005

2-1 讀取Excel檔：產假支薪

導言

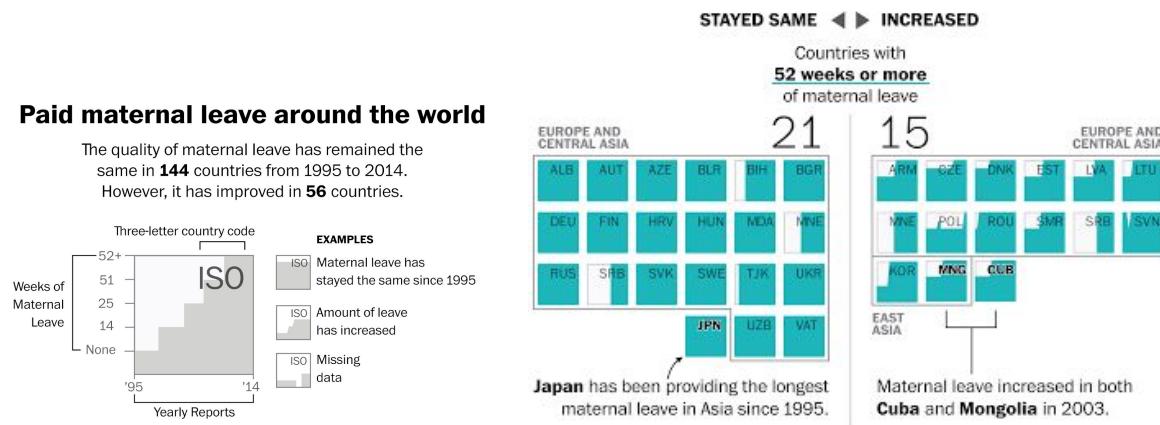
- 如何讀入Excel檔 (.xlsx, .xls) ？
- 運用R base套件篩選資料列與欄。如何從data.frame選出所需要的變項欄？如何篩出所要的資料列？
- 如何進行基礎的NA值處理？`is.na()`
- 如何繪製長條圖？如何繪製數個子圖？如何利用for-loop重複執行類似的程式碼？
`barplot()`

在這個案例中尤其要注意的是如何運用R的base套件選取所需的變項欄與篩取合乎條件的資料列，這是基本功，也是拿到資料、成功讀檔、確認變項變數型態後的下一件事，以及進入資料彙整（Summarization）前所需要思考的前置步驟。因為NA值的處理通常依照案例需求有不同的作法，繪圖更是需要根據案例、視資料類型、視覺化工具等進行調整。所以，我自己通常R寫久了以後，不見得會記得要怎麼繪圖（我會對我用過的函式留下一點點印象，然後用`ctrl-shift-f`查詢看看是否我在哪一個檔案曾經用過該函式。

讀取檔案 -> 確認變項變數型態 -> (處理NA) -> 選取變項欄／篩取資料列 -> 資料彙整 -> 視覺化

案例說明：產假支薪

本案例將利用R來重製華盛頓郵報在2016/08/13的一篇談論美國婦女產假支薪情形的報導。這個案例中將會應用到data.frame和基本的繪圖與資料摘要方法。（原始新聞來源：
Melissa Etehad & Jeremy C.F. Lin (August 13, 2016) The world is getting better at paid maternity leave. The U.S. is not. The Washington Post¹ :)



資料概況

matleave_95~matleave_13, total 19 years (columns)

197 countries (rows)	A	B	C	D	E	F	U	V	W
1	country	iso2	iso3	region	matleave_95	matleave_96	matleave_11	matleave_12	matleave_13
2	Afghanistan	AF	AFG	South Asia	2	2	2	2	2
3	Albania	AL	ALB	Europe & Central Asia	5	5	5	5	5
4	Algeria	DZ	DZA	Middle East & Africa	3	3	3	3	3
5	Andorra	AD	AND	Europe & Central Asia	2	2	3	3	3
6	Angola	AO	AGO	Sub-Saharan Africa	2	2	2	2	2

目標

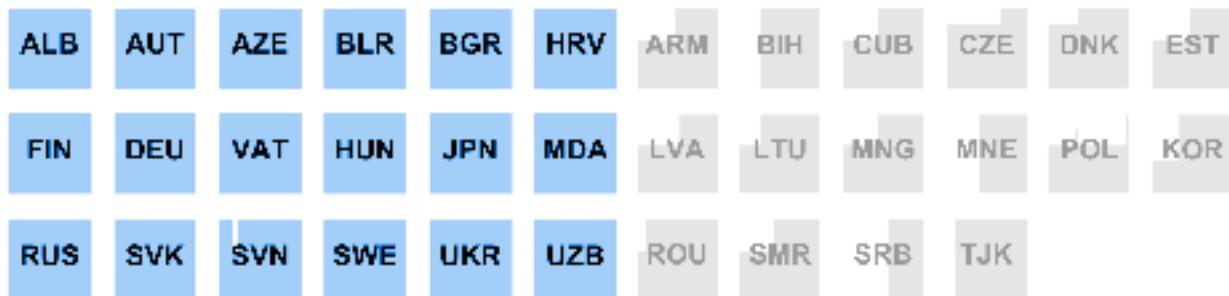
我希望仿照The Washington Post的報導，先以最後一年為基準分5, 4, 3, 2, 1（福利由高到沒有）數層來繪製。首先先繪製最後一年（matleave_13）為5的。然後再分左右兩半，

¹ Melissa Etehad & Jeremy C.F. Lin (August 13, 2016) The world is getting better at paid maternity leave. The U.S. is not. The Washington Post

https://www.washingtonpost.com/news/worldviews/wp/2016/08/13/the-world-is-getting-better-at-paid-maternity-leave-the-u-s-is-not/?utm_term=.060efaf71b59

左邊原為一開始有紀錄就是5的（包含最早matleave_95沒資料的如SRB和MNE），右邊原為一開始有紀錄就不是5的。但為了方便起見，我把問題簡化為「左半側為matleave_95為5的，右半側為matleave_95不是5的」如下圖，所以照我方法繪製出來的圖會有一點差異，MNE和SRB就被歸到右邊（灰色）第一年不是5的資料群。

而以下的範例程式，就以繪製出有紀錄的最後一年（matleave_13）為5，有資料的第一年（matleave_95）亦為5的區塊，也就是左半邊藍色的區塊。



步驟一：用readxl套件讀取Excel檔

```
# install.packages("readxl")
library(readxl)
options(stringsAsFactors = FALSE)

rawdata <- read_excel("data/WORLD-MACHE_Gender_6.8.15.xls", "Sheet1",
col_names=T)
```

- R的base套件中沒有內建讀取Excel檔的工具，所以需要安裝並載入讀取Excel檔的套件才有辦法讀取Excel檔。
- 為了避免讀取到文字型態chr的資料被程式自動轉為factor，使用`options(stringsAsFactors = FALSE)`自動將所有帶參數`stringsAsFactors`的參數值設為`FALSE`。
- 利用`?read_excel`查詢一下可以怎麼用該函式如下。help上提到，該函式的功能為讀取.xls與.xlsx檔，正確來說，是將.xls檔和.xlsx檔讀取後轉存為R的資料型態如`data.frame`或`list`。`read_excel()`函式中的`path`指的是檔案路徑、`sheet`指的

是哪一個資料表，可以給資料表名稱（如上例）或者指定第幾個資料表；skip則可以忽略掉一些Excel開頭可能包含的幾列不需要的詮釋資料，其他可以詳閱查詢help的說明。

```
read_excel(path, sheet = NULL, range = NULL, col_names = TRUE,
col_types = NULL, na = "", trim_ws = TRUE, skip = 0, n_max = Inf,
guess_max = min(1000, n_max))
```

- 通常讀取好資料後，我會習慣性地用View()指令來觀察該data.frame。

	country	iso2	iso3	region	wb_econ	matleave_95	matleave_96
1	Afghanistan	AF	AFG	South Asia		1	2
2	Albania	AL	ALB	Europe & Central Asia		2	5
3	Algeria	DZ	DZA	Middle East & North Africa		2	3
4	Andorra	AD	AND	Europe & Central Asia		4	2
5	Angola	AO	AGO	Sub-Saharan Africa		2	2
6	Antigua and Barbuda	AG	ATG	Americas		4	2

- 然後我會運用一些函式來協助我觀察一下該data.frame的特性。class()可以獲取資料型態、dim()可以獲知有幾列幾行（先列後行）、names()可以知道有哪些變項。
- 但我最常用的是str()，它告訴我這個其為一個data.frame，且有197個observation，也就是列，和156個變項。且str()會列出每個變數的變數名稱、變數型態（例如下方的iso2變數型態為chr，而wb_econ的變數型態為num，每個變數他會列出前幾筆資料。

```
> class(rawdata)
[1] "tbl_df"     "tbl"        "data.frame"

> dim(rawdata)
[1] 197 156

> names(rawdata)
[1] "country"      "iso2"        "iso3"
[4] "region"       "wb_econ"     "matleave_95"
[7] "matleave_96"   "matleave_97"   "matleave_98"
[10] "matleave_99"  "matleave_00"   "matleave_01"
```

```

...
> str(rawdata)
Classes 'tbl_df', 'tbl' and 'data.frame': 197 obs. of 156
variables:
 $ country           : chr "Afghanistan" "Albania" "Algeria"
 "Andorra" ...
 $ iso2              : chr "AF" "AL" "DZ" "AD" ...
 $ iso3              : chr "AFG" "ALB" "DZA" "AND" ...
 $ region            : chr "South Asia" "Europe & Central Asia"
 "Middle ... "
 $ wb_econ           : num 1 2 2 4 2 4 2 2 4 4 ...
 $ matleave_95        : num 2 5 3 2 2 2 2 3 1 5 ...
 $ matleave_96        : num 2 5 3 2 2 2 2 3 1 5 ...

```

步驟二：選取所需變項

select essential variables

```

matleave <- rawdata[ , c(3, 6:24)]
str(matleave)

```

- 選擇所需變項欄必須要把條件寫在`rawdata`中括號的逗號右側。我需要`iso3`的欄位作為國名的代號（`iso3`變項內的值為國家的三碼代號，例如台灣為TWN）。而`6:24`則為`matleave_95`至`matleave_13`的所有欄位。
- 我這邊是選出第三行、與六至二十四行。`c(3, 6:24)`為標準兩個vector相串連（concatinating）的結果，`6:24`相當於產生`c(6, 7, 8, ..., 24)`，而`c(3, 6:24)`則是把`3`擺在`6:24`之前。

步驟三：將NA值取代為0。

replace NAs as 0

```

matleave[is.na(matleave)] <- 0

```

- `NA`相當於「Not Available」或「Not a Value」的意思。通常若資料沒填到的話會有兩種情形，一種是什麼都沒有，系統會自動塞`NA`，另一種是系統會塞入一個長度為零的空字串「`""`」。
- `matleave[is.na(matleave)]`會選擇所有是`NA`的資料。`is.na(matleave)`會傳回一個包含`TRUE`與`FALSE`的向量。而`matleave`的中括號中若有`TRUE`或`FALSE`的向量，就會篩取出該位置為`TRUE`的資料。
- 該操作的目的是以`0`取代`NA`的資料格，以避免彙整運算或繪圖時產生錯誤。
- 有一些檢測是否`data.frame`或`vector`中還有沒有`NA`的方法。
 - 方法一：最簡單的是用`anyNA(matleave)`，如果還有`NA`的話，就會傳回`TRUE`；否則就會傳回`FALSE`。
 - 方法二：跑跑看`sum(is.na(matleave))`的結果會不會等於`0`。如果還有`NA`的話`is.na(matleave)`內的結果就還會包含`TRUE`。`TRUE`如果被當成數字加總總是`1`，`FALSE`為`0`。所以如果全部加總起來不是`0`，那就代表還有`NA`。

步驟四：篩出所需要的資料列

`filter rows by variable values`

```
m5 <- matleave[matleave$matleave_13 == 5, ]
m55<- m5[m5$matleave_95 == 5, ]
```

我先篩取出`matleave_13`年資料為5的欄位，再從中篩取出`matleave_95`年資料為5的資料。在這過程要特別留意，`matleave[matleave$'matleave_13' == 5,]`逗號前面為一個和`matleave`等大小、形狀相同的T/F向量。

建議若初學之時感到疑惑，在兩行程式碼前後可用`dim()`、`str()`、`nrow()`等函式觀察兩個階段分別剩下多少筆資料，或分別印出`matleave$matleave_13`、`matleave$matleave_13==5`、`length(matleave$matleave_13==5)`來觀察。

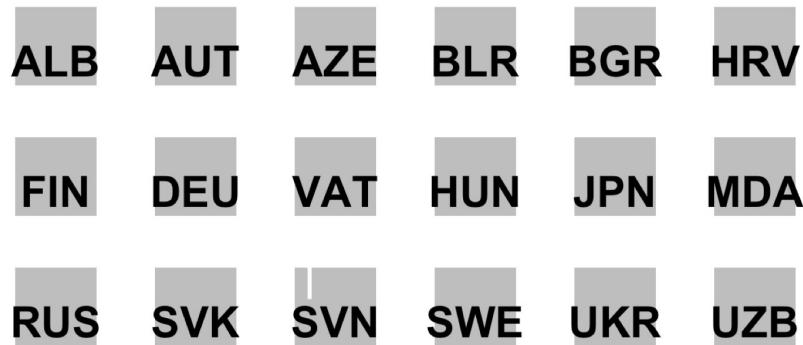
步驟五：繪圖

```
par(mfrow=c(4,6), mai= c(0.2, 0.2, 0.2, 0.2))
for (i in c(1:nrow(m55))){
```

```

    barplot(unlist(m55[i,-1]),
            border=NA, space=0, xaxt="n", yaxt="n", ylim =
c(0,5))
    title(m55[i,1], line = -4, cex.main=3)
}

```



基本繪圖很簡單，就只需要plot(x)或plot(x, y)，x與y之處分別給他向量，就可以繪製成散布圖（scatter）。但沒有調整過的圖通常很醜，所以，通常我會假想一個預期希望看見的樣子，然後朝那個目標逐一測試繪圖的參數（Arguments），過程概略如下。

先plot一列資料看看。

```

> barplot(m55[2, ])      # raise error
Error in barplot.default(m55[2, ]) :
  'height' must be a vector or a matrix

```

錯誤訊息說height必須要是一個vector，用?barplot查詢一下他的用法為
barplot(height, ...)，第一個參數必須要為height，顯然是個數值資料。因此自我檢查後，想起第一個變項欄為iso3國碼，因此我除了篩出第二列出來繪圖外，另外把第一欄給刪除。

```

> barplot(m55[2, -1])    # raise error
Error in barplot.default(m55[2, -1]) :
  'height' must be a vector or a matrix

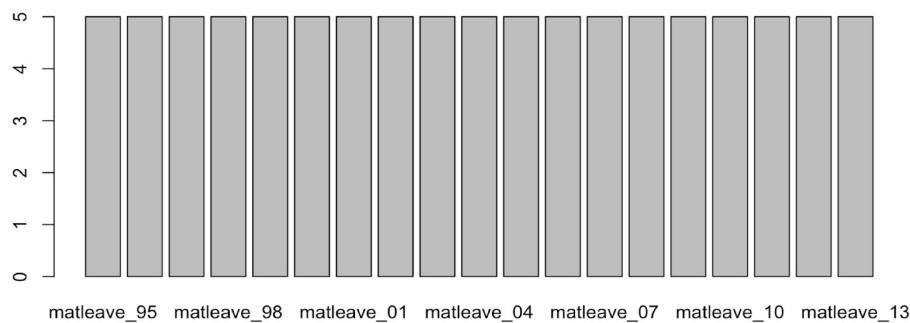
```

結果仍然是同樣的錯誤，顯然錯誤仍沒解決。因此，「經驗老道（意味著初學者可能沒辦法自己想到）」的我想到，搞不好這個資料型態不是vector或matrix，所以畫不出來。照道理來說height應該要是個numeric vector。因此用class()函式來測試一下資料的型態。

```
> class(matleave[2, -1])
[1] "tbl_df"     "tbl"        "data.frame"
```

果不其然，雖然取的是一橫列資料，但他不是一個vector，他相當於是一個只有一筆資料的data.frame，雖然我們「感覺上」那是一列資料，很像vector，但他不是就不是，想太多！我確定這樣篩出來是我要繪製的資料項目沒錯，但我需要它是一個vector而不是data.frame，此時就要用變數型態的轉換，一個無論多大的data.frame想要拆成一長條的vector，就是用unlist()來轉，便會產生vector。此時我便可利用barplot()將其繪製為長條圖如下。

```
> class(unlist(matleave[2, -1]))
[1] "numeric"
> barplot(unlist(m55[2, -1]))
```



接下來我會一一測試barplot()的參數把他畫成我要的樣子，他有哪些參數可查詢？
barplot。最後我所得到的長條圖如下。`ylim`是為了確保之後要繪製等級4至1時，每個小圖仍然是等高的。`space`指的是bar之間的間隙、`border`顧名思義、並利用`xaxt="n"`將x軸與y軸座標系隱藏起來。

```
barplot(unlist(m55[2, -1]))
barplot(unlist(m55[2, -1]), ylim=c(0, 5))
```

```
barplot(unlist(m55[2, -1]), ylim=c(0, 5), space=0)
barplot(unlist(m55[2, -1]), ylim=c(0, 5), space=0, border=NA)
barplot(unlist(m55[2, -1]), ylim=c(0, 5), space=0, border=NA, xaxt="n", yaxt="n")
```

但此時我不只要畫一張圖，數一數一共要畫18張子圖，先畫六張的程式碼如下。底下可以看見每一行非常相似且一致的特徵，僅有m55內的「資料列索引」由1被列出至6。

```
barplot(unlist(m55[1, -1]), ylim=c(0, 5), space=0, border=NA, xaxt="n", yaxt="n")
barplot(unlist(m55[2, -1]), ylim=c(0, 5), space=0, border=NA, xaxt="n", yaxt="n")
barplot(unlist(m55[3, -1]), ylim=c(0, 5), space=0, border=NA, xaxt="n", yaxt="n")
barplot(unlist(m55[4, -1]), ylim=c(0, 5), space=0, border=NA, xaxt="n", yaxt="n")
barplot(unlist(m55[5, -1]), ylim=c(0, 5), space=0, border=NA, xaxt="n", yaxt="n")
barplot(unlist(m55[6, -1]), ylim=c(0, 5), space=0, border=NA, xaxt="n", yaxt="n")
```

這種完全重複、只有索引相異的指令，最好的方法是用迴圈（for-loop）的方式將相同的程式碼，用一個新的變數來控制要畫哪一列，從1至6之間做六次。

```
for(i in 1:6){
  barplot(unlist(m55[i, -1]), ylim=c(0, 5), space=0, border=NA, xaxt="n", yaxt="n")
}
```

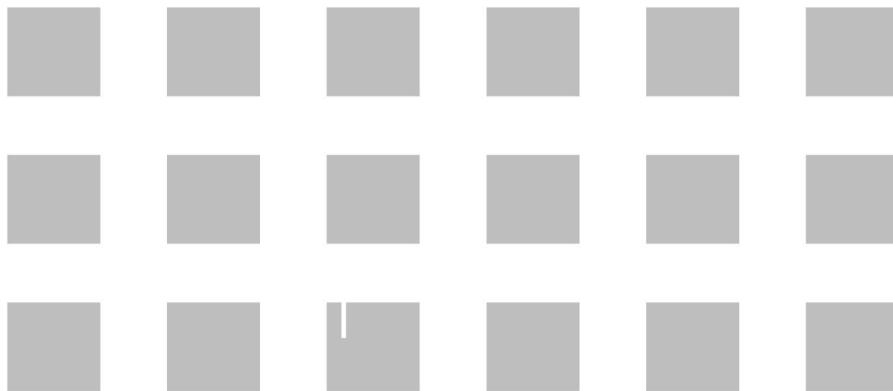
所以，確定可以用for迴圈幫我畫出六張圖，我就把1:6改成1:nrow(m55)，nrow(m55)為m55的資料筆數。

但是像上面這樣做只會讓程式碼重新畫十八張圖。因此要上網查詢看看如何繪製子圖。你可以google「subplot in R」或「multiple plot in R」應該都可以找到相同的結果。有一個網頁是「R Multiple Plot Using par() Function」（

<https://www.datamentor.io/r-programming/subplot/>），他告訴你說用par()這個函式可以控制把數個子圖畫在同一張圖上。你可以查詢看看?par的參數，並且找到mai、mfcol和mfrow等參數的說明如下，並據此繪製成下圖。

- mai: A numerical vector of the form c(bottom, left, top, right) which gives the margin size specified in inches.
- mfcol, mfrow: A vector of the form c(nr, nc). Subsequent figures will be drawn in an nr-by-nc array on the device by columns (mfcol), or rows (mfrow), respectively.

```
par(mfrow=c(3,2), mai= c(0.2, 0.2, 0.2, 0.2))
for(i in 1:nrow(m55)){
  barplot(unlist(m55[i, -1]), ylim=c(0, 5), space=0, border=NA, xaxt="n", yaxt="n")
}
```



從上圖中可以看見我幾乎快完成了。只差把國名，也就是`iso3`，打在該國的方塊上。在`barplot`上面打標題可以google「barplot title R」他會教你怎麼上title，但你也可以查詢「plot annotation R」也就是在圖上面繪製文字（和`barplot`分開繪製，會比較好控制字型，程式語言把這種上文字稱為annotation）。此時，若直接打`title(m55[i, 1])`已經能夠繪製文字，但不是我要的大小，我會進一步查詢`?title`怎麼控制文字大小和文字的「基線」，也就是把哪裡當成文字的底線。經查詢後，控制文字大小用`cex.main`、控制基線用`line`。然後我就自己測試看看大小和基線，直到我滿意如下圖。

```
par(mfrow=c(4,6), mai= c(0.2, 0.2, 0.2, 0.2))
for (i in 1:nrow(m55)){
  barplot(unlist(m55[i,-1]), ylim=c(0,5), space=0, border=NA, xaxt="n",yaxt="n")
  title(m55[i,1], line = -4, cex.main=3)
}
```



練習：完成其他的繪圖

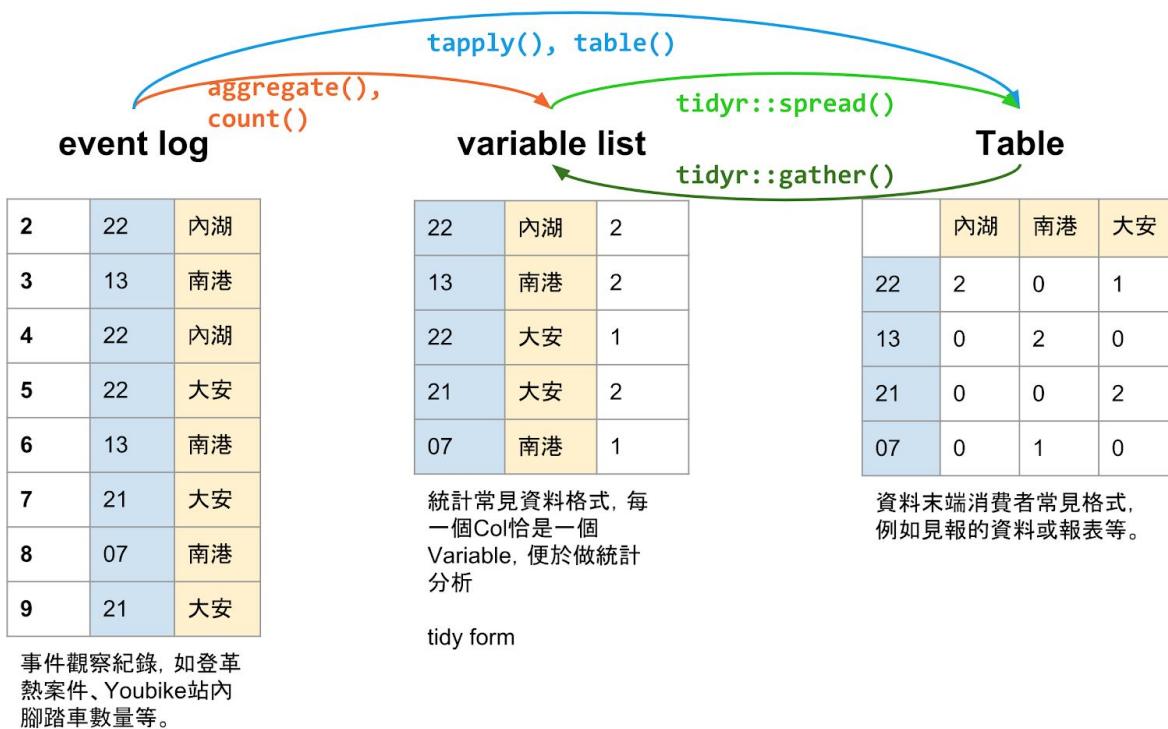
2-2 讀取CSV檔：台北市竊盜案

步驟一：摘要與提要

除了延續前節篩取資料列和選取變項欄外，本節的重點在於利用count()或tapply()等函式進行計數。一般來說，觀察得到的資料是觀察紀錄（例如在路口蹲點計算車流或者經過的行人類型）。但能夠被計算或比較的是觀察紀錄後的計量，也就是具目的性地抽取數個變項，然後看看各個變項組合有多少筆適才的觀察紀錄。最後，為了一般讀者閱讀，會經常把其中兩個變項抽出來，然後把一個變項當成欄，一個變項當成列，然後交叉呈現兩個變項下的資料計數。

讀取CSV檔

- (更改變項名稱為英文)
- 用substr()取出部分文字以產生新的變項
- 計數／彙整
- 繪圖



```
options(stringsAsFactors = FALSE)
```

步驟二：載入台北市竊盜案資料

```
url <-  
"http://data.taipei/opendata/datalist/datasetMeta/download?id=6878523  
1-d6c5-47a1-b001-77eec70bec02&rid=34a4a431-f04d-474a-8e72-8d3f586db3df"  
df <- read.csv(url, fileEncoding = "big5")
```

df <- read.csv(url, fileEncoding = "big5")這行的意思是把url用read.csv()這個函式讀取。讀取的同時，由於一般EXCEL的中文編碼為BIG5，所以該文件八成是該台北市政單位還用EXCEL在編資料，所以要跟程式碼講說，這個網址所指到的檔案編碼為BIG5。請自行嘗試看看如果沒有加入fileEncoding會有什麼錯誤訊息。

部分電腦可能為無法直接讀取，此時可以先自行把檔案下載下來後再讀取，唯獨記得要把該資料檔拖至本專案的資料夾內。例如以下面的例子來說，我把tptheft.csv拖至現在專案中的data資料夾，因此以下路徑正確而可以讀取。從不同的瀏覽器下載下來可能會產生不一樣的csv檔，如果無法開啟可以多嘗試不同的瀏覽器看看。

```
df <- read.csv("data/tptheft.csv", fileEncoding = "big5")
```

用View()瀏覽後可看見該data.frame的變項值與變項概略如下。

	編號	案類	發生.現.日期	發生時段	發生.現.地點
1	1	住宅竊盜	780210	10~12	台北市中山區新生北路3段31 ~ 60號
2	2	住宅竊盜	951016	19~21	台北市信義區三張里莊敬路341巷1 ~ 30號
3	3	住宅竊盜	970609	01~03	台北市信義區松隆里松山路615巷1 ~ 30號
4	4	住宅竊盜	990407	04~06	台北市萬華區雙園街61 ~ 90號
5	5	住宅竊盜	991013	22~24	台北市內湖區成功路4段331 ~ 360號
6	6	住宅竊盜	991024	13~15	台北市南港區東新里興南街52巷1 ~ 30號

步驟三：產生新的變項

「發生時段」我打算取出前面的數字來代表時間就好，「發生地點」我打算只取出行政區名，其他地址不要。邏輯上，我要把那串字取出第x個字到第y個字，所以要用substr()這個函式，或者未來會教到的stringr::str_sub()函式。請用?substr查詢其用法和意義「getting sub string since x to y」。

```
df$time <- substr(df$發生時段, 1, 2)
df$region <- substr(df$發生.現.地點, 4, 5)
```

另外我有發現前面幾年的資料特別少，所以我想要取出現在是幾年。我用str()觀察df後發現「發生日期」為一個整數，那代表我可以對其做加減乘除做運算。該變項的數值最多有7位，部分為6位，因為資料是從99年跨至100至104年。我若要取出這是幾年的資料，經觀察後我打算將該整數除以10000取商，剛好可以獲得年份的資料。除法取商的語法為%/%。

```
df$year <- df$發生.現.日期 %/% 10000
```

步驟四：整理、清理資料

該資料可發現，有幾年的資料在特別早之前，似乎直到104年才逐漸穩定，但不太確定。所以我首先要刪除那些資料太少的年份。所以我打算用 `tapply()` 這個函式看看究竟每個年份出現幾次。`tapply()` 的說明如下。`tapply` 可以將所有資料，根據某個離散變項進行分組，並且指定一個函式來計算。例如：我現在要算各年的資料筆數，我就只需要把所有編號（唯一），根據年，用 `length` 函式來計算所有編號在各年各出現幾次。

- Description: Apply a function to each cell of a ragged array, that is to each (non-empty) group of values given by a unique combination of the levels of certain factors.

從資料上可看得出來，應該只有104年後的資料是正常的。其他都只有個位數，所以我就篩取資料年份大於103年的資料，經過篩選後剩下2150筆資料。

```
> tapply(df$編號, df$year, length)
 78   95   97   99  100  101  102  103  104  105  106  107
 1     1     1     4     3     7     8     5   619   648   505   378
df <- df[df$year > 103, ]
```

題外話，我也可以不見得要用 `length`。例如我在 maternal leave 的案例中，我打算計算每個區域（北歐、東歐、俄羅斯）的平均產假支薪等級，那我就需要把所有的 `matleave_13` 根據資料中的 `region` 來做分組，並取平均。此時我的運算式會是 `tapply(rawdata$matleave_13, rawdata$region, mean)`，分別顯示出每個地域的平均。

```
> tapply(rawdata$matleave_13, rawdata$region, mean)
Americas          East Asia & Pacific        Europe & Central Asia
2.371429           2.281250            4.309091
Middle East & North Africa      South Asia       Sub-Saharan
Africa
                           2.421053           2.125000
2.500000
```

步驟五：計數／彙整／摘要：使用 `tapply()`

我們要回答的第一個數據問題通常是，那XXX的案例有幾個？例如大安區有多少竊盜案？買超過10000元訂單的客戶有多少人？男生和女生會修程式課的個別有多少人？這稱為計數。最基礎的計數如上述 `tapply()` 的形式。

```
> tapply(df$編號, df$time, length)
 00  01  02  04  06  07  08  10  12  13  14  16  18  19  20  22
 17 163  24 156  25 229  25 353  34 253  42 253  32 245  36 263
> tapply(df$編號, df$region, length)
中和 中山 中正 信義 內湖 北投 南港 士林 大同 大安 文山 松山 板橋 淡水 萬華
 1 263 159 141 204 223 118 242 111 190 146 145 4 1
202
```

此時應會發現有幾個時間點（00, 02, 06, 08, 12, 14, 18, 20）和幾個非台北市地區（中和、板橋、淡水）的資料比數比別人少很多，因此要把這些資料給篩除。因為這些不要的資料都是文字型態的資料，且有數個，所以我們可以使用%in%這個運算符號，他指的是該變項的值是不是後面這個vector中的其中一個值。但這樣是判斷資料有沒有在這些不完整的資料中，前面要加一個驚嘆號，!代表邏輯上的NOT。前面加上一個NOT相當於我篩取出該變項值不包含在後面向量中的其中一個。經過這樣的篩選後，就只剩下1914筆資料了。

```
df <- df[!df$time %in% c("00", "02", "06", "08", "12", "14", "18",
"20"),]
df <- df[!df$region %in% c("中和", "板橋", "淡水"), ]
```

上述案例只能根據一個資料維度或根據一個變項來做計數彙整。然而，我們希望知道，不同的時間和不同的地點，是否竊盜比例會有所不同，此時我們需要同時考慮兩個變項。寫法如下。list()是指把括號內的該兩個vectors合起來後一併轉為同一個vector。計算完後的結果如下圖：

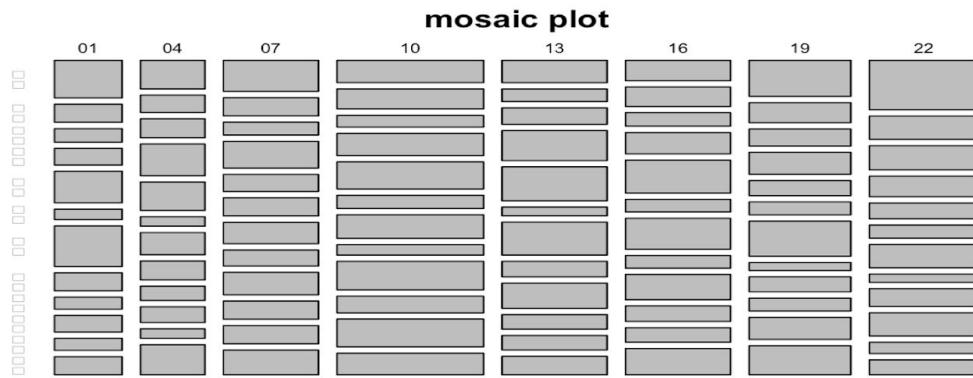
```
res <- tapply(df$編號, list(df$time, df$region), length)
```

	中山	中正	信義	內湖	北投	南港	士林	大同	大安	文山	松山	萬華
01	25	12	9	11	21	7	27	12	8	11	8	12
04	18	11	12	20	18	6	14	12	9	10	6	19
07	29	17	12	25	16	17	20	15	21	17	17	23
10	32	29	17	32	39	19	34	15	41	24	40	31
13	23	13	17	31	35	9	34	16	26	15	15	19
16	21	20	14	22	34	13	32	13	26	16	15	27
19	36	20	17	22	15	13	35	8	15	13	22	29
22	53	25	26	22	17	14	25	9	19	25	12	16

步驟六：視覺化

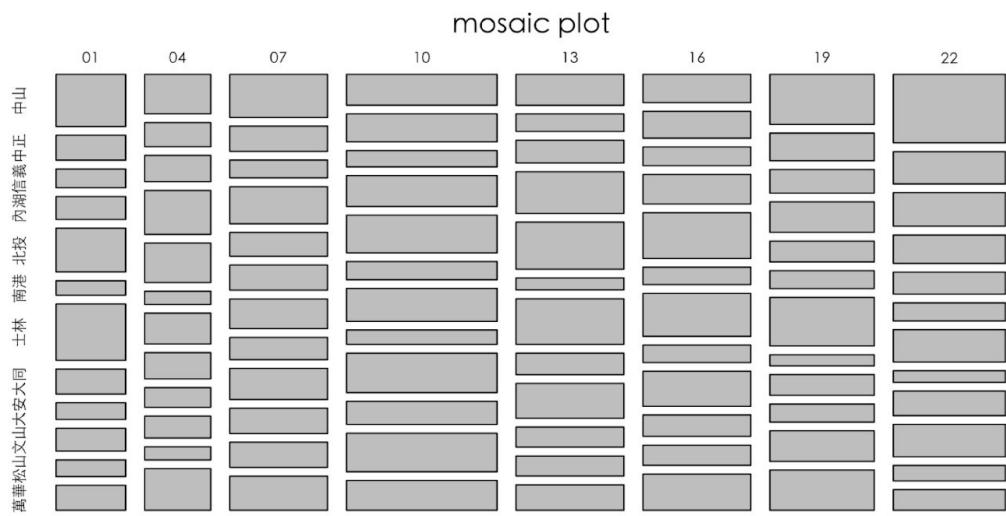
`mosaicplot()`非常擅長視覺化兩個離散變項，並透過自動化百分比的設置，可以讓資料探索者得知，哪些變項的大小分佈跟其他不同。

```
mosaicplot(res)
mosaicplot(res, main="mosaic plot")
```



