Seminar Series

2019 Fall

한국통계학회 여성위원회 세미나

Part1. R ladies in Korea 2019-10-30

장소 : 이화여자대학교 종합과학관 D동 DB101호

주제 : rtweet 패키지를 사용한 트위터 크롤링과 시각화

발표자 : 정혜윤

R Ladies

KOREAN STATISTICAL SOCIETY 한국통계학회 1971

Seminar Series

CONTENTS

1. rtweet 패키지 소개
2. 타임라인 크롤링과 시각화
3. 팔로워 크롤링과 시각화
4. 키워드 크롤링과 시각화

https://www.r-bloggers.com/extracting-sydney-transport-data-from-twitter/

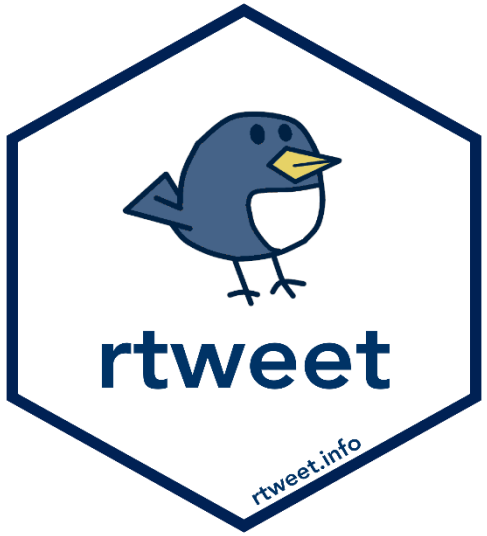
R Ladies

KOREAN STATISTICAL SOCIETY 한국통계학회 1971

rtweet 소개

Task	rtweet	twitteR	streamR	RTwitterAPI
Available on CRAN	✓	✓	✓	✗
Updated since 2016	✓	✗	✓	✗
Non-'developer' access	✓	✗	✗	✗
Extended tweets (280 chars)	✓	✗	✓	✗
Parses JSON data	✓	✓	✓	✗
Converts to data frames	✓	✓	✓	✗
Automated pagination	✓	✗	✗	✗
Search tweets	✓	✓	✗	?
Search users	✓	✗	✗	?
Stream sample	✓	✗	✓	✗
Stream keywords	✓	✗	✓	✗
Stream users	✓	✗	✓	✗
Get friends	✓	✓	✗	✓
Get timelines	✓	✓	✗	?
Get mentions	✓	✓	✗	?
Get favorites	✓	✓	✗	?
Get trends	✓	✓	✗	?
Get list members	✓	✗	✗	?
Get list memberships	✓	✗	✗	?
Get list statuses	✓	✗	✗	?

Task	rtweet	twitteR	streamR	RTwitterAPI
Get list subscribers	✓	✗	✗	?
Get list subscriptions	✓	✗	✗	?
Get list users	✓	✗	✗	?
Lookup collections	✓	✗	✗	?
Lookup friendships	✓	✓	✗	?
Lookup statuses	✓	✓	✗	?
Lookup users	✓	✓	✗	?
Get retweeters	✓	✓	✗	?
Get retweets	✓	✓	✗	?
Post tweets	✓	✓	✗	✗
Post favorite	✓	✗	✗	✗
Post follow	✓	✗	✗	✗
Post messsage	✓	✓	✗	✗
Post mute	✓	✗	✗	✗
Premium 30 day	✓	✗	✗	✗
Premium full archive	✓	✗	✗	✗
Run package tests	✓	✗	✗	✗



현존하는 R 트위터 크롤링 패키지 중에
가장 많은 기능을 수행 가능한 패키지

SydStats 크롤링과 시각화

매일 시드니의 열차 지연 정보 통계를 업로드 하는 트위터

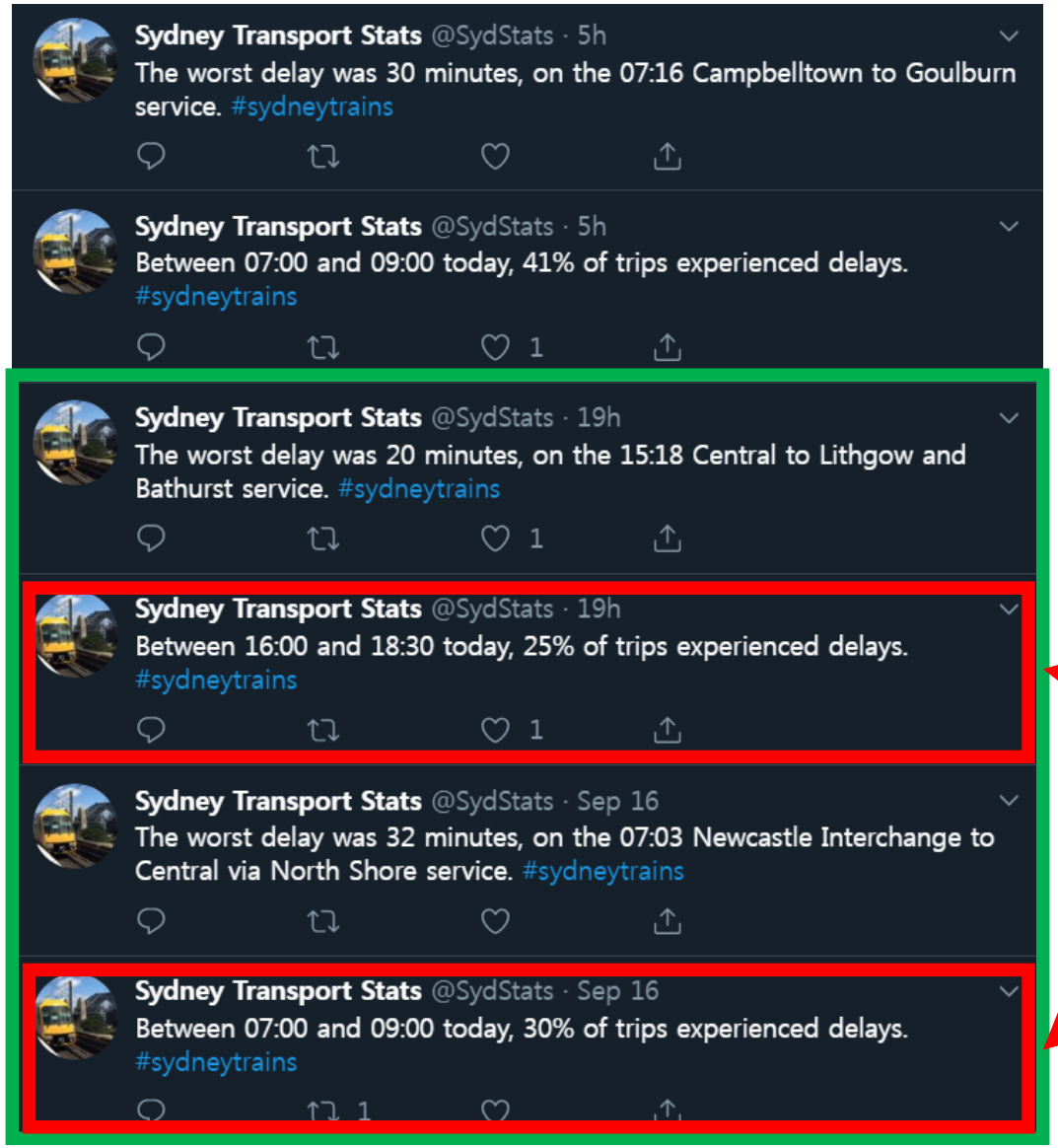
Sydstats 계정의 타임라인을 크롤링**(rtweet)**

트윗의 업로드 트윗 내용(지연된 역, 지연된 시간, 지연 확률)를 포함한 데이터 셋으로 정제**(tidyverse, knitr, lubridate)**

데이터셋 시각화 **(ggplot2)**

Sydstats계정에는 오전2개, 오후2개 매일 같은 양식으로 총 4개의 트윗이 업로드 된다.

