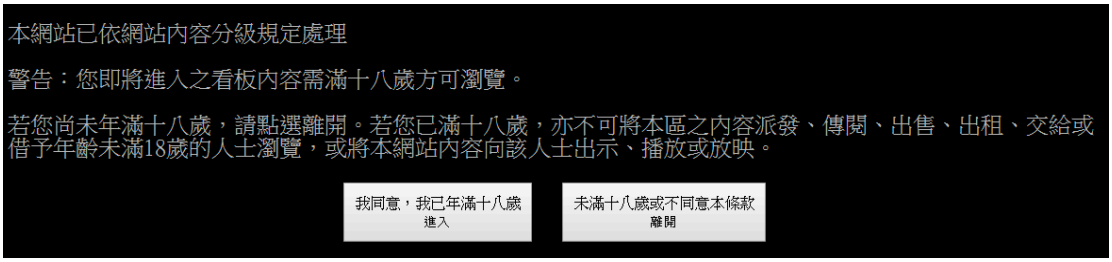
# R 軟體爬蟲和文字斷詞

李智慎 副統計分析師

這一期將教大家如何使用 R 軟體擷取網路頁面上的資料即俗稱的爬網或爬蟲，我們將以台灣最大的社群論壇 PTT 八卦版做示範，主要使用 rvest 套件擷取文章內容和 jiebaR 套件做斷詞。

在準備爬網前，必須先了解網頁整體的架構，所以首先讓我們進入 PTT 八卦版首頁 https://www.ptt.cc/bbs/Gossiping/index.html




由上圖可看到一開始進入 PTT 八卦版頁面時，會有一個是否年滿 18 歲的驗證頁面，按下我同意後才會進入八卦版文章的首頁，畫面如下


那要怎麼樣利用 R 軟體的程式執行驗證的頁面呢?首先準備好要使用的套件並引入

```
#install.packages(c("tidyverse", "rvest", "stringr", "jiebaR", "tmcn"))
library(tidyverse)
library(rvest)
```

```
library(stringr)
library(jiebaR)
library(tmcn)
```

各套件用途描述如下

- tidyverse: 內含資料處理套件(dplyr)和繪圖套件(ggplot2)等
- rvest: 網頁解析處理套件
- stringr: 字串處理套件
- jiebaR: 文字斷詞
- tmcn: 文字字庫

將 PTT 網址設定好使用 html_session 讀取並存為 gossiping.session

```
ptt.url <- "https://www.ptt.cc"
gossiping.url <- str_c(ptt.url, "/bbs/Gossiping")
gossiping.url
[1] "https://www.ptt.cc/bbs/Gossiping"
gossiping.session <- html_session(url = gossiping.url)
gossiping.session
<session> https://www.ptt.cc/ask/over18?from=%2Fbbs%2FGossiping%2Findex.html
  Status: 200
  Type:     text/html; charset=utf-8
  Size:     2411
```

可看到 url 指定八卦版的連結，但由於是初始進入而被導向至認證頁面，所以在 gossiping.session 的上方連結才會是認證頁面的連結。

接下來的步驟如下

1. 找到認證的表單(form)

```
gossiping.form <- gossiping.session %>%
  html_node("form") %>%
  html_form()
gossiping.form
<form> '<unnamed>' (POST /ask/over18)
  <input hidden> 'from': /bbs/Gossiping/index.html
  <button submit> 'yes
  <button submit> 'no
```

2. 填上確認數值(yes)

```
gossiping <- submit_form(
  session = gossiping.session,
  form = gossiping.form,
  submit = "yes"
)
gossiping
```

```
<session> https://www.ptt.cc/bbs/Gossiping/index.html
  Status: 200
  Type:     text/html; charset=utf-8
  Size:     8661
```

可看到在提交確認表單後，gossiping 的連結已改變為八卦版首頁，此時我們可以先進入八卦版首頁看一下整體網頁架構，而觀看原始碼的方式只需在頁面空白處按右鍵並點選檢視網頁原始碼，如下圖