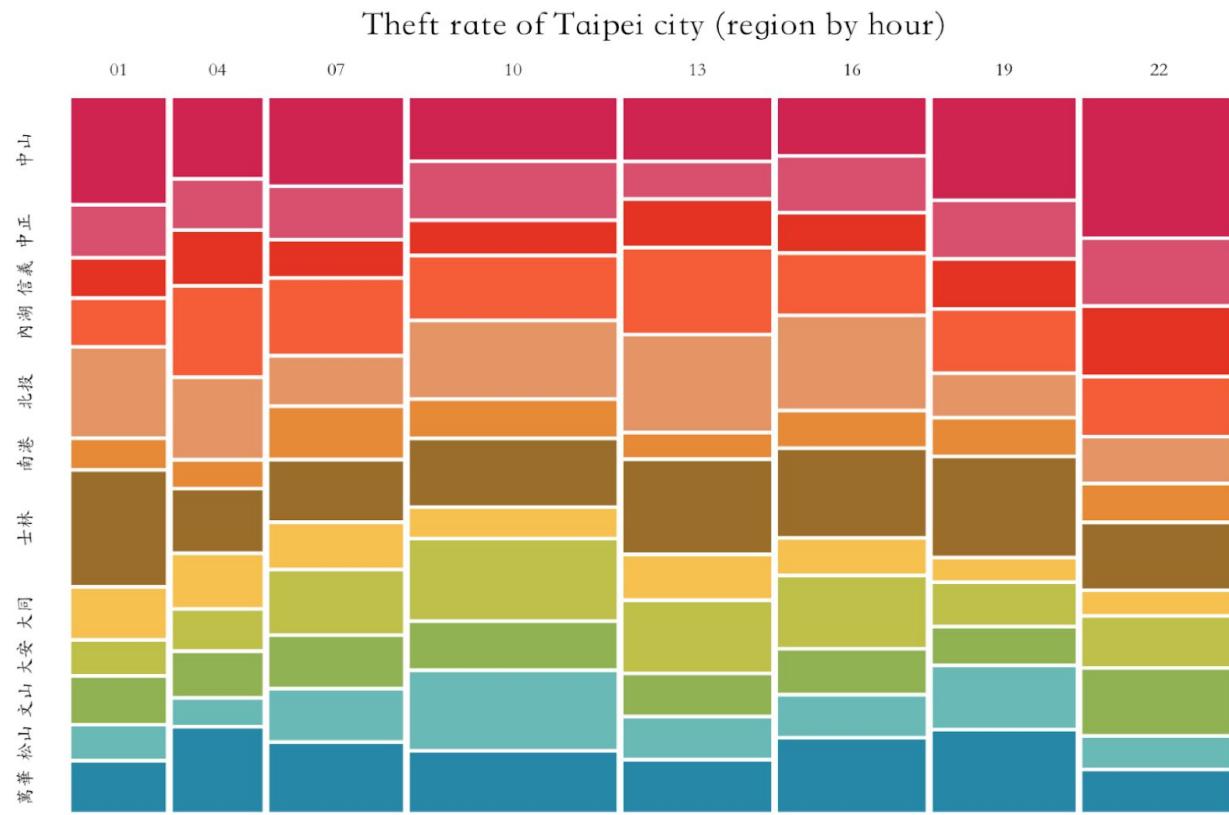
但是，類似上述繪圖無法顯示中文要怎麼辦？未來R的繪圖尤其是Mac會經常遇到無法顯示中文的問題，Windows未來在用`ggplot2`套件進行繪圖時也會遇到相同的問題。因此，我們需要告訴電腦要用什麼字體。例如Mac上可以用以下兩種字體。

```
par(family='Heiti TC Light')
# par(family='STKaiti')
mosaicplot(res, main="mosaic plot")
```



用自訂顏色來觀察會更清楚，一共有十二個區，就給予十二種顏色。

```
colors <- c( '#D0104C', '#DB4D6D', '#E83015', '#F75C2F',
           '#E79460', '#E98B2A', '#9B6E23', '#F7C242',
           '#BEC23F', '#90B44B', '#66BAB7', '#1E88A8')
par(family='STKaiti')
mosaicplot(res, color=colors, border=0, off = 3,
           main="Theft rate of Taipei city (region by hour)")
```



綜合：資料彙整（Data Summerazation）

在R中可以依照某個類別變數來計算次數、平均的函式有很多個。`tapply()`是原始base套件中的函式，其家族還有`sapply()`、`lapply()`、`apply()`等函式。但在快速的資料探索性分析中，反而會比較常用`table()`，而善於操作變項的統計書籍則常用`aggreage()`，後期開發的`dplyr`套件則有`count()`。在未來的練習中，最常用的是`count()`。

方法二：`table()`

以下為`table()`的用法，他也可以用於單一變項的計數，且他產生的結果並非一個`data.frame`而是一個資料型態稱為「table」，查詢看看`?table`以獲得更詳盡的說明。

```
> res <- table(df$time, df$region)
> class(res)
[1] "table"
> res
```

	中山	中正	信義	內湖	北投	南港	士林	大同	大安	文山	松山	萬華
01	25	12	9	11	21	7	27	12	8	11	8	12
04	18	11	12	20	18	6	14	12	9	10	6	19
07	29	17	12	25	16	17	20	15	21	17	17	23
10	32	29	17	32	39	19	34	15	41	24	40	31
13	23	13	17	31	35	9	34	16	26	15	15	19
16	21	20	14	22	34	13	32	13	26	16	15	27
19	36	20	17	22	15	13	35	8	15	13	22	29
22	53	25	26	22	17	14	25	9	19	25	12	16

方法三 : count()

以下為 `count()` 的用法。注意，將 `df` 根據 `time` 與 `region` 兩個類別變項計數後，會多一個 `n` 的變項，指出某個時間、某個區域有多少起竊盜案件。這樣的型態是統計學最常看見的型態，也就是 variable-based 的型態。但如果我們希望它變成一個「二維」的表格，亦即欄和列分別代表兩個不同的變項，那就需要用 `tidyR` 的函式 `spread()` 將你所要展開到欄的那個變項，和計數後的數字 `n` 作為參數，展開為一個二維的表格如下（你也可以選擇展開 `time`）。

關於 `tidyR:::spread()` 這樣的表示法，在 `:::` 前為套件名稱，當程式尚未用 `library(tidyR)` 載入該套件，或者你只打算用 `spread()` 這麼一次，可以用這種表示法，指出這個 `spread()` 為 `tidyR` 這個套件中的函式。據此，R 會自動載入該套件並執行該函式。

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

time <chr>	region <chr>	n <int>	time <chr>	中山 <dbl>	中正 <dbl>	信義 <dbl>	內湖 <dbl>	北投 <dbl>	南港 <dbl>	士林 <dbl>
01	中山	25	01	25	12	9	11	21	7	27
01	中正	12	04	18	11	12	20	18	6	14
01	信義	9	07	29	17	12	25	16	17	20
01	內湖	11	10	32	29	17	32	39	19	34
01	北投	21	13	23	13	17	31	35	9	34
01	南港	7	16	21	20	14	22	34	13	32
01	士林	27	19	36	20	17	22	15	13	35
01	大同	12	22	53	25	26	22	17	14	25
01	大安	8								
01	文山	11								

練習

- 用 `tapply()`、`table()` 和 `count()` 與 `spread()` 後所跑出來的表格看似都是欄和列分別有兩個變項，請用 `class()` 測試看看，用不同的方法所產生的資料型態有何差異。

2-3 讀取JSON檔

預覽與回顧

```
## Type I: Well-formatted JSON: UVI, AQI, Hospital_revisits
```

這類的資料以典型的`[{}, {}, {}]`形式儲存，以下方式就可直接轉為 `data.frame`
`df <- fromJSON(content(GET(url), "text"))`

```
## Type II: hierarchical JSON: rent591, facebook graph api, google map
```

這類的json資料為well-formatted，但要的資料儲存在比較深的階層中，代表其並非簡單地二維表格，還有其他更多的詮釋資料被擺在同一個JSON檔案中。解決策略：通常 `fromJSON()` 轉完後為 `list`，逐一就 `variable names` 查看資料在哪裡。

```
View(res$data$data)
```

```
## Type III: Ill-formatted JSON: food_rumors, ubike
```

這類的資料並非以典型的`[{}, {}, {}]`形式儲存，但仍是有序的二維數據。可將資料`unlist()`攤開，然後去除不必要的NA後，按欄位數目重建Matrix再轉回`data.frame`。解決策略：用`as.data.frame()`或`unlist()`硬轉成`data.frame`或`vector`來看資料樣貌是否有規律。

2-3-1 何謂JSON檔案？

JSON（JavaScript Object Notation）是一種資料的儲存與表示方法，也就是一種檔案格式。資料可儲存格式有很多種，如XML、JSON、CSV等常見格式。例如政府開放資料的紫外線資料<http://data.gov.tw/node/6076>就包含這三類資料格式。資料格式不同，用來存取資料的程式碼就會不同。

資料的核心概念是相同的，但可以用不同的方式來表述資料，亦即儲存為不同格式的資料檔案。比方說，就通訊錄而言，一個人就是一筆資料，這筆資料通常有姓名、住址、電話等等你所熟悉的詮釋欄位，問題是，要用什麼方式來儲存它，之後才會比較好利用？而不同的儲存方法，在取用上是否有所差異？在資料的儲存彈性上、可以容許的資料維度、可包含的詮釋欄位結構彈性又有多大。而CSV、JSON和XML/HTML（HTML與XML均為Mark-up Language）即是三種不同表述資料的格式。

CSV擅長儲存表格型態的二維資料，或者需要用長列表的型態儲存樹狀資料，JSON和XML則擅長儲存樹狀、階層或高維度資料。XML除了可以標示階層和欄位外，還易於設計類別，或者加入物件的詮釋項目。比較上，CSV彈性最低，而XML可儲存最複雜的資訊。

JSON格式利用兩個記號來紀錄資料的結構：

- `{}`為無序的欄位名稱到欄位值的對應 unordered key-to-value pair
- `[]`為有序的資料項目 ordered data entry

(Option) 在瀏覽器上或電腦上安裝JSON Viewer

在瀏覽器上可安裝JSON Viewer²，在電腦上可安裝SublimeText並安裝其JSON Reindent外掛套件。如何安裝可見此教學影片 <https://youtu.be/2M1ww-WTJ2k>。

² JSON Viewer:

<https://chrome.google.com/webstore/detail/json-viewer/gbmdgpbipfallnflgajpalibnhhdgobh>

Instructions of Installing SublimeText and JSON ReIndent

1. Download sublime text
2. Install
 - a. Tools > Command Palette
 - b. install package control
 - c. Package Control: Install Package
 - d. JSON Reindent
3. Reformat JSON
 - a. Open JSON file by sublime text
 - b. Tools> Command Palette
 - c. JSON Reindent

範例：空氣品質指標的JSON檔案

以下為一個空氣品質開放資料前兩筆資料的JSON版本，黃色底線部分為第一筆資料。

```
[{"AQI": "78", "CO": "0.22", "CO_8hr": "0.2", "County": "基隆市", "NO": "1.1", "NO2": "4.7", "NOx": "5.8", "O3": "54", "O3_8hr": "42", "PM10": "37", "PM10_AVG": "31", "PM2.5": "28", "PM2.5_AVG": "26", "Pollutant": "細懸浮微粒", "PublishTime": "2017-10-03 16:00", "SiteName": "基隆", "SO2": "1.4", "Status": "普通", "WindDirec": "81", "WindSpeed": "2.7"}, {"AQI": "78", "CO": "0.24", "CO_8hr": "0.3", "County": "新北市", "NO": "1", "NO2": "11", "NOx": "12", "O3": "46", "O3_8hr": "40", "PM10": "40", "PM10_AVG": "39", "PM2.5": "25", "PM2.5_AVG": "27", "Pollutant": "細懸浮微粒", "PublishTime": "2017-10-03 16:00", "SiteName": "汐止", "SO2": "2.6", "Status": "普通", "WindDirec": "36", "WindSpeed": "3"}, ]
```

2-3-2 比較JSON與CSV

例一：紫外線指標（UVI）

紫外線指標資料的CSV格式如下：

```
SiteName,UVI,PublishAgency,County,WGS84Lon,WGS84Lat,PublishTime
花蓮,5.59,中央氣象局,花蓮縣,"121,36,48","23,58,30",2015-11-23 12:00
馬祖,4.49,中央氣象局,連江縣,"119,55,24","26,10,09",2015-11-23 12:00
高雄,4.39,中央氣象局,高雄市,"120,18,57","22,33,58",2015-11-23 12:00
玉山,8.14,中央氣象局,南投縣,"120,57,34","23,29,15",2015-11-23 12:00
臺南,4.45,中央氣象局,臺南市,"120,12,17","22,59,36",2015-11-23 12:00
```

相對的紫外線指標資料JSON格式如下：

```
[
  {
    "SiteName": "花蓮",
    "UVI": "5",
    "PublishAgency": "中央氣象局",
    "County": "花蓮縣",
    "WGS84Lon": "121,36,18",
    "WGS84Lat": "23,58,37",
    "PublishTime": "2015-11-23 11:00"
  },
  {
    "SiteName": "馬祖",
    "UVI": "4",
    "PublishAgency": "中央氣象局",
    "County": "連江縣",
    "WGS84Lon": "119,55,23",
    "WGS84Lat": "26,10,10",
    "PublishTime": "2015-11-23 11:00"
  },
  {
    "SiteName": "高雄",
    "UVI": "4",
    "PublishAgency": "中央氣象局",
    "County": "高雄市",
    "WGS84Lon": "120,18,29",
    "WGS84Lat": "22,34,04",
    "PublishTime": "2015-11-23 11:00"
  }
]
```

例二：空氣品質指標（AQI）

CSV格式資料如下，下載自<https://data.gov.tw/dataset/6075>。

```
SiteName,County,AQI,Pollutant,Status,SO2,CO,CO_8hr,O3,O3_8hr,PM10,PM2
.5,N02,NOx,NO,WindSpeed,WindDirec,PublishTime,PM10_AVG,PM2.5_AVG
麥寮,雲林縣,104,細懸浮微粒,對敏感族群不良,
1.9,0.18,0.2,41,50,101,34,3.1,3.6,0.5,5.3,4.5,2017-10-03 16:00,79,37
關山,臺東縣,33,,良好,,,,,21,,,,,,1,179,2017-10-03 16:00,19,10
馬公,澎湖縣,55,細懸浮微粒,普通,
3.4,0.19,0.2,50,44,37,21,3.2,4.7,1.5,3.4,337,2017-10-03 16:00,28,17
金門,金門縣,101,細懸浮微粒,對敏感族群不良,
2.6,0.21,0.3,54,55,52,32,5,5.9,0.8,4.5,104,2017-10-03 16:00,64,36
馬祖,連江縣,68,細懸浮微粒,普通,
2.3,0.2,0.3,55,50,53,26,,,5,45,2017-10-03 16:00,47,22
...
...
```

若用R成功讀取，轉為data.frame後，用View()打開的局部截圖：

	SiteName	County	AQI	Pollutant	Status	SO2	CO	CO_8hr	O3	O3_8hr	PM10	PM2.5
1	二林	彰化縣	59	細懸浮微粒	普通	2.8	0.38	0.5	7	3	36	14
2	三重	新北市	35		良好	5.9	3.44	1.3	-		58	8
3	三義	苗栗縣	53	細懸浮微粒	普通	2.5	0.57	0.4	11	17	42	25
4	土城	新北市	31		良好	2.9	1.14	0.5	3.5	5	30	16
5	士林	臺北市	28		良好	3.6	1.1	0.7	1.8	4	41	ND
6	大同	臺北市	37		良好	4.8	2.45	1.3	-		45	14
7	大里	臺中市	60	細懸浮微粒	普通	2.7	0.88	0.6	2.2	2	36	9
8	大園	桃園市	57	細懸浮微粒	普通	1.8	0.4	0.5	34	14	33	14
9	大寮	高雄市	77	細懸浮微粒	普通	2.3	0.52	0.4	8.7	7	55	29
10	小港	高雄市	56	細懸浮微粒	普通	5	0.84	0.5	7.7	9	50	27
11	中山	臺北市	39		良好	4.7	1.45	0.8	2.1	5	37	7
12	中壢	桃園市	48		良好	3.9	1.82	0.9	1.9	6	51	22
13	仁武	高雄市	36		良好	4.3	0.75	0.5	6.5	8	50	7

JSON格式資料如下：

```
[
  {
    "SiteName": "二林",
    "County": "彰化縣",
    "AQI": "77",
    "Pollutant": "細懸浮微粒",
    "Status": "普通",
```

```

    "SO2": "3.7",
    "CO": "0.19",
    "CO_8hr": "0.2",
    "O3": "15",
    "O3_8hr": "27",
    "PM10": "64",
    "PM2.5": "37",
    "NO2": "8.5",
    "NOx": "8.8",
    "NO": "0.3",
    "WindSpeed": "1",
    "WindDirec": "171",
    "PublishTime": "2017-09-27 23:00",
    "PM10_AVG": "49",
    "PM2.5_AVG": "26"
  },
  {...},
  {...},
  ...
]

```

例三：就診後同日於同醫院因同疾病再次就診率

CSV: <https://data.gov.tw/dataset/18585>

指標名稱,年度季別,分區業務組,縣市別,醫事機構代碼,醫事機構名稱,院所指標值,所屬分區業務組指標值,全國指標值
就診後同日於同醫院因同疾病再次就診率,101Q4,臺北業務組,新北市新莊區,
0131060010,衛生福利部樂生療養院,0.0151,0.01060,0.00930
就診後同日於同醫院因同疾病再次就診率,102Q1,臺北業務組,新北市新莊區,
0131060010,衛生福利部樂生療養院,0.0147,0.01040,0.00920
就診後同日於同醫院因同疾病再次就診率,103Q1,臺北業務組,新北市新莊區,
0131060010,衛生福利部樂生療養院,0.0145,0.01030,0.00920
就診後同日於同醫院因同疾病再次就診率,100Q2,臺北業務組,新北市新莊區,
0131060010,衛生福利部樂生療養院,0.0152,0.00990,0.00870
就診後同日於同醫院因同疾病再次就診率,103Q2,臺北業務組,新北市新莊區,
0131060010,衛生福利部樂生療養院,0.0143,0.01060,0.00960
就診後同日於同醫院因同疾病再次就診率,100Q3,臺北業務組,新北市新莊區,

0131060010,衛生福利部樂生療養院,0.0169,0.01000,0.00880
就診後同日於同醫院因同疾病再次就診率,103Q3,臺北業務組,新北市新莊區,
0131060010,衛生福利部樂生療養院,0.0136,0.01040,0.00970
...

JSON:

```
[
  {
    "指標名稱": "就診後同日於同醫院因同疾病再次就診率",
    "年度季別": "105Q3",
    "分區業務組": "臺北業務組",
    "縣市別": "臺北市大同區",
    "醫事機構代碼": "0101090517",
    "醫事機構名稱": "臺北市立聯合醫院",
    "院所指標值": "0.0050",
    "所屬分區業務組指標值": "0.00970",
    "全國指標值": "0.00930"
  },
  {
    "指標名稱": "就診後同日於同醫院因同疾病再次就診率",
    "年度季別": "100Q2",
    "分區業務組": "臺北業務組",
    "縣市別": "臺北市大同區",
    "醫事機構代碼": "0101090517",
    "醫事機構名稱": "臺北市立聯合醫院",
    "院所指標值": "0.0048",
    "所屬分區業務組指標值": "0.00990",
    "全國指標值": "0.00870"
  },
  ...
]
```

2-3-3 JSON放在網路上的實際樣貌

書裡面方便你觀察的JSON格式

```
[
  {
    "SiteName": "二林",
    "County": "彰化縣",
    "AQI": "77",
    "Pollutant": "細懸浮微粒",
    "Status": "普通",
    "SO2": "3.7",
    "CO": "0.19",
    "CO_8hr": "0.2",
    "O3": "15",
    "O3_8hr": "27",
    "PM10": "64",
    "PM2.5": "37",
    "NO2": "8.5",
    "NOx": "8.8",
    "NO": "0.3",
    "WindSpeed": "1",
    "WindDirec": "171",
    "PublishTime": "2017-09-27 23:00",
    "PM10_AVG": "49",
    "PM2.5_AVG": "26"
  },
  {...},
  {...},
  ...
]
```

實際上的JSON格式

- 為何要排這麼密？因為檔案中連空白都佔空間，空白越少檔案就越小，傳輸就越快。
- 會不會沒空白和換行就出錯？JSON格式都是用{}和[]巢套包起來，所以沒換行不會錯。

```
[{"AQI": "78", "CO": "0.22", "CO_8hr": "0.2", "County": "基隆市",
  "NO": "1.1", "NO2": "4.7", "NOx": "5.8", "O3": "54", "O3_8hr": "42", "PM10": "37",
  "PM10_AVG": "31", "PM2.5": "28", "PM2.5_AVG": "26", "Pollutant": "細懸浮"}]
```

```

微粒", "PublishTime": "2017-10-03 16:00", "SiteName": "基隆
", "SO2": "1.4", "Status": "普通
", "WindDirec": "81", "WindSpeed": "2.7"}, {"AQI": "78", "CO": "0.24", "CO_8hr
": "0.3", "County": "新北市
", "NO": "1", "NO2": "11", "NOx": "12", "O3": "46", "O3_8hr": "40", "PM10": "40",
"PM10_AVG": "39", "PM2.5": "25", "PM2.5_AVG": "27", "Pollutant": "細懸浮微粒
", "PublishTime": "2017-10-03 16:00", "SiteName": "汐止
", "SO2": "2.6", "Status": "普通", "WindDirec": "36", "WindSpeed": "3"}, ,
...
]

```

2-3-4 自網路取得JSON並轉為R的資料型態

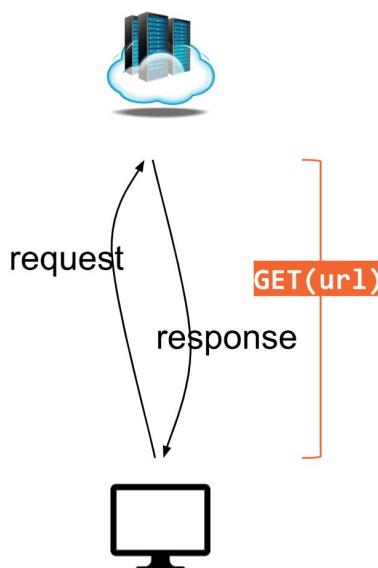
從網路上取得JSON檔案並剖析、轉為R的資料型態（`data.frame`或`list`）需要用到`httr`和`jsonlite`兩個套件，分別用以取回檔案和轉換文字檔案格式為R的物件（資料型態）。

須知JSON檔不像CSV強調用二維的表格來儲存資料，因此JSON檔編製的結果格式會十分不同。但可以知道的大原則是，`[]`中包含有`{}`就會被`fromJSON()`轉換為`data.frame`，而外層的`[]`或`{}`則會被轉換成有階層式名稱的`list`。

httr套件用以發送http request取得網頁檔案

最常用的是`GET()`與`POST()`函式，以及用`content(httprequest, "text")`把抓回來的數據解譯為單純的字串。

- 從網路上取資料共通的過程是，得把檔案給拿回來。拿回來的檔案，不管檔名是什麼，都是文字檔，看裡面的code就知道是CSV或JSON或html。
- 之後CSV用`read.csv()`讀，JSON用`jsonlite::fromJSON()`讀、html檔案則要用日後教的html parser來剖析。



這邊有個重要的觀念是「為何說是取得網頁檔案」？這必須要從拜訪網頁的實際操作來理解。所有的網頁，其實就是遠端伺服器上的「檔案」。而瀏覽器（如Chrome、Safari、Firefox等）實際上所做的事情是，用http request是依照網址向遠方伺服器要求取回檔案，當驗證是個合法存取後，伺服器便會將該檔案發送回本地電腦，而瀏覽器便按照這個檔案所寫的內容再向遠端伺服器要必須的檔案

(例如圖檔、CSS、JS）。都要回來後，瀏覽器便將這些檔案（主要是HTML、CSS、JS）組成你所看見的網頁。

所以，「**拜訪網頁**」這種好像到哪裡去遊歷的概念一整個是太過美好的想像，瀏覽器不過是把要的東西抓回本地電腦組成一個假想世界而已，這是多令人傷心的事。**上網說到底**只是機器在你電腦上重構了一個數據世界。你既沒有到哪裡去，實際上也沒人到你這裡來。你與他之間只是被天網交換訊息而已。

jsonlite套件：JSON轉為R的物件

jsonlite套件是把符合JSON格式規範的字串轉為R的物件，視JSON檔案內容結構而定，會轉為**data.frame**或**list**。正常的情形，一旦凡中括號[]包著數個{}裡面各有成對的key-to-value pairs，那**fromJSON()**這個函式就可以把他們轉為**data.frame**；但如果都是{}的話，就會被轉為**list**。

安裝jsonlite與httr套件

用下方的code來安裝第三方套件，如果你已經安裝過**tidyverse**，這兩個**httr**和**jsonlite**都有被**tidyverse**套件集合收錄，就不需要另行安裝，只需用**library()**加載之。

```
# pkgs <- c("jsonlite", "httr")
# pkgs <- pkgs[!(pkgs %in% installed.packages()[, "Package"])]
# if(length(pkgs)) install.packages(pkgs)
library(httr)
library(jsonlite)
options(stringsAsFactors = F)
```

2-3-5 狀況一：標準JSON編排，空氣品質指標資料

```
url <-
"http://opendata.epa.gov.tw/ws/Data/REWIQA/?$orderby=SiteName&$skip=0
&$top=1000&format=json"
df <- fromJSON(content(GET(url), "text", encoding = "utf-8"))
str(df)

'data.frame':   81 obs. of  23 variables:
 $ SiteName    : chr  "二林" "三重" "三義" "土城" ...
 $ County      : chr  "彰化縣" "新北市" "苗栗縣" "新北市" ...
```

```
$ AQI           : chr  "66" "47" "53" "33" ...
$ Pollutant    : chr  "細懸浮微粒" "" "細懸浮微粒" "" ...
$ Status        : chr  "普通" "良好" "普通" "良好" ...
$ SO2           : chr  "2.3" "2.4" "1.3" "1.5" ...
$ CO            : chr  "0.51" "1.28" "0.47" "0.52" ...
$ CO_8hr        : chr  "0.5" "1.6" "0.5" "0.7" ...
$ O3            : chr  "5" "-" "14" "5.1" ...
$ O3_8hr        : chr  "11" "" "17" "10" ...
$ PM10          : chr  "53" "29" "22" "20" ...
$ PM2.5         : chr  "16" "16" "14" "6" ...
$ NO2           : chr  "16" "39" "16" "20" ...
$ NOx           : chr  "18" "76" "19" "26" ...
$ NO             : chr  "2.3" "37" "3.8" "6" ...
$ WindSpeed     : chr  "0.9" "" "1" "0.8" ...
$ WindDirec     : chr  "130" "" "228" "126" ...
$ PublishTime: chr  "2019-03-25 01:00" "2019-03-25 01:00"
"2019-03-25 01:00" "2019-03-25 01:00" ...
$ PM2.5_AVG    : chr  "22" "15" "16" "10" ...
$ PM10_AVG      : chr  "48" "35" "29" "24" ...
$ SO2_AVG       : chr  "3" "3" "2" "1" ...
$ Longitude     : chr  "120.409653" "121.493806" "120.758833"
$ Latitude      : chr  "23.925175" "25.072611" "24.382942" "24.982528"
...
...
```

JSON格式轉R物件

`fromJSON(content(GET(url), "text", encoding = "utf-8"))`由內到外有三個函式：

步驟一：取得網路檔案

`httr::GET()`按照指定的url，向該伺服器發出`GET()` request把網頁檔案抓回來。若成功取得，他會回覆一個**HTML status code**

(<https://developer.mozilla.org/zh-TW/docs/Web/HTTP>Status>)。如果成功的話就是2開頭的數字例如**200 OK**代表該伺服器接受該請求並開始傳回檔案。

```
> response <- GET(url)
> class(response)
[1] "response"
```

```
> ??httr::GET
```

步驟二：轉出檔案內的文字

`httr::content(response, "text", encoding = "utf-8")`。可用`?content`查詢看看`content(response, "text")`的用途。其是把抓回來的檔案，轉為純文字的字串。`content()`是把抓回來的`response`解成純文字（JSON本身就是以純文字儲存，只是有約定熟成的格式，所有的工程師才知道要怎麼編織和讀取這種檔案）。

從上述回應的資料看他的`class`是一個`response`，但如果看Global Environment看來是個`list`，裡面裝載很多資料，而主要核心的內容在`content`這個欄位，但看來是用`binary code`裝起來的，而不是純文字。

因此，對於這個抓回來的檔案，我需要用`httr::content()`幫忙把純文字給解出來。經`help()`查詢可得知`content()`後面的參數有三類，其中可以要轉為純文字的就是`content(response, "text")`。因此偵測轉出來的變數會是長度為1的`character`。

```
> text <- content(response, "text", encoding = "utf-8")
> class(text)
[1] "character"
> ??httr::content
> length(text)
[1] 1
```

步驟三：將JSON格式文字轉為R物件

`jsonlite::fromJSON()`。因為我們用眼睛看就知道他是個JSON格式的檔案，所以用`fromJSON()`這個函式，把用JSON格式編成的字串轉為R的物件，有可能是`data.frame`或`list`。`fromJSON()`預期會把JSON中`[]`的每一個項目轉為一筆筆的資料，然後把`{}`的pair當成變項欄的變數名稱

`fromJSON()`將這個`character`轉為R的物件，也就是`data.frame`或`list`。注意，此時`text`是一個`character`。而我們會用`fromJSON()`轉，那是我們知道他是用JSON格式編寫的文字檔，就像我們知道`.csv`檔是用逗號分隔表示法一樣，JSON就是用巢套的`[{}, {}]`記號來表述資料。

並要提醒初學者，`.json`或`.csv`都只是幫助程式初步篩選檔案的副檔名罷了，裡面的究竟是不是個完整的json檔這都要去看、去測。我自然也可以在`.json`的檔案裡頭用逗號分隔模式撰寫。

```
df <- fromJSON(text)
?fromJSON
```

2-3-6 狀況二：多元、彈性但階層化的JSON資料：104求職網

第二類是最常會見到的例子，解出來的資料是個很多階層的list，通常一筆資料傳回來時多會附加一些metadata，比方說，一共幾筆資料、下一個資料區塊在哪裡，好讓使用者或者本地端的瀏覽器能夠繼續取得下一筆資料。因此，資料通常會在樹狀節點的某一個子節點。以下面的例子來說，就是存在res\$data\$list這個節點中。

```
url_104 <-
"https://www.104.com.tw/jobs/search/list?ro=0&keyword=%E8%B3%87%E6%96%
99%E5%88%86%E6%9E%90&area=6001001000&order=1&asc=0&kwop=7&page=2&mod
e=s&jobsource=n104bank1"

res <- fromJSON(content(GET(url_104), "text", encoding = "utf-8"))

df <- res$data$list
# head(df)

str(df)
'data.frame':   20 obs. of  34 variables:
 $ jobType      : chr  "2" "0" "2" "2" ...
 $ jobNo        : chr  "6362240" "10840395" "7776001" "10971798" ...
 $ jobName      : chr  "資深資料分析師." "【知名遊戲網路公司】資料分析人
員_工作環境優！(KAS_834)" "C.顧問類-顧問/資深顧問/經理(鑑識服務與資料分
析)" "S-資料分析師(2019校園徵才)" ...
 ...
```

Option: 取回資料並寫在硬碟

有時候寫爬蟲尤其是在爬會即時更新的資料時，會需要反覆定時地抓資料，這時候通常會先通通抓回來再慢慢合併整理。此時要特別注意如何保持每次抓回來的資料都是獨特的一個資料。以下面的例子來講，因為每次檔名都是一樣的，他會一直覆蓋過去，所以再怎麼抓，都不會是歷時性資料。通常會自動讀取當下時間當成檔名的一部分，這樣就不會重複了。這將在日後youbike的例子中用到。

```
response <- GET(url_104,
                 write_disk("data/url_104.json",
                            overwrite=TRUE))
```

2-3-7 狀況三：非典型的JSON資料結構：食品闢謠

來自衛福部的食品藥物管理署所開放的「食品闢謠」資料可能是個沒好好編過JSON的單位所編出來的案子。它資料內容應該很簡單，照道理來說，每一筆資料因為含有數個變項的值，因此每一筆資料會是一個{}裡面帶著多個資料變項與資料值的對應，（由於每一筆資料沒有特別再有一個對應的編號），這些{}會統一有序地存在一個[]中。但，食品闢謠的結構卻是一個[]裡面有329個data.frame，且每個data.frame只有對角線有資料，然後每一筆資料就一個data.frame，這是個資料編排的失誤。

附帶一提，類似的資料還有都用{{}, {}, ...}來存放資料的台北市Youbike資料，但台北市Youbike的資料編排並沒有失誤，只是最外框是{}，然後多了一個key作為站台編號指到每筆資料。

其JSON資料內容概況如下，黃色、綠色分別是第一、二筆資料，藍色是刪除資料值僅留下變項名稱的結果，最後一筆資料是該結構的示意圖。請仔細觀察其規律性。最外面是一個[]，每一筆資料為一個[]所包裹的資料，但每一筆資料的每個變項都個別是一個{}，而不是在一個{}指定有很多個變項。

```
[[{"分類": "醫療器材"}, {"標題": "坊間流傳電療器材可以治百病，是真的嗎？"}, {"內容": "解答： (1)目前食藥署針對電位治療器...(另開視窗)"}, {"附檔連結": ""}, {"發布日期": "07 29 2015"}], [{"分類": "食品"}, {"標題": "可樂會殺精，是真的嗎？"}, {"內容": "解答： (1) 網路上流傳「可樂可以殺精？」的謠言，...."}, {"附檔連結": ""}, {"發布日期": "07 31 2015"}], [{"分類": ""}, {"標題": ""}, {"內容": ""}, {"附檔連結": ""}, {"發布日期": ""}], [{"{}": {}}, {"{}": {}}, {"{}": {}}, {"{}": {}}], ...
]]
```

以程式嘗試讀取之。

```
url <- 'http://data.fda.gov.tw/cacheData/159_3.json'
safefood <- fromJSON(content(GET(url), 'text'))
str(safefood)
class(safefood)
```

```
[1] "list"
class(safefood[[1]])
[1] "data.frame"
View(safefood[[1]])
```

資料在「Environment Panel」看起來長得像這樣子。

Data	
safefood	Large list (363 elements, 963.4 Kb)
:'data.frame': 5 obs. of 5 variables:	
..\$ 分類 : chr [1:5] "醫療器材" NA NA NA ...	
..\$ 標題 : chr [1:5] NA "坊間流傳電療器材可以治百病，是真的嗎？" NA NA ...	
..\$ 內容 : chr [1:5] NA NA "解答：(1)目前食藥署針對電位治療器核准之適應症範圍僅有「..."	
..\$ 附檔連結: chr [1:5] NA NA NA "" ...	
..\$ 發布日期: chr [1:5] NA NA NA NA ...	
:'data.frame': 5 obs. of 5 variables:	
..\$ 分類 : chr [1:5] "食品" NA NA NA ...	
..\$ 標題 : chr [1:5] NA "可樂會殺精，是真的嗎？" NA NA ...	
..\$ 內容 : chr [1:5] NA NA "解答：(1) 網路上流傳「可樂可以殺精？」的謠言，經查此則..."	
..\$ 附檔連結: chr [1:5] NA NA NA "" ...	
..\$ 發布日期: chr [1:5] NA NA NA NA ...	