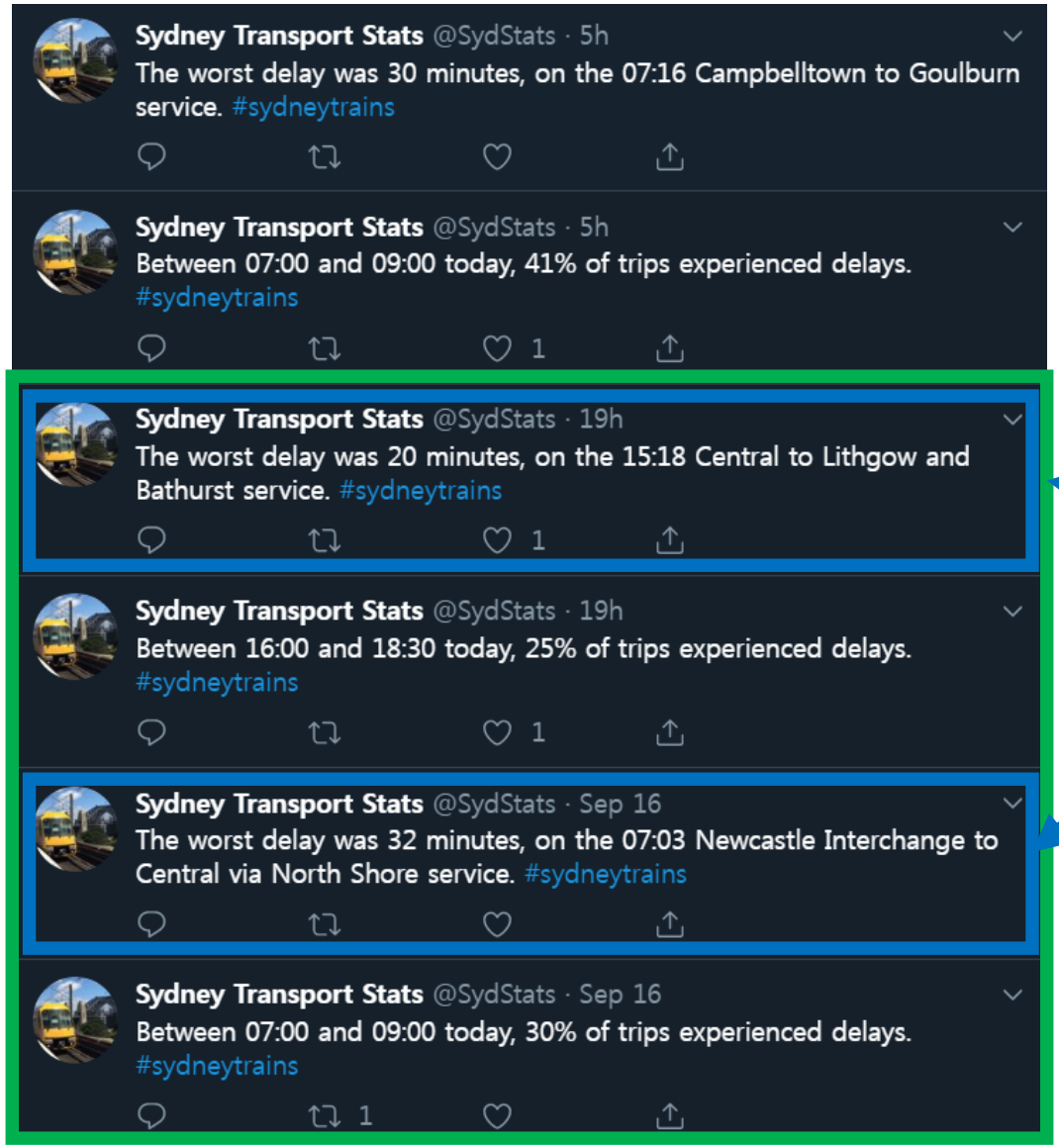
트윗(Tweet)은 각각의 글들을 의미한다.



Sydstats계정에는 오전2개, 오후2개 매일 같은 양식으로 총 4개의 트윗이 업로드 된다.

첫번째 양식

출근시간(7:00~9:00)과 퇴근시간(16:00~18:30)에 몇 퍼센트의 열차가 연착되는지 나타낸 트윗 → **데이터1**



Sydstats계정에는 오전2개, 오후2개 매일 같은 양식으로 총 4개의 트윗이 업로드 된다.

두번째 양식

오전과 오후에 각각 가장 많이 연착된 노선 구간과 연착 시간을 나타낸 트윗 → **데이터2**

SydStats 크롤링

```
32 library(rtweet)
33 sydstats <- get_timeline("SydStats", n = 3200) #n은 최대 행의 갯수
```

```
> dim(sydstats)
[1] 1275   90
```


```
> colnames(sydstats)
 [1] "user_id"                "status_id"              "created_at"
 [4] "screen_name"            "text"                   "source"
 [7] "display_text_width"     "reply_to_status_id"     "reply_to_user_id"
[10] "reply_to_screen_name"   "is_quote"               "is_retweet"
[13] "favorite_count"         "retweet_count"          "quote_count"
[16] "reply_count"            "hashtags"               "symbols"
[19] "urls_url"               "urls_t.co"              "urls_expanded_url"
[22] "media_url"              "media_t.co"             "media_expanded_url"
[25] "media_type"             "ext_media_url"          "ext_media_t.co"
[28] "ext_media_expanded_url" "ext_media_type"         "mentions_user_id"
[31] "mentions_screen_name"   "lang"                   "quoted_status_id"
[34] "quoted_text"            "quoted_created_at"      "quoted_source"
[37] "quoted_favorite_count"  "quoted_retweet_count"   "quoted_user_id"
[40] "quoted_screen_name"     "quoted_name"            "quoted_followers_count"
[43] "quoted_friends_count"   "quoted_statuses_count"  "quoted_location"
[46] "quoted_description"     "quoted_verified"        "retweet_status_id"
[49] "retweet_text"           "retweet_created_at"     "retweet_source"
[52] "retweet_favorite_count" "retweet_retweet_count"  "retweet_user_id"
[55] "retweet_screen_name"    "retweet_name"           "retweet_followers_count"
[58] "retweet_friends_count"  "retweet_statuses_count" "retweet_location"
[61] "retweet_description"    "retweet_verified"       "place_url"
[64] "place_name"             "place_full_name"        "place_type"
[67] "country"                "country_code"           "geo_coords"
[70] "coords_coords"          "bbox_coords"            "status_url"
[73] "name"                   "location"               "description"
[76] "url"                    "protected"              "followers_count"
[79] "friends_count"          "listed_count"           "statuses_count"
[82] "favourites_count"       "account_created_at"     "verified"
[85] "profile_url"            "profile_expanded_url"   "account_lang"
[88] "profile_banner_url"     "profile_background_url" "profile_image_url"
```

타임라인에 올라온 1275개의 트윗을 가져올 수 있고 각 트윗의 정보가 90개의 변수에 저장된다.

SydStats 크롤링

아래와 같이 트윗들이 잘 저장되었다.

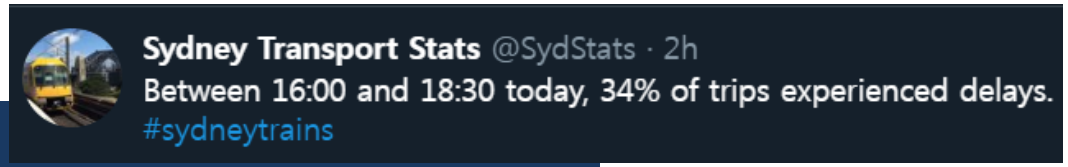
```
> head(sydstats$text)
[1] "The worst delay was 6 minutes, on the 08:24 City to Berowra via Gordon service.
 #sydneytrains"
[2] "Between 07:00 and 09:00 today, 31% of trips experienced delays. #sydneytrains"

[3] "The worst delay was 26 minutes, on the 15:59 Macarthur to City Circle via Airpor
t service. #sydneytrains"
[4] "Between 16:00 and 18:30 today, 30% of trips experienced delays. #sydneytrains"

[5] "The worst delay was 65 minutes, on the 06:24 Leppington to City Circle via Granv
ille service. #sydneytrains"
[6] "Between 07:00 and 09:00 today, 36% of trips experienced delays. #sydneytrains"
```