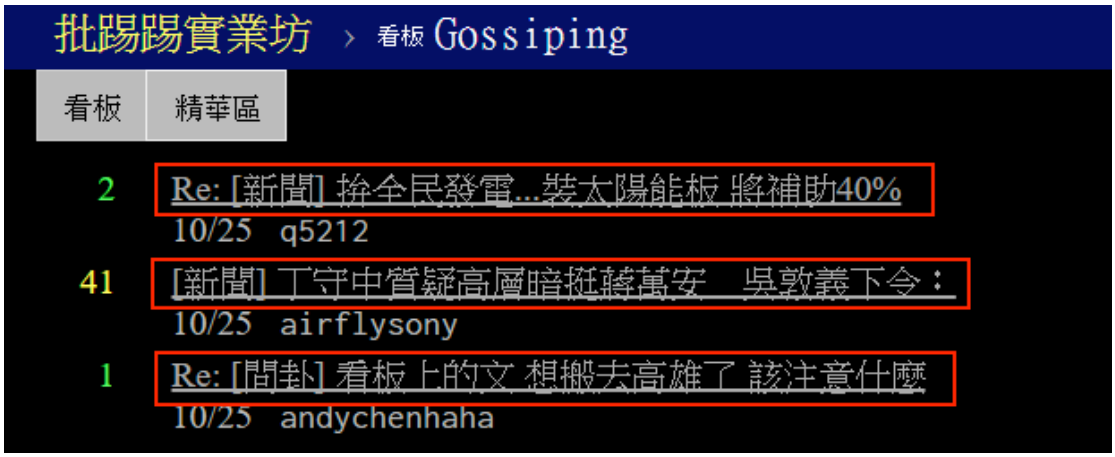

點選後原始碼頁面如下

```
<div class="r-ent">
    <div class="nrec"><span class="hl f2">2</span></div>
    <div class="mark"></div>
    <div class="title">
        <a href="/bbs/Gossiping/M.1508920363.A.FFB.html">Re: [新聞] 拚全民發電...裝太陽能板 將補助40%</a>
    </div>
    <div class="meta">
        <div class="date">10/25</div>
        <div class="author">q5212</div>
    </div>
</div>

<div class="r-ent">
    <div class="nrec"><span class="hl f3">41</span></div>
    <div class="mark"></div>
    <div class="title">
        <a href="/bbs/Gossiping/M.1508920365.A.43F.html">[新聞] 丁守中質疑高層暗挺蔣萬安　吳敦義下令：</a>
    </div>
    <div class="meta">
        <div class="date">10/25</div>
        <div class="author">airflysony</div>
    </div>
</div>

<div class="r-ent">
    <div class="nrec"><span class="hl f2">1</span></div>
    <div class="mark"></div>
    <div class="title">
        <a href="/bbs/Gossiping/M.1508920370.A.3EB.html">Re: [問卦] 看板上的文 想搬去高雄了 該注意什麼</a>
    </div>
    <div class="meta">
```

上圖紅色框內的 a 元素就是下圖網頁上標題的部份




而接下來的任務就是將這些連結一一儲存起來，然後再進入連結內文蒐集各部分的文字內容，但一頁只會顯示 10 多篇文章，所以還必須經過跳頁來蒐集文章的連結，而網址連結有一個規則性，我們只需利用這規則性到各個頁面蒐集文章連結即可，

首頁連結: https://www.ptt.cc/bbs/Gossiping/index.html

首頁上頁連結: https://www.ptt.cc/bbs/Gossiping/index25436.html

可看到 index 後面會接數字，所以只需要依照數字遞減，依序的往前作連結即可蒐集各頁面的文章連結，程式如下

```
page.latest <- gossiping %>%
  html_nodes("a") %>%
  html_attr("href") %>%
  str_subset("index[0-9]{2,}\\.html") %>%
  str_extract("[0-9]+") %>%
  as.numeric()
```

page.latest 是最新頁碼數字，必須從主頁 gossiping 裡去尋找步驟依序如下

1. html_nodes("a"): 擷取所有 a 元素
2. html_attr("href"): 擷取 a 元素裡的 href 屬性也就是該連結網址
3. str_subset("index[0-9]{2,}\\.html"): 篩選符合 index 後面接一串數字的連結
4. str_extract("[0-9]+ "): 擷取連結內數字的部分
5. as.numeric(): 轉換為數字格式