View(safefood[[1]])每一筆資料看起來像是這個樣子：

	分類	標題	內容	附檔連結	發布日期
1	醫療器材	NA	NA	NA	NA
2	NA	坊間流傳電療器材可以治百病，是真的嗎？	NA	NA	NA
3	NA	NA	解答：(1)目前食藥署針對電位治療器核准之適應症範圍僅...	NA	NA
4	NA	NA	NA		NA
5	NA	NA	NA	NA	07 29 2015

處理非典型的JSON檔

但這時候也不難觀察到其規律性。既然每個data.frame是一筆資料，且資料都是照順序出現在對角線，那我就把data.frame給unlist()拆成vector後，把NA給移除了，那剩下的就是我們要的資料了。

但，由於對整筆資料`unlist()`，那整筆資料會變成一個很長的`vector`，不過我們知道每五個元素就是一筆資料。所以我可以嘗試用`matrix()`的指令，讓資料每五個就折成一筆資料。

程序大致上是

1. `safefood.v <- unlist(safefood)` 把資料`unlist()`為`vector`。
2. `safefood.v <- safefood.v[!is.na(safefood.v)]` 剔除NA值
3. `safefood.m <- matrix(safefood.v, byrow = T, ncol = 5)` 照列來折，因為每五個就一筆資料，所以是照列折，然後用`ncol = 5`來指定五個一折。

```
> safefood.v <- unlist(safefood)
> head(safefood.v)
    分類1      分類2      分類3      分類4      分類5      標題1
"醫療器材"        NA        NA        NA        NA        NA

> anyNA(safefood.v) # anyNA() to check if NAs still exist
[1] TRUE

> sum(is.na(safefood.v)) # check if NAs exist
[1] 7240

> safefood.v <- safefood.v[!is.na(safefood.v)] # remove NAs
> anyNA(safefood.v) # double-check NAs
[1] FALSE

# convert vector to matrix
> safefood.m <- matrix(safefood.v, byrow = T, ncol = 5)
> class(safefood.m)
```

轉為`matrix`後，已經跟`data.frame`的資料樣貌夠像了，所以再把它轉為`data.frame`，並作後續的資料清理和變項選取。

```
# convert matrix to dataframe
safefood.df <- as.data.frame(safefood.m)

# delete the 4th column
safefood.df <- safefood.df[-4]
```

```
# rename the data.frame
names(safefood.df) <- c('category', 'question', 'answer',
'timestamp')
```

各個JSON案例

以下的網站均以JSON作為連至後端資料庫後，透過前端AJAX語法寫回網頁的範例。

```
url_AQI <-
"http://opendata.epa.gov.tw/ws/Data/REWIQA/?$orderby=SiteName&$skip=0&$top=1000&format=json"
url_foodRumor <- "http://data.fda.gov.tw/cacheData/159_3.json"
url_ubike <- "http://data.taipei/youbike"
url_rent591 <-
"https://rent.591.com.tw/home/search/rsList?is_new_list=1&type=1&kind=2&searchtype=1&region=1"
url_dcard <- "https://www.dcard.tw/_api/forums/girl/posts?popular=true"
url_cht <-
"https://www.googleapis.com/customsearch/v1element?key=AIzaSyCVAXiUzRYsML1Pv6RwSG1g
unmMikTzQqY&rsz=1&num=20&hl=zh_TW&prettyPrint=false&source=gcsc&gss=.com&sig=0c3990
ce7a056ed50667fe0c3873c9b6&cx=013510920051559618976:klssxyhsnf7g&q=%E9%85%92%E9%A7%9
5&lr=&filter=1&sort=&googlehost=www.google.com&callback=google.search.Search.apiary
7677&nocache=1481218832065"
url_pchome <-
"http://ecshweb.pchome.com.tw/search/v3.3/all/results?q=X100F&page=1&sort=rnk/dc"
url_udn <- "https://video.udn.com/realtimedata/general"
url_104 <-
"https://www.104.com.tw/jobs/search/list?ro=0&keyword=%E8%B3%87%E6%96%99%E5%88%86%E
6%9E%90&area=6001001000&order=1&asc=0&kwop=7&page=2&mode=s&jobsource=n104bank1"
res <- fromJSON(content(GET(url), "text"))
```

2-4 讀取XML檔

(待補) 相較於HTML檔，XML為比較規則的資料，所以有好用的函式庫可以迅速轉換資料，而不用像HTML要設計剖析器來剖析之。

2-X 讀取資料庫

```

library(RSQLite)
library(tidyverse)
library(stringr)
library(rjson)
options(stringsAsFactors = F)

fnames <- list.files("data/", full.names = T)
filter_in <- c('dropbox', 'dictionary', 'facebook', 'mail', 'youtube',
'toasty', 'comic', 'dm5', 'mobile01')

myfetchData <- function(dbname){
  conn = dbConnect(RSQLite::SQLite(),
    dbname=dbname)

  # for mac
  # res <- dbSendQuery(conn, "SELECT urls.url, visits.visit_time,
  visits.visit_duration, visits.id, visits.from_visit, visits.transition,
  visits.segment_id, urls.id FROM visits INNER JOIN urls ON
  visits.url=urls.id;")

  # for win
  res <- dbSendQuery(conn, "SELECT urls.url,
    CAST(visits.visit_time as TEXT) as visit_time,
    CAST(visits.visit_duration as TEXT) as visit_duration,
    visits.id, visits.from_visit, visits.transition,
  visits.segment_id, urls.id
    FROM visits INNER JOIN urls ON visits.url=urls.id;")

  data <- dbFetch(res, n=-1)
  data
}

data <- myfetchData(fnames[1])
str(data)

```

SQL query list and review

在進行查詢時轉換資料的型態

```
res <- dbSendQuery(conn, "SELECT urls.url,
                           CAST(visits.visit_time as TEXT) as visit_time,
                           CAST(visits.visit_duration as TEXT) as visit_duration,
                           visits.id, visits.from_visit, visits.transition,
```

2-X 讀取一般文字檔

```
fout <- file("html/unlabel_comments_tsai_byFans.html")
html <- readChar("html/template.html",
file.info("html/template.html")$size)
res <- sprintf(html, all.str)
writeLines(res, fout)
close(fout)
```

2-X 寫入檔案

write_disk() 將GET()的檔案直接寫到Local端

```
url <- "http://data.taipei/youbike"
GET(url, write_disk("../data/ubikeSample.json", overwrite=TRUE))
```

將xml_document寫為HTML

```
doc <- read_html(links[3])
write_html(doc, "test.html")
system("open test.html")
```

write() 將GET()或POST()結果在尚未讀成xml_document前寫為檔案

相當於把字串直接寫為HTML，只要他是字串即可，所以說不定也可以用下述writeLines()的方法

```
res <- POST(url, body = form)
doc.str <- content(res, "text") # convert result to string
write(doc.str, "test.html") # for testing
file.show("test.html") # for testing
system("open test.html")
```

writeLines() 將資料寫為文字檔

```
fout <- file("html/unlabel_comments_tsai_byFans.html")
html <- readChar("html/template.html", file.info("html/template.html")$size)
res <- sprintf(html, all.str)
writeLines(res, fout)
close(fout)
```

2-A Practices & Assignments

Practice 2-3-1 用JSON表示資料

下列為一個假的CSV資料，嘗試用JSON表示之。

```
name, height, weight
R2-D2, 100, 400
C-3PO, 180, 500
```

Practice 2-3-2 查找並抓取網路上的JSON檔

下列這些網路文件應該都是json檔，嘗試把他抓回來看看。

```
url_rent591 <-
"https://rent.591.com.tw/home/search/rsList?is_new_list=1&type=1&kind
=2&searchtype=1&region=1"
url_reHospital <-
"http://data.nhi.gov.tw/Datasets/DatasetResource.ashx?rId=A21030000I-
E30008-002&ndctype=JSON&ndchnid=18585"
url_dcard <-
"https://www.dcard.tw/_api/forums/girl/posts?popular=true"
url_pchome <-
"http://ecshweb.pchome.com.tw/search/v3.3/all/results?q=X100F&page=1&
sort=rnk/dc"
url_104 <-
"https://www.104.com.tw/jobs/search/list?ro=0&keyword=%E8%B3%87%E6%96
%99%E5%88%86%E6%9E%90&area=6001001000&order=1&asc=0&kwop=7&page=2&mod
e=s&jobsource=n104bank1"
url_ubike <- "http://data.taipei/youbike"
```

Practice 2-3-3 抓取並轉譯JSON文件為data.frame

請嘗試獲取多筆（至少一筆）政府開放資料的連結，其資料格式為JSON檔。用`httr`套件直接獲取該檔案，並接續用`jsonlite`解析為R的檔案格式。當完成後，請用程式列印出該資料的主要特徵（例如用`str()`, `dim()`, `head()`等），並用View觀察你所抓下來剖析完成的資料，將其用螢幕擷取程式複製該畫面儲存為圖檔，一併壓縮繳交至課程平台。

2-B 本章節有用的系統指令

執行command line commands

```
system("ls ../../dataset/pxmartchannel")
system("open test.html")
```

列出資料夾底下所有的檔案

59

```
files <- list.files("../.../dataset/pxmartchannel", full.names = T)
```

第三章、爬蟲設計

在教各位撰寫網頁爬蟲前，必須提醒各位，不當的爬蟲可能會觸犯法律。通常觸犯法律會是以下幾項要件：

- 該資訊服務為了避免第三方使用者爬取他的資料，所以設計了一些檢核機制。若第三方使用者刻意用不同方法繞過這些檢核機制，即有觸法的可能。例如偽造IP等。
- 因為大量爬取資訊，從每小時、每日、每週、每月的流量尺度來算產生大量的request（比方說1%都是來自於同一個使用者），那就會有觸法的嫌疑。主訴是因為這樣的大量操作會影響服務提供方的服務品質。

3-1 使用Chrome DevTools檢查網頁頁面操作

在2-3節讀取JSON檔的內容中，我讀取了數個網址的資料，並告知各位這些網頁背後的運作都是JSON的檔案格式。那接下來的問題是，這些網址似乎都不是一般人會知道的網址，我究竟為何會知道，資料就存放在這個地方？

```
url_dcard <-
"https://www.dcard.tw/_api/forums/girl/posts?popular=true"
url_pchome <-
"http://ecshweb.pchome.com.tw/search/v3.3/all/results?q=X100F&page=1&
sort=rank/dc"
url_104 <-
"https://www.104.com.tw/jobs/search/list?ro=0&keyword=%E8%B3%87%E6%96%
99%E5%88%86%E6%9E%90&area=6001001000&order=1&asc=0&kwop=7&page=2&mod
e=s&jobsource=n104bank1"
url_ubike <- "http://data.taipei/youbike"
```

Chrome DevTools

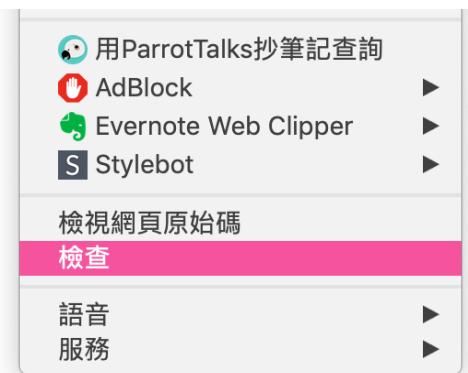
為了找到該網址，我們通常會用各瀏覽器的開發者工具來檢查、檢視這些網頁如何運作（FireFox、Safari都有類似功能，甚至會有不同的瀏覽器產生的結果不同）。以Chrome而言，要使用Chrome DevTools，可查看網頁說明

[https://developers.google.com/web/tools/chrome-devtools/。](https://developers.google.com/web/tools/chrome-devtools/)

以下將以104找工作網站來展示，爬取於104網站搜尋「爬蟲」的結果，並介紹如何找到所需的JSON資料連結。

步驟一：選檢查開啟Chrome DevTools

首先開啟該頁面（我選擇<https://www.104.com.tw/jobs/main/>），找空白處按滑鼠右鍵選擇「檢查」。「檢查」後的彈出分頁便是Chrome DevTools。



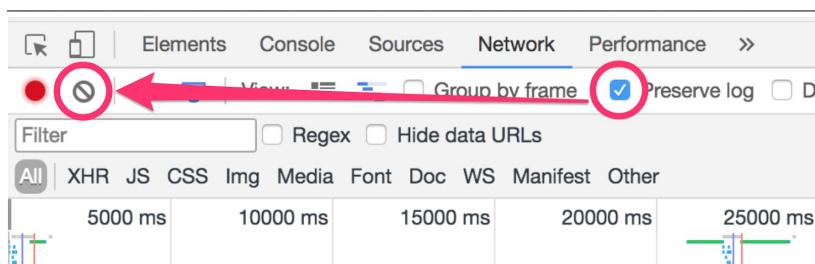
Chrome DevTools上有數個Tabs，第一個是Elements，通常是前端工程師最常用的，用以檢查頁面上的元素、CSS和XPath；Console通常是偵測用的，可以檢查JavaScript是否有效執行，不然JavaScript本身並沒有工具可以告訴其撰寫了錯誤的代碼。Source通常會顯示目前頁面所載入的HTML或CSS檔案；最後Network是本單元最需要的功能，可檢視在瀏覽器輸入URL，按下Enter後，瀏覽器抓了哪些檔案，做了哪些事。



步驟二：選Network Panel進行前置步驟

打開Network Panel後約有以下三個步驟，請嘗試跟著做，避免錯誤。

- 先勾取Preserve Log的選項，代表中間若有一些作為參數傳遞用的檔案，勾選Preserve Log意味著要把那些過程留下來，便於觀察。
- 第二個步驟是點選清除目前紀錄的功能，為一個禁止符號。
- 第三個步驟是按下該頁重新載入的功能，或者把游標放到網址列處再按Enter，即可以重新載入該頁面。



步驟三：載入頁面並觀察

確定已經輸入「爬蟲」作為關鍵字後，便可照上述步驟，進行搜尋。此時，應該可以看見不少圖檔、.js、.css或者各式各樣的檔案跑出來。

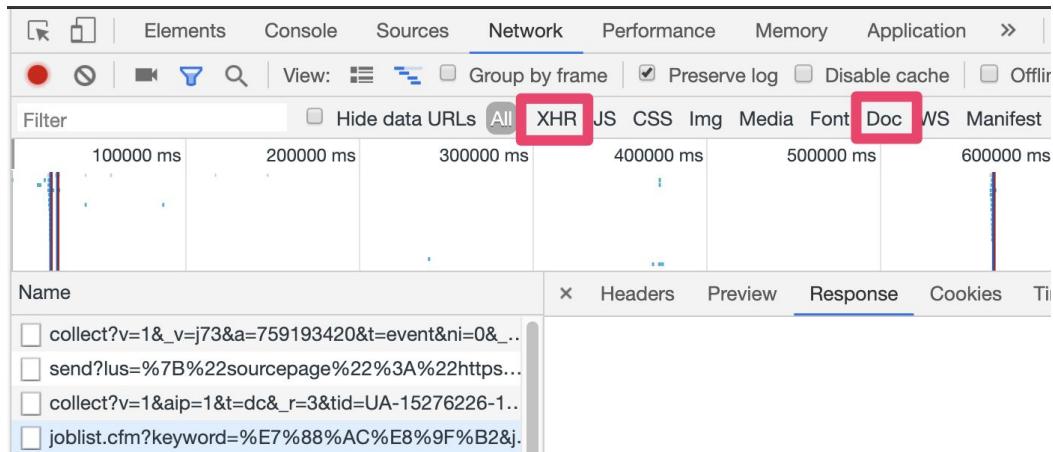
The screenshot shows a job search interface with a search bar for '爬蟲'. Below the search bar are filters for location ('台中市中區'), experience ('2年以上'), and education ('專科'). The results list two job posts:

- 3/25 爬蟲工程師** at **謝輝廣告有限公司** | 廣告行銷公關業
台北市信義區 | 2年以上 | 大學
1. 負責傳統網頁包含影片/文字/圖片等各種網站訊息高效採集與正確解析 2. 總結分析不同網站，網頁的結構特點及規律，負責爬蟲架構設計與研發，獨立完成爬蟲核心算法和策略優點 3. 設計策略和算法提升網頁抓取的效率和質量，進一步解決系統網頁的重待遇面議
- 3/27 爬蟲工程師** at **創視科技有限公司** | 電腦軟體服務業
台中市西屯區 | 經歷不拘 | 學歷不拘
1.以php、javascript開發 2.負責傳統網頁包含影片/文字/圖片等各種網站訊息高效採集與正確解析 3.總結分析不同網站，網頁的結構特點及規律，負責爬蟲架構設計與研發，獨立完成爬蟲核心算法和策略優點 4.具備相關網頁知識以協助
月薪 40,000~60,000元

To the right, the Chrome DevTools Network panel is displayed. The 'Network' tab is selected. The 'JS' tab is highlighted with a red box around the 'conversion_async.js' entry. Other visible entries include various image files (1_42117.jpg, 1_42313.jpg, 15_14387.gif, 15_14401.jpg) and other JavaScript files.

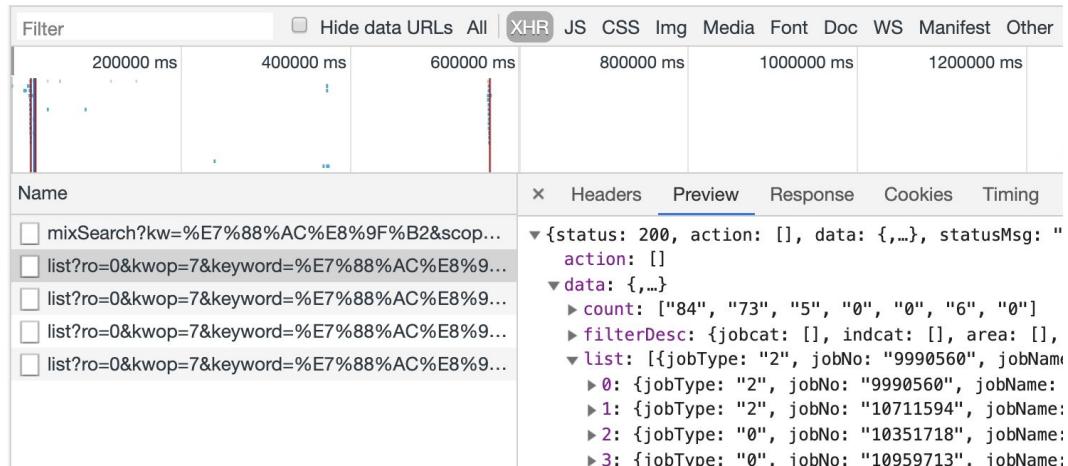
步驟四：找尋所需的JSON檔案

此時，Chrome DevTools的Network Panel中看起來應該如圖。不斷刷新的就是一個又一個的檔案。但這些檔案事實上是有分類的，All自然指的是全部的檔案，CSS和JS是網頁前端和前後端用的，Img和Media就是該頁面上的圖檔和影像檔。而我們所需要的通常是在XHR或者Doc中（其中如果是JSON的話尤其更常見於XHR中）。



此時，從XHR和Doc Tab應該找不到JSON的資料檔（但其實我知道他是一個JSON檔），因此，我就找看看第二頁、第三頁的資料在哪裡。要找第二頁和第三頁的資料，就要把滑鼠往下滾，要不繼續往下滾讓他觸發下一頁的資料，要不就是點選第二頁、第三頁的超鏈結，來找尋固定的規則。而104是屬於第一種方法，一直往下滾，他就會把新的資料傳回來。

一直往下滾的過程會發現XHR跳出好幾個回傳的JSON檔案。



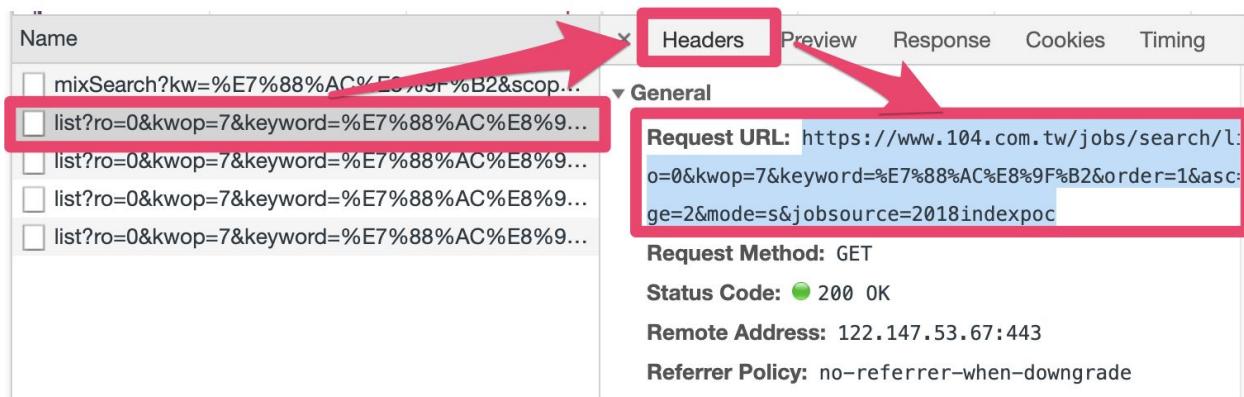
步驟五：複製JSON檔案連結。

點選該網頁後，不要停留在Preview Tab，點選Headers Tab可以看出你自己的網頁是發出了什麼Request給伺服器一端。Headers裡面所記錄的就是當發出一個Request時，伺服器端回應了什麼Response。Request中可以跟著發出去的有：

1. **Referer:** 進到這一頁前，是從哪一頁跳過來的。
2. **User-Agent:** Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.86 Safari/537.36。User-Agent為該支手機用以上網的瀏覽器型號。

複製下來的網址如下：

```
https://www.104.com.tw/jobs/search/list?ro=0&kwop=7&keyword=%E7%88%AC%E8%9F%B2&order=1&asc=0&page=2&mode=s&jobsouce=2018indexpoc
```



3-3 逐頁爬取資料-104人力銀行

前節介紹了如何用Chrome DevTools找到JSON的資料連結。但這通常是某一分頁內的資料，如果要爬取所有的資料的話，必須要知道如何依照搜尋結果的分頁來逐頁獲取資料並整合在一起。通常，每一個頁面的資料，大多是跟隨著使用者的操作載入，例如下拉或者是點選下一頁。而不同的頁面就會有不同的網址，但多半這些網址是有規律性的，所以要做的事情便是**觀察網址的規律性**。觀察規律性的兩大要點：

1. **第二頁在哪裡？**你就拉到第二頁看看，看會跳出什麼新資料，就知道第二頁的URL為何和規律性為何。
2. **最後一頁在哪裡？**一般來說，如果是用JSON來儲存資料，通常最後一頁的頁碼可能會有三種情形：
 - a. 直接寫在url上

- b. 在JSON檔的內容中會有該欄位（例如104便是、鉅亨網也是）
- c. 每一頁都存著下一頁的網址，取到不能取為止（新手感到最麻煩的方法）

3-3-1 載入讀取JSON所需函式庫

以下以104人力銀行的網頁，打撈「爬蟲」作為關鍵字的結果。

```
library(httr)
library(jsonlite)
options(stringsAsFactors = F)
```

3-3-2 取得第一個頁面的JSON資料

104的案例必須要先載入第二頁的資料才會知道第一頁的網址在哪裡

Name	Status	Type	Initiator	Size	T...	Waterfall
mixSearch?kw=%E7%88%...	200	xhr	jquery.min.j...	288 B	5...	
list?ro=0&kwop=7&keyword...	200	xhr	jquery.min.j...	9.4 KB	1...	
list?ro=0&kwop=7&keyword...	200	xhr	jquery.min.j...	10.4 ...	2...	
list?ro=0&kwop=7&keyword...	200	xhr	jquery.min.j...	8.6 KB	2...	

```
url2 <-
"https://www.104.com.tw/jobs/search/list?ro=0&kwop=7&keyword=%E7%88%A
%C8%9F%B2&order=1&asc=0&page=2&mode=s&jobsource=2018indexpoc"

url3 <-
"https://www.104.com.tw/jobs/search/list?ro=0&kwop=7&keyword=%E7%88%A
%C8%9F%B2&order=1&asc=0&page=3&mode=s&jobsource=2018indexpoc"

result2 <- fromJSON(content(GET(url2), "text"))
df2 <- result2$data$list
```

	jobType	jobNo	jobName	jobNameSnippet	jobRole	jobRo
1	0	10778115	python程式開發工程師(台北)	python程式開發工程師(台北)	1	1
2	0	10778116	python程式開發工程師(新竹)	python程式開發工程師(新竹)	1	1
3	2	10775838	數據分析師(新媒體事業部/產品發展中心/平台開發組)	數據分析師(新媒體事業部/產品發展中心/平台開發組)	1	1
4	0	10188442	.NET全端工程師	.NET全端工程師	1	1
5	0	10959713	資深雲端服務開發工程師	資深雲端服務開發工程師	1	1
6	0	10351718	數據應用處-大數據分析副理	數據應用處-大數據分析副理	1	1

看起來這網址的規律性是由page=後面的數字來決定，因此偷改一下原本的page=2為page=1看看結果，確實也抓得到資料。

```
url1 <-
"https://www.104.com.tw/jobs/search/list?ro=0&kwop=7&keyword=%E7%88%A
%C8%9F%B2&order=1&asc=0&page=1&mode=s&jobsource=2018indexpoc"
result1 <- fromJSON(content(GET(url1), "text"))
df1 <- result1$data$list
```

3-3-3 合併兩個頁面的資料

但是以下程式碼卻會出錯。

```
all.df <- bind_rows(df1, df2) # will raise error
# Error in bind_rows_(x, .id) :
#   Argument 31 can't be a list containing data frames
```

出錯的原因在於，`bind_rows()`或者`rbind()`等可以把一個`data.frame`後面接上另一個`data.frame`的函式對於資料格式有很嚴格的要求，避免合併錯誤。最起碼的規定便是兩個`data.frame`的變數必須相同，但是`bind_rows()`對於變數不同可以自行產生新的變數，並填為`NA`，所以會比`rbind()`好用。然而，如果有其中一個變數，其資料型態仍是階層性的`list`或者`data.frame`的話，那就會無法使用`bind_rows()`，因此必須要把它扁平化（我把它稱為`flatten`）。

3-3-4 「扁平化」

扁平化有三種做法或典型案例：

1. 丟掉 (Drop) 仍有階層的變項丟掉或留下沒有階層的變項：如果沒階層的變項很多，有階層的變項很少的話，那就把有階層的變項丟了；如果沒階層的變項其實不多，就直接選擇所要的變項會比較快。
2. 階層中仍有資料需要保留：那就要設法把資料給複製出來給予新的變項名稱，然後丟掉原本仍有階層的變項。
3. 使用轉換資料型態的函式直接把整個階層化的變項拆掉（最麻煩也最常出錯）。

如果把NULL指派給某一個data.frame的變項的話，就可以把該變項丟掉，記得前後兩個data.frame都要丟掉。

```
df1$link <- NULL
df1$tags <- NULL
df2$link <- NULL
df2$link <- NULL
all.df <- bind_rows(df1, df2)
```

更好的方法是善用dplyr套件的select()。

```
df1 <- result1$data$list
df2 <- result2$data$list
df1 <- select(df1, -tags, -link)
df2 <- select(df2, -tags, -link)
all.df <- bind_rows(df1, df2)
```

3-3-5 找到最後一頁的頁碼

亦即找到最後一個資料區段的網址，這個104的案例其實可以在每個回傳的JSON內容中找到。

```
last_page_num <- result1$data$totalPage
```

3-3-6 黏貼頁碼組成新的url並測試

測試看看找到的頁碼是否確實能夠抓到資料。這邊使用paste0(string1, var1, string2)的方式來把頁碼貼入原本的url內容中。確實能夠抓到資料。

字串相黏也經常會使用sprintf()挖空填變項的方法，但這個方法在這個案例中行不同，因為中文字有很多類似sprintf()會用到的控制符號如%E等

```
url.last_page <-
paste0("https://www.104.com.tw/jobs/search/list?ro=0&kwop=7&keyword=%
E7%88%AC%E8%9F%B2&order=1&asc=0&page=", last_page_num,
"&mode=s&jobsouce=2018indexpoc")
result.last_page <- fromJSON(content(GET(url.last_page), "text"))
```

3-5-7 用for迴圈抓回所有資料

此時我僅把頁碼列印出來，和每一個抓回來的`data.frame`的資料筆數列印出來，我並沒有合併抓回來的資料。

```
for(p in 2:last_page_num){
  url <-
  paste0("https://www.104.com.tw/jobs/search/list?ro=0&kwop=7&keyword=%
E7%88%AC%E8%9F%B2&order=1&asc=0&page=", p,
"&mode=s&jobsouce=2018indexpoc")
  result <- fromJSON(content(GET(url), "text"))
  temp.df <- select(result$data$list)
  print(paste(p, nrow(temp.df)))
}
```

3-5-8 合併抓回來的資料

```
url1 <-
"https://www.104.com.tw/jobs/search/list?ro=0&kwop=7&keyword=%E7%88%AC%E8%
F%B2&order=1&asc=0&page=1&mode=s&jobsouce=2018indexpoc"
result1 <- fromJSON(content(GET(url1), "text"))
last_page_num <- result1$data$totalPage
all.df <- select(result1$data$list, -link, -tags)

for(p in 2:last_page_num){
  url <-
  paste0("https://www.104.com.tw/jobs/search/list?ro=0&kwop=7&keyword=%E7%88%
AC%E8%9F%B2&order=1&asc=0&page=", p, "&mode=s&jobsouce=2018indexpoc")
```

```

result <- fromJSON(content(GET(url), "text"))
temp.df <- select(result$data$list)
all.df <- bind_rows(all.df, temp.df)
print(paste(p, nrow(all.df)))
}

```

3-4 剖析HTML檔案

目前本章已介紹了讀取XLSX、CSV、JSON檔等方法，已經可以讀取幾乎所有的開放資料，也可讀取104、信義房屋、Dcard、facebook、Google Map API、Flickr API、Twitter Rest API等資料。但仍有些網頁是直接由伺服器端傳回整個網頁而非資料檔，例如**PTT網頁版、不動產實價登錄網站、政府標案決標資訊**等等。類似這樣的網頁，我們除了要爬取HTML檔案外，尚須撰寫一個HTML剖析器來獲得其中的資料。絕大部分的網頁都需要用多個<div>或者<table>或等規劃版面以擺放導覽元件、廣告、標題、分類等等各種資訊，資料通常僅佔其中的一小部分，因此我們必須要撰寫HTML剖析器，找到目標的HTML標籤，將之爬取回來。

這類的網站如下，可看出HTML的標籤層層巢套不一，甚至可以為了防止被抓取而動態更改資料巢套的階層，那要如何從這些網站中取得資料呢？雖然說看似複雜，但是也不難看出個規律性，例如新聞搜尋的結果似乎就有標題、簡要內文、時間與圖片。通常，我們的瀏覽器在發出搜尋的要求後，其傳回來的也是一個HTML檔案，也會傳回一些CSS或Javascript來告訴瀏覽器要怎麼視覺化這個HTML檔。所以，這個HTML和CSS是在傳回你的瀏覽器後視覺成該模樣，如果看起來是有規律性的，那意味著必定有一套規律性是設計來讓程式自己知道要怎麼視覺化這些標題或內容，好讓你看起來有規則。而這套規則，簡單地說就是由HTML標籤與屬性所構成的規則，讓我們得以用CSS去選取相同規則的元件，將之視覺化為相同的樣子。

因此，這裡我們除了要弄懂HTML和CSS外，也要介紹CSS Selector和XPath這兩種可以選取HTML元素的方法，**CSS Selector和XPath是用來定義一個路徑，選取HTML的一個或者多個條件相同的元素。**

批踢踢實業坊 > 看板 Gossiping

聯絡資訊 關於我們

看板 精華區

最舊 (上頁) 下頁 最新

搜尋文章…

4 [新聞] 檢舉食安高市府拒發獎金 法院判他應得216
qqq5566 4/07 ...

[問卦] 小英現在到底是黑還是白？
genie529 4/07 ...

[問卦] 今年的大甲媽
bec0396 4/07 ...

Re: [問卦] 宜蘭人是不是覺得好險沒蓋六輕？
OhwadaAkira 4/07 ...

農產品交易行情 <https://amis.afa.gov.tw>

產品	上價	中價	下價	平均價 (元/公斤)	跟前 交易 比
小計	90.0	90.0	90.0	32.6	-
FAO 其他花類	90.0	90.0	90.0	90.0	-
FA1 黃秋葵	113.2	63.0	30.9	66.6	-
FB11 花都葵 青梗 留梗柄	33.8	22.4	14.3	23.1	-
FC1 胡瓜 黑刺	28.7	18.5	12.9	19.4	-
FDO 花胡瓜 其他	42.7	31.8	20.1	31.6	-
FD1 花胡瓜	41.2	29.6	17.8	29.6	-
FE1 冬瓜 白皮	13.4	9.1	6.3	9.4	+
FF0 絲瓜 其他	37.0	21.4	9.3	22.1	-
FF1 絲瓜	26.0	17.1	8.9	17.2	-

Network tab of the browser developer tools showing network requests for the page.