데이터1) 출근 시간과 퇴근시간에 몇 퍼센트의 열차가 지연되는지

크롤링한 데이터 정제

❖ 〈주의〉 시드니의 시간은 utc시간 보다 10시간 빠르다!

```
sydstats$created_at<-sydstats$created_at+hours(10)
```




```
delays_data <- sydstats %>%
  #가장 앞에Between이 들어가고 2019년인 타임라인출력
  filter(grepl("^Between", text),
         year(created_at) == 2019) %>%
  select(created_at, text) %>% #created_at(작성시간),text(트윗내용)만 포함한 데이터
  mutate(Date = as.Date(created_at),
         Start = str_match(text, "^Between\\s+(.*?)\\s+")[, 2], # 시작 시간
         End = str_match(text, "and\\s+(.*?)\\s+today")[, 2], # 끝 시간
         delayed = str_match(text, "\\s+(\\d+)%")[, 2] %>% as.numeric(), #연착율
         dtstart = ymd_hm(paste(Date, Start)), #연월일(작성시간)+시작시간
         dtend = ymd_hm(paste(Date, End)), #연월일(작성시간)+끝시간
         ystart = year(Date), #년도(작성시간)
         mstart = month(dtstart), #월
         wstart = isoweek(dtstart), #주(2019년의 몇번째 주인지)
         dstart = factor(wday(dtstart, label = TRUE, week_start = 1)),#요일
         peak = factor(ifelse(hour(dtstart) == 7, "morning", "afternoon"))) %>%
  select(dtstart, dtend, ystart, mstart, wstart, dstart, delayed, peak)
```

#Between으로 맨 앞부터 시작(^)하며
2019년에 작성된 트윗만 filter함수를 이용하여 선택

str_match :매칭 문자열 추출 및 행,열 반환(stringr 패키지)

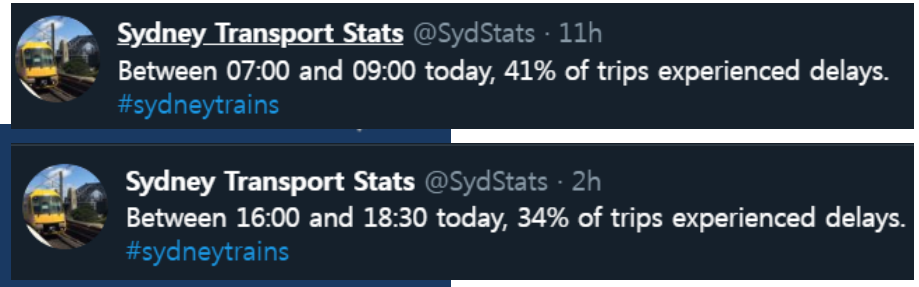
```
> a
[1] "Between 16:00 and 18:30 today, 33% of trips experienced delays."
> str_match(a, "^Between\\s+(.*?)\\s+")  #Between 공백 적어도 한번 (임의 개수의 임의문자 많아야 한번) 공백 적어도 한번
     [,1]              [,2]
[1,] "Between 16:00 " "16:00"
> str_match(a, "and\\s+(.*?)\\s+today")   #and 공백이 적어도 한번 (임의 개수의 임의문자 많아야 한번) 공백 적어도 한번 today
     [,1]               [,2]
[1,] "and 18:30 today" "18:30"
> str_match(a, "and\\s+(.*.)\\s+(today)") #and 공백이 적어도 한번 (임의 개수의 임의문자 많아야 한번) 공백 적어도 한번(today)
     [,1]               [,2]    [,3]
[1,] "and 18:30 today" "18:30" "today"
> str_match(a, "\\s+(\\d+)%")             #공백 적어도 한번  (숫자가 적어도 한개) %
     [,1]   [,2]
[1,] " 33%" "33"
```

```
#^: 문자열 시작 위치를 매칭
#\\s : 간격, ` `
#+: 적어도 1 번 매칭한다.(+ 앞에 있는게)
#?:많아야 한번 매칭된다.(? 앞에 있는게)
#.*:임의 문자를 임의 갯수만큼 매칭한다
#[:digit:] 혹은 \\d : 숫자, 0,1,2,3,4,5,6,7,8,9, 동등한 표현 [0-9]
```

R 정규표현식: https://statkclee.github.io/ds-authoring/regex-r.html

크롤링한 데이터 정제

데이터1) 출근 시간과 퇴근시간에 몇 퍼센트의 열차가 지연되는지




```
delays_data <- sydstats %>%
  #가장 앞에Between이 들어가고 2019년인 타임라인출력
  filter(grepl("^Between", text),
         year(created_at) == 2019) %>%
  select(created_at, text) %>% #created_at(작성시간),text(트윗내용)만 포함한 데이터
  mutate(Date = as.Date(created_at),
         Start = str_match(text, "^Between\\s+(.*?)\\s+")[, 2], # 시작 시간
         End = str_match(text, "and\\s+(.*?)\\s+today")[, 2], # 끝 시간
         delayed = str_match(text, "\\s+(\\d+)%")[, 2] %>% as.numeric(), #연착율
         dtstart = ymd_hm(paste(Date, Start)), #연월일(작성시간)+시작시간
         dtend = ymd_hm(paste(Date, End)), #연월일(작성시간)+끝시간
         ystart = year(Date), #년도(작성시간)
         mstart = month(dtstart), #월
         wstart = isoweek(dtstart), #주(2019년의 몇번째 주인지)
         dstart = factor(wday(dtstart, label = TRUE, week_start = 1)),#요일
         peak = factor(ifelse(hour(dtstart) == 7, "morning", "afternoon"))) %>%
  select(dtstart, dtend, ystart, mstart, wstart, dstart, delayed, peak)
```

데이터1) 출근 시간과 퇴근시간에 몇 퍼센트의 열차가 지연되는지

크롤링한 데이터 정제

```
> delays_data %>% slice(1:6)
# A tibble: 6 x 8
  dtstart             dtend               ystart mstart wstart dstart delayed peak
  <dttm>              <dttm>               <dbl>  <dbl>  <dbl> <ord>    <dbl> <fct>
1 2019-10-25 07:00:00 2019-10-25 09:00:00   2019     10     43 금          31 morning
2 2019-10-24 16:00:00 2019-10-24 18:30:00   2019     10     43 목          30 afternoon
3 2019-10-24 07:00:00 2019-10-24 09:00:00   2019     10     43 목          36 morning
4 2019-10-23 16:00:00 2019-10-23 18:30:00   2019     10     43 수          33 afternoon
5 2019-10-23 07:00:00 2019-10-23 09:00:00   2019     10     43 수          34 morning
6 2019-10-22 16:00:00 2019-10-22 18:30:00   2019     10     43 화          30 afternoon
```

```
> tail(delays_data)
# A tibble: 6 x 8
  dtstart             dtend               ystart mstart wstart dstart delayed peak
  <dttm>              <dttm>               <dbl>  <dbl>  <dbl> <ord>    <dbl> <fct>
1 2019-01-03 16:00:00 2019-01-03 18:30:00   2019      1      1 목          17 afternoon
2 2019-01-03 07:00:00 2019-01-03 09:00:00   2019      1      1 목          23 morning
3 2019-01-02 16:00:00 2019-01-02 18:30:00   2019      1      1 수          25 afternoon
4 2019-01-02 07:00:00 2019-01-02 09:00:00   2019      1      1 수          22 morning
5 2019-01-01 16:00:00 2019-01-01 18:30:00   2019      1      1 화          25 afternoon
6 2019-01-01 07:00:00 2019-01-01 09:00:00   2019      1      1 화          45 morning
```

크롤링한 데이터 정제

데이터2) 오전,오후에 각각 가장 많이 지연된 노선구간과 연착시간


```
service_data <- sydstats %>%
  filter(grepl("^The", text), #The로 시작하는 데이터
         year(created_at) == 2019) %>% #2019년에 작성된 트윗
  select(created_at, text) %>% #created_at(작성시간),text(트윗내용)만 포함한 데이터
  mutate(Date = as.Date(created_at), #작성날짜
         Delay = str_match(text, "was\\s+(.*?)\\s+minutes")[, 2] %>% as.numeric(),#지연시간
         dtime = str_match(text, "\\s+the\\s+(.*?)\\s+")[, 2], #열차 시간
         service = str_match(text, ":\\d+\\s+(.*?)\\s+service")[, 2]) #노선 구간
```