透過以上步驟即可得到最新頁面的號碼數 page.latest

得到最新的頁碼數 page.latest 後，利用迴圈的方式對每一頁做文章連結的收集

```
links.article <- NULL
page.length <- 5
for (page.index in page.latest:(page.latest - page.length)) {
  link <- str_c(gossiping.url, "/index", page.index, ".html")
  print(link)
  links.article <- c(
    links.article,
    gossiping %>%
      jump_to(link) %>%
      html_nodes("a") %>%
      html_attr("href") %>%
      str_subset("[A-z]\\.[0-9]+\\.[A-z]\\.[A-z0-9]+\\.html")
  )
}
```

```
[1] "https://www.ptt.cc/bbs/Gossiping/index25439.html"
[1] "https://www.ptt.cc/bbs/Gossiping/index25438.html"
[1] "https://www.ptt.cc/bbs/Gossiping/index25437.html"
[1] "https://www.ptt.cc/bbs/Gossiping/index25436.html"
[1] "https://www.ptt.cc/bbs/Gossiping/index25435.html"
[1] "https://www.ptt.cc/bbs/Gossiping/index25434.html"
```

links.article 為存放文章連結的空間，page.length 為想要存放的頁數，迴圈內容如下

1. link: 當下迴圈輪到的連結
2. jump_to(link): 跳頁至新頁面
3. html_nodes("a") & html_attr("href"): 找 a 元素的 href 屬性
4. str_subset("[A-z]\\.[0-9]+\\.[A-z]\\.[A-z0-9]+\\.html"): 篩選符合文章連結格式的連結

```
links.article <- unique(links.article)
```

為了避免連結重複用 unique 重新做處理。

經過以上迴圈處理後得到每一頁內的文章連結 links.article，接下來進入 links.article 內各連結即可爬文

```
push.table <- tibble() #  建立推文儲存空間
article.table <- tibble() #  建立文章儲存空間
for (temp.link in links.article) {

  article.url <- str_c(ptt.url, temp.link) #  文章網址
  temp.html <- gossiping %>% jump_to(article.url) #  連結至文章網址
  article.header <- temp.html %>%
    html_nodes("span.article-meta-value") %>% #  開頭部分元素
    html_text()
  article.author <- article.header[1] %>% str_extract("^[A-z0-9_]+") #  作者
  article.title <- article.header[3] #  標題
  article.datetime <- article.header[4] #  時間
  article.content <- temp.html %>%
    html_nodes( #  內文部分
      xpath = '//div[@id="main-content"]/node()[not(self::div|self::span[@class="f2"])]'
    ) %>%
    html_text(trim = TRUE) %>%
    str_c(collapse = "")
  article.table <- article.table %>% #  合併文章資料
    bind_rows(
      tibble(
        datetime = article.datetime,
        title = article.title,
        author = article.author,
```

```
            content = article.content,
            url = article.url
          )
      )

  article.push <- temp.html %>% html_nodes("div.push") #  擷取推文
  push.table.tag <- article.push %>% html_nodes("span.push-tag") %>% html_text(trim =
TRUE) #  推文種類
  push.table.author <- article.push %>% html_nodes("span.push-userid") %>% html_text(trim
= TRUE) #  作者
  push.table.content <- article.push %>% html_nodes("span.push-content") %>%
html_text(trim = TRUE) %>% str_sub(3) #  推文內容
  push.table.datetime <- article.push %>% html_nodes("span.push-ipdatetime") %>%
html_text(trim = TRUE) #  推聞時間

  push.table <- push.table %>% #  合併推文資料
    bind_rows(
      tibble(
        tag = push.table.tag,
        author = push.table.author,
        content = push.table.content,
        datetime = push.table.datetime,
        url = article.url
      )
    )

}
article.table <- article.table %>% #  格式整理清除 NA
  mutate(
    datetime = str_sub(datetime, 5) %>% parse_datetime("%b %d %H:%M:%S %Y"),
    month = format(datetime, "%m"),
    day = format(datetime, "%d")
  ) %>%
  filter_all(
    all_vars(!is.na(.))
  )
push.table <- push.table %>% #  格式整理清除 NA
  mutate(
```

```
    datetime = str_c("2017/", datetime) %>% parse_datetime("%Y/%m/%d %H:%M"),
    month = format(datetime, "%m"),
    day = format(datetime, "%d")
  ) %>%
  filter_all(
    all_vars(!is.na(.))
  )
```