政府招標決標資訊下載（XML）：

<http://web.pcc.gov.tw/tpsreport/transfer/dataTransfer.do?method=getOkfnOpenDataXml>

政府招標決標資訊查詢：

<https://web.pcc.gov.tw/tps/pss/tender.do?method=goSearch&searchMode=common&searchType=advance&searchTarget=ATM>

項 次	種類	機關 名稱	標案案號	標案名稱	招標公 告日期	決標或 無法 決標公 告	截止投 標日期	公開開 覽/徵求 起迄日期	預 告 公 告 日 期
1	招標	臺北 市政 府環 境保 護局	106E008	竹子湖地區公共廁所 工程	106/11/16	公告	106/11/27		
2	招標	臺北 市政 府環 境保 護局	L10629	鍋爐過熱器及下端鍋 爐飛灰螺旋輸送機清 木柵 理 垃圾 焚化 廠	106/11/16	公告	106/11/21		
3	招標	臺北 市政 府農 委會	10721	本局107年度商品提貨	106/11/16	八生	106/11/27		

3-4-1 HTML的結構

HTML檔案的結構大致如下：

- 首先會有一個檔案類別的宣告`<!DOCTYPE html>`，用以告訴第三方瀏覽器或應用程式說這是一個HTML5檔案；
- 再來是成對標籤所組成的巢套結構，下例即有一對`<html></html>`包著一對`<head></head>`和一對`<body></body>`。
- 另外`<!-->`包著的內容為註解，瀏覽器或程式遇到該區段的內容會略過不處理。

```

註解Comments -----><!--This is an html 5 document-->
Document Type declaration -----><!DOCTYPE html>
                                         <html>
                                         <head>
                                         <title>your website title</title>
                                         <!-- put your css and js here -->
                                         </head>
                                         <body>
                                         <!--put your main content here-->
                                         </body>
                                         </html>

```

Hierarchical paired tags

下圖可用以說明HTML檔案的巢套（一層包一層）結構（圖片來源
https://www.w3schools.com/html/html_intro.asp）。

```

<html>
  <head>
    <title>Page title</title>
  </head>

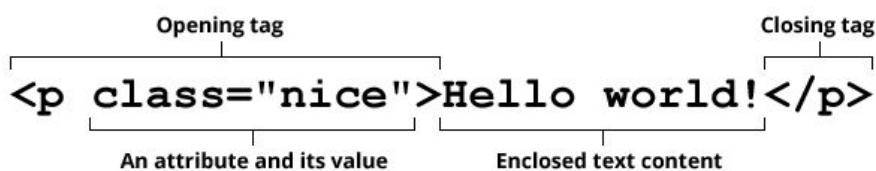
  <body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
    <p>This is another paragraph.</p>
  </body>
</html>

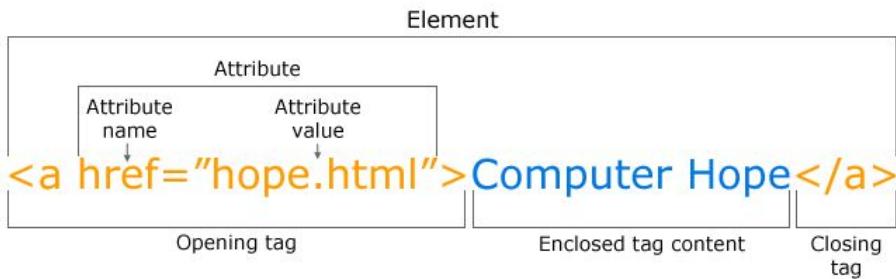
```

元素與屬性 Elements and Attributes

每一對標籤，我們把它稱為一個Element（HTML元素）。下二圖為兩個元素（elements），一對「標籤（tags）」前後包夾稱為一個「元素」，在Opening tag中除了標籤名外的其他資訊為「屬性（Attribute）」，一個屬性包含屬性名稱（Attribute name）和屬性值（Attribute value），屬性值多為文字，所以需要用雙引號括起來。

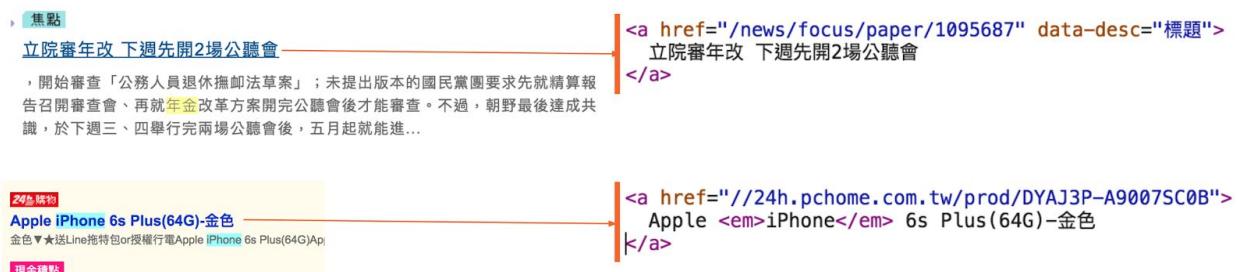
Anatomy of an HTML element





範例 Hyperlink <a>

例如：以下自由時報新聞關鍵字查詢的結果以及pchome24h購物中心產品查詢的結果。可見到<a>為一對elements、而網址則以attribute value指給href這個既定名稱的<a>的屬性。



HTML中的資料在哪裡？

那我們要的資料通常會怎麼放在HTML上呢？

- <p> 通常如果是本文多會是用<p></p>, <p>的意義為paragraph；
- <h1>如果是要取出新聞或文章標題的話，通常會是<h1>、<h2>等headings（標題）；
- 若回傳的搜尋結果的話，多半會是用內包著表示，為unordered list（無序列表）、為list item（列表中的項目）
- <table>若是查詢資料結果為網頁「表格」多半用<table></table>包裹，裡面會有<th></th>、<tr></tr>、<td></td>等，分別為table header、table row、與table data。
- <div>網頁中巢套的最厲害的是<div></div>為division的縮寫，也就是網頁的區塊，通常被用來區隔頁面中的區塊，是進行排版和視覺化的好幫手。

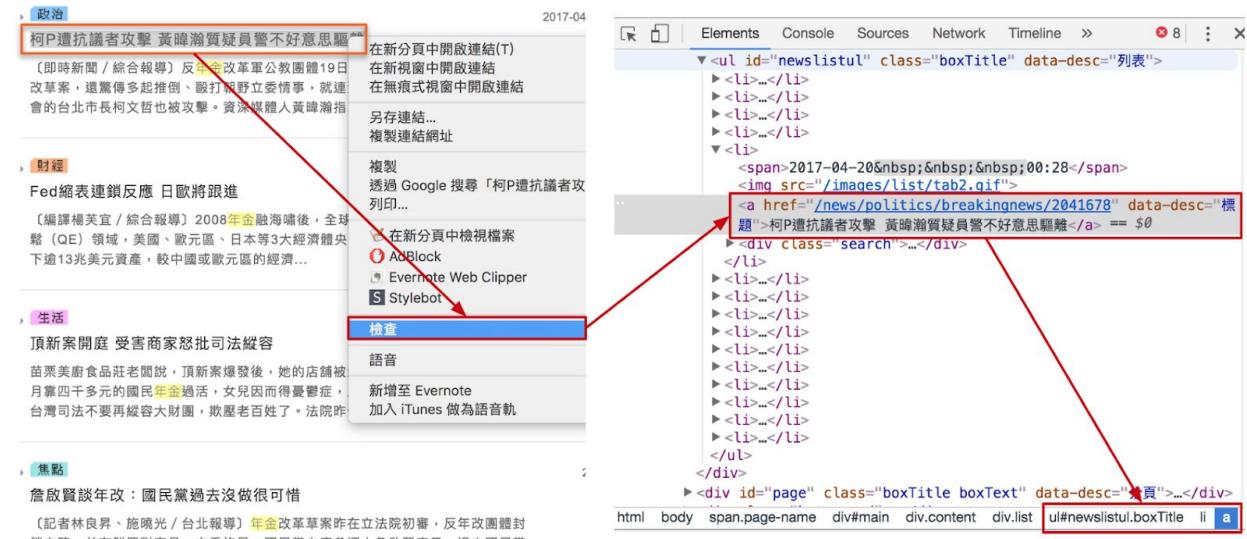
Id and class

在一個html檔案中，`id`是唯一的，而`class`則可以出現在好幾個地方。因此，若你要的資料外包一個有`id`的區塊，應該非常高興。例如下面的例子的CSS Selector可縮寫為

`#newslistul li`



Using DevTools to get data path

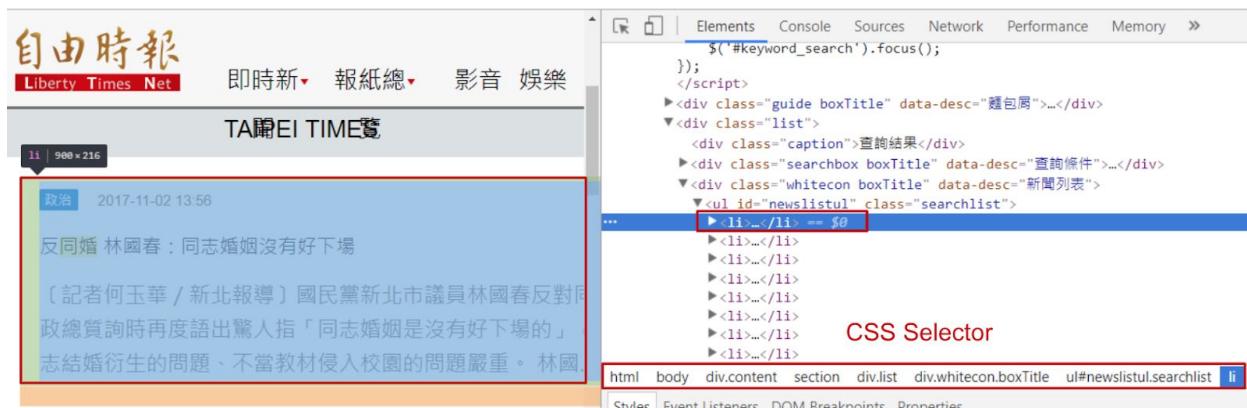


3-4-2 XPath and CSS selector

CSS和XPath的用途：CSS Selector和XPath是用來定義一個路徑，以選取HTML中的一個或者多個條件相同的元素。例如說，定義一條路徑來選取頁面上所有的標題，或選取所有標題背後的超鏈結，或選取頁面上搜尋結果的圖片路徑、或選取某PTT Web上討論版某篇文章所有回文者的ID。

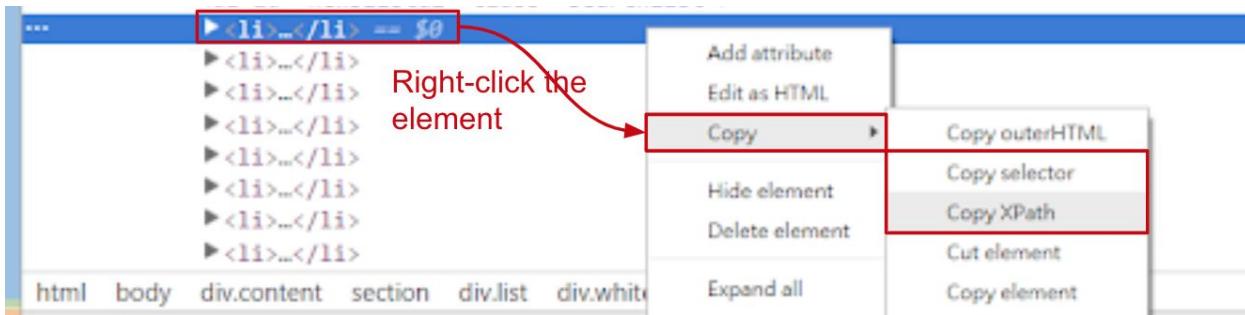
一般來說，要獲取HTML文檔中的內容有兩種途徑，一種是CSS Selector，和CSS語法、概念完全相同；另一種是XPath，不是那麼常看見，但卻有支援一些CSS Selector沒支援的功能（例如用text()取出沒有HTML tag僅有純文字的部分）。

使用Chrome DevTools找尋CSS Selector與XPath



對著你所要複製資料路徑（或者元素路徑）的元素按右鍵，就可以複製CSS selector與XPath路徑，自由時報的搜尋結果為例，其在搜尋結果頁面的一則文章其CSS selector與XPath分別為

- CSS selector: #newslistul > li:nth-child(1)
- XPath: /*[@id="newslistul"]/li[1]



一般化路徑與簡化路徑

注意看上面的例子會發現他是從`#newslistul`這個id開始寫起，因為id在整個HTML內理論上是唯一的，所以前面的路徑都可以省了。而複製的路徑`#newslistul > li:nth-child(1)`

表示`#newslistul`這個id底下的第一個``，可以進一步修改路徑以 表示`#newslistul`底下的``我全都要了`#newslistul li`。

若以XPath來看的話，複製的路徑：`//*[@id="newslistul"]/li[1]`

- `//`表示選取所有可能的情形
- `*`表示任意一個element。因為有id的關係，所以可以忽略它本身是個甚麼物件，所以用`/*`代表即可
- `*[@id="newslistul"]`表示任意一個id為`newslistul`的元素
- `/li[1]`在前述標籤底下的表示第一個``

可更改路徑為選取全部的元素，而非只是第一個：`//*[@id="newslistul"]//li`

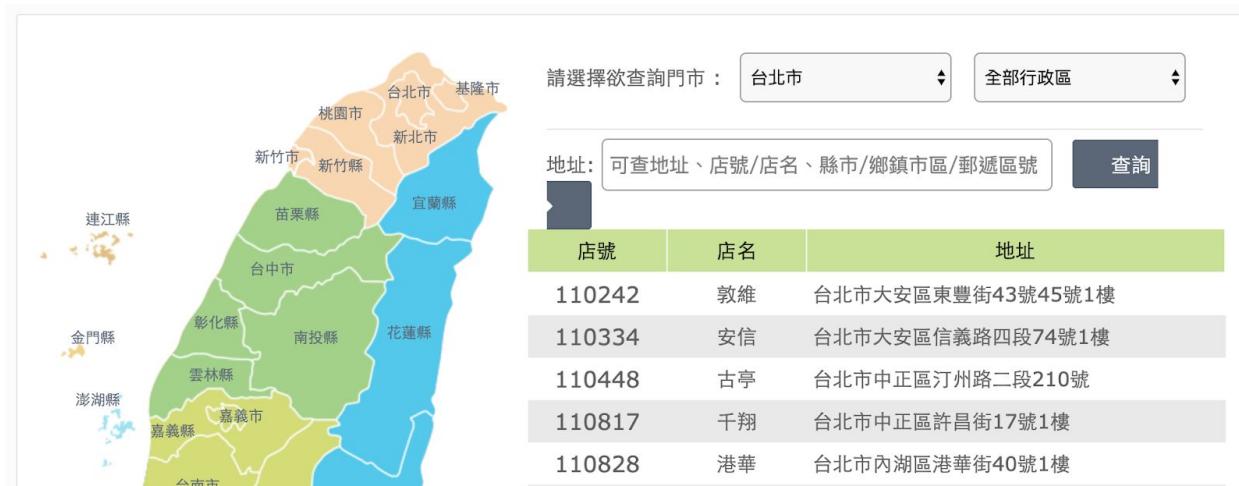
- `//li`表示在前述標籤底下的所有``我都要

3-5 爬取ibon資料

我打算把ibon資料打下來做分析，我把它視為7-11的門市，我的目的是依照鄉鎮市區來計算出每個地區的ibon數，以作為鄉鎮市區的發展指標。

ibon的查詢（http://www.ibon.com.tw/retail_inquiry.aspx#gsc.tab=0）介面如下。該介面可以根據每個不同的縣市查詢其所有的ibon所在地。所以我需要先：

1. 抓下來看有哪些縣市（提示：需要爬取並剖析HTML，用`GET()`）。
2. 根據每個縣市，爬取其有哪些ibon，一樣要剖析HTML，但要用`POST()`。
3. 把各縣市的資料合併起來，然後用`str_sub()`抽取出每個ibon的鄉鎮市區來做`summarize()`。



3-5-1 使用`GET()`爬取縣市

匯入所需函式庫

```
library(httr)
library(rvest)
options(stringsAsFactors = F)
```

不使用pipeline的寫法

```
url <- "http://www.ibon.com.tw/retail_inquiry.aspx#gsc.tab=0"
doc <- read_html(url)
nodes <- html_nodes(doc, "#Class1 option")
text <- html_text(nodes)
```

使用pipeline的寫法

```
url <- "http://www.ibon.com.tw/retail_inquiry.aspx#gsc.tab=0"
counties <- read_html(url) %>%
  html_nodes("#Class1 option") %>%
  html_text()
```

counties

```
[1] "台北市" "新北市" "基隆市" "宜蘭縣" "桃園市" "新竹市" "新竹縣" "苗栗縣"
[9] "台中市" "彰化縣" "南投縣" "雲林縣" "嘉義市" "嘉義縣" "臺南市" "高雄市"
[17] "屏東縣" "花蓮縣" "台東縣" "澎湖縣" "金門縣" "連江縣" "南海諸島"
```

有了所有的縣市名稱後，就可以進入下一步撈取所有縣市的ibon所在地址。

3-5-2 用POST()撈取一個縣市的ibon地址

打開Chrome DevTools後，開啟network分頁後，在原先的介面切換並載入新北市的資料。結果發現只跳出三個頁面，資料在最後一個頁面。觀察一下，他的Request Method是POST，而不是GET。只要是POST的，代表你得傳送一些表單資料（Form data）給他，把Headers分頁拉到最下方，可以找到Form data。這邊可以看見，網頁送出了新北市和COUNTY兩個表單資料給該兩個欄位。

Name	Value
strTargetField	COUNTY
strKeyWords	新北市

接下來，我們要模仿POST送出表單資料的過程把資料撈回來。從Headers可以找到Request URL http://www.ibon.com.tw/retail_inquiry_ajax.aspx。我們可以用content()來觀察一下抓回來的東西對不對。

我可以content()把抓回來的response轉字串，然後用print()列印出來。但是如果HTML檔案的內容過多的時候，會很容易找不到目標資料。另一個方法是，用cat()指令。cat()指令的用法和print()幾乎相同，唯獨cat()不會自動插入一個換行符號在列印結果的最末。除此之外，cat()指令可以把文字「列印」到檔案中，如以下的指令就把

抓回來的文字，寫到一個html中，我們就可以用電腦的瀏覽器把他打開來觀看。打開後應該就可以所要抓的表格。

```
url <- "http://www.ibon.com.tw/retail_inquiry_ajax.aspx"
res <- POST(url, body = list(strTargetField='COUNTY',
                             strKeyWords='台北市'))
print(content(res, "text"))
cat(content(res, "text"), file="temp.html")
```

至於實際把資料抓回來的部分，我選擇把每個表格的地址欄儲存為character vector，反正地址欄都有縣市鄉鎮市區名稱，我可以之後再解析即可。

```
doc <- read_html(content(res, "text"))
addr.node <- html_nodes(doc, "table tr td:nth-child(3)")
addrs <- html_text(addr.node) %>% trimws()
```

Options. 使用html_table()抓回整個表格

但事實上，你也可以用`html_table()`這個r vect的指令把整個table抓回來並轉為一個data.frame，

```
doc <- read_html(content(res, "text"))
table.node <- html_node(doc, "table")
html_table(table.node)
      X1     X2          X3
1 店號   店名        地址
2 110242 敦維  台北市大安區東豐街43號45號1樓
3 110334 安信  台北市大安區信義路四段74號1樓
4 110448 古亭  台北市中正區汀州路二段210號
```

3-5-3 用for-loop撈回所有縣市的ibon地址

上述方法可以撈回一個縣市的所有ibon地址，於是用for迴圈把所有縣市跑過，並且把所有抓到的地址組合成一個大的vector。注意下列程式碼，除了亮黃底色的程式碼外，其他都與前述程式碼相同。一共會撈回5459個ibon地址。

```
all_addr <- c()
url <- "http://www.ibon.com.tw/retail_inquiry_ajax.aspx"
for(county in counties){
  res <- POST(url, body = list(strTargetField='COUNTY',
```

```

        strKeyWords=county))
doc <- read_html(content(res, "text"))
addr.node <- html_nodes(doc, "table tr td:nth-child(3)")
addrs <- html_text(addr.node) %>% trimws()
print(sprintf("%s: %d stores", county, length(addrs)))
all_addr <- c(all_addr, addrs)
}

```

3-5-4 清理資料並取出鄉鎮市區

接下來其實是這個練習最困難的一部分，要取出鄉鎮市區。主要是因為

1. 有些鄉鎮市區的名字是四個字（如「太麻里鄉」、「阿里山鄉」）、兩個字（如「東區」）
2. 縣市級的「市」和鄉鎮市區的「市」可能會誤判。
3. 路名裡面會有「鎮」導致誤判，例如「XX區鎮海路」，程式可能會誤判為有一個鎮叫做「XX區」。

因此，你必須要有非常清晰的邏輯，並設計一些檢驗措施，讓你的鄉鎮市區名不會誤判。請自己先寫寫看再參考最後的答案。

先把它轉為dplyr實際上的tibble格式（tibble很像data.frame，幾乎可以照data.frame的方式來用，但稍微嚴謹一點點）。每個縣市的表格都有標題欄位「地址」，所以會多好幾個「地址」的資料，需要將其濾除。

```

library(tidyverse)
seven <- tibble(addr = all_addr) %>%
  slice(-1) %>%
  mutate(county = str_sub(addr, 1, 3)) %>%
  filter(!str_detect(addr, "地址")) %>% head %>% View

```

接下來，我先取代掉會造成分析麻煩的「楠梓加工區」，然後，我偵測縣市兩個字到後面鄉鎮市區前的字，而且我用.{1, 3}限定只能取1~3個字，這樣包準不會取到太後面卻還有「鎮」或「市」的路名。在前半部的縣市名稱，我需要加一個問號，代表Non-greedy的版本，不會一路取到後面鄉鎮市區的市，所以寫法是.*?[縣市]。而我實際上要取出的

就是鄉鎮市區的那幾個字，所以那幾個字的範圍我才加上小括號。最後用 `\\1` 取出這個小括號的範圍來作為鄉鎮市區的名字。

為了要知道我有沒有做對，我計算了鄉鎮市區的名字長度來做觀察。結果會發現，我取到的部分鄉鎮市區仍會多取到開頭為「市」或「鎮」的路名，如「市民大道」或「鎮海路」。

```
library(tidyverse)
seven <- tibble(addr = all_addr) %>%
  slice(-1) %>%
  mutate(county = str_sub(addr, 1, 3)) %>%
  filter(!str_detect(addr, "地址")) %>%
  mutate(addr = str_replace(addr, "楠梓加工區", "楠梓區")) %>%
  mutate(town = str_replace(addr, ".*?[縣市](.{1,3}[鄉鎮市區]).*", "\\1"))
%>%
  mutate(townlen = nchar(town)) %>% View
```

	addr	county	town	townlen
1	台北市松山區市民大道四段105號	台北市	松山區市	4
2	台北市中正區市民大道三段2號B2樓	台北市	中正區市	4
3	台北市信義區市府路45號B1之53	台北市	信義區市	4
4	新北市板橋區區運路21號	新北市	板橋區區	4
5	新北市汐止區鄉長路二段16號	新北市	汐止區鄉	4
6	宜蘭縣五結鄉鎮安村西河路4-6號	宜蘭縣	五結鄉鎮	4

因此，我針對這些長度為4的，我只取前三個字。但這樣會不小心把「阿里山鄉」給改為「阿里山」，所以我最後用手動操作的方式，把「阿里山」給恢復成「阿里山鄉」，這樣字的例子還有「太麻里」與「三地門」。

經過這樣的程序，就可以把鄉鎮市區的名字給完整切出來，你可以想想看有沒有比較簡單的邏輯或想法，想到的話跟我說。

```
library(tidyverse)
seven <- tibble(addr = all_addr) %>%
  slice(-1) %>%
  mutate(county = str_sub(addr, 1, 3)) %>%
```

```

filter(!str_detect(addr, "地址")) %>%
  mutate(addr = str_replace(addr, "楠梓加工區", "楠梓區")) %>%
  mutate(town = str_replace(addr, ".*?[縣市](.{1,3}[鄉鎮市區]).*", "\\\1"))
%>%
  mutate(town = ifelse(nchar(town)==4, str_sub(town, 1, 3), town)) %>%
  mutate(town = str_replace(town, "太麻里", "太麻里鄉"),
         town = str_replace(town, "阿里山", "阿里山鄉"),
         town = str_replace(town, "三地門", "三地門鄉")) %>%
  mutate(townlen = nchar(town)) %>%
  mutate(site_id = paste0(county, town))

```

3-6 爬取PTT討論版

在爬取HTML之前，你要先知道的是，資料如何被放在巢套Element的HTML檔案中，而Attribute（尤其是id和class和的href）又如何提供線索給CSS Selector或XPath來選取我們所要的Element，最後要知道XPath和CSS Selector的用途是用來選取你所需要的HTML中的資料。

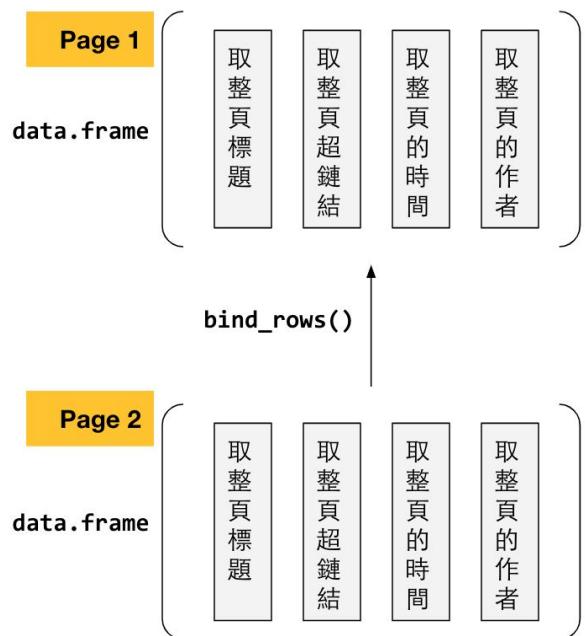
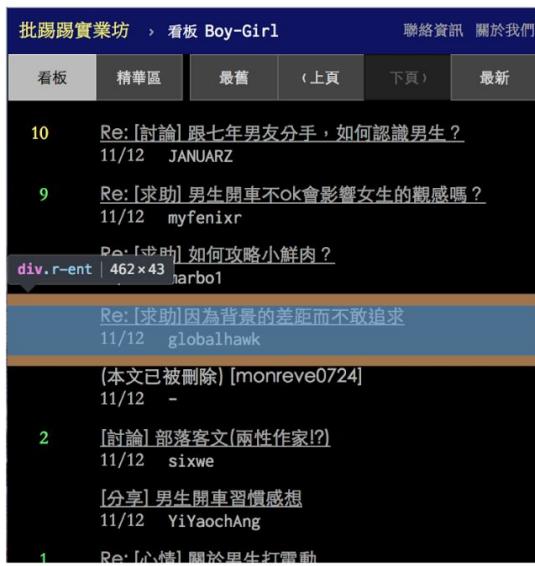
以下以ptt boy-girl版為例，展示如何以爬蟲擷取網頁上的資料，並整理成data.frame。PTT的網頁分為兩種主要類型，

- 其一稱為鏈結頁或索引頁，為文章的超鏈結，例如
<https://www.ptt.cc/bbs/Boy-Girl/index.html>。
- 其二為每一篇文章的內容頁
<https://www.ptt.cc/bbs/Boy-Girl/M.1523994970.A.71C.html>。

對於這種網頁，要設計兩階段的爬蟲，第一階段是把所有所需鏈結撈回來，第二階段是根據撈回來的鏈結去打撈文章，並把裡面的內文整理出來。

對於第一階段而言，概念大概如下面的示意圖，其實爬取HTML的概念都差不多是這樣，但以下示意圖以上述的第一階段為例：

用XPath或CSS Selector「分別」選取出整個頁面中的標題、超鏈結、時間、和作者，然後用data.frame()組合成一個data.frame；Page 2也如此炮製，然後把Page 2的data.frame用bind_rows()附加在Page 1的数据frame後面。



Step 1. 載入所需套件

在這個例子中，不僅要用 `httr` 來取得網頁頁面，還要用 `rvest` 套件（其實是呼叫 `xml2`）來剖析網頁。網頁內容並不像 json 一樣可以直接轉為 `data.frame` 或 `list`。網頁內容可以是資料、視覺化元素、也可以是架構元素，相對於 json 而言複雜也冗贅許多。因此需要一個套件能夠剖析 HTML 標籤例如 `xml2`，而 `rvest` 內則應用了 `xml2` 套件的內容來剖析網頁，另外，我會加載 `dplyr` 套件來用 `bind_rows()` 將過程中所抓取的資料進行合併。

```
library(rvest)
library(httr)
library(dplyr)
options(stringsAsFactors = F)
```

Step 2. 取回並剖析HTML檔案

在從 HTML 檔案中打撈出我們所要的資料有三個重要的 `rvest` 函式（步驟），分別為：

1. `read_html()` 依照網址將網頁取回並轉為 `xml_document`。

2. `html_nodes()` 用CSS Selector或XPath選擇所需的資料節點，另外`html_node()`是只取滿足條件的第一個節點。
3. `html_text()`或`html_attr()`或`html_table()`、`html_tags()`取出所要抓的節點的目標資料，可能是一個表格、一個標籤內容、或者是一個屬性值。

Step 2-1. `read_html()` 將網頁取回並轉為xml_document

- `read_html()`內部包含了`GET()`與`content()`等的實作，其主要的功能是將取回來的response轉為xml_document，所以若以`class(doc)`觀察其型態，會是`xml_document xml_node`。
- 使用`browseURL(url)`可以用瀏覽器打開該網址並瀏覽。

```
url <- "https://www.ptt.cc/bbs/Boy-Girl/index.html"
doc <- read_html(url)
class(doc)
## [1] "xml_document" "xml_node"
browseURL(url)
```

Step 2-2 以`html_nodes()` 以選擇所需的資料節點

html的檔案還包含了相當多其他視覺、互動、排版的標籤，因此通常只有少部分是資料，且存在層層的html元素中。因此，獲取到該網頁並轉為xml_document後，便要用`html_nodes()`或`html_node()`根據所給的CSS Selector或XPath來選擇所要取出的節點中的資料。要獲取該元素的CSS Selector可以利用Chrome DevTool或者是Firefox。用法是對著該網頁空白處按右鍵選擇檢查（inspect）。

以下用CSS Selector抽取：

- `#`指的是`id`、`.`指的是`class`。
- `#main-container`意思是，某個`id`為`main-container`的元素。
- `.title`指的是某個`class`為`title`的元素。
- `div.title`指的是`class`為`title`的`div`（排版元素）。
- `Div.r-list-container.action-bar-margin.bbs-screen`指的是同時具有`r-list-container`、`action-bar-margin`、`bbs-screen`三個`class`的`div`元素。

```
css <- "#main-container > div.r-list-container.action-bar-margin.bbs-screen
> div > div.title > a"
node.a <- html_nodes(doc, css)
```

```
class(node.a) # "xml_nodeset"
length(node.a)
```

用XPath抽取：

```
path <- '//*[@id="main-container"]/div[2]//div/div[3]/a'
node.a <- html_nodes(doc, xpath = path)
links <- html_attr(node.a, "href")
```

Step 2-2 棉充說明與XPath、CSS Selector的最佳化

用CSS Selector和XPath抽取有一些經驗法則，以PTT為案例來說的話，原本複製得來的CSS Selector和XPath分別為

- CSS Selector : `#main-container >`
`div.r-list-container.action-bar-margin.bbs-screen >`
`div:nth-child(3) > div.title > a`
- XPath : `//*[@id="main-container"]/div[2]/div[3]/div[2]/a`

```

▼ <div class="r-ent">
  ▶ <div class="nrec">...</div>
  <div class="mark"></div>
  ▼ <div class="title">
    ...
      <a href="/bbs/Boy-Girl/M.1494254996.A.BC9.html">Re: [討論] 不過就一本小說  

      有這嚴重？</a> == $0
    </div>
    ▶ <div class="meta">...</div>
    </div>
    ▶ <div class="r-ent">...</div>
    ▶ <div class="r-ent">...</div>
  
```

html body div#main-container div.r-list-container.action-bar-margin.bbs-screen div.r-ent div.title a

到html之間，有一個id為#main-container的div。所以從div#main-container開始取即可。通常確定有id後，我就會開始找重複項，這邊的重複項是class為r-ent的div，找到重複項後，我就會去找重複項後面到我要的資料的路徑，而我們要的就在class為title的div中。所以一個簡化的路徑是div#main-container div.r-ent div.title a。但在這邊，既然有id，就不在乎是什麼element有id，所以可以只寫#main-container，而也只有資料在的div才有.r-ent的class，所以也不用強調前面的div，至於.title前面一定是搭div的element，所以我可以簡化到寫為#main-container .r-ent .title a。甚至，由於.r-ent一定在#main-container裡面，所以連前面的id我都可以省略掉。

如果用XPath來寫的話，照上面的邏輯應該要寫成

```
//*[@id="main-container"]//div[@class="r-ent"]/div[@class="title"]/a。
```

注意到第二個div前有兩個斜線//，原因是#main-container和.r-ent之間還有一層div，兩個斜線代表前後兩者間還有其他層，如果沒有兩個斜線而只有一個斜線的話，那就是代表#main-container下面一層馬上就要是.r-ent，若你把它改成單斜線的話，就會發現取不到資料。

以政府招標為範例來解釋

- 最終要抓的資料節點為何？主要為[``](#)的[href](#)屬性值和[`<a>`](#)底下[`<div>`](#)中的內容。
- 觀察資料在哪個節點「逐筆」出現？這邊是[`<tr>`](#) table row, [`<td>`](#)則是資料欄位
- 利用id和class來辨識：遇到有id就從id開始取就好，例如[`#searchResult`](#)。
- 中間的節點多可以忽略：例如這個CSS selector可省略寫為[`#searchResult tr a`](#)

tag is highlighted with a blue border. The path to this element is shown in the breadcrumb navigation bar at the bottom: tr > td > table > tbody > tr > td > #searchResult > tbody > tr > td > a."/>

```
▼<td class="T12" style="text-align:left">
  ▼<a href="/tps/tpam/main/tps/tpam/
    tpam_tender_detail.do?
    searchMode=common&scope=F&primaryKey=52333966"> == $6
    "106E008"
    <div class="wordwrap">竹子湖地區公共廁所工程</div>
  </a>
</td>
```

Step 2-3 `html_text()`或`html_attr()`轉出所要的資料

我們所要的資料為[`\[心情\]`](#) 看到自己喜歡女生跟別的男生走很近好難過[``](#)中的超鏈結和標題文字。

- `html_text()`: 在[`<a>`](#)與[``](#)之間的[心情] 看到自己喜歡女生跟別的男生走很近好難過稱為[`<a>`](#)的元素內容，要用`html_text(node.a)`來抽取。
- `html_attr():`* 在[`<a>`](#)內的[`href="/bbs/Boy-Girl/M.1523983903.A.71E.html"`](#)稱為[`<a>`](#)的屬性，該屬性名稱為[`href`](#)，屬性值為[`/bbs/Boy-Girl/M.1523983903.A.71E.html`](#)。要用`html_attr(node.a, "href")`來抽取（相當於指定某個Element的[`href`](#)屬性的內容）。

取出元素節點的內容，相當於取出<a>間所夾的內容。

```
texts <- html_text(node.a)
length(texts)
```

取出元素節點某個屬性的值，這邊是取出[href](#)這個屬性的值，也就是超鏈結。

```
links <- html_attr(node.a, "href")
class(links)
# character
links[1]
# "/bbs/Boy-Girl/M.1555188846.A.D5F.html"
```

但這些超鏈結只有後半段，點開一篇文章觀察其網址，發現我們所抓到的網址少了前面那一段，因此用[paste0\(\)](#)黏上前綴的網址前段，便可得到完整網址，可用
[browseURL\(links\[1\]\)](#)觀察。

```
pre <- "https://www.ptt.cc"
links <- paste0(pre, links)
links[1]
# [1] "https://www.ptt.cc/bbs/Boy-Girl/M.1555188846.A.D5F.html"
```

重組上列程式碼如下：

```
pre <- "https://www.ptt.cc"
url <- "https://www.ptt.cc/bbs/Boy-Girl/index.html"
doc <- read_html(url)
css <- "#main-container div.r-ent div.title a"
node.a <- html_nodes(doc, css)
```

題外話，因為[rvest](#)與[httr](#)均支援tidyverse的程式寫作，因此可改為以下pipeline的形式。但通常我不見得會這麼寫，比如說<a>這個Element我可能不僅會取出其超鏈結，還打算取出標題文字，為了避免重複操作，我不見得會用pipeline來寫。

```
pre <- "https://www.ptt.cc"
url <- "https://www.ptt.cc/bbs/Boy-Girl/index.html"
links <- url %>% read_html %>%
  html_nodes("#main-container div.r-ent div.title a") %>%
  html_attr("href") %>%
  paste0(pre, .)
```

Step 3. 用for迴圈打撈多頁的連結

我們可以觀察到PTT該版的鏈結頁的網址規則如下

- 最新頁：<https://www.ptt.cc/bbs/Boy-Girl/index.html>
- 倒數第二頁：<https://www.ptt.cc/bbs/Boy-Girl/index3902.html>
- 倒數第三頁：<https://www.ptt.cc/bbs/Boy-Girl/index3901.html>
- 倒數第四頁：<https://www.ptt.cc/bbs/Boy-Girl/index3900.html>
- 最新一頁因此可類推出為<https://www.ptt.cc/bbs/Boy-Girl/index3903.html>

因此，我打算寫一個for-loop，讓他幫我（先）抓最後10頁，那就是3894到3903頁。並且，把頁數當成網址的參數，用sprintf()或paste0()組合出網址，以下分別提供兩種版本。你可以把它print出來且點選看看是否是你所要的網頁。

```
for(p in 3894:3903){
  url <- sprintf("https://www.ptt.cc/bbs/Boy-Girl/index%s.html", p)
  # url <- paste0("https://www.ptt.cc/bbs/Boy-Girl/index", p, ".html")
}
```

接下來，我要用一個all_links來存放所有的網址，並且把每一個頁面抓到的網址們都用vector的concatenation粘在一起 all_links <- c(all_links, links)。

```
all_links <- c()
for(p in 3894:3903){
  url <- sprintf("https://www.ptt.cc/bbs/Boy-Girl/index%s.html", p)
  all_links <- c(all_links, links)
}
```