데이터2) 오전,오후에 각각 가장 많이 지연된 노선구간과 연착시간

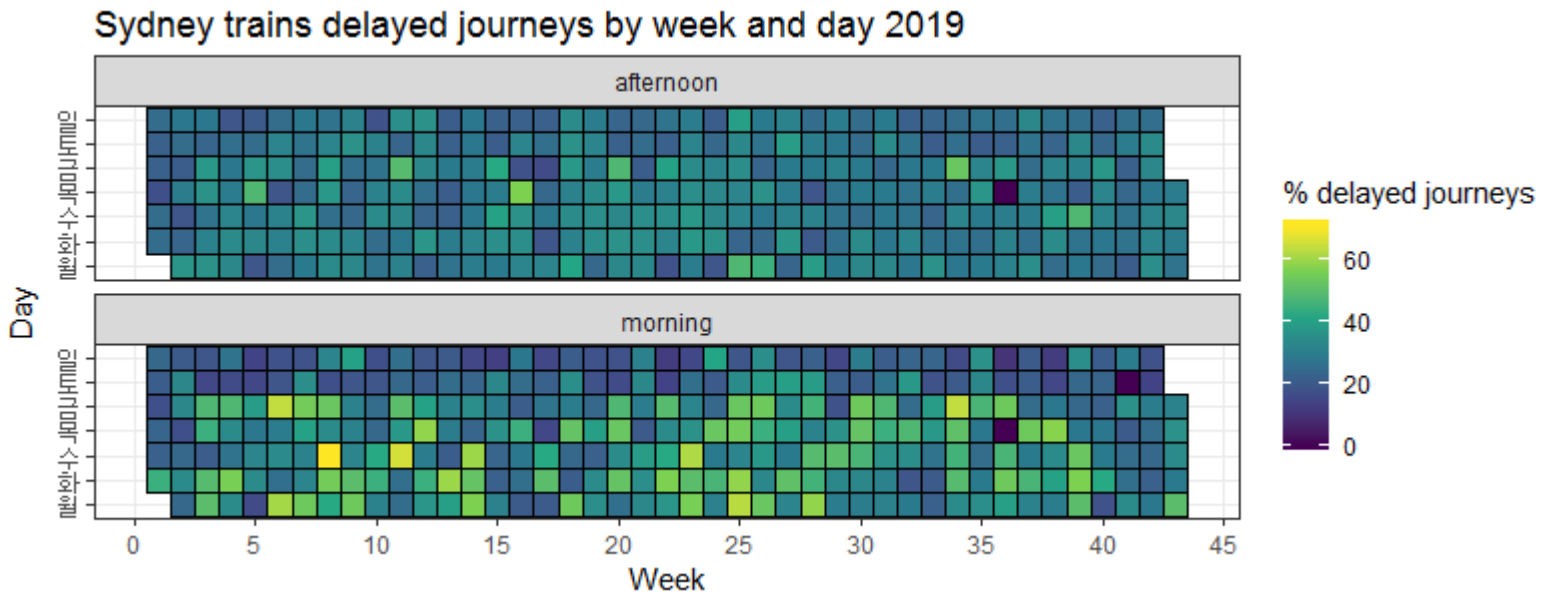
크롤링한 데이터 정제

```
> service_data %>% select(-text) %>% slice(1:6)
# A tibble: 6 x 5
  created_at          Date       Delay dtime service
  <dttm>              <date>     <dbl> <chr> <chr>
1 2019-10-25 08:08:05 2019-10-25     6 08:24 City to Berowra via Gordon
2 2019-10-24 17:38:53 2019-10-24    26 15:59 Macarthur to City Circle via Airport
3 2019-10-24 08:08:02 2019-10-24    65 06:24 Leppington to City Circle via Granville
4 2019-10-23 17:38:39 2019-10-23    57 14:56 City to Emu Plains
5 2019-10-23 08:07:58 2019-10-23    11 06:49 Campbelltown to Goulburn
6 2019-10-22 17:38:30 2019-10-22     7 15:09 City to Berowra via Gordon
> tail(service_data %>% select(-text))
# A tibble: 6 x 5
  created_at          Date       Delay dtime service
  <dttm>              <date>     <dbl> <chr> <chr>
1 2019-01-03 17:52:17 2019-01-03    12 15:12 Newcastle Interchange to Central via Strathfield
2 2019-01-03 08:20:19 2019-01-03    29 05:02 Newcastle Interchange to Central via Strathfield
3 2019-01-02 17:56:46 2019-01-02    10 14:15 Central to Newcastle Interchange via Strathfield
4 2019-01-02 08:23:44 2019-01-02     3 07:13 Lidcombe to City Circle via Bankstown
5 2019-01-01 17:46:29 2019-01-01    44 16:12 Berowra and Hornsby to City
6 2019-01-01 08:11:57 2019-01-01   135 06:23 Leppington to Richmond
```

크롤링 데이터 시각화

```
103 delays_data %>%
104   ggplot(aes(wstart, dstart)) +
105   geom_tile(aes(fill = delayed), color = "black") +
106   scale_fill_viridis_c(name = "% delayed journeys") +
107   scale_x_continuous(breaks = seq(0, 52, 5)) +
108   facet_wrap(~peak, ncol = 1) +
109   coord_equal() +
110   labs(x = "Week",
111        y = "Day",
112        title = "Sydney trains delayed journeys by week and day 2019")
```

데이터1(열차 지연율)을 바탕으로 **geom_tile**함수를 사용하여 일별 지연율을 나타낼 수 있다.

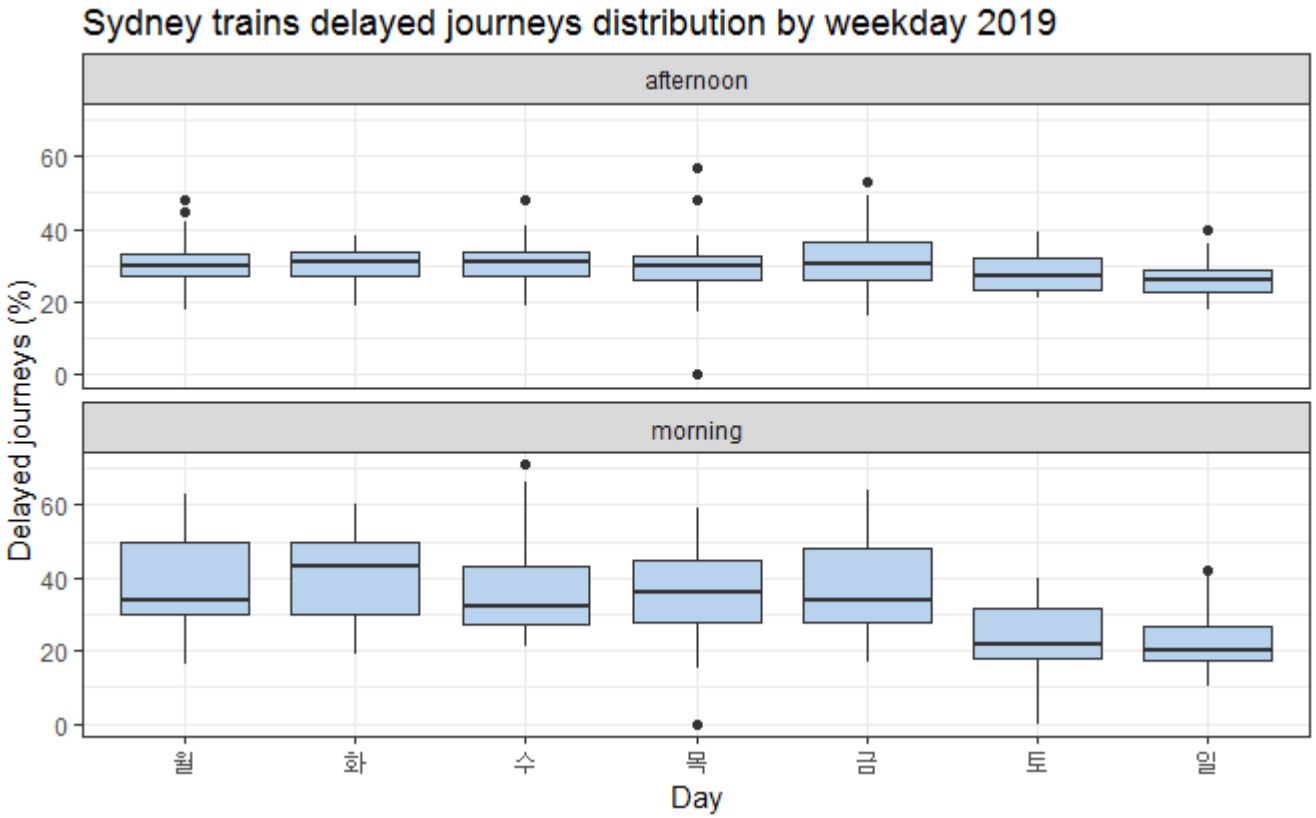


- ✓ 오전이 오후보다 지연율이 높음
- ✓ 토요일과 일요일이 지연율이 특히 낮음

크롤링 데이터 시각화

데이터1(열차 지연율)을 바탕으로 요일별 지연율을 박스 플롯으로 나타낼 수 있다.

```
ggplot(data=delays_data,aes(dstart, delayed)) +
  geom_boxplot(fill = "slategray2") +
  facet_wrap(~peak, ncol = 1) +
  labs(x = "Day",
       y = "Delayed journeys (%)",
       title = "Sydney trains delayed journeys distribution by weekday 2019")
```