經過以上步驟，可得到 2 資料

1. article.table: 文章資料(標題、作者、時間、內文)
2. push.table: 推文資料(類型、作者、時間、內文)

接下來我們可以使用 jiebaR 這套件做斷詞如下，

```
library(jiebaR)
jieba.worker <- worker()
```

jieba.worker 是一個斷詞工具，可和 segment 搭配使用，有了斷詞工具後就可以來對每天八卦版的文章內容做斷詞，如下

```
article.date <- article.table %>%
  group_by(date) %>% # 以每日做分組
  do((function(input) {
    freq(segment(input$content, jieba.worker)) %>% # 斷詞後計算詞頻
      filter(
        !(char %in% toTrad(stopwordsCN())), # 過濾 stopword
        !str_detect(char, "[A-z0-9]"), # 過濾英文數字
        nchar(char) > 1 # 過濾單個字
      ) %>%
      arrange(desc(freq)) %>% # 以詞頻排序
      slice(1:100) %>% # 取前 100
      return
  })(.)) %>%
  ungroup
article.date.words <- freq(article.date$char) %>%
  rename(freq.all = freq)
```

最後可得每日出現最多次前 100 名的詞，我們藉此統計並過濾出每日特有的詞最後當日的前 5 大代表詞，程式如下

```
article.everyday <- article.date %>%
  left_join( # 比對全部詞
    article.date.words,
    by = 'char'
  ) %>%
  group_by(date) %>% # 以每日做分組
  arrange(freq.all) %>% # 每組的詞頻做排序由小到大
  slice(1:5) %>% # 取每組前 5
  summarise( # 合併詞並對詞頻加總
    char = str_c(char, collapse = ", "),
    freq = sum(freq)
  ) %>%
  ungroup
```