最後，我就將上述抓到網址的方法填入這個for-loop中，並把抓到的網址存為links，就會隨著每回合的for-loop逐漸把抓到的網址整理在一起。

```
pre <- "https://www.ptt.cc"
all_links <- c()
for(p in 3894:3903){
  url <- sprintf("https://www.ptt.cc/bbs/Boy-Girl/index%s.html", p)
  print(url)
  doc <- read_html(url) # Get and parse the url
  css <- "#main-container div.r-ent div.title a"
  node.a <- html_nodes(doc, css)
  links <- html_attr(node.a, "href")
  links <- paste0(pre, links) # Recover links
  all_links <- c(all_links, links)
}
length(all_links)
```

Step 4. 根據連結取回所有貼文

前面是針對每一個頁面的網址取回該頁面中所有的貼文鏈結，所以我現在all_links中是所有的貼文鏈結。我可以仿照前面的做法，就每一個貼文鏈結，取回貼文內容，貼文內容可能包含作者、時間、標題、版別、內文等資料欄位

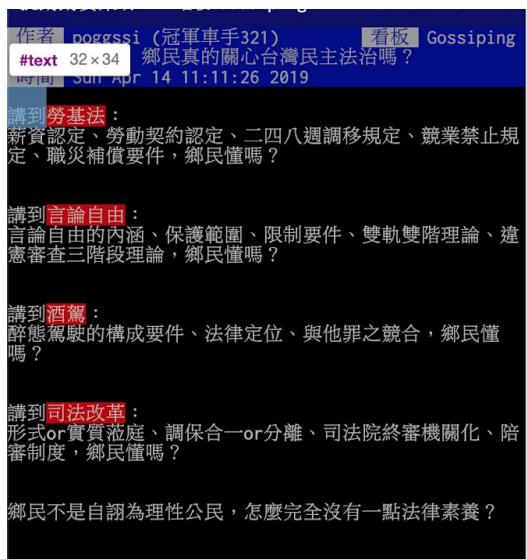
不失一般性地，用第一篇貼文的鏈結來做資料抓取實驗，之後再抓第二篇、第三篇即可。

首先，對於這每篇文章，我要爬取作者（author）、標題（title）、時間（time）、看板（board）和內容（content）五個欄位，包含原本文章的連結url一共六個欄位。

從以下的圖發現，每篇文章的內容放置在div#main-content中，之後一連跟著四個div.article-meteline，裡面分別有作者、看板、標題與時間，但從class名稱看不出來哪個，但我們要的值會是在span.article-meta-value中。所以我的規劃是，一次把所有的.article-meta-value給抓回來，然後再去分別指定哪個是作者、看板、標題或時間。所以我用

```
link <- all_links[1]
doc <- read_html(link)
meta.css <- "#main-content div.article-meteline span.article-meta-value"
```

```
metadata <- html_text(html_nodes(doc, meta.css))
```



```
<div id="main-content" class="bbs-screen bbs-content">
  <div class="article-metaline">
    <span class="article-meta-tag">作者</span>
    <span class="article-meta-value">poggssi (冠軍車手321)</span>
  </div>
  <div class="article-metaline-right">
    <span class="article-meta-tag">看板</span>
    <span class="article-meta-value">Gossiping</span>
  </div>
  <div class="article-metaline">
    <span class="article-meta-tag">標題</span>
    <span class="article-meta-value">[問卦] 鄉民真的關心台灣民主法治嗎？</span>
  </div>
  <div class="article-metaline">
    <span class="article-meta-tag">時間</span>
    <span class="article-meta-value">Sun Apr 14 11:11:26 2019</span>
  </div>
  "講到"
  <span class="b1">勞基法</span>
  "：
  薪資認定、勞動契約認定、二四八週調移規定、競業禁止規定、職災補償要件，鄉民懂嗎？

  講到"
  <span class="b1">言論自由</span>
  "：
  言論自由的內涵、保護範圍、限制要件、雙軌雙階理論、違憲審查三階段理論，鄉民懂嗎？
```

另外，從上面的文字中可發現，主文的部分被包含在

#main-content

中，但在上述的作者、標題等後面，若這時候我們用CSS Selector搭配html_node()和html_text()把

#main-content

的內容都給取出來，會連上述的作者、標題等都一起取出來。最好的方法是用XPath的text()來只取出文字的部分，而不取出有tag的部分。

而且這樣取出來會是每個paragraph都是一個character vector中的值，所以要把他們合併成一篇文章，此時要用paste()指令，paste()指令可以指定把一個character vector給串接成一個string，輸入的變數是一個character vector，然後設定參數collapse = "", 代表串接起這些character時中間不要有空白。

```
post.xpath <- '//*[@id="main-content"]/text()'
post.paragraph <- html_text(html_nodes(doc, xpath = post.xpath))
post <- paste(post.paragraph, collapse = "")
```

最後，我們知道metadata變數的第一個是作者、第二個是標題、第三個是時間、便依序指派給個別的變數後，組成data.frame如下，這樣所組成的data.frame是為只有一筆資料的data.frame，但有五個變項。

```
link <- all_links[1]
```

```

doc <- read_html(link)
meta.css <- "#main-content div.article-meteline span.article-meta-value"
metadata <- html_text(html_nodes(doc, meta.css))
post.xpath <- '//*[@id="main-content"]/text()'
post.paragraph <- html_text(html_nodes(doc, xpath = post.xpath))
post <- paste(post.paragraph, collapse = "")
post.df <- data.frame(post,
                      uid = metadata[1],
                      title = metadata[2],
                      timestamp = metadata[3],
                      url = link
)

```

這樣我們取得的第一篇文章存放在post.df中，之後，我稍微修改一下上述的程式，就可以用for-loop抓取第2至n篇文章，每一篇都存在名為temp.df的数据框中，然後用bind_rows()依序和post.df銜接在一起，除了黃色的部分是新增的之外，都跟前面的程式幾乎一模一樣。

```

for(link in all_links[2:10]){
  doc <- read_html(link)
  meta.css <- "#main-content .article-meteline .article-meta-value"
  metadata <- html_text(html_nodes(doc, meta.css))
  post.xpath <- '//*[@id="main-content"]/text()'
  post.paragraph <- html_text(html_nodes(doc, xpath = post.xpath))
  post <- paste(post.paragraph, collapse = "")

  temp.df <- data.frame(post,
                        uid = metadata[1],
                        title = metadata[2],
                        timestamp = metadata[3],
                        url = link
  )
  post.df <- bind_rows(post.df, temp.df)
}

```

uid	uname	timestamp	post	link
[分享]她答應了!她答應了!	candyzz (噜拉拉)	Sun Dec 10 02:37:15 2006	作者candyzz (噜拉拉)看板Boy-Girl標題大小姐時間Sun De...	https://www.ptt.cc/bl
2 abcc122333 (小巴)	[分享]她答應了!她答應了!	Mon Dec 25 01:11:34 2006	作者abcc122333 (小巴)看板Boy-Girl標題[分享]她答應了!...	https://www.ptt.cc/bl
3 soppy158 (Rosy)	(兩性成長)兩性平等與錯誤投射	Tue Dec 26 10:11:03 2006	作者soppy158 (Rosy)看板Boy-Girl標題(兩性成長)兩性平...	https://www.ptt.cc/bl
4 voicespq (男稿的人)	[分享]人際關係 永續發展	Mon Jan 8 11:45:20 2007	作者voicespq (男稿的人)看板Boy-Girl標題[分享]人際關係 ...	https://www.ptt.cc/bl
5 flower319 (*花*)	[討論]我該怎樣跟我家閃光開口	Sun Jan 14 13:46:24 2007	作者flower319 (*花*)看板Boy-Girl標題[討論]我該怎樣跟...	https://www.ptt.cc/bl
6 MercNick (跌倒後才能體會痛)	[閒聊] 你一定要幸福囉 !	Tue Jan 16 21:35:12 2007	作者MercNick (跌倒後才能體會痛)看板Boy-Girl標題[閒聊] ...	https://www.ptt.cc/bl
7 twoyearsold (老二最近酷酷的)	Re: [求助] 留學代表一定要分手嗎	Tue Feb 6 03:44:19 2007	作者twoyearsold (老二最近酷酷的)看板Boy-Girl標題Re: [...]	https://www.ptt.cc/bl
8 laura4 (反 嫦 訓 真)	[分享] 愛情 by李敖	Tue Feb 6 22:06:37 2007	作者laura4 (反 嫦 訓 真)看板Boy-Girl標題[分享] 愛情 by...	https://www.ptt.cc/bl
9 laura4 (反 嫦 訓 真)	[分享] Boy & Girl	Tue Feb 6 23:49:06 2007	作者laura4 (反 嫦 訓 真)看板Boy-Girl標題[分享] Boy & Gi...	https://www.ptt.cc/bl
10 larukas (迷霧中的王者 N)	[分享] 喜歡與愛的不同	Fri Feb 9 03:03:16 2007	作者larukas (迷霧中的王者 N)看板Boy-Girl標題[分享] 喜...	https://www.ptt.cc/bl
11 iloveBridge (我要BJ單身日記)	[閒聊] 自己的失戀經歷	Sun Feb 11 04:40:19 2007	作者iloveBridge (我要BJ單身日記)看板Boy-Girl標題[閒聊] ...	https://www.ptt.cc/bl
12 yujihinata ()	Re: [討論]有女生跟我一樣嗎..很怕嫁到窮的男生	Sun Feb 11 11:34:12 2007	作者yujihinata ()看板Boy-Girl標題Re: [討論]有女生跟我一...	https://www.ptt.cc/bl
13 rabbitmimi (掃塵除垢)	Re: [討論]有女生跟我一樣嗎..很怕嫁到窮的男生	Sun Feb 11 14:50:50 2007	作者rabbitmimi (掃塵除垢)看板Boy-Girl標題Re: [討論]有...	https://www.ptt.cc/bl
14 MIKANANA (焰)	[轉載]爺爺的情書	Mon Feb 12 16:55:42 2007	作者MIKANANA (焰)看板Boy-Girl標題[轉載]爺爺的情書時...	https://www.ptt.cc/bl

補充(1) 較好的寫法

一個比較好的寫法是，不用先讀第一篇文章，而是用`data.frame()`初始化一個空的`data.frame`，之後可以利用`bind_rows()`可自動增添缺少的變數的特性，自然就會補上所需要的變數。因此，只要改寫黃色的部分。

但這樣的寫法仍會有一個缺點，也就是當文章數越來越多時會越跑越慢。原因是，假設現在你已經抓了9999篇文章，你這次的for-loop要抓第10000篇，然後用`bind_rows()`合併第10000篇，此時，`post.df`已經有9999篇非常肥大，等號右邊的`bind_rows()`跑完後會變成10000篇，此時又要把原本很肥大的`post.df`覆蓋掉，所以會非常費時。

```
post.df <- data.frame()
for(link in all_links[1:10]){
  doc <- read_html(link)
  meta.css <- "#main-content .article-meteline .article-meta-value"
  metadata <- html_text(html_nodes(doc, meta.css))
  post.xpath <- '//*[@id="main-content"]//text()'
  post.paragraph <- html_text(html_nodes(doc, xpath = post.xpath))
  post <- paste(post.paragraph, collapse = "")

  temp.df <- data.frame(post,
                        uid = metadata[1],
                        title = metadata[2],
                        timestamp = metadata[3],
                        url = link
  )
  post.df <- bind_rows(post.df, temp.df)
}
```

```
}
```

補充(2) 最佳的寫法

採用list先將每一個data.frame存放起來，然後跑完所有的for-loop後才用bind_rows()將所有資料合併為data.frame。此時，p指的是第幾個連結，而前例的link就相當於下方的all_links[p]。

```
post.list <- list()
for(p in 1:length(all_links)){
  doc <- read_html(all_links[p])
  meta.css <- "#main-content .article-meteline .article-meta-value"
  metadata <- html_text(html_nodes(doc, meta.css))
  post.xpath <- '//*[@id="main-content"]/text()'
  post.paragraph <- html_text(html_nodes(doc, xpath = post.xpath))
  post <- paste(post.paragraph, collapse = "")

  post.list[[p]] <- data.frame(post,
                                uid = metadata[1],
                                title = metadata[2],
                                timestamp = metadata[3],
                                url = all_links[p]
                                )
}
post.df <- bind_rows(post.list)
```

3-7 rtweet 使用API獲取資料

since_id - Returns results with an ID greater than (that is, more recent than) the specified ID. There are limits to the number of Tweets which can be accessed through the API. If the limit of Tweets has occurred since the since_id, the since_id will be forced to the oldest ID available. max_id - Returns results with an ID less than (that is, older than) or equal to the specified ID.

<https://developer.twitter.com/en/docs/tweets/timelines/guides/working-with-timelines>

通常用rtweet套件時，他傳回的會是從最近的往前回傳，所以最後一筆資料通常最老。而tweet的編號是照時間順序邊的，越接近現在編號越大。因此，如果抓完一個資料區段想延續把剩下還沒抓完的資料區段抓完，那就是把最後一筆資料（status id最小的資料）的id當成max_id，就可以往前抓。

抓取的時候用max_id所抓到的會是max_id往前抓的內容，若用since_id所抓到的會是since_id往最近抓的內容。

Step 1. 獲取資料權限

- **登入**：用你自己的twitter帳號登入twitter的開發者網站
<https://developer.twitter.com/>。並申請成為開發者帳號。申請時請詳實填寫可能的用途，和所需的資料內容。
- **新建App**：在右上角可以找到一個APP的選項，點選並選擇「Create An App」以建立一個新的App來獲取用資料的權限。
- 要填寫網站的地方可以填系所網站或台大網站，不要亂填信箱。
- 最後你可以拿到一個權限碼如下圖所示，包含Consumer Key、Consumer secret、Access token與Access token secret。屆時要複製下來貼入程式碼。

The screenshot shows the 'Keys and tokens' tab of the Twitter Developer app details page. It displays two sets of API keys: Consumer API keys and Access token & access token secret.

Consumer API keys:

- API key: MR[REDACTED]tCbBit
- API secret key: 7I4pDR[REDACTED]wV9D

Access token & access token secret:

- Access token: 53118769-[REDACTED]DRC3sEp6H
- Access token secret: 21EFpqBAj-[REDACTED]C8zWxu

Below the tokens, there are 'Regenerate' and 'Revoke' buttons for each set of keys.

Step 2. Environment settings

```
# install.packages("rtweet")
library(rtweet)
library(stringr)
library(tidyverse)
library(tidytext)
options(stringsAsFactors = F)
```

Setting-up tokens

```
token <- create_token(app = "YOUR_APP_NAME",
                      consumer_key = "MRFPZ...CbBit",
                      consumer_secret = "714pDFQEnG...wV9D")
```

Step 3. Getting tweets

```
rt <- search_tweets( q, n = 1000000,
                      retryonratelimit = TRUE, include_rts = T)
```

請查閱rtweet套件以獲取更多可用的函式。 <https://rtweet.info/index.html>

Step 4. Getting user timeline

```
user_timeline <- get_timelines(c("speakerpelosi"), n = 3200)
```

補充：使用youtube API

- <https://github.com/soodoku/tuber>

3-8 定時爬取即時資料

Youbike：排得很好（反正拆開組回來就是data.frame），但是即時資料，每分鐘更新，需要定時抓，每隔多久抓一次？這有兩種策略：策略一是邊抓邊組合，抓下來就組合成data.frame；策略二是抓下來後先全部存成檔案，等抓到夠再讀檔組合。而實用上都會用策略二，因為你不知道你的程式會不會當掉或你不小心把它關掉，導致於你前面抓的都前功盡棄。策略二包含三個步驟：

1. 抓取檔案，並讀取系統時間，作為檔案名稱命名
2. 讓程式自動休息n秒
3. 用for-loop讓程式自動跑上面的事情

Step 0. Loading packages

```
library(httr)
library(jsonlite)
options(stringsAsFactors = FALSE)
url <- "http://data.taipei/youbike"
```

Step 1. Getting data twelve times per 5 mins

`Sys.sleep()`以秒數計時，5分鐘相當於300秒。

```
for(i in 1:12){
  GET(url, write_disk("ubike"), overwrite=TRUE)
  Sys.sleep(300)
}
```

Step 2. Renaming files by timestamp

由於這樣抓下來每個檔案名稱都是ubike，又設定`overwrite=TRUE`的參數，最後會導致資料只剩下一筆，不符合需求。因此，要對每一次抓下來的資料做妥善命名，保持其唯一性。這通常有兩種方法，其一是編流水號，其二是利用時間。我比較喜歡用時間，反正時間不會重頭，用時間鐵定不會重複。

```
for (i in 1:30) {
  ctime <- format(Sys.time(), "%Y%m%d%H%M%S")
  GET(url, write_disk(paste0("data/ubike", ctime)))
  print(ctime)
```

```
Sys.sleep(60)
}
```

`Sys.time()`所抓取的是當下的時間，其本身是R的時間物件`POSIXct`。由於已經是R的時間物件，所以可以用`format()`把該時間轉為字串，只要給定其所希望的時間格式，就可以轉為所希望的日期與時間字串格式。相關的控制詞彙（如`%Y%m%d%H%M%S`）可查詢[?strptime](#)。

在此稍微複習一下，在程式語言中，時間通常有三種格式，一種是該程式語言用來做時間運算的時間物件，Python有Python的時間物件、R有R的時間物件；另一種是該時間物件可以被展示為字串；最後一種是該時間物件實際上被儲存的方式多半為一個整數，之後再把該整數用數學公式轉為時間（例如用自1900年1月1日0分0秒至今日的總秒數的整數來作為時間的儲存單位）。

```
> Sys.time()
[1] "2019-04-08 02:49:49 CST"
> class(Sys.time())
[1] "POSIXct" "POSIXt"
> format(Sys.time(), "%Y%m%d%H%M%S")
[1] "20190408025352"
> format(Sys.time(), "%Y-%m-%d %H:%M:%S")
[1] "2019-04-08 02:54:26"
```

Step 3. Full code

若我每5分鐘撈一次，那每天需要撈288次，兩天就需要576次，如果要撈一週，就要撈 288×7 次，不是個非常大的數字，但因為每撈一次要休息五分鐘，保證開了以後你不關電腦你的電腦就會撈一週。

```
for (i in 1:576) {
  ctime <- format(Sys.time(), "%Y%m%d%H%M%S")
  GET(url, write_disk(paste0("ubike", ctime)))
  print(ctime)
  Sys.sleep(300)
}
```

3-X Miscellaneous

- 內政部不動產實價交易平台 (<http://lvr.land.moi.gov.tw>)。該範例的特色在於，進網頁時會有認證過程，資料的頁面為HTML Table，但是一次傳回所有的資料，只是資料表格被分頁呈現而已
 - https://github.com/suensummit/RCrawlers/blob/master/CaseStudies/Case8_LandMoiGov/try.R

3-A Practices & Assignments

Practice 3-1-1：找到這些網站背後的JSON檔

1. 公視新聞網<https://news.pts.org.tw/dailynews.php>
2. 鉅亨網新聞 <https://news.cnnes.com/news/cat/headline?exp=a>
3. 空氣品質監測站（開放資料常常當，他自己網頁不會當）
<https://taqm.epa.gov.tw/taqm/tw/default.aspx>
4. 在Dcard Girl版找到資料檔案的連結
<https://www.dcard.tw/f/girl>
5. 在pchome的24h下查詢後找到相對應的資料檔的連結
<http://24h.pchome.com.tw/index/>
6. 在104下查詢後，找到相對應的資料連結
<https://www.104.com.tw/>
7. 信義房屋 (<http://buy.sinyi.com.tw/list/Taipei-city/116-zip/1.html>)：需要點按第二頁後才會找到JSON。藉此可以回去找到第一頁。
8. 中央社 <https://www.cna.com.tw/>

Practice 3-1-2

請問在Alexa網站流量調查「Top Sites in Taiwan」所列的Top 100網站中，哪些網站背後的運作可能也是一個JSON檔案？請利用Chrome DevTools試著尋找之，並撈取回你找的那頁的資料，在程式碼中用函式簡要印出該資料的特徵（例如`str()`等）。

Practice 3-3-1

請爬取除了dcard、104之外，以json作為資料儲存格式的網頁，至少包含五個頁面或超過100則以上的資料（也就是說你不能只爬一頁，必須要嘗試觀察規則，爬比較多頁）。這邊建議可以嘗試看看鉅亨網的即時新聞、信義房屋、中央社的新聞查詢、蘋果日報的新聞查詢等，或者是政府一些開放查詢的資料。

繳交規定：

1. 程式碼的for-loop中必須要以類似`print(p, nrow(all.df))`的做法列印出抓取每一頁後的資料總筆數。
2. 以.rmd檔撰寫，並輸出為.html檔。以`save(variables, file=filename)`將資料儲存為.rda檔。最後，將這三個檔案壓縮並上傳至Ceiba。

Practice 3-4-1 查找XPath與CSS Selector

在觀看過對於XPath和CSS Selector的簡介後，請找到下列頁面或滿足下列需求的XPath與CSS（請注意，如果在XPath或CSS Selector中有特定的id或class的話，必須要盡量採用id或class來簡化XPath或CSS Selector）。

1. 選取PTT Gossiping版最新頁面的所有標題之CSS Selector與XPath

<https://www.ptt.cc/bbs/Gossiping/index.html>

The screenshot shows the PTT Gossiping board index page. A specific post is highlighted with a blue box. The post title is "[問卦] FB是不是大型詐騙平台阿？". The browser's developer tools are open, showing the HTML structure of the highlighted post. The title is located within a `[問卦] FB是不是大型詐騙平台阿？` element. The surrounding HTML includes meta information and other post details.

```

    ><div class="meta">...</div>
    </div>
    <div class="r-ent">
        <div class="nrec"></div>
        <div class="title">
            <a href="/bbs/Gossiping/M.1554643566.A.5A5.html">[問卦] FB是不是大型詐騙平台阿？</a>
        </div>
        <div class="meta">...</div>
        <div class="r-ent">
            <div class="title">
                <a href="/bbs/Gossiping/M.1554643566.A.5A5.html">[問卦] FB是不是大型詐騙平台阿？</a>
            </div>
        </div>
    </div>

```

html body #main-container div.r-list-container.action-bar-margin.bbs-screen div.r-ent div.title a

2. 選取這則文章所有回文者的ID之CSS Selector與XPath

<https://www.ptt.cc/bbs/Gossiping/M.1554642394.A.0DA.html>



3. 中時電子報搜尋結果第一個回傳頁面的所有標題

<https://www.chinatimes.com/search/柯文哲?page=2&chdtv>

Practice 3-5-1 爬取新聞搜尋結果

- 嘗試撈取自由時報某關鍵字近一年的搜尋結果，注意到自由時報規定一次只能查閱三個月的新聞，請問要如何解決之（提示：用sprintf()修改你要撈取的網址，然後用for-loop撈取）。如果你嘗試過無法撈取近一年的新聞，你可以只撈近三個月即可。
- 自訂你要撈取的關鍵字，用nrow()印出你所撈取的則數。
- 請順便在RMD中回答，你認為撈取新聞可以做什麼樣的專案或研究？我可能可以打撈什麼類型的關鍵字（列出你認為該包含的關鍵字）來回答什麼樣的問題？例如：我想撈「中國」、「台灣」這樣的關鍵字來觀察近幾年來新聞在談論中國或台灣時，其用字有何改變？而中國或台灣又有哪些相關新聞越來越受重視？可否從新聞用字發覺近期民眾對中國與台灣的感受或意識形態。
- 嘗試即可，請勿以爬蟲大量下載，避免造成對方服務塞車，影響對方權益。
- <http://news.ltn.com.tw/search?keyword=%E6%9F%AF%E6%96%87%E5%93%B2>

Practice 3-5-2 爬取並比較新聞媒體

- 嘗試撈取數個電子報的的搜尋頁面，例如自由時報、蘋果日報、聯合時報、中時電子報。

第四章 tidy風格R程式寫作

4-0 tidy型態的資料

從這章節開始，將大量採用tidyverse的系列套件來代替原本R基礎套件所提供的功能。tidyverse套件其實是一群有共同規範的套件的集合，例如httr、dplyr、jsonlite、tidytext、stringr等套件都是該套件集合的成員之一。應用tidyverse系列套件來寫R的程式意味著以下主要變化：

1. dplyr套件的函式可使選取變項或資料更加清楚簡潔、也提供強大的資料彙整、樞紐運算的相關函式，例如以下的select()、mutate()、filter()等函式。



2. 用magrittr套件的pipeline來處理資料，不但使得程式碼流程清楚，也可以降低新變數的個數。
3. 用tidy風格作為操作data.frame的規範，強化以變項為基礎來操作資料的精神。tidy風格和一般風格的資料如下。各位在前面章節的案例中應該不難回想起，產假支薪的資料便是以下所提到的典型的Messy form，因為它將每一年作為變數，並在每一年的變數底下列出該國當年的產假支薪水平；但tidy風格會希望將「年」獨立為一個變項，也跟統計的想法較為接近。

Messy form

year	artist	track	time	date.entered	wk1	wk2	wk3
2000	2 Pac	Baby Don't Cry	4:22	2000-02-26	87	82	72
2000	2Ge+her	The Hardest Part Of ...	3:15	2000-09-02	91	87	92
2000	3 Doors Down	Kryptonite	3:53	2000-04-08	81	70	68
2000	98°0	Give Me Just One Nig...	3:24	2000-08-19	51	39	34
2000	A*Teens	Dancing Queen	3:44	2000-07-08	97	97	96
2000	Aaliyah	I Don't Wanna	4:15	2000-01-29	84	62	51
2000	Aaliyah	Try Again	4:03	2000-03-18	59	53	38
2000	Adams, Yolanda	Open My Heart	5:30	2000-08-26	76	76	74

tidy form

year	artist	time	track	date	week	rank
2000	2 Pac	4:22	Baby Don't Cry	2000-02-26	1	87
2000	2 Pac	4:22	Baby Don't Cry	2000-03-04	2	82
2000	2 Pac	4:22	Baby Don't Cry	2000-03-11	3	72
2000	2 Pac	4:22	Baby Don't Cry	2000-03-18	4	77
2000	2 Pac	4:22	Baby Don't Cry	2000-03-25	5	87
2000	2 Pac	4:22	Baby Don't Cry	2000-04-01	6	94
2000	2 Pac	4:22	Baby Don't Cry	2000-04-08	7	99
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-02	1	91
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-09	2	87
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-16	3	92
2000	3 Doors Down	3:53	Kryptonite	2000-04-08	1	81
2000	3 Doors Down	3:53	Kryptonite	2000-04-15	2	70
2000	3 Doors Down	3:53	Kryptonite	2000-04-22	3	68
2000	3 Doors Down	3:53	Kryptonite	2000-04-29	4	67
2000	3 Doors Down	3:53	Kryptonite	2000-05-06	5	66

The case (<http://varianceexplained.org/r/trump-tweets/>) is written by David Robinson, author of the book "R for text mining", author of library tidytext, data scientist at StackOverflow.

- Link of github:
https://github.com/dgrtwo/dgrtwo.github.com/blob/master/_R/2016-08-09-trump-tweets.Rmd
- Link of the article: <http://varianceexplained.org/r/trump-tweets/>

作者在該篇文章的結語中寫道

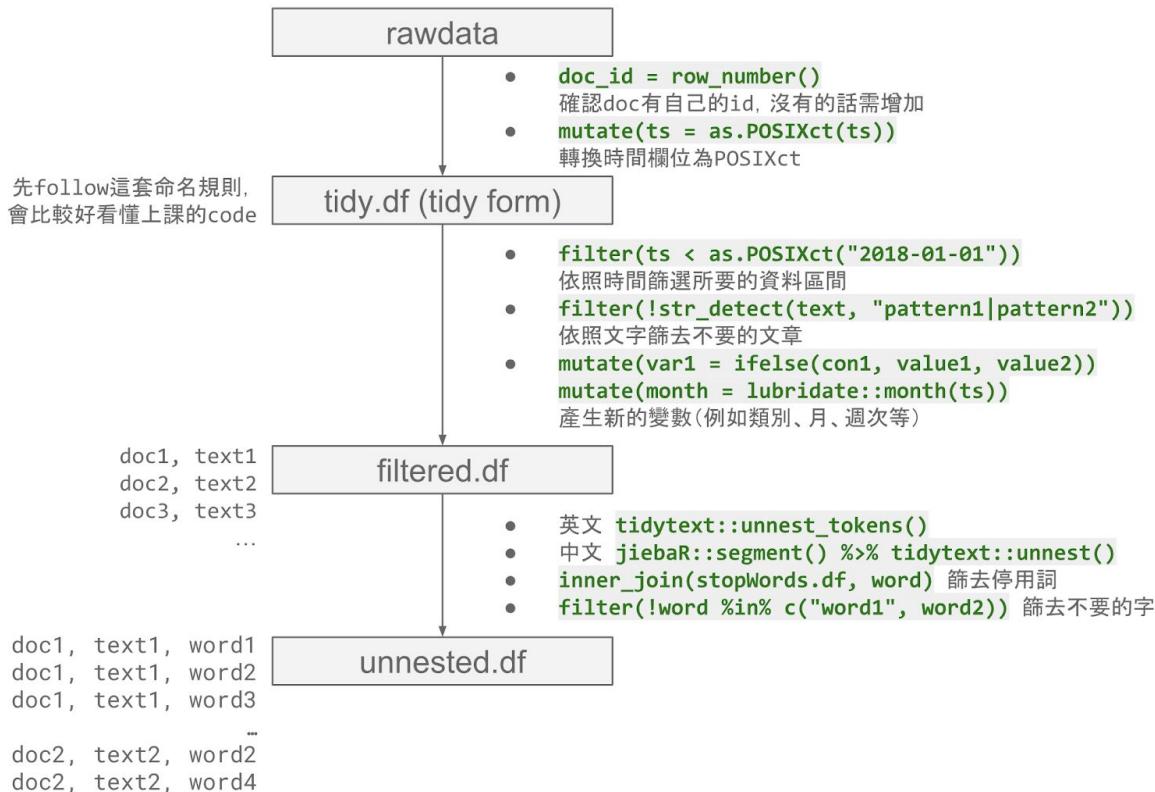
My analysis, shown below, concludes that the Android and iPhone tweets are clearly from different people, posting during different times of day and using hashtags, links, and retweets in distinct ways. What's more, we can see that the Android tweets are angrier and more negative, while the iPhone tweets tend to be benign announcements and pictures. Overall I'd agree with @tvaziri's analysis: this lets us tell the difference between the campaign's tweets (iPhone) and Trump's own (Android).

而在進行分析之前，可以先邀請各位想想，在一則tweet中可以包含哪些「變數項」？

- timestamp -> year, months, week, day, hour, hour+mins
- Actions -> retweet or not, retweeted or not
- wordings -> negative/positive, part-of-speech, ...
- media -> with pic or not, with link or not, ...