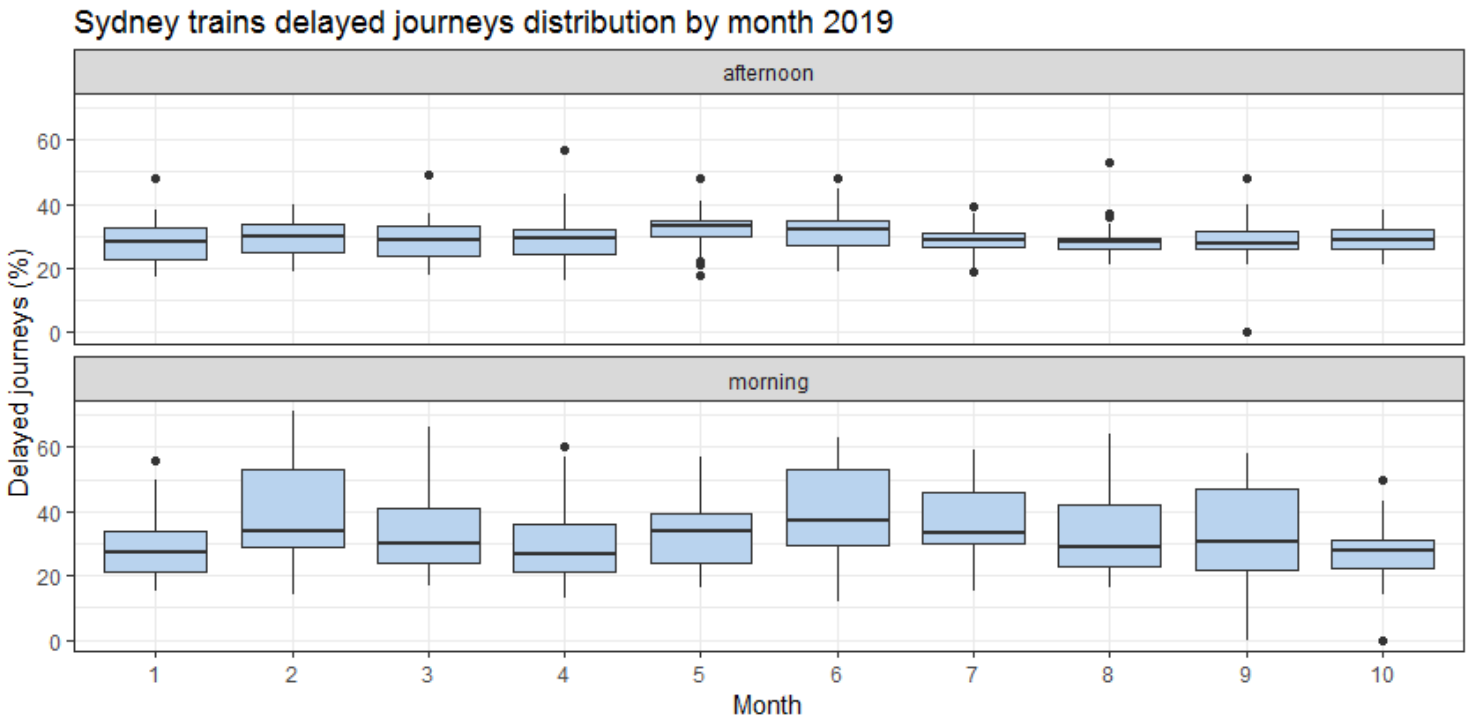


- ✓ 오전이 오후보다 지연율이 높음
- ✓ 토요일과 일요일이 지연율이 특히 낮음

크롤링 데이터 시각화

```
delays_data %>%
  mutate(mstart = month(dtstart, label = TRUE)) %>%  # month를 수치=>요인으로 변경
  ggplot(aes(mstart, delayed)) +
  geom_boxplot(fill = "slategray2") +
  facet_wrap(~peak, ncol = 1) +
  labs(x = "Month",
       y = "Delayed journeys (%)",
       title = "Sydney trains delayed journeys distribution by month 2019")
```

데이터1(열차 지연율)을 바탕으로 월별 지연율을 박스 플롯으로 나타낼 수 있다.

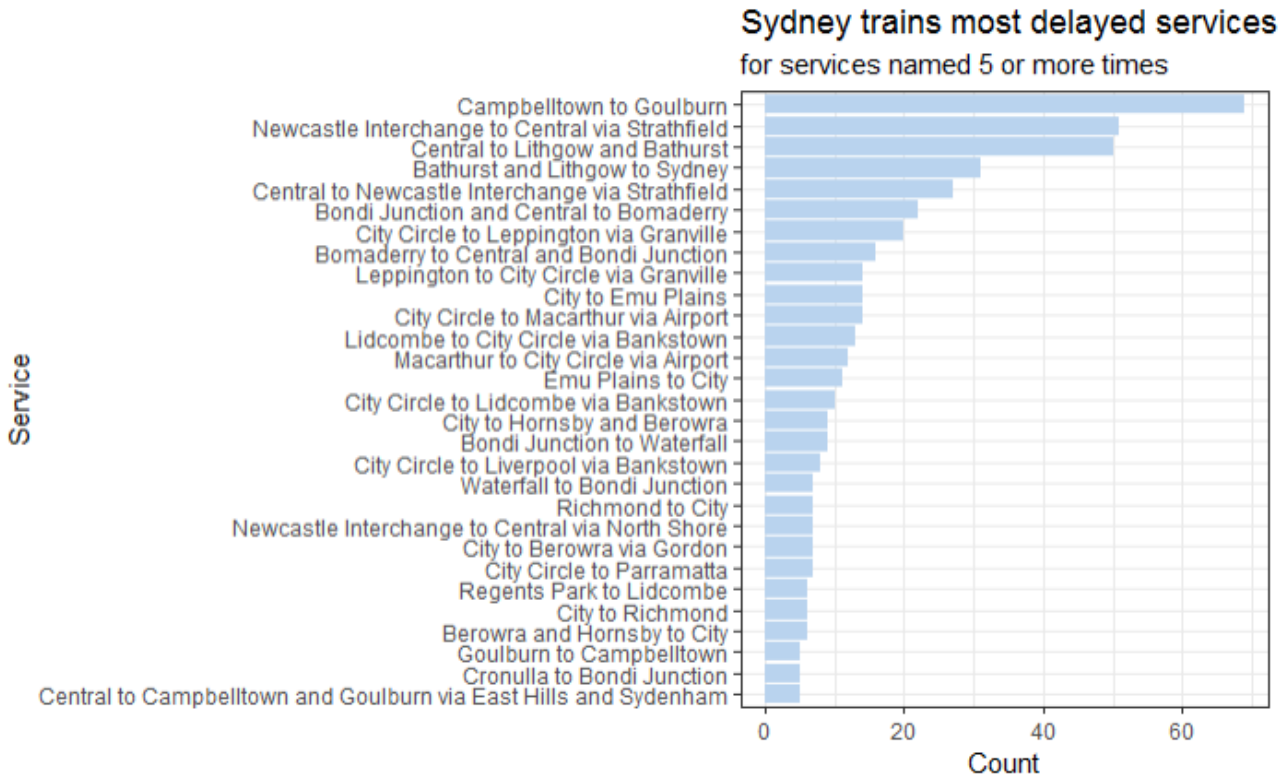


- ✓ 오전이 오후보다 지연율이 높음
- ✓ 오전의 경우 2, 6월에 지연율이 높음

크롤링 데이터 시각화

```
151 service_data %>%
152   count(service, sort = TRUE) %>% #구간별로 몇번 지연됐는지
153   filter(n > 4) %>% #4번 이상 지연된 구간만 저장
154   ggplot(aes(reorder(service, n), n)) +
155   geom_col(fill = "slategray2") + coord_flip() +
156   labs(x = "Service",y = "Count",
157        title = "Sydney trains most delayed services 2019",
158        subtitle = "for services named 5 or more times")
```