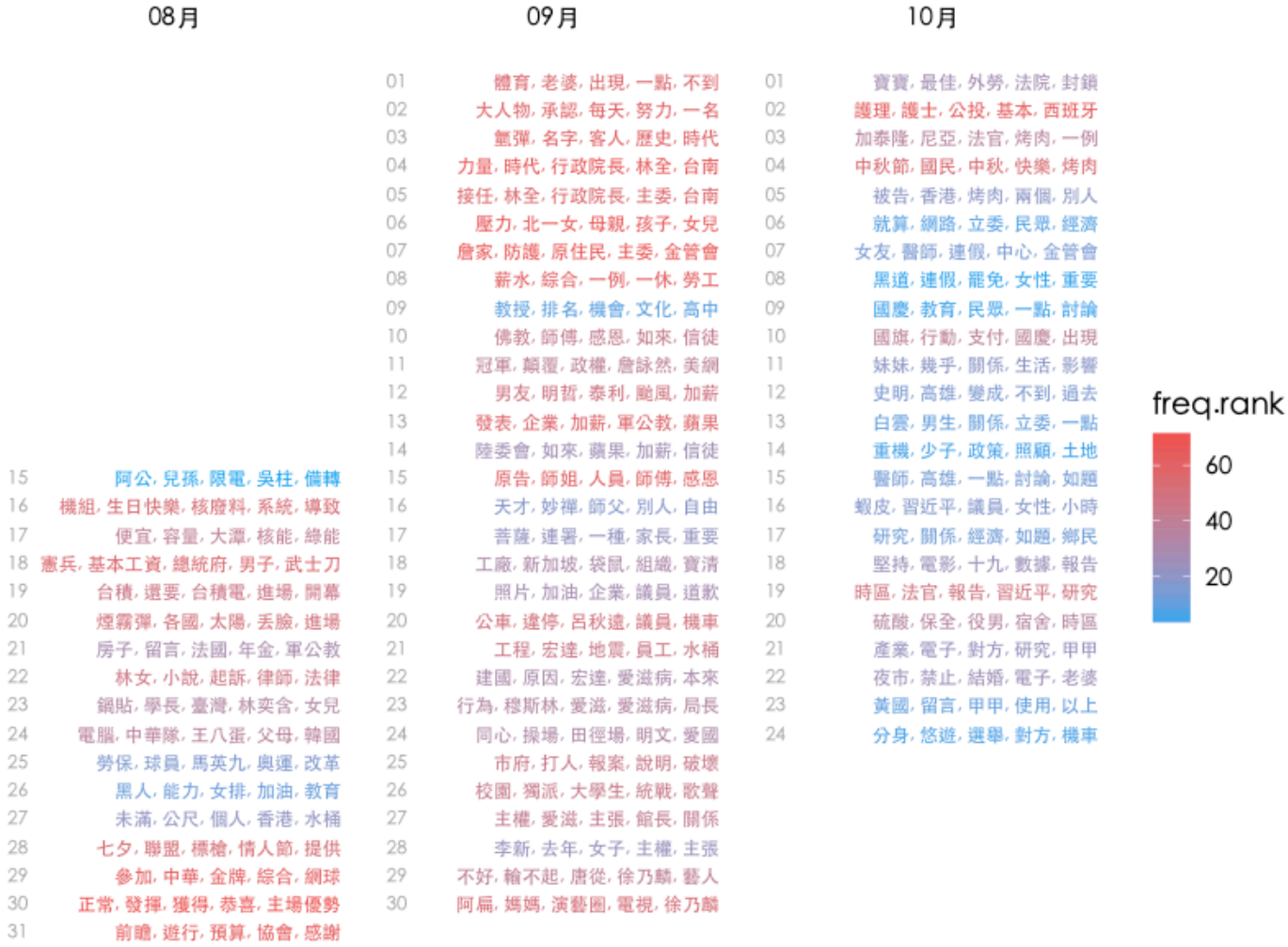
article.every 資料檢視前 10 筆

| 日期 | 詞頻前 5 | 詞頻加總 |
|---|---|---|
| 2017-10-24 | 分身，悠遊，選舉，對方，機車 | 453 |
| 2017-10-23 | 黃國，留言，甲甲，使用，以上 | 467 |
| 2017-10-22 | 夜市，禁止，結婚，電子，老婆 | 576 |
| 2017-10-21 | 產業，電子，對方，研究，甲甲 | 557 |
| 2017-10-20 | 硫酸，保全，役男，宿舍，時區 | 671 |
| 2017-10-19 | 時區，法官，報告，習近平，研究 | 820 |
| 2017-10-18 | 堅持，電影，十九，數據，報告 | 567 |
| 2017-10-17 | 研究，關係，經濟，如題，鄉民 | 475 |
| 2017-10-16 | 蝦皮，習近平，議員，女性，小時 | 519 |
| 2017-10-15 | 醫師，高雄，一點，討論，如題 | 479 |

最後我們可以以上資料做個簡單的圖表，並利用顏色做熱門程度的區別

```
article.everyday %>%
  mutate( # 計算月日和頻率排名
    month = str_c(format(date, "%m"), "月"),
    day = format(date, "%d") %>% parse_number(),
    freq.rank = rank(freq)
  ) %>%
  ggplot() +
  geom_text(
    aes(
```

```
      x = 1,
      y = day,
      label = char,
      color = freq.rank
    ),
    hjust = 1,
    size = 3
  ) +
  geom_text(
    aes(
      x = 0,
      y = day,
      label = format(date, "%d")
    ),
    hjust = 0,
    size = 3,
    alpha = 0.4
  ) +
  scale_color_continuous(low = "#03A9F4", high = "#EF5350") +
  scale_y_reverse() +
  facet_grid( ~ month) +
  theme_void()
```




可看到就整體資料 8~10 月部份，以 8 月底到 9 月初的關注最熱門，可能是受世大運的影響，基本上文字探勘還算是一項很新的技術，該怎麼定義什麼是熱門詞彙、什麼是代表詞彙的方法有很多，可以從各種面向去做篩選，要用什麼樣方式過濾也可以看個人的想法與創意。