- source from -> android, web, or iphone,

在本案例中，實際上進行的是個文字探勘、文字分析的案例，因此我們將經過以下的歷程操作文字資料，正式地說是「將文本合理地拆解為可供運算的單元字詞」，以進行後續的文字運算。因此，在本案例中將經過以下處理文字的流程。

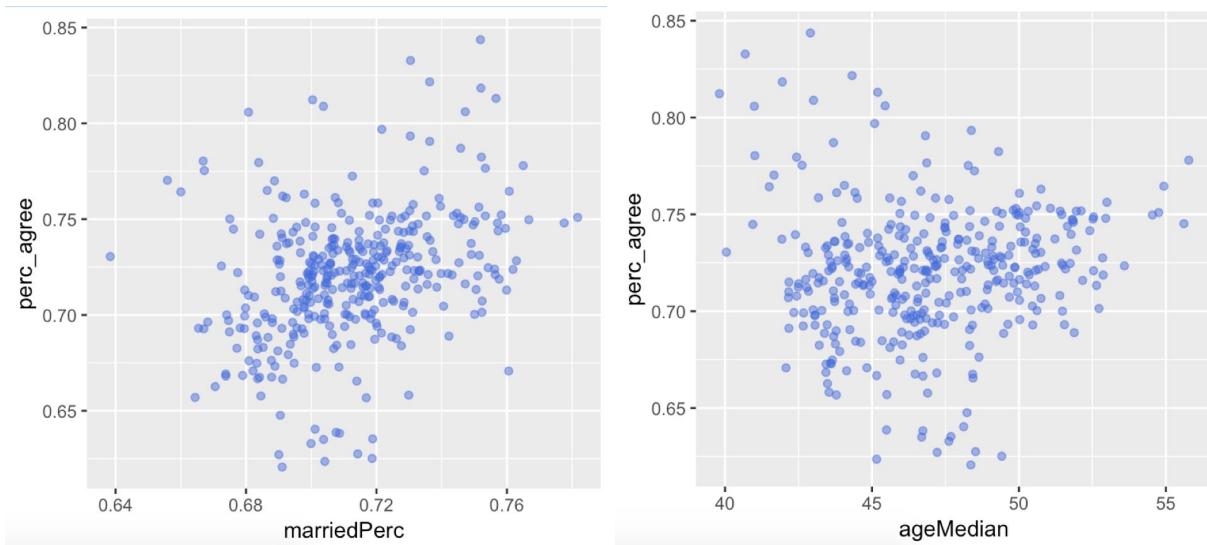


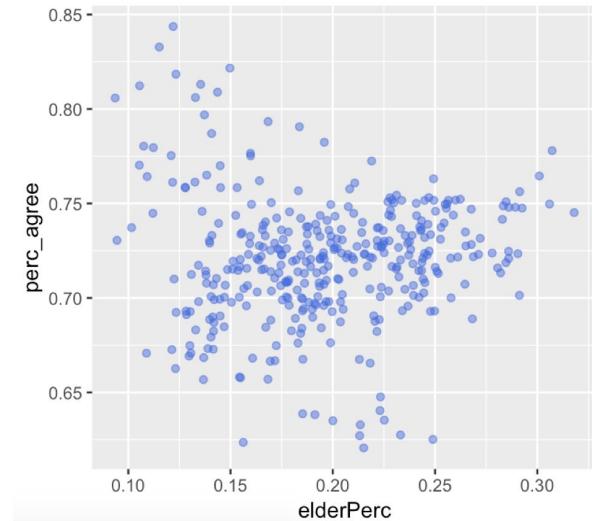
4-1 台灣2018年公投案例分析簡介

以下的例子將以彙整縣市、鄉鎮市、村里等級的性別、年齡分佈，並嘗試與2018年九合一公投與選舉投票結果結合，最後繪製為地圖。

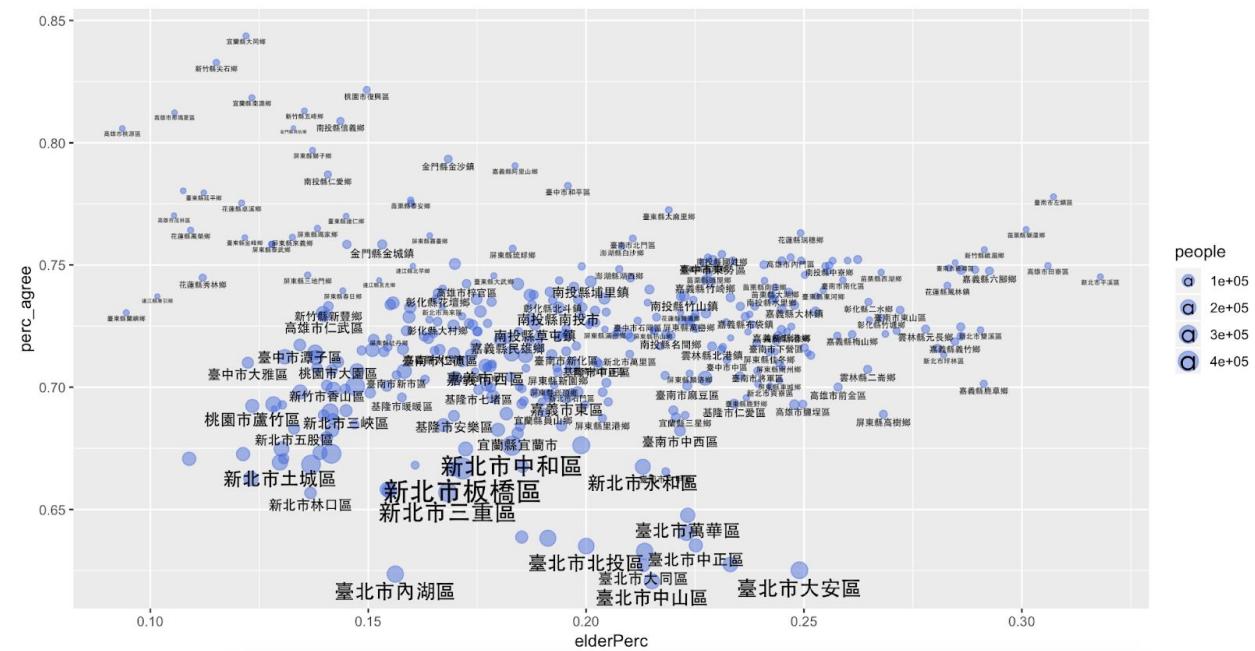


該節的目的仿照「台灣公民投票2018（<https://rfrd-tw.github.io/>）」的方式，產生以下的圖表，來看結婚率、老年人口比例、教育程度、年齡中位數等和公投各案同意比例的相關性（以下尚未補齊教育程度）。





我的版本（圓圈大小為鄉鎮的人數）：大鄉鎮市區是老年人口比例越高，通過第10案的比例也越低，小鄉鎮市區是老年人口比例越高，通過第10案比例越高。



4-1-1 下載資料

內政部各村里教育程度資料：

<https://data.moi.gov.tw/MoiOD/Data/DataDetail.aspx?oid=EC96FA1A-005C-4801-BBCD-5CD246CC9D80>

內政部現住人口性別、年齡、婚姻狀況：15歲以上的年齡以每5年進行統計

<https://data.moi.gov.tw/MoiOD/Data/DataDetail.aspx?oid=5E3DD3A4-3658-4A29-94C2-C3FDD2554493>

內政部遷出遷入統計表

<https://data.moi.gov.tw/MoiOD/Data/DataDetail.aspx?oid=C21E9008-2AC7-4574-BC7D-933E17F4418F>

內政部15歲以上現住人口按性別、年齡、婚姻狀況及教育程度分（已經是tidy form）

<https://data.moi.gov.tw/MoiOD/Data/DataDetail.aspx?oid=4E7FFDCC-17EC-4E5C-9DD7-780C2890AF6B>

各位可下載內政部現住人口性別、年齡婚姻狀況的統計來進行練習。

4-1-2 讀取資料並觀察資料

```
library(tidyverse)
options(stringsAsFactors = F)
raw <- read.csv("twdata/opendata107Y030.csv")
```

資料如下，每一個district_code代表某地區的某個里。例如新北市瑞芳區碩仁里是65000120017，瓜山里是65000120024，前面沒變只有改變最末數據。而每一個欄位，是一個代表單身與否、年齡層、離婚與否變項。

statistic_yyy	district_code	site_id	village	single_age_15down_m	single_age_15_19_m	single_age_20
107	65000120017	新北市瑞芳區	碩仁里	4	0	3
107	65000120024	新北市瑞芳區	瓜山里	7	0	2
107	65000240008	新北市平溪區	望古里	4	0	1
107	65000240012	新北市平溪區	新寮里	6	0	1
107	10002100001	宜蘭縣三星鄉	員山村	3	0	8
107	10002110007	宜蘭縣大同鄉	太平村	0	0	0
107	10008130014	南投縣仁愛鄉	榮興村	0	0	0
107	10015020012	花蓮縣鳳林鎮	森榮里	1	0	2

Tidy型態的資料

我們之前在談資料的「觀察、統計、和二維表格」三種型態時，曾經談到統計型態和二維表格型態間的差異。當時所提到的「統計型態」，也就是每個變項欄恰好是我們所認知的單一變項（如每一個變項欄恰是人口統計變項的年齡、性別、教育程度、數量），會有助於進行統計分析，也就是tidy型態的資料。相較之下，上述的表格是把資料攤成二維的型態，每一個變項是某個年齡層的某種性別的某種婚姻狀況，包含了三個人口統計變項，是方便一般大眾讀的，但不是適合進行統計的tidy型態。

這類的資料tidyverse的相關套件把它稱為tidy form。遵守tidy form形式的資料是，每一個欄恰好一個變項。例如內政部15歲以上現住人口按性別、年齡、婚姻狀況及教育程度分。<https://data.moi.gov.tw/MoiOD/Data/DataDetail.aspx?oid=4E7FFDCC-17EC-4E5C-9DD7-780C2890AF6B>中，每個變數（年齡、婚姻狀況、教育程度、人口數等等）均各自為一個欄上的變項。

```
statistic_yyy,district_code,site_id,village,age,marital_status,edu,sex,population
統計年,區域別代碼,區域別,村里,年齡,婚姻狀況,教育程度,性別,人口數
107,65000010001,新北市板橋區,留侯里,15~19歲,未婚,博畢,男,0
107,65000010001,新北市板橋區,留侯里,15~19歲,未婚,碩畢,男,0
107,65000010001,新北市板橋區,留侯里,15~19歲,未婚,大畢,男,0
107,65000010001,新北市板橋區,留侯里,15~19歲,未婚,專畢,男,0
107,65000010001,新北市板橋區,留侯里,15~19歲,未婚,高中畢,男,10
107,65000010001,新北市板橋區,留侯里,15~19歲,未婚,國中畢,男,25
107,65000010001,新北市板橋區,留侯里,15~19歲,未婚,國小畢以下,男,1
```

4-1-3 分析企劃

1. 建立各鄉鎮市區的老年人口比例
2. 建立各鄉鎮市區的年齡中位數
3. 讀取所有公投案的結果
4. 視覺化年齡與公投結果間的相關性
5. 讀取ibon資料作為鄉鎮市區發展指標
6. 視覺化各鄉鎮市區的ibon數量與公投結果間的相關性
7. 繪製地圖

4-2 用base清理與彙整資料

由於我想要推知，有沒有結過婚、年齡中位數是否會影響2018年公投第10案的同意比例，初步我將以村里資料作為單位資料（意即每一列為一個村里），然後計算每個村里曾結過婚的人口比例、老年人口比例、年齡中位數等來觀察這些指標是否和第10案同意比例有相關性。因此，我希望能夠從前述資料中加總以造出下表，總計出每個里曾結過婚的人口、總人口數、超過65歲的人口數等等，以進行統計分析。

	statistic_yyy	district_code	site_id	village	married	npeople	elder	age0	age15
2	107	65000010001	新北市板橋區	留侯里	920	1649	343	218	70
3	107	65000010002	新北市板橋區	流芳里	826	1564	333	227	73
4	107	65000010003	新北市板橋區	赤松里	463	840	170	133	42
5	107	65000010004	新北市板橋區	黃石里	632	1136	238	203	55
6	107	65000010005	新北市板橋區	挹秀里	1061	1904	398	273	77

Step 1. 去除第一行CSV的標題列，另外保留前四個文字變項

```
raw <- read.csv("twdata/opendata107Y030.csv")
df1 <- raw[-1,] # delete the 1st row
df.stat <- df1[,1:4] # select 1 to 4 columns
```

Step 2. 轉變多個變項的資料欄位 <- Very important

在進入資料彙整運算前，要先注意第五欄到最後一欄的資料都是文字型態，所以如果要計算各鄉鎮的總人口數，或曾結婚的人口數之前，必須要先把文字資料轉為數值型態。所以我必須要把第五欄到最後一欄都轉為數值型態 (`ncol(df1)`為156)。過去若要把一個變項轉為數值的方法是用

```
df1$single_age_15down_m <- as.numeric(df1$single_age_15down_m)
```

但這只是一個變項，我現在有152個變項這樣一個一個寫還得了？一定有解決之道，不然就不用寫程式了，我用Google sheet或Excel拉還比較快。此時，我要把「每一變項欄」

轉為數值，我可以使用`apply()`這個函式。`apply()`這個函式是用來「將某幾列或某幾行套用後方的函式」。此時，我設定參數`MARGIN=2`代表要針對每一個變項欄操作，若`MARGIN=1`則代表要針對每一列資料操作。`function(x) as.numeric(x)`代表要把每一欄或者每一列當成這個`x`，然後進行計算，在此由於`MARGIN=2`所以是把每一欄當成`x`，然後用`as.numeric(x)`把`x`轉變為數值型態。轉完後資料由於都是數值被轉為`matrix`，所以用`as.data.frame()`將其轉回`data.frame`。

```
m1 <- apply(df1[, 5:156], MARGIN = 2, function(x) as.numeric(x))
df2 <- as.data.frame(m1)
?apply
```

Step 3. 計算各村里的人口數

接下來我打算計算各鄉鎮市區的人口數。一樣用`apply()`，但這次我要統計的是每一列的總和，所以`MARGIN=1`。因為`df2`和`df.stat`都是從原本`df1`取出的變項欄，所以每一列的資料應該相等，所以在`df2`做完運算後，直接assign給`df.stat`作為新的變項。

```
df.stat$npeople <- apply(df2, MARGIN=1, function(x) sum(x))
```

Step 4. 計算各村里曾結婚過的人口總數

請自己仔細觀察資料`df2`的資料，概況如下。由於我們要取的是曾經結過婚的人口數（好看看是不是結過婚會比較支持2018年公投案的第十案），所以是先取39:152欄，然後像前一個步驟一樣，針對列來進行加總後，assign給`df.stat`作為新的變項。

- 1:19欄為19個年齡層的單身男性 (`single_m`) 、
- 20:38欄為19個年齡層的單身女性 (`single_f`) 、
- 39:57欄為19個年齡層的已婚男性 (`married_m`) 、
- 58:76欄為19個年齡層的已婚女性 (`married_f`) 、
- 77:95欄為19個年齡層的離婚男性 (`divorced_m`) 、
- 96:114欄為19個年齡層的離婚女性 (`divorced_f`) 、
- 115:133欄為19個年齡層的喪偶男性 (`widowed_m`) 、
- 134:152欄為19個年齡層的喪偶女性 (`widowed_f`)

```
df.stat$married <- apply(df2[, 39:152], 1, function(x) sum(x))
```

Step 5. 計算超過六十五歲的高齡人口數

從上述的資料分布，我們可以知道高齡人口數是12:19欄、31:38欄等，所以我先一一選取出df2中的這些變項欄，然後一樣用上述的加總方法來進行加總。

```
df.stat$elder <- apply(df2[,c(12:19, 31:38, 50:57, 69:76, 89:95,
97:114, 126:133, 145:152)], 1, function(x) sum(x))
```

Step 6. 計算比例

除以總人口數以計算比例

```
df.stat$elderPerc <- df.stat$elder / df.stat$npeople
df.stat$marriedPerc <- df.stat$married / df.stat$npeople
```

Step 7. (cont.)以下我們將改用dplyr來處理資料

4-3 用dplyr清理並產生tidy form資料

以下將用dplyr的函式來處理上述的資料。

Step 1. 產生村里的全名

我先使用`slice(-1)`減去第一行中文欄位名稱。再來，目前縣市鄉鎮市區和村里分別是兩個變項，由於不同的鄉鎮市可能會有相同的村里名，所以我把`site_id`與`village`粘接起來成為完整的村里名`vname`。

這邊我多加了一行程式碼讓`vname`可以排到前面一點的變項欄，我把它放在`statistic_yyy`變項後面，可以用`select()`達到這個目的，我之後的變項欄的還要寶劉，所以我多打一個`everything()`就可以把剩下的變項欄都擺放在後面。因此這個重排變項欄的完整程式碼為`select(statistic_yyy, vname, everything())`。

我最後將資料pipe給View()來看一下結果，之後在進行運算時記得取消掉，不然每次都會打開View的視窗，無法繼續往下執行。

```
raw <- read.csv("twdata/opendata107Y030.csv") %>%
  slice(-1) %>%
  mutate(vname = paste0(site_id, village)) %>%
  select(statistic_yyy, vname, everything()) %>% View
```

Step 2. 用gather()收合變數產生tidy form資料

接下來，我要把表格型態的資料轉為tidy型態資料。原本的資料是這樣的型態。

district_code	site_id	village	single_age_15down_m	single_age_15_19_m	single_age_20_24_m	single_age_25_29_m
10008080008	南投縣中寮鄉	中寮村	35	28	19	18
10008080009	南投縣中寮鄉	廣福村	27	19	17	6
10008080010	南投縣中寮鄉	永福村	58	54	53	46
10008080011	南投縣中寮鄉	義和村	58	42	57	34
10008080012	南投縣中寮鄉	清水村	40	24	42	22
10008080013	南投縣中寮鄉	龍安村	33	19	20	24
10008080014	南投縣中寮鄉	龍岩村	28	24	12	20
10008080015	南投縣中寮鄉	永和村	17	12	11	17

我要將後方的數值變項欄（single_age_15down_m等）轉為單一變項key的值，再把其所對應到的資料值，也轉為單一變項value。請注意看上圖和程式碼後方結果圖的顏色區塊。南投縣中寮鄉中寮村（綠色）被複製且展開為多列。而原本多個年齡層和資料的變數項（紅色）變成一個變項欄的資料，分別對應到其原本對應的數值（藍色）。

相對於spread()把變項展開成欄，gather()函式可以收合被展開的變項，在此將要收合的變數名稱統一稱為key，並將該變數所對應到的數值稱為value。並且我用6:ncol(.)來指定我要收合哪些變項欄。ncol(.)的「.」代表從前面%>% pipe進來的那個data.frame。gather()後資料列從7760增加至1,179,520。（灰底部分用來觀察結果用）

```
tidy_data <- raw %>%
  gather("key", "value", 6:ncol(.)) %>%
  arrange(vname) %>% head(20) %>% View
```

	statistic_yyy	district_code	site_id	village	key	value	vname
1	107	10008080008	南投縣中寮鄉	中寮村	single_age_15down_m	35	南投縣中寮鄉中寮村
2	107	10008080008	南投縣中寮鄉	中寮村	single_age_15_19_m	28	南投縣中寮鄉中寮村
3	107	10008080008	南投縣中寮鄉	中寮村	single_age_20_24_m	19	南投縣中寮鄉中寮村
4	107	10008080008	南投縣中寮鄉	中寮村	single_age_25_29_m	18	南投縣中寮鄉中寮村
5	107	10008080008	南投縣中寮鄉	中寮村	single_age_30_34_m	8	南投縣中寮鄉中寮村
6	107	10008080008	南投縣中寮鄉	中寮村	single_age_35_39_m	16	南投縣中寮鄉中寮村
7	107	10008080008	南投縣中寮鄉	中寮村	single_age_40_44_m	9	南投縣中寮鄉中寮村
8	107	10008080008	南投縣中寮鄉	中寮村	single_age_45_49_m	7	南投縣中寮鄉中寮村
9	107	10008080008	南投縣中寮鄉	中寮村	single_age_50_54_m	6	南投縣中寮鄉中寮村
10	107	10008080008	南投縣中寮鄉	中寮村	single_age_55_59_m	5	南投縣中寮鄉中寮村

Step 3. 切割key欄位為數個變數

由於每一列恰好是一種婚姻狀態、一個年齡層和一個性別，所以，我們可以把key中的婚姻狀態、年齡層和性別切割出來做為變數。觀察key欄位發現其格式有一些規律性，主要是婚姻狀態_年齡下界_年齡上界_性別的形式。標準的範例如married_15_10_m或widowed_25_29_f，但有一些並非這種形式，例如：

- single_age_15_19_m：其中single_age之間多了一個底線，所以把single_age取代為single就好。
- married_15down_m：因為是15down少了一個底線，所以取代為0_14。
- Married_100up_f：因為100up少了一個底線，所以取代為100_105。

之後，我使用tidyverse::separate()函式將key切成四個變項，分別為married、ageLower、ageUpper、gender。separate()有一個參數是remove=T（預設值），意思是說，當把key變項切割為四個變項後，預設把key變項給丟棄；但如果未來你還會用到key變項的話，你可以把remove改為FALSE，代表切割完後，還保留key變項。

- tidyverse::separate() : Given either regular expression or a vector of character positions, separate() turns a single character column into multiple columns.

此時我清理出來的資料大致如下：

	statistic_yyy	vname	district_code	site_id	village	married	ageLower	ageUpper	gender	value
1	107	新北市板橋區留侯里	65000010001	新北市板橋區	留侯里	single	0	14	m	118
2	107	新北市板橋區流芳里	65000010002	新北市板橋區	流芳里	single	0	14	m	119
3	107	新北市板橋區赤松里	65000010003	新北市板橋區	赤松里	single	0	14	m	60
4	107	新北市板橋區黃石里	65000010004	新北市板橋區	黃石里	single	0	14	m	113
5	107	新北市板橋區挹秀里	65000010005	新北市板橋區	挹秀里	single	0	14	m	123
6	107	新北市板橋區湧興里	65000010006	新北市板橋區	湧興里	single	0	14	m	351
7	107	新北市板橋區新興里	65000010007	新北市板橋區	新興里	single	0	14	m	169
8	107	新北市板橋區社後里	65000010008	新北市板橋區	社後里	single	0	14	m	318
9	107	新北市板橋區香社里	65000010009	新北市板橋區	香社里	single	0	14	m	248
10	107	新北市板橋區中正里	65000010010	新北市板橋區	中正里	single	0	14	m	313

Step 4. 轉換資料為數值型態並建立65歲以上的變項指標

最後就剩零星的操作，包含轉換資料為數值型態、或者你也可以在這裡建立新的指標（例如年齡平均）。最後加上一個arrange(vname)讓他按照村里的全名排序。

```
tidy_data <- raw %>%
  gather("key", "value", 6:ncol(.)) %>%
  mutate(key = str_replace(key, "15down", "0_14")) %>%
  mutate(key = str_replace(key, "100up", "100_105")) %>%
  mutate(key = str_replace(key, "single_age", "single")) %>%
  separate(key, c("married", "ageLower", "ageUpper", "gender")) %>%
  mutate(ageLower = as.numeric(ageLower),
        ageUpper = as.numeric(ageUpper),
        value = as.numeric(value))
  ) %>%
  arrange(vname)
```

4-4 建立鄉鎮市區與村里指標

4-4-1 使用group_by()建立村里指標

將資料轉換為tidy型態後，接下來要做的事情是建立村里、鄉鎮市區、縣市的分級指標。針對每個村里，我希望計算出總人口數people（原本依據年齡與性別、婚姻情形分

割)、老年人總數`elderSum`、曾結婚人口總數`marriedSum`。之後再分別除以該村里的總人口數`people`，老年人的人口比例`elderPerc`以及結婚的人口比例`marriedPerc`。

因為一個村里的資料會根據不同性別、不同婚姻情形、不同年齡層被切割為不同的資料列，共 $2 \times 4 \times 19$ 個資料列。因此，如果我想知道一個村里的總人口數或相關統計資料，就不需彙整這些資料列。`dplyr`有非常強大的`group_by()`可以根據群組來進行運算，我用村里代號（`district_code`）來做群組運算，所以是`group_by(district_code)`或用我們所產生的`vname`作為群組基準來運算`group_by(vname)`。

語法上，通常`group_by()`之後經常會跟著`summarise()`，跟`mutate()`的語法有點像，都會產生新變數，但因為這邊用`group_by()`針對某個或某幾個變數做彙整，相當於`base`套件的`apply()`函式，因此會根據每個不同的群組做組內的數值彙整運算。比方說，在以下的程式碼中，我用`sum(num)`計算了該群組內的總人數，然後同樣累計了年齡大於等於65歲的總人數，以及婚姻狀態不為`single`的總人數。

簡單地說，相當於按照不同的村里（`district_code`）各別做`value`的加總（該村里的總人口數）、篩選出年齡65歲以上的人口組別進行加總、篩選出不是單身者的人口組別進行加總。

```
...
  group_by(district_code) %>%
  summarise(
    people = sum(value),
    elderSum = sum(value[ageLower >= 65]),
    marriedSum = sum(value[!married %in% ("single")])
  ) %>%
...

```

之後會加一個`ungroup()`解開群組的設定（就像你在powerpoint中群組化圖像物件一樣，若你要個別使用的話，要解開群組）。

最後，算完每個村里的老年人口總數和曾結過婚的總數，用`mutate()`去除以總人數，就可以知道每個村里的老年人口比例和曾結過婚的比例。

```
village_stat <- tidy_data %>%
  filter(ageLower >= 20) %>%
  group_by(district_code) %>%
  summarise(
    people = sum(num),
```

```

elderSum = sum(num[ageLower >= 65]),
marriedSum = sum(num[!married %in% ("single")])
) %>%
ungroup() %>%
mutate(elderPerc = elderSum / people,
       marriedPerc = marriedSum / people)

```

我通常會將原本資料raw中的詮釋資料（中文名、縣市名、鄉鎮市名等）再根據district_code為索引left_join回來如下。

- `join()`系列的函式都為了要合併某兩個data.frame，而且這兩個data.frame一定有共通的一個變項。和`bind_cols()`不同，`bind_cols()`只要求列的數目要一樣。`join()`是參考資料庫的設計，兩個data.frame如果列的資料順序不同，他可以依照某一個data.frame的變項作為索引抽取另一個data.frame的變項來做合併。
- `left_join()`代表根據左邊的data.frame（也就是pipeline下來的data.frame）的項目來做合併，所以`left_join()`的結果項目會和原本pipe下來的data.frame列的項目與順序相同。根據左邊的data.frame來做合併的話，因為有指定索引變項，所以右邊的data.frame會依照左邊的索引變項的出現順序重排序以結合。
- 注意到我`left_join()`的對象是`raw %>% select(statistic_yyy, district_code, vname, site_id, village)`，也就是說我可以在式子中挑選我要的那幾個變項後再來做合併。
- `by="district_code"`代表根據兩個data.frame的`district_code`為索引來合併。

```

village_stat <- village_stat %>%
  left_join(raw %>% select(statistic_yyy, district_code, vname,
site_id, village),
            by = "district_code")

```

4-4-2 建立鄉鎮市區指標

剛剛是根據村里（village）來建立指標，現在要根據鄉鎮市區來建立指標。走過前方的邏輯後，我們只需要把原本用來做`group_by()`的村里變項`district_code`改為鄉鎮市區的變項`site_id`，就可以完成這件事，其他都一樣，你發現沒？

```

town_stat <- tidy_data %>%
  filter(ageLower >= 20) %>%
  group_by(site_id) %>%

```

```

summarise(
  people = sum(value),
  elderSum = sum(value[ageLower >= 65]),
  marriedSum = sum(value[!married %in% ("single")])
) %>%
ungroup() %>%
mutate(elderPerc = elderSum / people,
      marriedPerc = marriedSum / people)

```

4-4-3 計算組距資料的中位數

參考中位數的計算公式

- <https://www.mathsisfun.com/data/frequency-grouped-mean-median-mode.html>
- <https://stackoverflow.com/questions/18887382/how-to-calculate-the-median-on-grouped-dataset>

Step 1. 建立鄉鎮市區之於年齡的二階層tidy型態資料

目前的資料如下,

1	statistic_yyy	vname	district_code	site_id	village	key	married	value	sex	ageRange	ageLower
1	107	南投縣中寮鄉中寮村	10008080008	南投縣中寮鄉	中寮村	single_age_0_14_m	single	35	m	0_14	0
2	107	南投縣中寮鄉中寮村	10008080008	南投縣中寮鄉	中寮村	single_age_15_19_m	single	28	m	15_19	15
3	107	南投縣中寮鄉中寮村	10008080008	南投縣中寮鄉	中寮村	single_age_20_24_m	single	19	m	20_24	20
4	107	南投縣中寮鄉中寮村	10008080008	南投縣中寮鄉	中寮村	single_age_25_29_m	single	18	m	25_29	25
5	107	南投縣中寮鄉中寮村	10008080008	南投縣中寮鄉	中寮村	single_age_30_34_m	single	8	m	30_34	30
6	107	南投縣中寮鄉中寮村	10008080008	南投縣中寮鄉	中寮村	single_age_35_39_m	single	16	m	35_39	35
7	107	南投縣中寮鄉中寮村	10008080008	南投縣中寮鄉	中寮村	single_age_40_44_m	single	9	m	40_44	40
8	107	南投縣中寮鄉中寮村	10008080008	南投縣中寮鄉	中寮村	single_age_45_49_m	single	7	m	45_49	45
9	107	南投縣中寮鄉中寮村	10008080008	南投縣中寮鄉	中寮村	single_age_50_54_m	single	6	m	50_54	50
10	107	南投縣中寮鄉中寮村	10008080008	南投縣中寮鄉	中寮村	single_age_55_59_m	single	5	m	55_59	55
11	107	南投縣中寮鄉中寮村	10008080008	南投縣中寮鄉	中寮村	single_age_60_64_m	single	1	m	60_64	60

因為我要計算的是根據每鄉鎮市區的年齡中位數，但是我目前有的是各村里每個年齡層的人口數，而且還區分婚姻狀況和性別，所以我要先把每鄉鎮市區的每個年齡層的人口數給加總起來。希望能夠產生以下的資料：

- `ageLower`是每組的下界；
- `value`是該鄉鎮市區中該組年齡的不同性別、婚姻狀況、不同里的總和；

- Cf 則是 value 的累加 (cumulative sum)。

	site_id	ageLower	value	cf
1	南投縣中寮鄉	15	780	780
2	南投縣中寮鄉	20	892	1672
3	南投縣中寮鄉	25	879	2551
4	南投縣中寮鄉	30	808	3359
5	南投縣中寮鄉	35	987	4346
6	南投縣中寮鄉	40	932	5278

底下我一共 group_by() 兩次，第一次是為了加總出每個鄉鎮市區的人口數有多少人，第二次的 group_by() 是為了讓每個鄉鎮市區依照年齡由小到大重新排序，並計算累加結果（為了求中位數）。

- 第一次 group_by()：因為 site_id 和 age_lower 都可以視為類別，所以我一次 group_by() 兩個變項，好計算不同鄉鎮市區的不同年齡層總數有多少人。然後我根據不同組來加總 value，並 assign 給一個新變項 value。
- 兩次 group_by() 中間我篩除了年齡小於 15 歲的組。
- 第二次 group_by()：因為我累加來計算中位數要每個鄉鎮市區都各自算，所以我 group_by(site_id)，然後用 arrange(ageLower) 根據 ageLower 由小到大做排序。再用 mutate() 搭配 cumsum(value) 產生 value 的累加結果。
- 最後我再用 arrange(site_id, ageLower) 就鄉鎮市區和年齡重新做排序。

```
town_age <- tidy_data %>%
  group_by(site_id, ageLower) %>%
  summarize(value = sum(value)) %>%
  ungroup() %>%
  filter(ageLower >= 15) %>%
  group_by(site_id) %>%
  arrange(ageLower) %>%
  mutate(cf = cumsum(value)) %>%
  ungroup() %>%
  arrange(site_id, ageLower)
```

Step 2. 計算組距固定下的年齡中位數

觀察 town_age 如下，假設中位數是發生在第七列也就是 ageLower 為 45 的那組，那也就是該鄉的總人口數除以二 ($\max(cf)/2$) 必須要大於 40 那組的 cf 5278，然後小於 45 那組的

`cf` 6377。但是運算式很難處理「前一列」，所幸這就是累加的結果，只要把每一列的 `cf-value` 就可以得到40那組的 `cf`。所以判斷式可以寫為

`(cf-value) <= max(cf)/2 & max(cf)/2 <= cf`。且一定只有一個組會符合條件，因此 `ageLower[(cf-value) <= max(cf)/2 & cf >= max(cf)/2]` 就會是中位數那組的下界，也就是 `L`。

	site_id	ageLower	value	cf
1	南投縣中寮鄉	15	780	780
2	南投縣中寮鄉	20	892	1672
3	南投縣中寮鄉	25	879	2551
4	南投縣中寮鄉	30	808	3359
5	南投縣中寮鄉	35	987	4346
6	南投縣中寮鄉	40	932	5278
7	南投縣中寮鄉	45	1099	6377
8	南投縣中寮鄉	50	1399	7776

公式為：`Estimated Median = L + ((n/2) - B)/G * W`

- `L`為中位數那組的下界
- `n`為總人口數（各組的總和），所以 `n = max(cf)`。
- `B`為中位數那組之前的人口總和（前面組別的總和）所以 `B = cf - value`。
- `G`為中位數那組的人口數，所以就是 `left_join()` 以後的 `value`。因為 `left_join()` 只 `join` 了包含每個鄉鎮市區包含中位數那組。
- `W`為組距（這份資料固定為5）

在 `left_join()` 之前每個鄉鎮市區都只會留下中位數那組，但這時候沒有該組 `value` 和 `cf`，就必須用 `left_join(town_age, by = c("site_id" = "site_id", "L" = "ageLower"))` 重新找回該組的人口和累積人口數（如果這點你聽得懂，代表你對於資料的操作和抽象思考能力已經不錯了）。在這之後才有辦法計算 `B` 和 `G`。也就是在 `left_join()` 之前是在找到中位數的組別，在 `left_join()` 之後才是計算中位數。

```
town_age_median <- town_age %>%
  group_by(site_id) %>%
  summarize(L = ageLower[(cf - value) <= max(cf)/2 & cf >= max(cf)/2],
            n = max(cf)
            ) %>%
  ungroup() %>%
  left_join(town_age, by = c("site_id" = "site_id", "L" = "ageLower"))
```

```
%>%
  mutate(B = cf - value, G = value, w = 5) %>%
  mutate(ageMedian = L + (n/2-B)/G*w)
```

Step 3. 合併新產生的年齡中位數

最後，我把前面算出來的`town_stat`結合我剛剛算出來的每個鄉鎮市區的中位數。

```
town_stat <- town_stat %>% left_join(town_age_median %>%
  select(site_id, ageMedian))
```

4-5 讀取公投資資料

4-5-1 讀取資料並重新命名

```
ref10 <- read_csv("twdata/referendum_byTown/第10案：你是否同意民法婚姻  
規定應限定在一男一女的結合?.csv")
names(ref10) <- c("no", "county", "town", "n_agree", "n_disagree",
  "n_valid", "n_invalid", "n_ticket", "n_people", "perc_ticket",
  "perc_agree", "perc_disagree", "agree_disagree")
```

4-5-2 產生同意票的比例

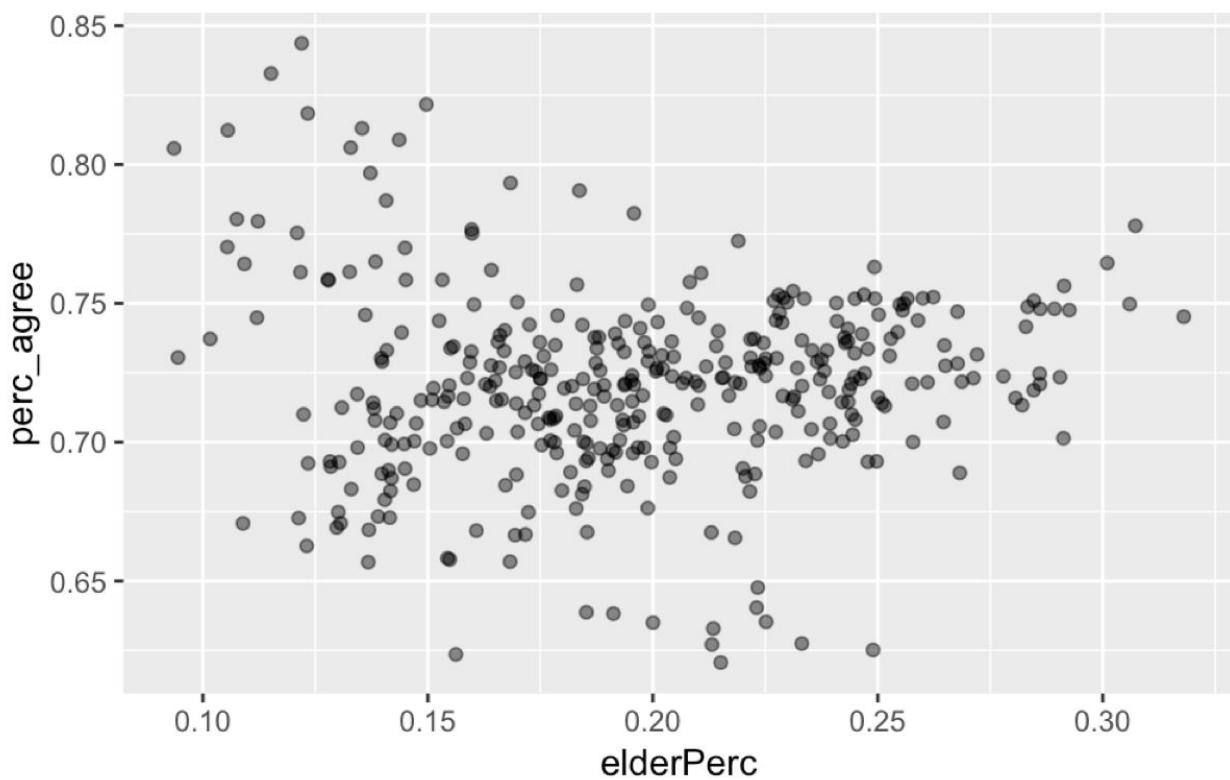
```
town_ref10 <- ref10 %>%
  filter(!is.na(town)) %>%
  mutate(site_id = paste0(county, town)) %>%
  mutate(perc_ticket = n_ticket / n_people,
    perc_agree = n_agree / n_ticket,
    perc_disagree = 1 - perc_agree)
```

鄉鎮市區資料和公投資資料結合後再繪圖

- `str_replace()`只會取代第一個找到的pattern，如果要取代所有字串內的空白，要用`str_replace_all()`，這邊是把多餘的空白 " " 通通取代為空的字串 ""。
- `aes()`用以指定x軸與y軸的變項。

3. `geom_jitter()`功能近似`geom_point()`，但它會將重疊的點加一點隨機參數讓他看起來不是完全重疊而能便於觀察。

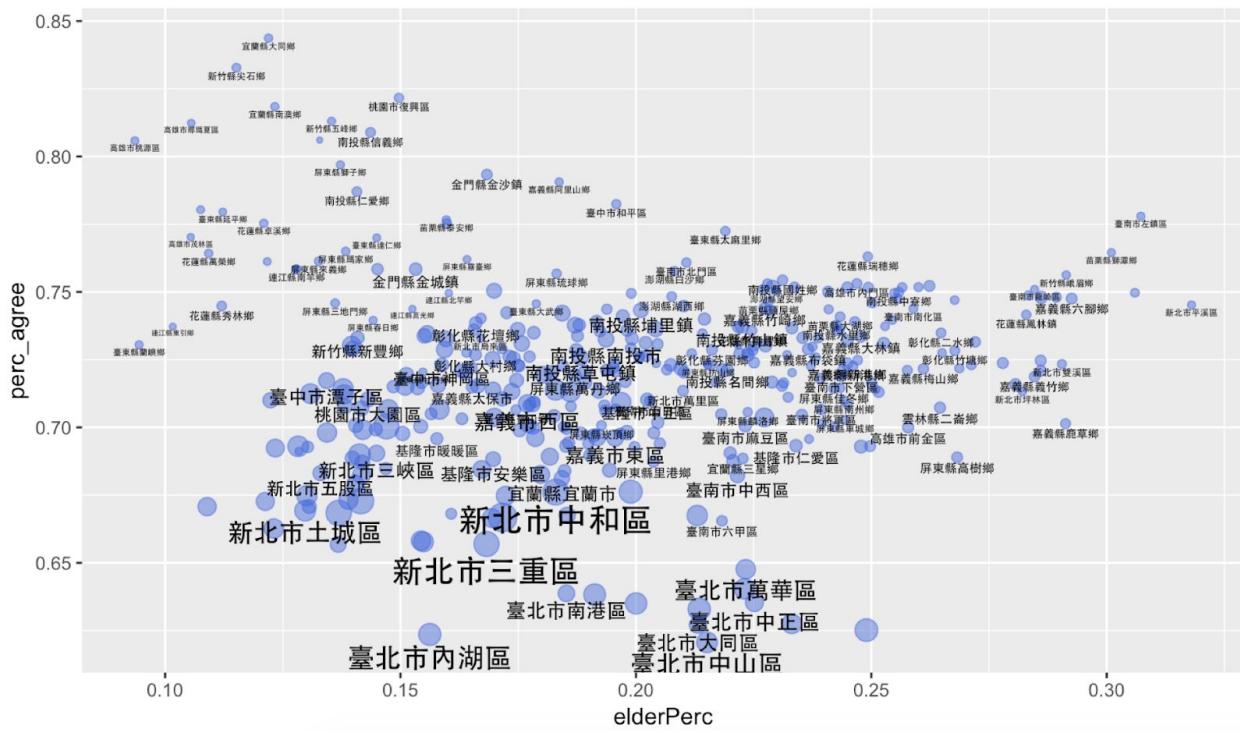
```
town_stat %>%
  mutate(site_id = str_replace_all(site_id, " ", ""))
  left_join(town_ref10) %>%
  ggplot() +
  aes(elderPerc, perc_agree) +
  geom_jitter(alpha = 0.5)
```