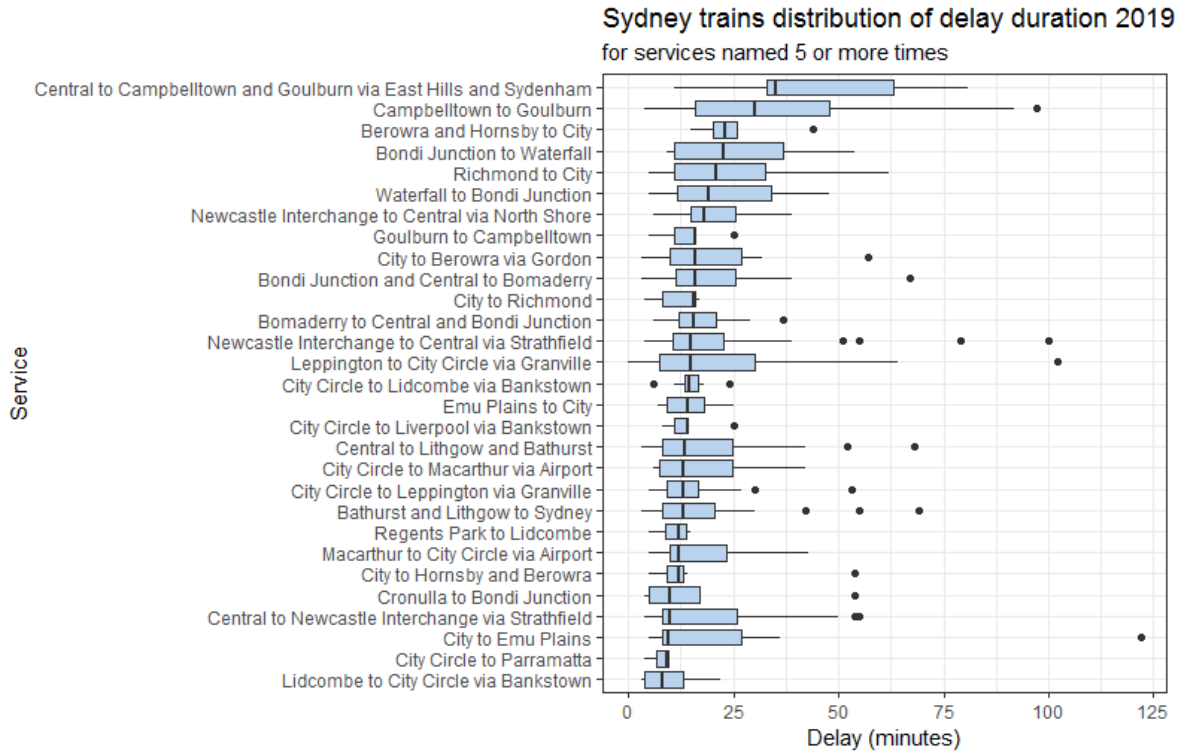
데이터2(일별로 많이 지연된 노선구간)을 바탕으로 노선 구간별 지연건수를 막대 그래프로 볼 수 있다.



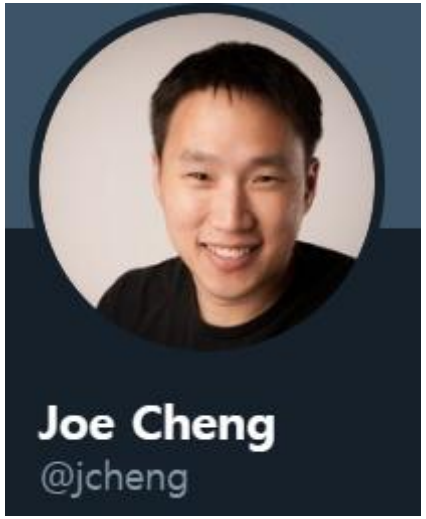
크롤링 데이터 시각화

```
164 service_data %>%
165   group_by(service) %>% filter(n() > 4) %>% #4번 이상 지연된 구간만 저장
166   ungroup() %>%
167   ggplot(aes(reorder(service, Delay, median), Delay)) +
168   geom_boxplot(fill = "slategray2") +
169   coord_flip() +
170   labs(x = "Service", y = "Delay (minutes)",
171        title = "Sydney trains distribution of delay duration 2019",
172        subtitle = "for services named 5 or more times")
```

데이터2(일별로 많이 지연된 노선구간)을 바탕으로 노선 구간별 지연시간 박스 플롯으로 볼 수 있다.



2. 팔로워 크롤링과 시각화



R 전문가와 R 관련 단체들의 팔로워를 크롤링**(rtweet)**

↓

각 팔로워의 ID와 각 ID의 팔로우 정보 데이터셋 생성**(tidyverse)**

↓

얻어온 정보로 시각화 **(UpSetR)**

팔로워 크롤링

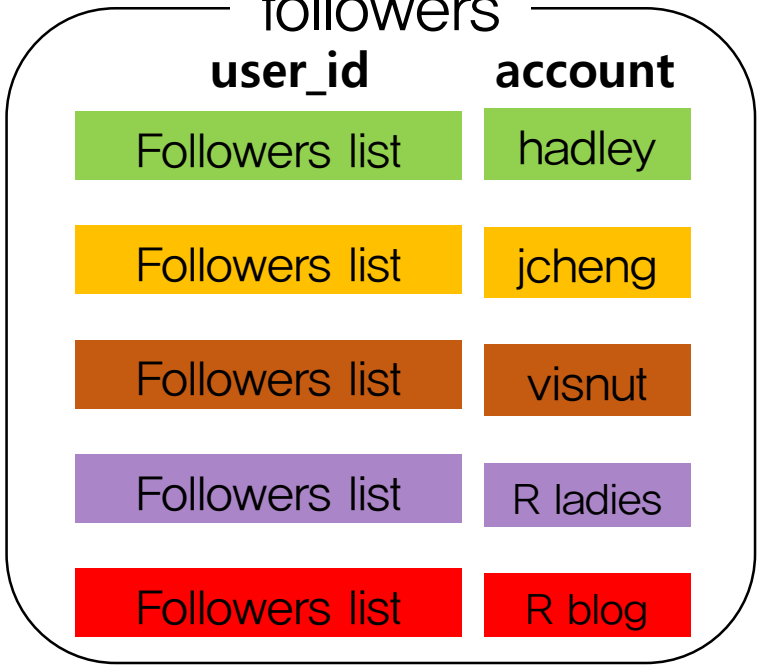
```r
#r 전문가 및 관계인
rstaters <- c("hadleywickham", "jcheng","visnut", "RLadiesGlobal", "Rbloggers")

#map_df는 데이터 프레임을 만드는 함수
followers <- map_df(rstaters, ~ get_followers(.x, n = 100000, retryonratelimit = TRUE)
                    %>% mutate(account = .x))
```

#트위터 주소

리턴할 팔로워 수(15분에 75000명)

출력 처리과정 출력 여부

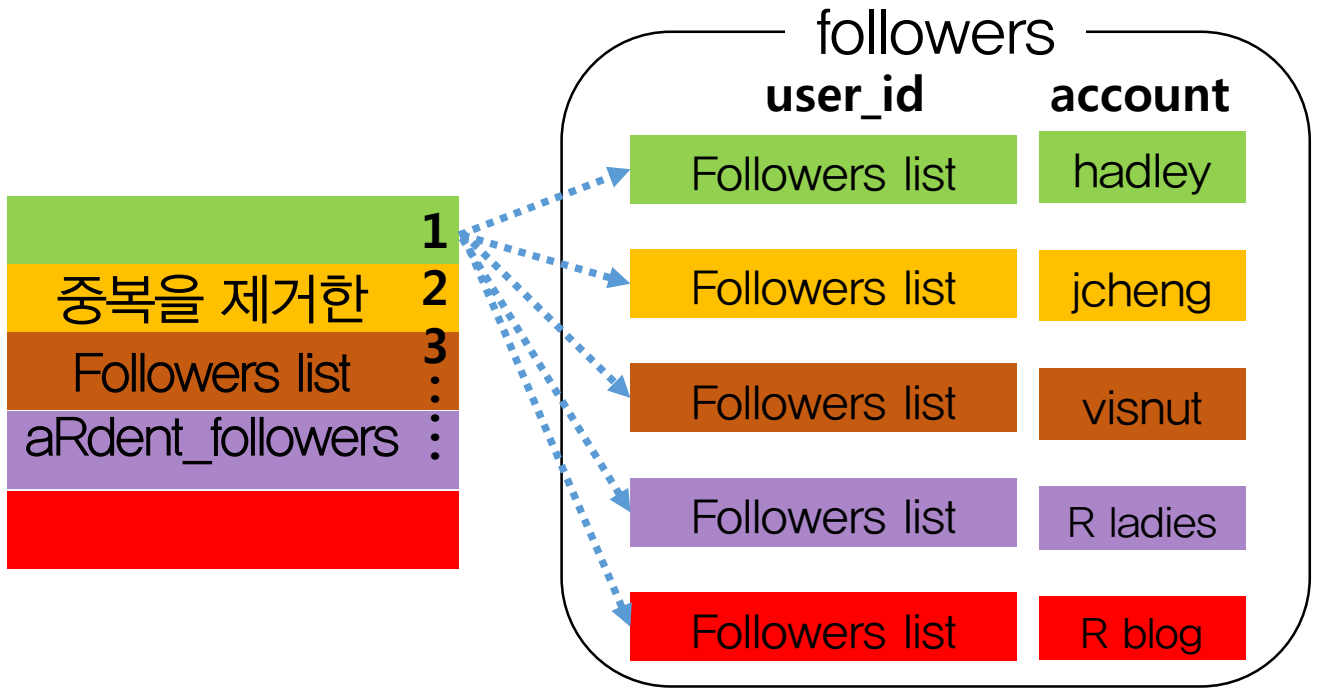
```
> followers
# A tibble: 196,200 x 2
   user_id             account
   <chr>               <chr>
 1 1140501638219620352 hadleywickham
 2 210805768           hadleywickham
 3 97988992            hadleywickham
 4 901976761           hadleywickham
 5 1174113347496624128 hadleywickham
 6 2858749436          hadleywickham
 7 983404853513932800  hadleywickham
 8 853476476           hadleywickham
 9 56046092            hadleywickham
10 2738875572          hadleywickham
# ... with 196,190 more rows
```