4-5-3 視覺化：加入人口數，並描繪為點的大小

1. 在典型的`geom_point()`中，`aes()`除了X/Y軸的變項外還可以塞入其他的變項，最常見的為`size`和`color`。所以最多可以餵給他四個變項。
2. 我可以用`geom_text()`來標上每個點的文字。參數`check_overlap`可避免文字重疊，但會讓一些文字消失。`vjust`是調整文字相對於點的垂直高低，`family`必須指定中文字型，否則`ggplot()`預設畫不出中文字。

```
town_stat %>%
  mutate(site_id = str_replace_all(site_id, " ", "")) %>%
  left_join(town_ref10) %>%
  ggplot() +
  aes(elderPerc, perc_agree, size = people) +
  geom_jitter(alpha = 0.5, color = "royalblue") +
  geom_text(aes(label = site_id), check_overlap = TRUE, vjust =
  1.5, family="黑體-繁 中黑")
```



4-5-4 讀取所有公投資料並存檔

```
files <- list.files("twdata/referendum_byTown/", full.names = T)
referendum_town <- data.frame()
for(f in files){
    ref <- read_csv(f)
    names(ref) <- c("refno", "county", "town", "n_agree",
"n_disagree", "n_valid", "n_invalid", "n_ticket", "n_people",
```

```
"perc_ticket", "perc_agree", "perc_disagree", "agree_disagree")

town_ref <- ref %>%
  filter(!is.na(town)) %>%
  select(refno, county, town, n_agree, n_disagree, n_valid,
n_invalid, n_ticket, n_people) %>%
  mutate(townfull = paste0(county, town)) %>%
  mutate(perc_ticket = n_ticket / n_people,
    perc_agree = n_agree / n_ticket,
    perc_disagree = 1 - perc_agree)

referendum_town <- bind_rows(referendum_town, town_ref)
}
```

refno	county	town	n_agree	n_disagree	n_valid	n_invalid	n_ticket	n_people	townfull
column 0: rownames									
1 第10案	連江縣	南竿鄉	2485	691	3176	101	3277	6461	連江縣南竿
2 第10案	連江縣	北竿鄉	832	234	1066	44	1110	2125	連江縣北竿
3 第10案	連江縣	莒光鄉	592	182	774	22	796	1481	連江縣莒光
4 第10案	連江縣	東引鄉	432	142	574	12	586	1138	連江縣東引
5 第10案	金門縣	金城鎮	10065	2610	12675	596	13271	37815	金門縣金城
6 第10案	金門縣	金沙鎮	5217	1043	6260	316	6576	18560	金門縣金沙

儲存為rda檔

我想把案子的名稱從中文改為ref7到ref16，避免日後要把每個案子獨立為變項時，變相名稱會變成中文。所以我取出「第」和「案」之間的數字，並把他填到ref後面。最後我把這整份資料pipe給`save()`儲存。

```
referendum_town %>%
  mutate(refno = str_replace(refno, "第([0-9]+)案", "ref\\1")) %>%
  save(file = "twdata/referendum_town.rda")
```

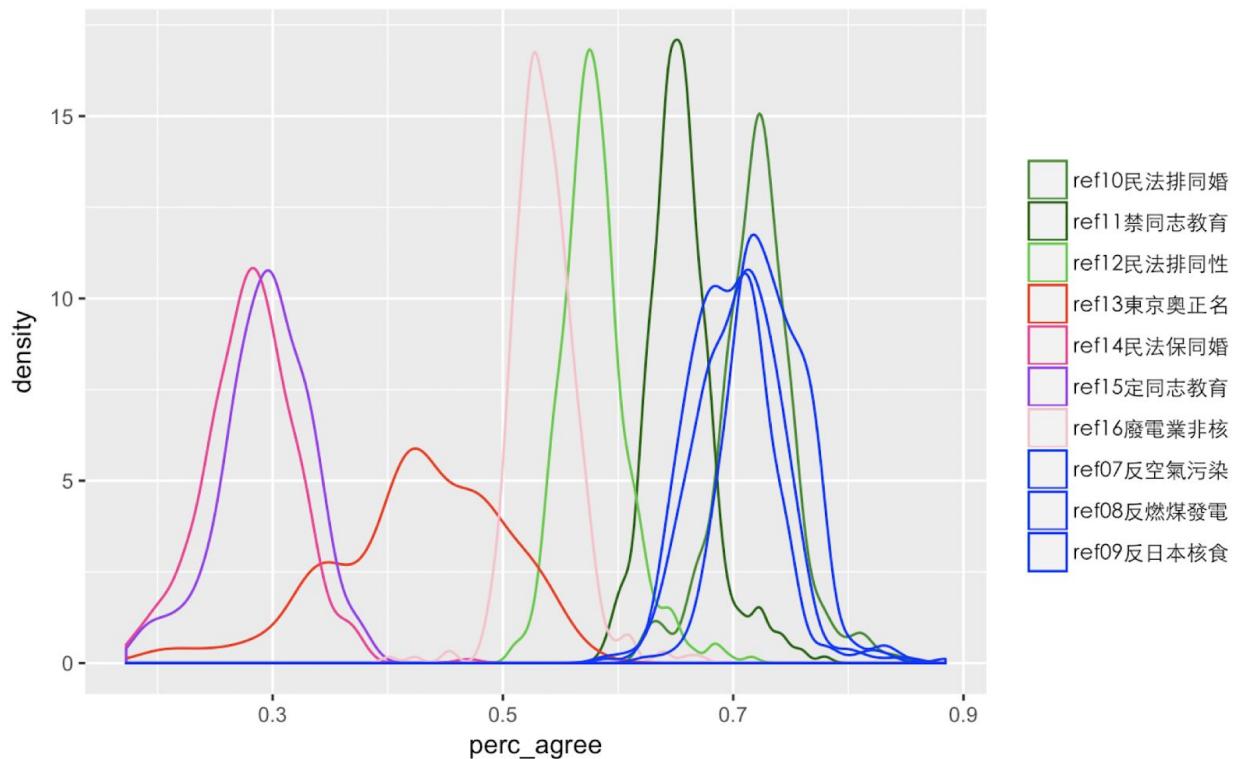
4-5-5 視覺化各案公投比例

```
ref_labels <- c("ref10民法排同婚", "ref11禁同志教育", "ref12民法排同性",
"ref13東京奧正名", "ref14民法保同婚", "ref15定同志教育", "ref16廢電業非核",
"ref07反空氣污染", "ref08反燃煤發電", "ref09反日本核食")
referendum_town %>%
  mutate(refno = str_replace(refno, "第([0-9]+)案", "ref\\1")) %>%
```

```

select(countytown, refno, perc_agree) %>%
ggplot() +
aes(perc_agree, color = refno) +
geom_density() +
scale_color_manual(name = "", labels = ref_labels,
values = c("ForestGreen", "DarkGreen",
          "LimeGreen", "red",
          "DeepPink ", "purple", "pink",
          "#0000FF", "#0000FF", "#0000FF")) +
theme(legend.text = element_text(family="Heiti TC Light"))

```



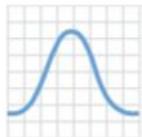
4-6 ggplot2整理

單一連續變數

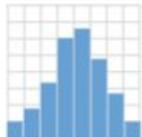
- 鄉鎮市區的公投第10案同意比例

一般來說，我會用`geom_density()`來做快速觀察單一變項的分佈。我比較少用`geom_histogram()`，因為他的`binwidth=5`需要指定要有幾個`bin`（也就是要有幾個長條）。

注意。如果我想要兩組資料有不同的`geom_density()`卻畫在同一張圖上，此時，我要用`color`來做分組，只要把分組變項指定給`color`，就會自動進行分組。如果我是用`geom_histogram()`或`geom_col()`的話，那我就要用`fill`來做分組，而不是`color`。因為`color`對長條圖來說只是框線而已。依此類推，`geom_jitter()`與`geom_point()`是用`color`來做分組，而非`fill`。



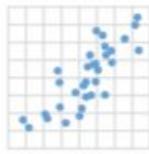
c + geom_density(kernel = "gaussian")
x, y, alpha, color, fill, group, linetype, size, weight



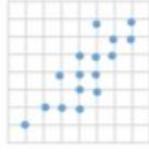
c + geom_histogram(binwidth = 5)
x, y, alpha, color, fill, linetype, size, weight

X和Y均為連續變數

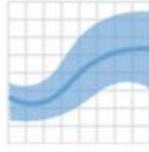
- 各鄉鎮市區的年齡中位數和2018年公投第10案的關係



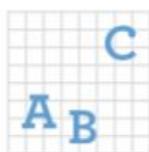
e + geom_jitter(height = 2, width = 2)
x, y, alpha, color, fill, shape, size



e + geom_point(), x, y, alpha, color, fill, shape, size, stroke



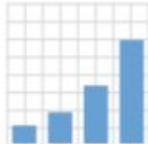
e + geom_smooth(method = lm), x, y, alpha, color, fill, group, linetype, size, weight



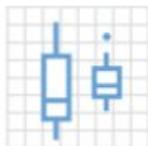
e + geom_text(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE), x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

X為離散、Y為連續變項。

- 每個字的出現頻率
- 各組的平均數、四分位數。



f + geom_col(), x, y, alpha, color, fill, group, linetype, size



f + geom_boxplot(), x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight

4-7 應用 - 網民行為分析

網軍的故事

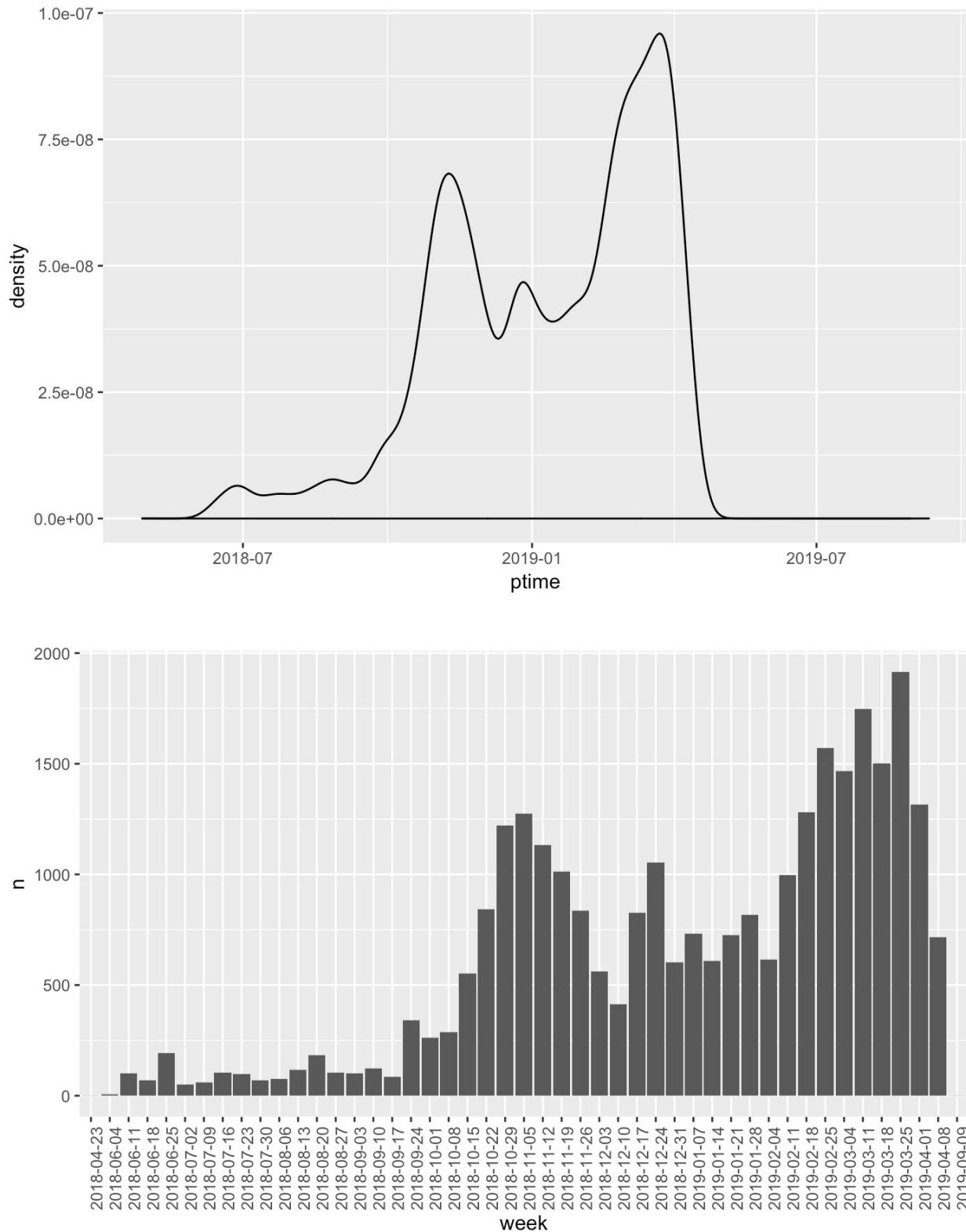
- <https://disp.cc/b/163-aRmH>
- <https://www.ianalyseur.org/user/LukeSkywaker/>
- <https://www.ianalyseur.org/heatmap/CenaC/>
- <https://www.ianalyseur.org/ip/60.251.182.146/>

短網址分析

- 韓國瑜臉書貼文中的google短網址 (<https://goo.gl/VMFG2k>)，留下了蛛絲馬跡：從這個短網址導入〈與米魯一起發大柴〉影片的流量中，共有31%的流量來自境外
- <https://goo.gl/VMFG2k> → https://goo.gl/#analytics/goo.gl/VMFG2k/all_time

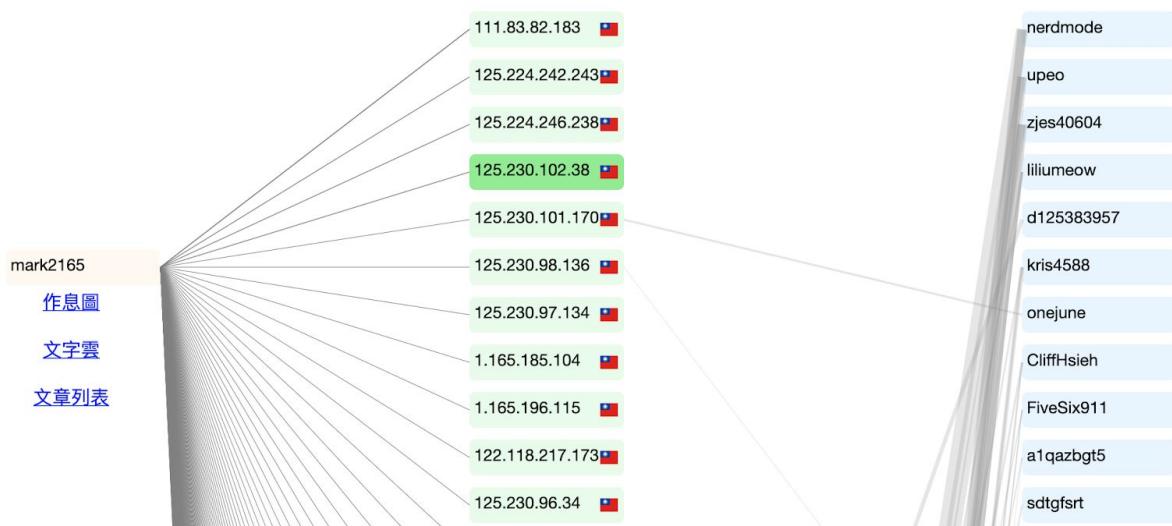
分析策略

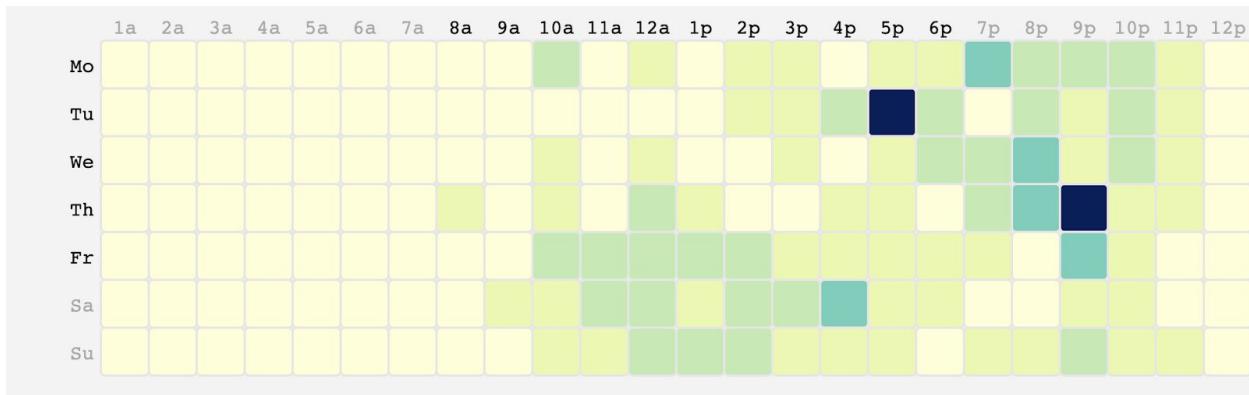
1. 如何抓到包含政治人物的貼文？
2. 如何篩選政治人物的貼文？
3. 發文時間、發文IP（如何轉成地點？）



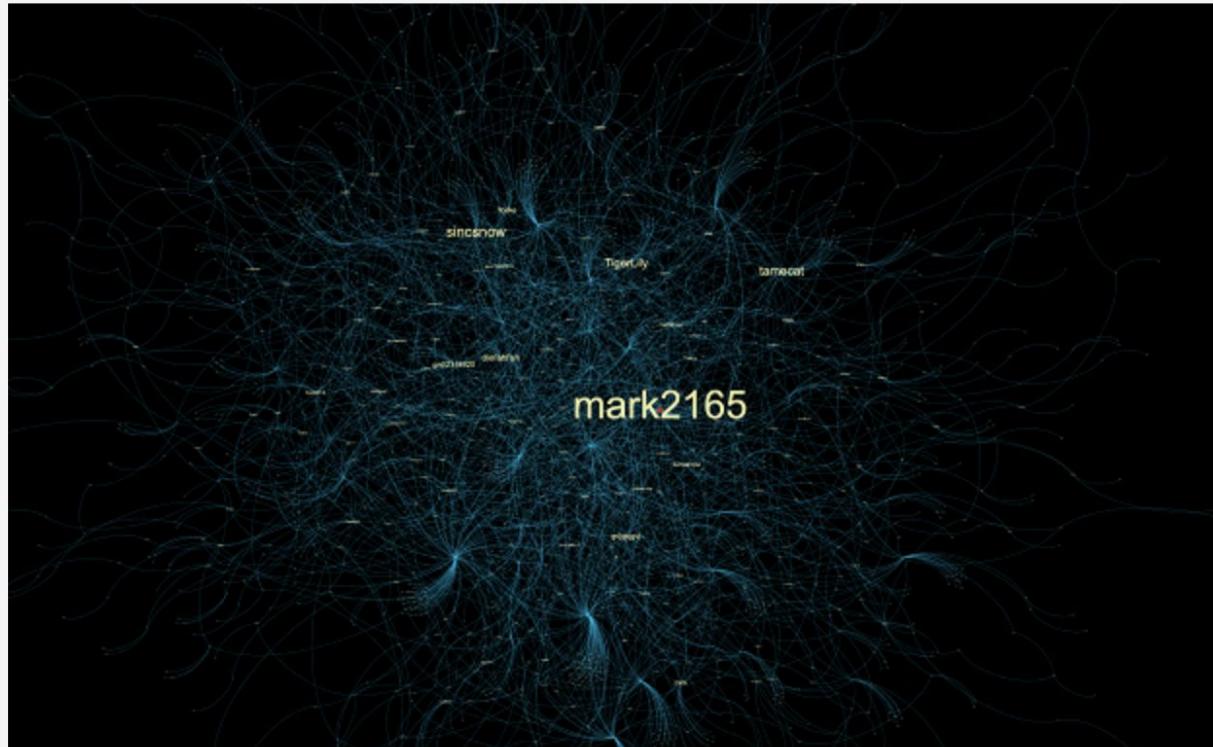
	poster_id	n
1	mark2165	212
2	ivorysoap	199
3	kolod546	147
4	c1951	120
5	fantasy14	110
6	filmystry	108
7	gn02118620	105
8	jackllyl	105
9	Whitening	101
10	diefishfish	98
11	richshen	96
12	sincsnow	90
13	Makubex82	89

<https://www.ianalyseur.org/user/mark2165/>





藉由貼文的ID帳號，我們循線畫出「韓流」的社群網絡圖。網絡中的黃色字體愈大，代表該使用者的發文篇數愈多。而藍色線條則為消息散播的途徑，代表該帳號與其他使用者的連結，線條愈粗，代表互動次數愈多。

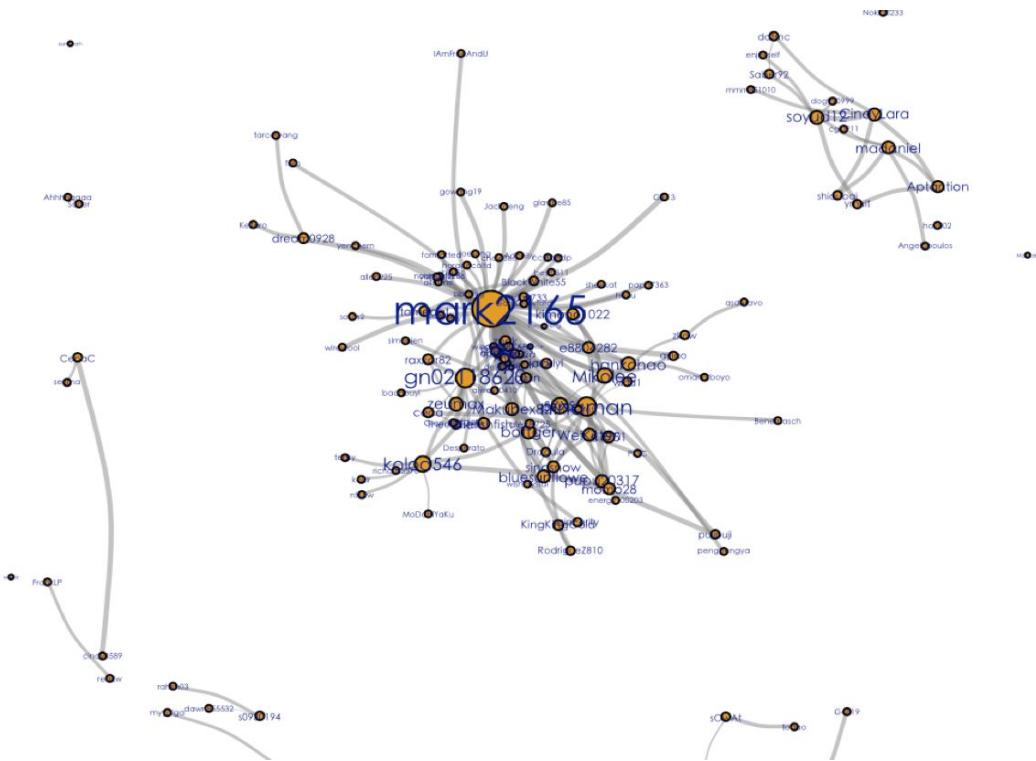
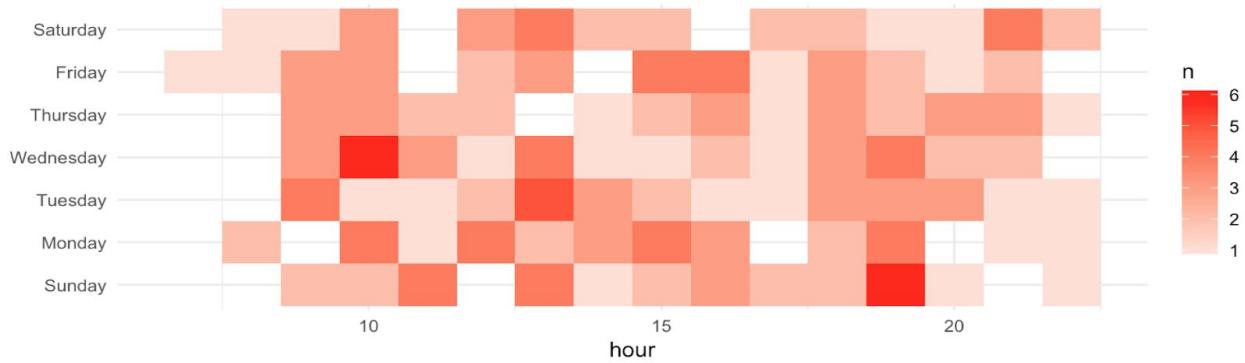


PTT上有關「韓國瑜」的使用者帳號 (資料蒐集區間：2018.4.9.韓國瑜宣布參選至2018.12.24.選後一個月)

6	a9zeros	159.65.131.29	3	cat886688	12	United States
7	a9zeros	159.65.138.183	3	cat886688	10	United States
8	a9zeros	159.65.141.71	3	cat886688	5	United States
9	a9zeros	206.189.84.145	2	artisi	1	United States
10	a9zeros	206.189.84.145	2	cat886688	3	United States
11	a9zeros	206.189.84.147	2	cat886688	3	United States
12	a9zeros	206.189.84.22	2	cat886688	8	United States
13	a9zeros	206.189.84.22	2	pttloser5566	1	United States
14	a9zeros	159.65.131.30	1	cat886688	6	United States
15	a9zeros	159.65.131.35	1	cat886688	1	United States
16	a9zeros	159.89.195.245	1	cat886688	8	Canada
17	a9zeros	206.189.36.78	1	cat886688	2	United States
18	a9zeros	206.189.84.142	1	cat886688	5	United States

7	manpc	163.26.71.6	25	AESA	9	Taiwan, Province of China
8	manpc	163.26.71.6	25	npapi	6	Taiwan, Province of China
9	manpc	163.26.71.6	25	poem5566	1	Taiwan, Province of China
10	manpc	163.26.71.6	25	shared	14	Taiwan, Province of China
11	manpc	163.26.71.6	25	xxlt	12	Taiwan, Province of China
12	manpc	163.26.71.6	25	ymuit	1	Taiwan, Province of China

46	xxlt	163.26.71.6	12	AESA	9	Taiwan, Province of China
47	xxlt	163.26.71.6	12	manpc	25	Taiwan, Province of China
48	xxlt	163.26.71.6	12	npapi	6	Taiwan, Province of China
49	xxlt	163.26.71.6	12	poem5566	1	Taiwan, Province of China
50	xxlt	163.26.71.6	12	shared	14	Taiwan, Province of China
51	xxlt	163.26.71.6	12	ymuit	1	Taiwan, Province of China



第四章練習

練習4-1-1 base and dplyr

請將上述4-1-3或4-1-4的課程教學內容的dplyr程式碼用base套件撰寫之。

練習4-1-2 讀取村里的教育水平

請讀取內政部村里的教育程度資料，清理資料為tidy型態，並依照你的想法，彙整出單一指標（例如大學畢業以上人口比例、不識字人口比例），每一列為一個村里。

練習4-1-3 彙整鄉鎮市區的教育水平

接續練習4-1-2，請嘗試根據鄉鎮市區彙整其教育水平指標，也就是每一列為一個村里。

練習4-1-4 將ibon所在地資料轉為鄉鎮市區的發展指標

練習4-1-5 用視覺化方法找出公投案和各個鄉鎮市區指標間的關係。

從前面應該累積了不少年齡中位數、老年人口數、教育水平、地區發展指標，也抓到了公投案的投票結果。請嘗試用ggplot2的繪圖函式，尋找這些人口統計資料和各個公投案之間的關係。指出你認為值得探討的分佈。

4-Truncated-1-4

Step 3. 用字串處理函式string::str_sub()取出性別

在這資料中最容易處理的是性別，因為現在的資料已經把男性和女性區分開，就是key的最後一個字元，`m`為男生，`f`為女生，所以僅需要把最後一個字元切出來，然後用`mutate()`指定給新的變項名稱做為變項即可，而最後一個字元意即「該字串長度數值所在的那個字元，所以是`nchar(key)`」。

```
tidy_data <- raw %>%
  gather("key", "value", 6:ncol(.)) %>%
  mutate(sex = str_sub(key, nchar(key))) %>%
  arrange(vname) %>% head(20) %>% View
```

Step 4. 用tidyverse::separate()切割字串（未婚、結婚、離婚、喪偶）

未婚 (single)、結婚 (married)、離婚 (divorced)、喪偶 (widowed) 等資料記錄在 key 字串的第一部份，後面用底線 (_) 與年齡分隔，可用 `separate()` 將其切割開來。

- 用 `c("married")` 指定我只取出一個一個變數，且該變數為 `married`。
- `extra = "drop"` 代表其他切出來的部分我都不要，只要第一個。
- `remove = F` 參數設定讓原本的變項 `key` 在切割後要保留下來，不要刪除。
- 建議用 `?separate` 查詢一下 `separate()` 的用法。此時會多出一個新變數。

```
tidy_data <- raw %>%
  gather("key", "value", 6:ncol(.)) %>%
  mutate(sex = str_sub(key, nchar(key))) %>%
  separate(key, c("married"), extra = "drop", remove = F) %>%
  arrange(vname) %>% head(20) %>% View
```

Step 5. 用RE偵測年齡群組的邊界

接下來我打算偵測出年齡群組的邊界。我觀察到年齡群組的紀錄方式為 `_15_19`、`_20_25`，以 5 為間隔，以底線和其他資料隔開，所以我打算用 Regular Expression 來偵測每個年齡群組的下界與上界。

我發現有兩個群組（`15down` 與 `100up`）並非 `15_19` 的形式，會造成 RE 偵測不到，所以我先行用 `str_replace()` 將之分別取代為 `0_14` 與 `100_105`，就會和其他資料一樣，都以數字作為上下界。

```
...
mutate(key = str_replace(key, "15down", "0_14")) %>%
mutate(key = str_replace(key, "100up", "100_105")) %>%
...
```

接下來，我要偵測 `_15_19` 這樣的特徵。

- `([0-9]+_[0-9]+)`：() 小括號代表我要取出的內容。至少一個數字表示為 `[0-9]+`，[] 中括號在此代表或的意思，也就是 0 到 9 之間任意一個數，+ 號代表一個或一個以上，所以 `[0-9]+` 代表一個或一個以上的數字。
- 接下來是要取出的內容周遭有什麼，前面有一個底線，更前面有一個或一個以上的任意字，所以是 `.+`；上述特徵後面可能底線跟著一個 `f` 或 `m`（代表 female 或 male），因此應寫為 `_[fm]`，代表底線後面有 `f` 或 `m`。

```
...
```

```
mutate(ageRange = str_replace(key, ".+_[0-9]+_[0-9]+)_[_fm]", "\\\1")) %>%
...

```

最後，我用`separate()`這個函式，把上述取到的`15_19`、`20_24`、`25_29`切割為兩個變項欄，並分別命名為`ageLower`與`ageUpper`。

```
tidy_data <- raw %>%
  gather("key", "value", 6:ncol(.)) %>%
  mutate(sex = str_sub(key, nchar(key))) %>%
  separate(key, c("married"), extra = "drop", remove = F) %>%
  mutate(key = str_replace(key, "15down", "0_14")) %>%
  mutate(key = str_replace(key, "100up", "100_105")) %>%
  mutate(ageRange=str_replace(key, ".+_[0-9]+_[0-9]+)_[_fm]", "\\\1")) %>%
  separate(ageRange, c("ageLower", "ageUpper"), remove = FALSE) %>%
```

第五章 文字探勘

5-1 Trump's tweets 字串操作與視覺化

5-1-1 載入資料與套件

```
library(dplyr)
library(ggplot2)
load(url("http://varianceexplained.org/files/trump_tweets_df.rda"))
names(trump_tweets_df)
[1] "text"          "favorited"      "favoriteCount"  "replyToSN"
[5] "created"       "truncated"      "replyToSID"     "id"
[9] "replyToUID"    "statusSource"   "screenName"    "retweetCount"
[13] "isRetweet"     "retweeted"     "longitude"    "latitude"
```

可發現資料項中的`statusSource`可指出哪些tweets是用iphone或用Android手機發的。

	truncated	replyToSID	id	replyToUID	statusSource	screenName	retweetCount
15:20:44	FALSE	NA	762669882571980801	NA	<a href="http://twitter.com/download/android" rel="..."	realDonaldTrump	3107
13:28:20	FALSE	NA	762641595439190016	NA	<a href="http://twitter.com/download/iphone" rel="..."	realDonaldTrump	2390
00:05:54	FALSE	NA	762439658911338496	NA	<a href="http://twitter.com/download/iphone" rel="n..."	realDonaldTrump	6691
23:09:08	FALSE	NA	762425371874557952	NA	<a href="http://twitter.com/download/android" rel="..."	realDonaldTrump	6402
21:31:46	FALSE	NA	762400869858115588	NA	<a href="http://twitter.com/download/android" rel="..."	realDonaldTrump	11717
13:49:29	FALSE	NA	762284533341417472	NA	<a href="http://twitter.com/download/android" rel="..."	realDonaldTrump	9892
02:19:37	FALSE	NA	762110918721310721	NA	<a href="http://twitter.com/download/iphone" rel="n..."	realDonaldTrump	5784
02:03:39	FALSE	NA	762106904436961280	NA	<a href="http://twitter.com/download/iphone" rel="n..."	realDonaldTrump	7930
01:53:45	FALSE	NA	762104411707568128	NA	<a href="http://twitter.com/download/android" rel="..."	realDonaldTrump	24663
20:04:08	FALSE	NA	762016426102296576	NA	<a href="http://twitter.com/download/iphone" rel="n..."	realDonaldTrump	7903
18:11:50	FALSE	NA	761988164382756864	NA	<a href="http://twitter.com/download/android" rel="..."	realDonaldTrump	8561
14:48:14	FALSE	NA	761936929902452740	NA	<a href="http://twitter.com/download/android" rel="..."	realDonaldTrump	13129

4-1-2 字串抽取與正規表示式

用正規表示式搭配`stringr`的函式可以從`statusSource`取出該筆tweet是用android或iphone發的，觀察一下他的規則，

statusSource

```
<a href="http://twitter.com/download/android" rel="nofollow">Twitter for Android</a>
<a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a>
<a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a>
<a href="http://twitter.com/download/android" rel="nofollow">Twitter for Android</a>
<a href="http://twitter.com/download/android" rel="nofollow">Twitter for Android</a>
<a href="http://twitter.com/download/android" rel="nofollow">Twitter for Android</a>
```

Step 1. 先選擇所需要的變項欄（請記得是在逗號的右手邊）

```
tweets <- tweets[, c("id", "source", "text", "created_at")]
```

Step 2: 用`sub()`函式來抽出`Twitter for`後面的Android或iPhone。`sub()`的作用是`substitute`, 實際是在做文字取代用的, 因此他應該會有三個輸入,

- 第一個是哪一個變數內容要被取代, 如下方的`tweets$source`
- 第二個是要把什麼取代為另外一個字串, 如下方的`".*Twitter for (.*)<.*"`。該描述為一個正規表示式 (Regular Expression), 簡單地說, 上面這段的意義是, 我不管`Twitter for`前面有啥, 也不管`<`後面有啥, 我要抓`Twitter for`和`<`中間的這段任意文字出來。
 - `.`表示的是所有字
 - `*`表示的是任意次數 (大於等於零次)
 - `.*`代表任意字出現任意次數
 - `.*Twitter`表示我不管前面有什麼, 我只看`Twitter`後的東西。
 - `<.*`會這麼寫代表我抓到`Twitter for`後面的字串直到遇到`<`。
 - `.*?`後面加一個?代表這個正規表示式不是貪婪的 (non-greedy), 也就是他會傾向於找到第一個制定好的特徵就停下來, 不然預設的正規表示式是 Greedy的, 他能夠探測到的字串越長越好, 沒有特別指定non-greedy的話, 如果很後面的資料還有`<`這個符號, 他就會傾向於找到最後一個`<`。
 - 最後`(.*?)`所表示的是, 我等一下要把這個小括號內部的東西留下來
- 第三個是要把要被取代的字串, 取代為什麼, 如下方的`"\1"`。該符號指的是前面的第一組小括號, 也就是他要把第一組小括號所抓出來的內容, 覆蓋掉原本該欄位的所有資料。而第一組小括號就是下面的`(.*?)`。

```
tweets$source <- sub(".*Twitter for (.*)<.*", "\\\1", tweets$source)
```

Step 3: Select source columns are "iPhone" or "Android" : 從上述結果中應該可以發現可以成功地抽出在source欄位中的iPhone或Android。注意這邊是要篩取資料列，所以要把篩取的條件寫在逗號之前。因為經過前述的操作，source這個變項只會有iPhone、Android或者其他其他的值（沒有Matching到的，你可觀察一下會是什麼樣的值），因此，我只需要篩出，source為iPhone或Android的資料，因此我用了%in%這個運算符號，指出如果source是iPhone或Android的其中一個，便會回傳True。

```
tweets <- tweets[tweets$source %in% c("iPhone", "Android"), ]
```

4-1-3 用dplyr來改寫

底下用dplyr的函式來改寫上述的程式，過程包含了選取變項、產生新的變項、最後篩出符合條件的資料列。

其中，我在字串取代的地方，base函式庫會用gsub()或sub()，而我這邊是用stringr的函式str_replace()，凡是進行取代概念都大同小異，都一樣要給三個參數（哪個變數、取代啥、取代為啥），但是給的順序不太一樣。

Dplyr的函式有兩個良好的性質：其一是其基本上所處理的主要資料對象都是data.frame（雖然它實際上是tipple）；其二是，每個函式多把輸入的data.frame作為第一個參數，如下列在函式中的temp.df1與temp.df2。首先是做選取變項指給temp.df1、接下來是在把temp.df1當成輸入以產生新的變項source，之後後指給temp.df2，再把temp.df2給當作輸入後篩出所需的資料列便指給temp.df2。

```
temp.df1 <- select(trump_tweets_df, id, statusSource, text, created)
temp.df2 <- mutate(temp.df1,
                    source = str_replace(statusSource,
                                         ".*Twitter for (.*)<.*", "\\\1"))
tweets <- filter(temp.df2, source %in% c("iPhone", "Android"))
```

4-1-4 用dplyr + magrittr的pipeline風格來改寫

但顯然這樣產生中間變項`temp.df1`和`temp.df2`有點冗贅，就有個神奇的傢伙設計了一套函式庫`magrittr`，讓我們在寫R的時候可以省略上述`temp.df1`和`temp.df2`。所以，你完全可以把下面的程式當成是省略掉上述`temp.df1`和`temp.df2`後的寫法。

```
tweets <- trump_tweets_df %>%
  dplyr::select(id, statusSource, text, created) %>%
  mutate(source = str_replace(statusSource,
                               ".*Twitter for (.*)<.*", "\\\1")) %>%
  # extract(statusSource, "source", "Twitter for (.*)<") %>%
  filter(source %in% c("iPhone", "Android"))
```

5-2 視覺化

4-2-1 視覺化：比較發文時間分佈

這些`tweets`是一段時間區間的資料，通常最容易想到的第一個假設就是，那不同手機（iPhone vs. Android）的發文時間分佈會不會不一樣？「發文時間分佈」是一個太籠統的說法，需要界定兩件事，首先，發文時間要以什麼為單位？用秒為單位鐵定不管用，因為每一篇文章幾乎都在以秒為單位的時間軸上是唯一的，那以分呢？就能統計出這時間區間的每分鐘有幾篇`tweets`，但這裡最好是以小時為單位，因為我們可以看出，不同載具，在一天24小時內，是否有個別發文的偏好時刻。所以，所謂的「發文時間分布」在此的具體定義應該是「以24小時為單位的`tweets`篇數分布」。

首先，要先把原本的時間取出「小時」來處理。原本的`created`原為R的時間物件。因此可以用`lubridate`函式庫的`hour()`來取出「小時」，之後便可用`hour()`搭配`with_tz()`來根據所指定的時區轉換時間。在此所指定的是EST³美東時間。通常這類全球性的服務會以UTC+0作為標準時間，UTC稱為「世界協調時間」，其設定的目標是盡量接近格林威治時間（GMT），大部分認為UTC能跟GMT互換，雖然GMT目前已經缺乏科學認證。

關於R時間物件可參考第二章時間物件，以及第三章定時爬取youtube資料的說明。

```
library(lubridate) # for hour()
> class(tweets$created)
```

³ https://en.wikipedia.org/wiki/Eastern_Time_Zone

```
[1] "POSIXct" "POSIXt"
tweets %>%
  mutate(hour = hour(with_tz(created, "EST")))
```

當抽出時間後會產生一個新變數`hour`，紀錄是哪一個小時發的。然後使用`count(source, hour)`來計算出不同的載具在不同的時間發布幾則貼文，做完這個步驟後會多一個變數`n`。因為`n`目前為次數，通常觀察比例會比較好觀察，所以我用`mutate()`產生一個新的變數，為`n`除以總數`sum(n)`。這是把數據轉為比例的做法之一。你可以在這些指令的`%>%`後面加上`View`，執行後就可以直接觀察執行到該步驟所產生的結果，例如：

```
library(lubridate) # for hour()
tweets %>%
  mutate(hour = hour(with_tz(created, "EST")))) %>%
  count(source, hour) %>%
  mutate(percent = n / sum(n)) %>% View
```

(Options) 這時候是把`n`除以總數，但和原本沒除可能沒差別，因為如果Android的`tweets`比較多，除以Android和iPhone總`tweets`還是一樣，比例不會變。因此，若要避免Android和iPhone的`tweets`數影響，應該分別除以兩個載具的發文數。`dplyr`在這點上非常容易處理，可用`group_by()`的指令讓算式依照群組計算，然後再用`ungroup()`回到原本的結果，例如下列程式碼：

```
library(lubridate) # for hour()
tweets %>%
  count(source, hour = hour(with_tz(created, "EST")))) %>%
  group_by(source) %>%
  mutate(percent = n / sum(n)) %>%
  ungroup() %>% View
```

最後是用`ggplot2`視覺化。`ggplot`視覺化主要包含三個部分，用`ggplot()`宣告要繪圖、用`aes()`來決定X-Y軸分別要畫什麼以及分組、用`geom_`開頭的函式來決定要畫什麼圖。

這邊我打算畫折線圖，以`hour`為X軸、以比例`percent`為Y軸，並在繪圖上以`source`做分組，這樣可以根據不同的`source`各自繪製一條折線。

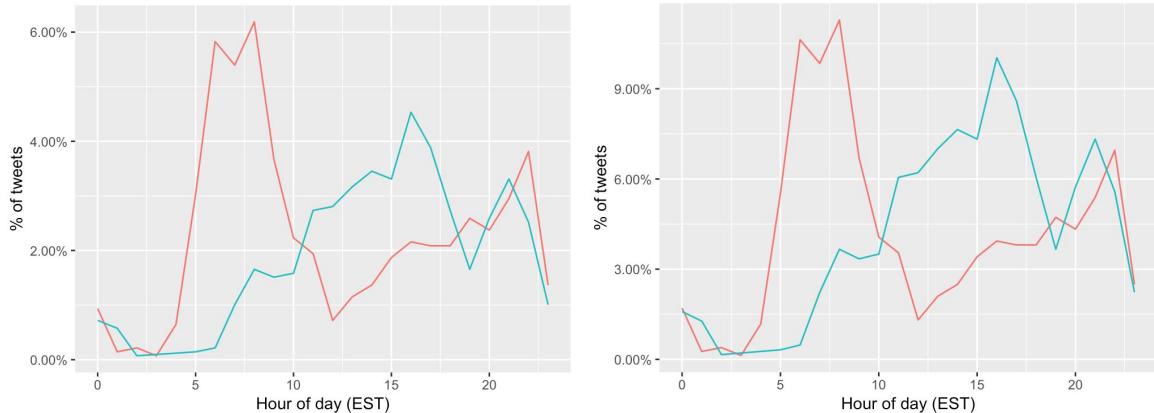
```
library(lubridate) # for hour()
library(scales) # for percent_format()
```

```

tweets %>%
  count(source, hour = hour(with_tz(created, "EST"))) %>%
  group_by(source) %>%
  mutate(percent = n / sum(n)) %>%
  ungroup() %>%
  ggplot() +
  aes(hour, percent, color = source) + # 定義x-y軸與分組
  geom_line() + # 決定繪圖種類
  scale_y_continuous(labels = percent_format()) + # 繪圖微調
  labs(x = "Hour of day (EST)", y = "% of tweets", color = "")

```

下列左圖為沒有`group_by()`的結果，右圖為有`group_by()`的結果，因為這個例子實際上Android和iPhone數量沒有差太多（例如差10倍以上），所以差異不會很大。但若要比較兩者的趨勢，應該用右圖來比較會比較正確。



4-2-2 視覺化：比較有無插圖

該分析假設，Android和iPhone發文時貼圖的次數可能會有差異。因此，要抽取出每一則`tweets`是否有貼圖。判斷是否有貼圖十分容易，就判斷是否有<http://t.co...>的超鏈結即可，有`t.co`的話就代表有貼圖。

在做這件事情前，由於如果是retweet或引用別人的話若被算進來的話，那會影響分析結果的可信度，因此應該是把retweet給拿掉。由於retweet別人時，該`tweets`會是雙引號開頭，所以我只要把雙引號開頭的`tweets`拿掉就可以了。此時，一樣可以用正規表示式來達到這樣的需求。由於要判斷是否有雙引號開頭，所以字串要用單引號來表示，而在單引號中、雙引號前的^的符號代表字串的開頭之意。也就是說，下面這句話就是，如果開頭有雙引號的，那就`str_detect()`成功了，但前面有一個`!`表示這樣的情形我不要。

```
tweets %>%
  filter(!str_detect(text, '^\"')) %>%
```

另外，可以上<https://regex101.com/> 測試RE（正規表示式的用法）驗證自己有沒有寫錯。

等濾除了retweet後，我要產生一個變項，如果有偵測到超鏈結，那該變項值為 Picture/link、如果沒偵測到，就是No picture/link。所以這是一個「三元」表示式，如果條件成立的話，那麼A就等於B，不然的話，A就等於C。

R的三元表示式為 `A = ifelse(cond, B, C)`，條件就是 `str_detect(text, "t.co")`；條件為真，A就等於B、不為真，A就等於C。

```
tweets %>%
  filter(!str_detect(text, '^\"')) %>%
  mutate(picture = ifelse(str_detect(text, "t.co"),
                           "Picture/link", "No picture/link")) %>%
```

最後，用 `count(source, picture)` 計算不同類型的手機中有發圖和沒發圖的各有幾個 tweets。由於手機只有兩種類型，而有圖沒圖只有兩種情形，因此統計起來一共有四個值。此時應該繪製為長條圖，所以要使用 `geom_col()`，然後把手機種類 `source` 和發文數 `n` 分別作為 X-Y 軸，最後，依照 `picture` 變項標示是否有圖來分組 `fill = picture` 並繪圖。繪圖時可選擇 `position = "dodge"`，他會把兩組的數據分開打。

`geom_col()` 應讀作 geom column，為繪製長條圖的工具。我認為比 `geom_bar()` 好用，因為可以控制我要給他幾個變數，比方說這個圖我就要給他兩個 X-Y 軸的變數。

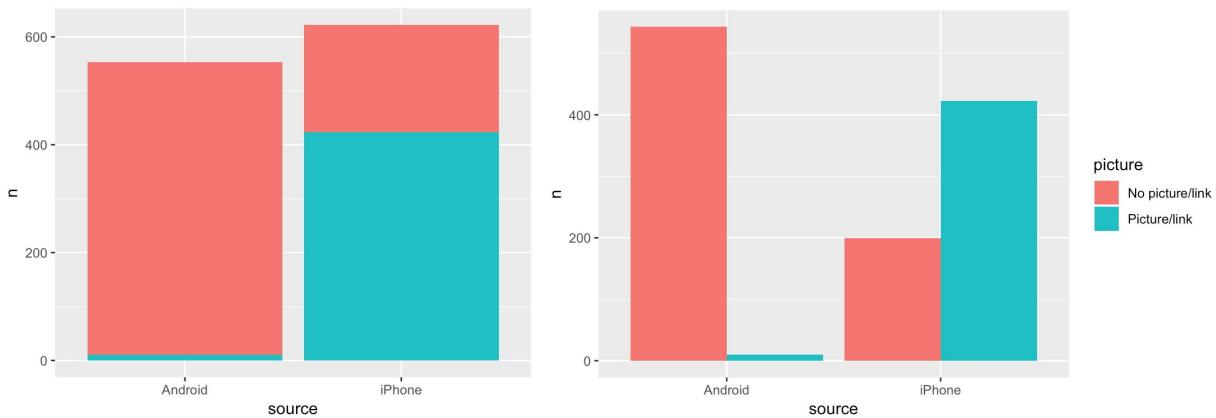
```
library(stringr)
tweets %>%
```

```

filter(!str_detect(text, '^"')) %>%
mutate(picture = ifelse(str_detect(text, "t.co"),
                        "Picture/link", "No picture/link")) %>%
count(source, picture) %>%
ggplot() +
aes(source, n, fill = picture) +
geom_col(position="dodge")

```

`geom_col(position="dodge")`為右圖，左圖是預設值，是`geom_col()`，或者是`geom_col(position="stack")`。



5-3 文字探勘

4-3-1 用字比較

```

library(tidytext) # unnest_tokens()
library(stringr)      # str_detect(), str_replace_all()

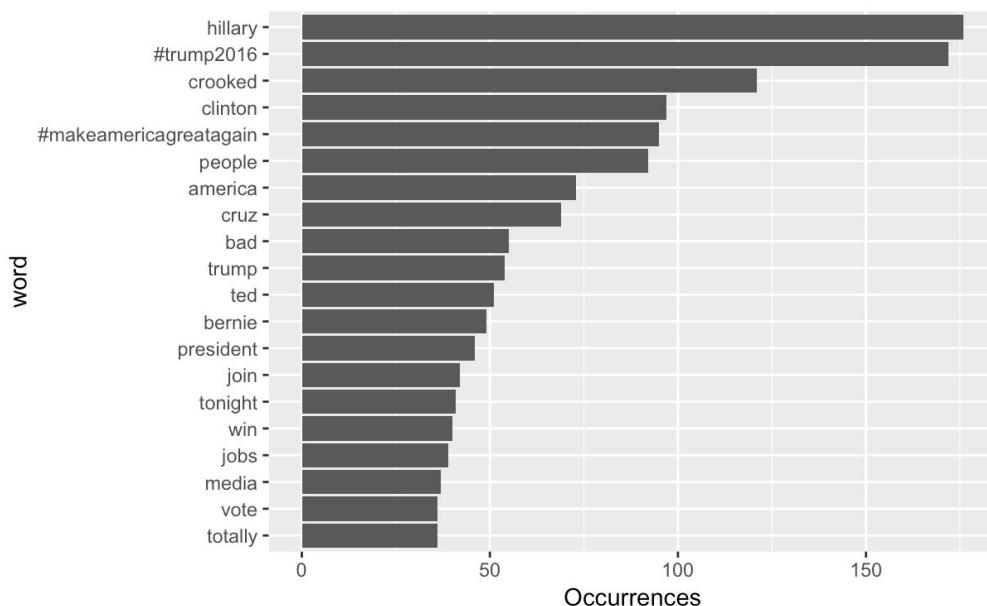
# View(test)
# stop_words$word
tweet_words <- tweets %>%
  filter(!str_detect(text, '^"')) %>%
  mutate(text = str_replace_all(text,

```

```
"https://t.co/[A-Za-z\d]+|&quot;, "") %>%
  unnest_tokens(word, text) %>%
  unnest_tokens(word, text, token = "regex", pattern =
"[^A-Za-z\d#@']+") %>%
  filter(!word %in% stop_words$word,
         str_detect(word, "[a-z]"))
```

4-3-2 熱門用字的視覺化

```
tweet_words %>%
  count(word, sort = TRUE) %>%
  head(20) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n)) +
  geom_col() +
  ylab("Occurrences") +
  coord_flip()
```



4-3-3 不同載具的用字差異

```
word_by_source <- tweet_words %>%
  count(word, source) %>%
```

```

filter(n >= 5) %>%
spread(source, n, fill = 0) %>%
ungroup()

sum(word_by_source$iPhone)
sum(word_by_source$Android)

android_iphone_ratios <- word_by_source %>%
  mutate(iPhone = (iPhone+1)/sum(iPhone+1)) %>%
  mutate(Android = (Android+1)/sum(Android+1)) %>%
  # mutate_at(.cols = vars(iPhone, Android),
             # .funs = funs((. + 1) / sum(. + 1))) %>%
  mutate(logratio = log2(Android / iPhone)) %>%
  arrange(desc(logratio))

```

簡化版視覺化

```

android_iphone_ratios %>%
  mutate(word = reorder(word, logratio)) %>%
  ggplot() +
  aes(word, logratio, fill=logratio < 0) +
  geom_col() +
  coord_flip()

```

清楚明瞭的視覺化

```

android_iphone_ratios %>%
  group_by(logratio > 0) %>%
  top_n(10, abs(logratio)) %>%
  ungroup() %>%
  mutate(word = reorder(word, logratio)) %>%
  ggplot(aes(word, logratio, fill = logratio < 0)) +
  geom_col() +
  coord_flip() +
  ylab("Android / iPhone log ratio") +
  scale_fill_manual(name = "", labels = c("Android", "iPhone"),
                    values = c("red", "lightblue"))

```

用字情緒分析

5-A Practices & Assignments

第六章 - 視覺化

6-1 ggplot

6-1-X Mac上的中文字體修正

```
theme(axis.text.y=element_text(colour="black", family="Heiti TC Light")) +  
geom_node_text(aes(label = name), repel = F, family = "Heiti TC Light") +
```

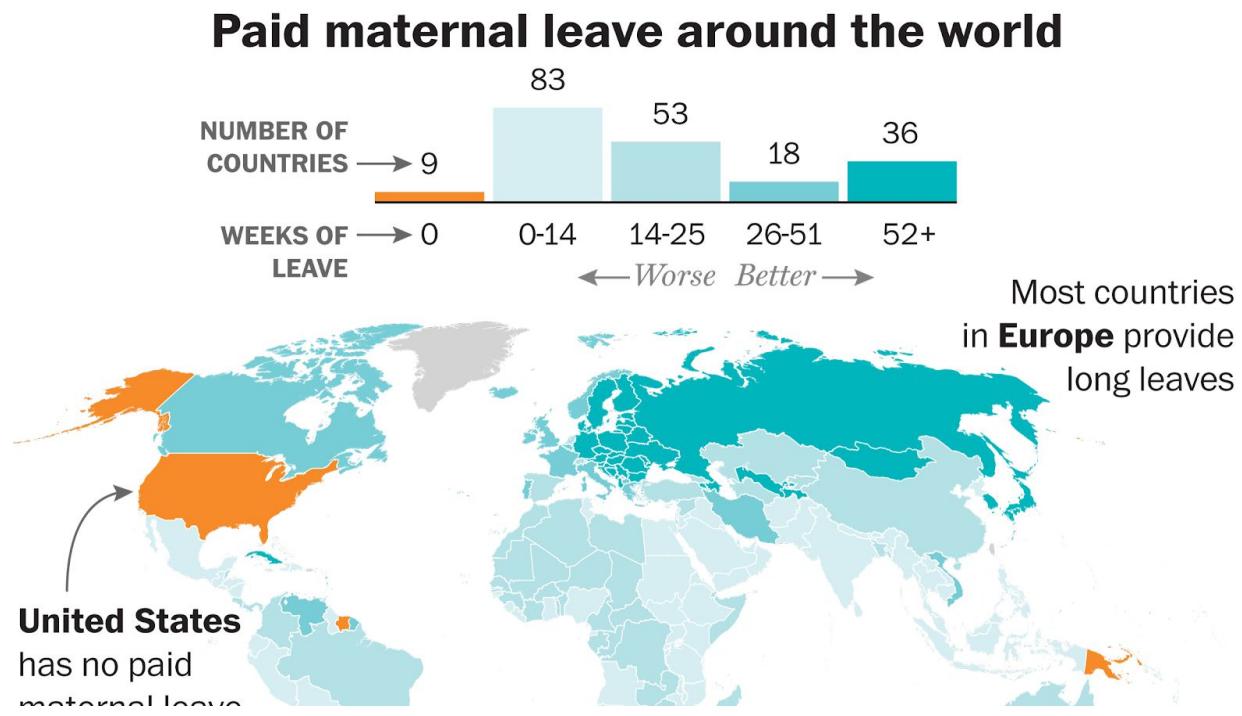
```
theme(legend.text = element_text(family = "Heiti TC Light"))
geom_text(aes(label = name), size = 3, family = "Heiti TC Light") +
  geom_text(aes(PC1, PC2, label=name), family="黑體-繁 中黑") # STKaiti
  theme(text=element_text(family="黑體-繁 中黑", size=12),
axis.text.x=element_text(angle=45))
  theme(text=element_text(family="STKaiti"))
```

6-Map 用rworldmap繪製產假支薪的世界地圖



在這個案例中，將完成在第二章中提及的產假支薪報導⁴繪製地圖的部分如下圖。以最後一年 (`matleave_13`) 的資料，將世界各國的情形（1至5共5個等級）繪製在地圖上，並視覺上凸顯出沒有產假支薪的國家，其他國家按照支薪週數的等級用顏色由淺至深填色。

⁴ Melissa Etehad & Jeremy C.F. Lin (August 13, 2016) The world is getting better at paid maternity leave. The U.S. is not. The Washington Post
https://www.washingtonpost.com/news/worldviews/wp/2016/08/13/the-world-is-getting-better-at-paid-maternity-leave-the-u-s-is-not/?utm_term=.060efaf71b59



載入所需套件與資料

載入所需套件

```
library(readxl)
library(rworldmap) # for drawing rworldmap
options(stringsAsFactors = F)
```

載入資料相關資料請見第二章產假支薪的說明

```
rawdata <- read_excel("data/WORLD-MACHE_Gender_6.8.15.xls", "Sheet1",
col_names=T)
mapdata <- rawdata[,c(3, 6:24)]
```

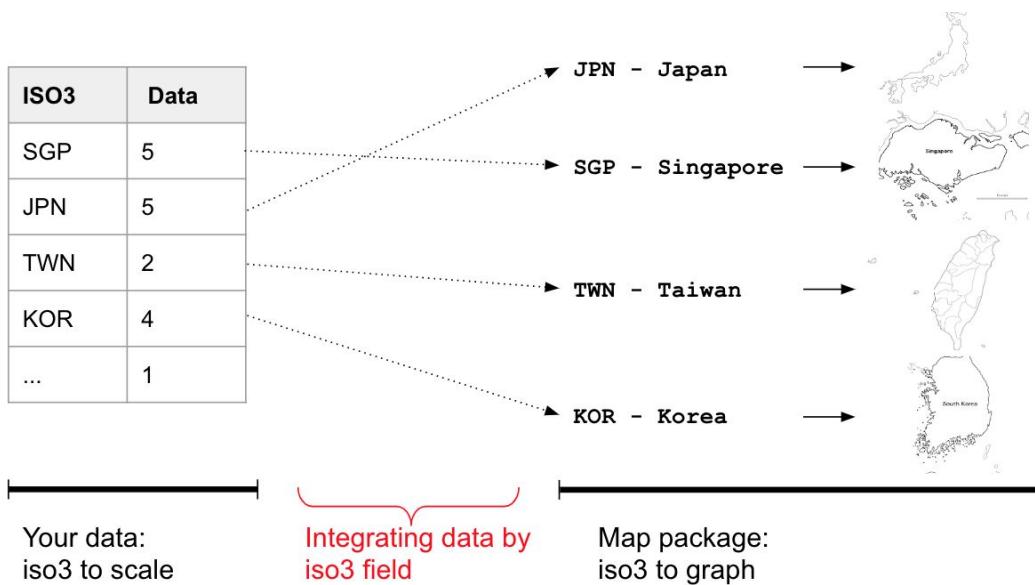
結合資料與地圖檔

```
> myMap <- joinCountryData2Map(mapdata, joinCode = "ISO3",
nameJoinColumn = "iso3")
196 codes from your data successfully matched countries in the map
1 codes from your data failed to match with a country code in the map
```

47 codes from the map weren't represented in your data

繪製地圖通常要有地圖檔與資料兩個資料來源。以世界各國地圖來解釋地圖檔的話，概念如下圖。地圖檔通常會將各國家名、國家代碼分別對應到一組X-Y座標所組成的vector，該組座標若描繪在圖面上恰為該國家的形狀。而資料檔，通常就像下圖一樣，國家名稱、國碼對應到一筆資料（例如`matleave_13`或人口數）。地圖檔內有國碼和邊界、資料檔內有國碼和資料，第一件事便是要將兩者結合（join）起來。

如果今天拿到的只是地圖檔，那就要自己寫程式join，例如用`left_join()`。但是`rworldmap`套件內建有join資料與地圖檔的工具，且提供好幾種據以join的資料形式，例如國名、`ISO2`、`ISO3`等國碼標準（可以查詢`?joinCountryData2Map`）。在此，`joinCode = "ISO3"`指定了用`ISO3`這個國碼標準來join資料與地圖檔；而`nameJoinColumn = "iso3"`則用以告訴該函式，合乎`ISO3`的變項名稱為何。



繪製地圖

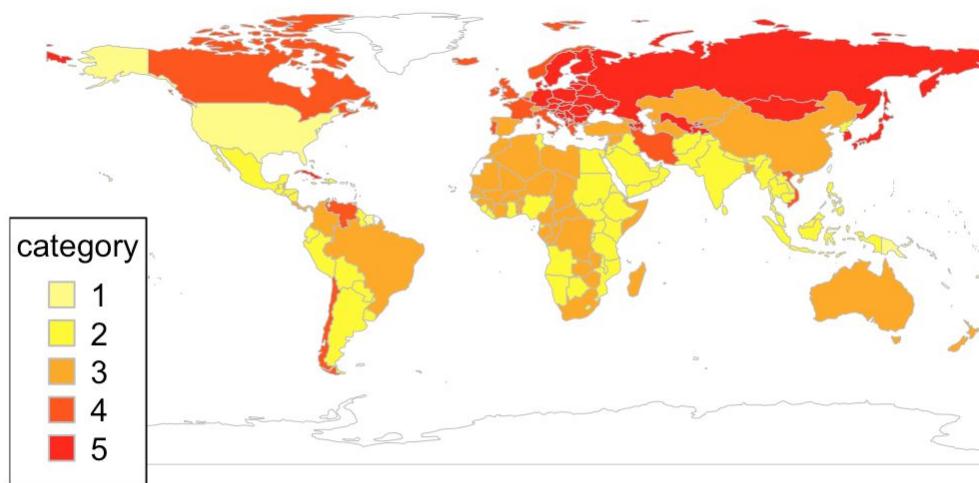
繪製地圖時，要決定每一個國家要上什麼顏色。因為我現在一共有五個尺度5至1，所以我把它當成類別而非連續變項來打，因此我要指定`catMethod = "categorical"`。若查詢`?mapCountryData`還可以找到其他兩種方法，比方說，如果我希望照人口數來打漸層，那我就要用第二種的其中一個；如果我希望自訂資料分組，例如將人口數0~10K分為一

組、10K~1M為一組，1M~10M為一組、10M至100M為一組，這樣，我就要使用第三種方法，並輸入 `c(0, 10000, 1000000, 10000000)` 等作為切分區間的數值依據。

1. "categorical" - each unique value is treated as a separate category
2. for numeric data : "pretty", "fixedWidth", "diverging", "logFixedWidth", "quantiles"
3. a numeric vector defining breaks e.g. `c(0:5)`, note that a value of 2 goes into 1-2 not 2-3, uses `cut(include.lowest=TRUE)`

```
mapCountryData(myMap
               , nameColumnToPlot="matleave_13"
               , catMethod = "categorical"
)
```

matleave_13



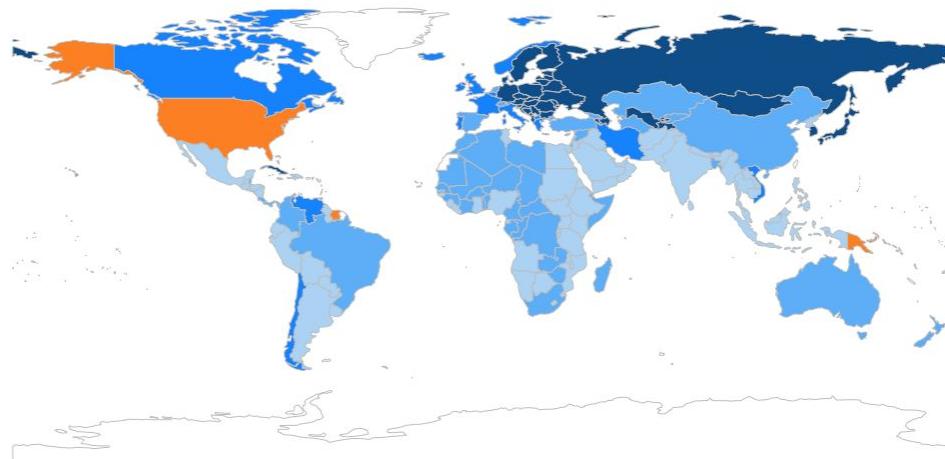
自定義顏色：此時，我希望自行定義顏色，讓顏色除了沒有產假支薪的國家外，顏色由淺至深。那可使用 `colourPalette = colors` 參數來直接定義顏色。因為一共有五個Level，所以一共要給五種顏色。你可以先輸入一些你比較有把握的顏色，之後逐一修改看看，看看他的顏色怎麼排。

```
colors <- c("#FF8000", "#A9D0F5", "#58ACFA", "#0080FF", "#084B8A")
mapCountryData(myMap
               , nameColumnToPlot="matleave_13"
               , catMethod = "categorical")
```

```

        , colourPalette = colors
        , addLegend="FALSE"
)

```

matleave_13

Practice 5-Map-1

就產假支薪資料，繪製1995至2013年的地圖於同一張圖片中，以概觀世界各國近年來的變化。（使用`par()`來控制子圖的繪製）

```

par(mfrow=c(4,5), mai= c(0.2, 0.2, 0.2, 0.2))
for(i in 51:69){
  mapCountryData(myMap
    , nameColumnToPlot=names(myMap)[i]
    , catMethod = "categorical"
    , colourPalette = colors
    , addLegend="FALSE"
  )
}

```

matleave_95 matleave_96 matleave_97 matleave_98 matleave_99



matleave_00 matleave_01 matleave_02 matleave_03 matleave_04



matleave_05 matleave_06 matleave_07 matleave_08 matleave_09



matleave_10 matleave_11 matleave_12 matleave_13

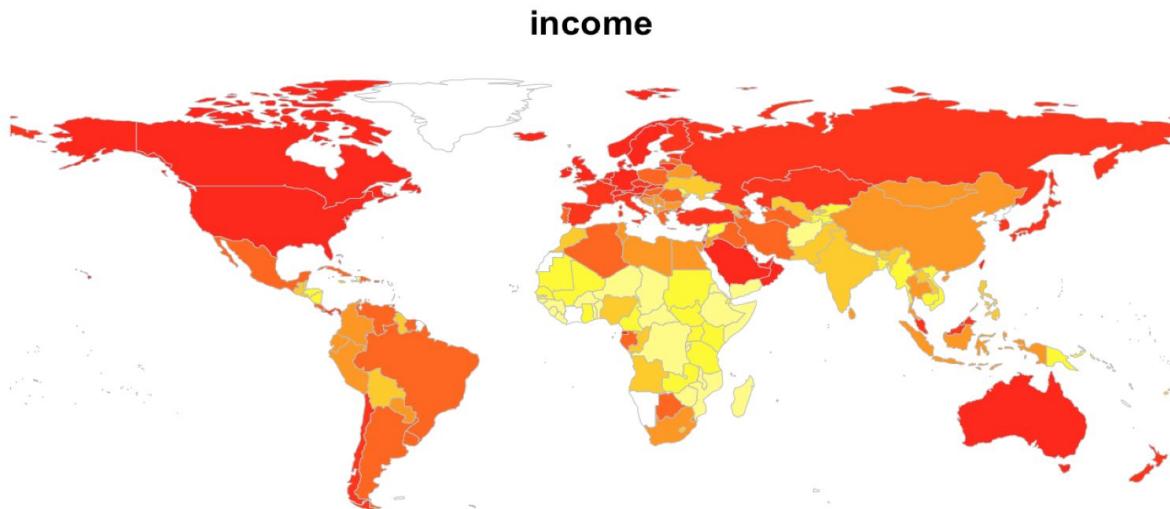


Practice 5-Map-2

在World Inequality Data中有記錄每個國家的income，但在其資料表中只有國名和income，在meta table中有country code。請嘗試用rworldmap繪製income map。使用?
[mapCountryData](#)查詢如何繪製數值（連續變項）的顏色，rworldmap也可以用iso2繪製世界地圖。

```
myMap <- joinCountryData2Map(ndata, joinCode = "ISO2", nameJoinColumn =
"iso2")
```

若還無法自行用程式處理該資料，可以用google sheet merge起來後另存為CSV，先把data與metadata兩個資料表最前面不屬於資料的header部分手動刪除



輸出資料至HTML

有時候用RStudio的data.frame觀察介面並不好觀察長篇的文字內容。一個好的方式是把整個data.frame寫到html檔就會便於觀看。

```
DT:::saveWidget(DT:::datatable(hot_items), 'hot_items.html')
```

或者用pipeline的方式寫

```
df %>%
DT:::datatable() %>%
DT:::saveWidget("rebought.html")
```

6-A Practices & Assignments

Appendix - Rmarkdown

R Markdown

```
output:  
  html_notebook:  
    code_folding: hide  
    number_sections: true  
    fig_caption: yes  
    highlight: zenburn  
    theme: simplex  
    toc: yes  
  editor_options:  
    chunk_output_type: console
```

R presentation - xaringan

```
output:  
  xaringan::moon_reader:  
    self_contained: True  
  nature:  
    highlightStyle: github  
    highlightLines: true  
    ratio: "16:10"
```