크롤링한 데이터 정제

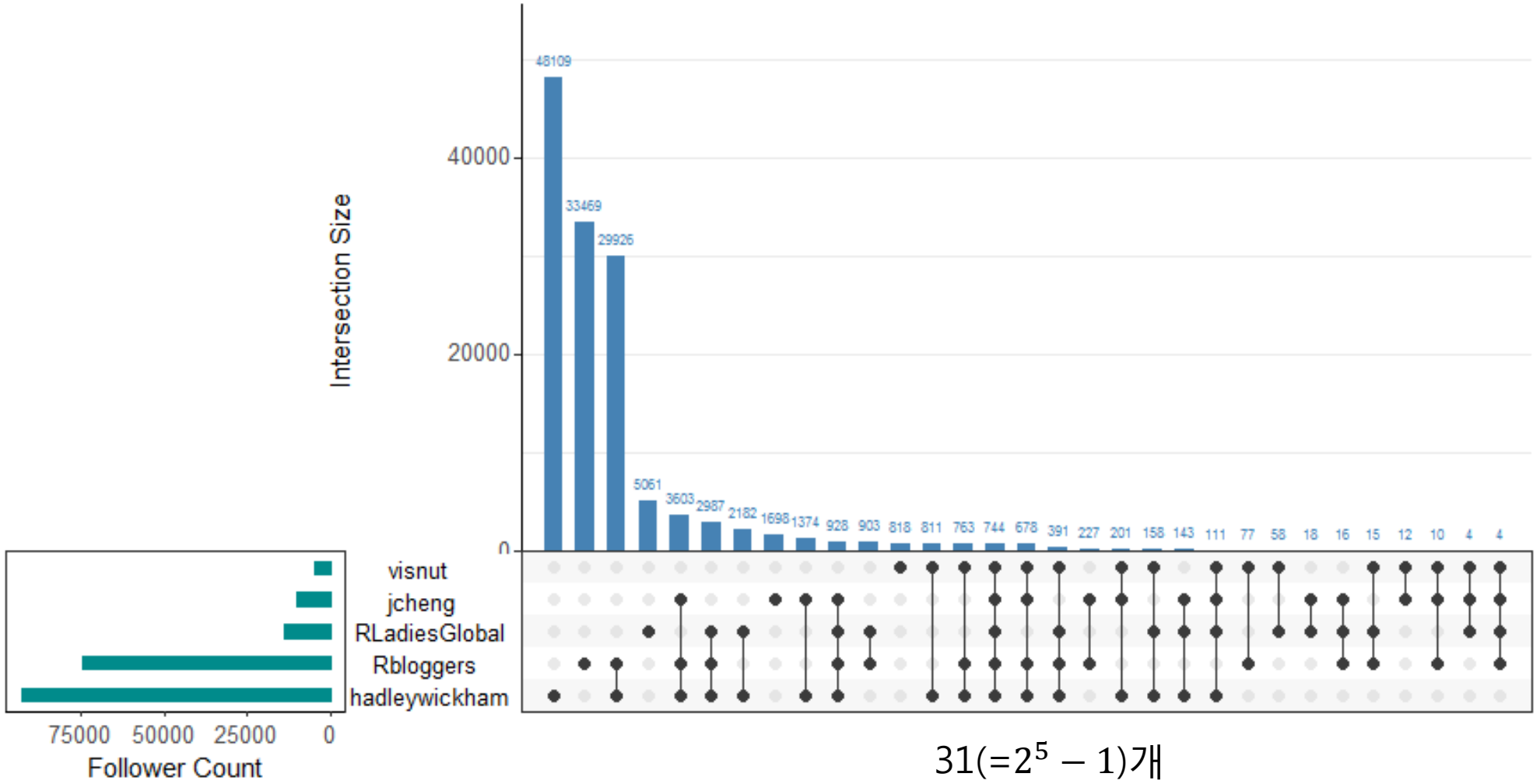
```r
# 5개의 계정의 팔로워 목록 중 중복을 제거
aRdent_followers <- unique(followers$user_id)
#각 팔로워가 해당 계정을 팔로우 하면 1 아니면 0을 출력하도록 함
binaries <- rstaters %>%
  map_dfc(~ ifelse(aRdent_followers %in% filter(followers, account == .x)$user_id, 1, 0) %>%
            as.data.frame) #데이터 프레임으로 변환
# 열이름 지정
names(binaries) <- rstaters
```

```
> binaries
   hadleywickham jcheng visnut RLadiesGlobal Rbloggers
1              1      0      0             0         0
2              1      0      0             0         0
3              1      0      0             0         1
4              1      0      0             0         0
5              1      0      0             0         1
6              1      0      0             1         1
7              1      0      0             0         1
8              1      0      0             0         0
9              1      0      0             0         0
10             1      0      0             0         0
11             1      0      0             0         0
12             1      0      0             0         1
13             1      0      0             0         0
14             1      0      0             0         0
15             1      0      0             0         0
16             1      0      0             0         0
17             1      0      0             0         0
18             1      0      0             0         0
```



크롤링한 데이터 시각화

```
41 upset(binaries, nsets = 5, main.bar.color = "SteelBlue", sets.bar.color = "DarkCyan",
42       sets.x.label = "Follower Count", text.scale = c(rep(1.4, 5), 1), order.by = "freq")
```



$31(=2^5-1)$개

3. 키워드 크롤링과 시각화

토큰 필요!!

↓

“koo” 가 들어간 트윗을 크롤링
(rtweet)

↓

각 트윗 작성자의 위도,경도 출력
(rtweet)

↓

얻어온 위치 정보로 시각화
(ggmap,ggplot2)

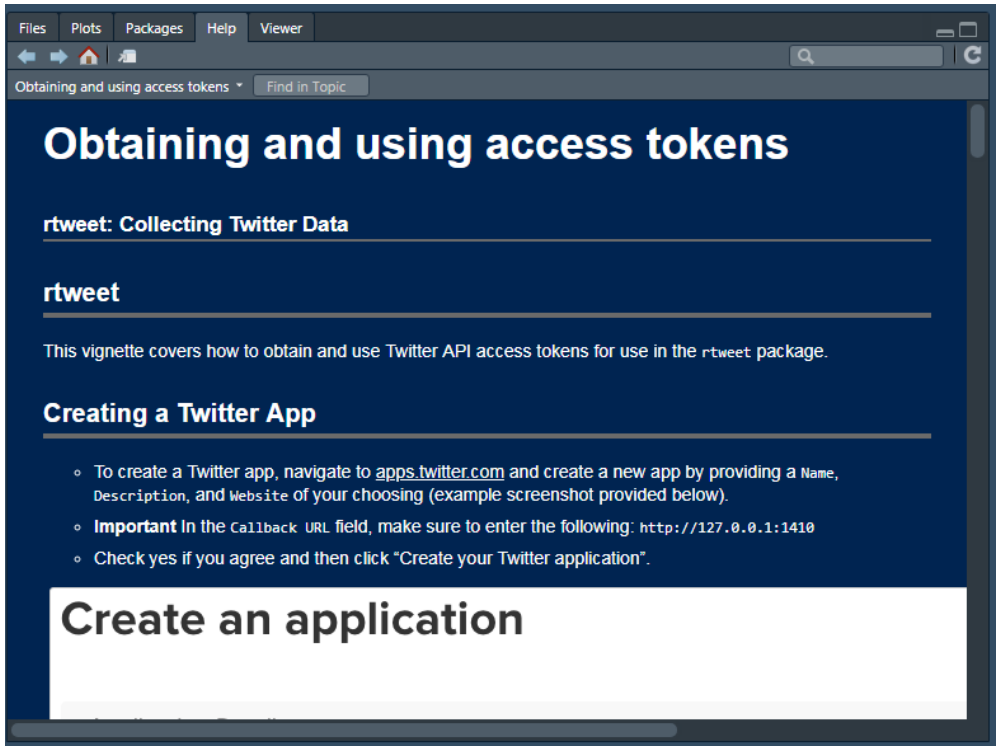
토큰 받는법

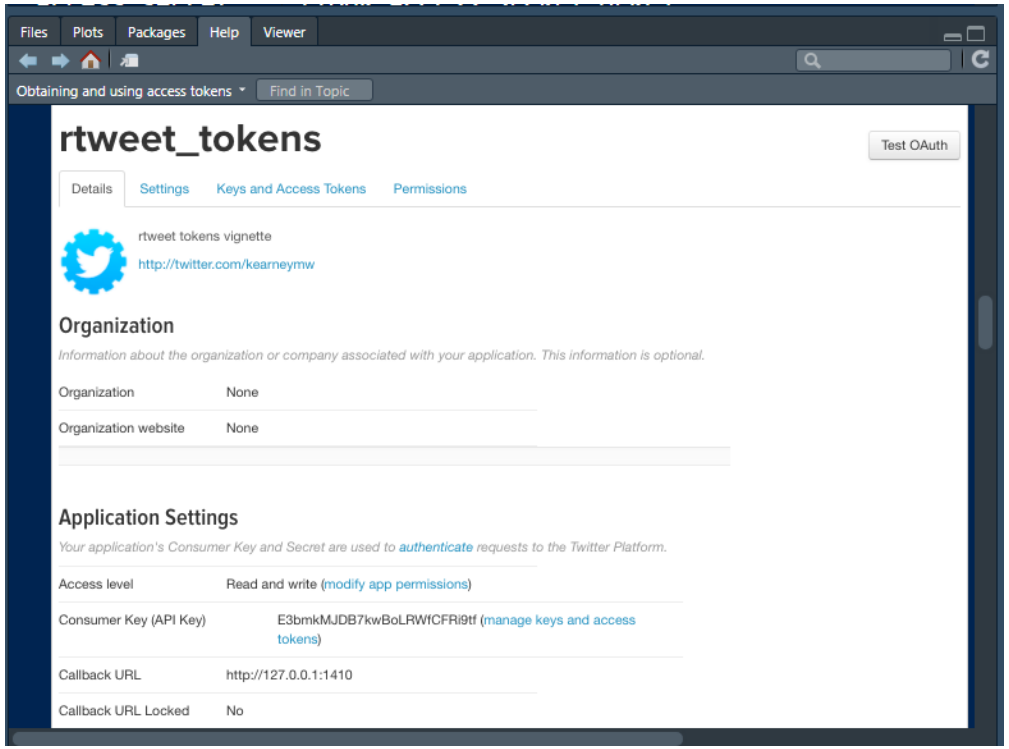
```
47 create_token(consumer_key = "xMKt1o
48 consumer_secret = "CmSxtalZaOB9YJgE
49 access_token  = "117235794626666496
50 access_secret = "XSiIWmqYxG6fQLrs2J
```

R Ladies

```
WhOAZJr ",
GIVrVKI7NiAKRHWr6poGcm",
wtlRxOW01hbFtinoyJm8",
u7duj9cDwJ6hOJz")
```

`Vignette("auth", package = "rtweet")` ✓ Help창에 토큰 받는 법이 상세하게 적혀 있다.





키워드 크롤링

```
71 rt1 <- search_tweets(q="koo ", include_rts=FALSE, "lang:en",
72                      geocode = lookup_coords("usa"),
73                      n = 10000, type="mixed")
```

```
> head(rt1$text,3)
[1] "@Say_Koo You can guess who gave me this card \U0001f602"



[2] "@Kpop_Herald @BigHitEnt I don’t see what is wrong with Koo dating. He is in his 20’s !He
 should be doing things 20 something do! As for tattoos why not if I’m not so terrified of nee
dle I would get one. Is his body he can do what he wants. #PCAs #BTS #TheConcertTour #LoveYour
selfSpeakYourself  @BTS_twt"
[3] "Closing my eyes on the way to jsq this morning because my dad is driving in this traffic
 like hes in Manila. \U0001f62b buhaaay koo! Lmfao"
```

키워드 크롤링

```
71 rt1 <- search_tweets(q="koo ", include_rts=FALSE, "lang:en",
72                      geocode = lookup_coords("usa"),
73                      n = 10000, type="mixed")
```

✓ 최대 9일전 자료까지 밖에 받아올 수 없다.

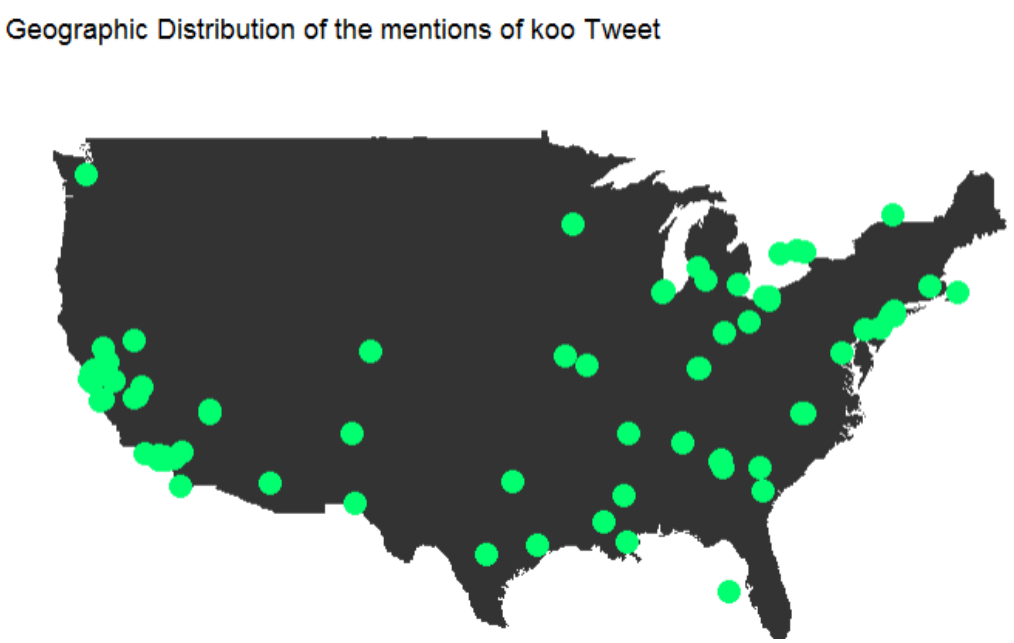
```
> summary(ymd_hms(rt1$created_at))
                Min.                 1st Qu.                  Median                    Mean
"2019-09-09 14:15:10" "2019-09-11 17:18:11" "2019-09-14 02:28:42" "2019-09-13 22:28:47"
             3rd Qu.                    Max.
"2019-09-16 05:48:28" "2019-09-17 17:13:00"
```

✓ 받을 수 있는 트윗의 개수가 제한되어 있다.

koo가 언급된 트윗이 10만 이상이더라도 한번에 받을 수 있는 트윗은 18000개이다.

위도,경도 변수 생성과 시각화

```
59 rt1 <- lat_lng(rt1)
60
61 world <- ggplot2::map_data("usa")
62 ggplot() +
63   geom_polygon(data = world, aes(x = long, y = lat, group = group)) +
64   geom_point(data = rt1, aes(x = lng, y = lat), color = "#01FF70", size = 5) +
65   coord_quickmap() +
66   labs(title = "Geographic Distribution of the mentions of koo Tweet") +
67   theme_void()
```



✓ 해당 트윗이 어디서 작성되었는지 볼 수 있다.

Thank you

https://github.com/jhy6219

Seminar Series

R Ladies

KOREAN STATISTICAL SOCIETY 한국통계학회 1971