데이터 과학

5주차

fitbit

201202156 오희빈

1. 걸음수 TOP10 구하기

1) 데이터 가공과정

```
import pandas as pd
import json
import os
import csv

dates = pd.date_range('20160401', '20160520')
dates05 = pd.date_range('20160501','20160520')
step = []
name = []

user_name = 'A0'
num = 1
```

=> 걸음수 Top10을 구하기 위해서 유저의 이름과 총 걸음수를 저장하기 위해 step과 name의 list를 만들고 user_name을 A0로 해서 num을 하나씩 증가시켜 유저의 번호를 찾는다. 또한 fitbit의 기간인 20160401 ~ 20160520과 5월부터 있는 user 때문에 20160501 ~ 20160520까지의 범위를 만든다.

```
for x in range(1,99):
    i = 0
    if os.path.exists('sokulee/A0'+str(x)) == True:
        if os.path.exists('sokulee/A0'+ str(x) + '/A0' + str(x) + '_' + '20160401' + '_steps.json') == True:
            for date in dates:
                fname = 'sokulee/A0'+ str(x) + '/A0' + str(x) + '_' + date.strftime('%Y%m%d') + '_steps.json'
                try: f = open(fname)
                except IOError as e:
                    print(str(e))
                else:
                    date_data = json.loads(f.read())
                    if (list(date_data.keys()) == ['errors', 'success'] or list(date_data.keys()) == ['success','errors']) == False :
                        i = i + int(date_data['activities-steps'][0]['value'])
        else :
            for date in dates05:
                fname = 'sokulee/A0'+ str(x) + '/A0' + str(x) + '_' + date.strftime('%Y%m%d') + '_steps.json'
                try: f = open(fname)
                except IOError as e:
                    print(str(e))
                else:
                    date_data = json.loads(f.read())
                    if (list(date_data.keys()) == ['errors', 'success'] or list(date_data.keys()) == ['success','errors']) == False :
                        i = i + int(date_data['activities-steps'][0]['value'])
            name.append(user_name + str(x))
            step.append(i)
```

=> 유저의 번호가 총 98번까지 있기에 for문을 98번 돌리면서 폴더 안에 존재하는 유저를 찾는다. 유저가 있다면 if문에 들어가고 없다면 다음 번호로 가서 유저의 번호를 확인한다. 그리고 그 유저의 steps 파일의 존재 여부를 확인하고 4월달 user정보가 없다면 else로 가서 5월 부터의 정보를 받아서 총 걸음수를 계산한다. 4월 부터의 정보가 있다면 4월1일 부터의 fitbit정보를 불러와서 open후 그 파일이 error인지의 존재 여부를 list(date_data.keys()) == [‘errors’,‘success’] 가 false이면 error 파일이 아니기에 date_data[‘activities-steps’][0][‘value’]인 하루 총 걸음수를 I에 저장하면서 모든 날짜를 저장해서 그 값을 list에 저장하고 한명의 user의 총 걸음수를 저

장한 후 I의 값을 0으로 만들어 다음 user의 총 걸음수를 list에 저장시켜 나간다.

```
with open('step.csv','w',newline='\n') as csvfile:
    fieldnames = ['name','total_steps']
    writer = csv.DictWriter(csvfile,fieldnames=fieldnames)
    writer.writeheader()
    for x in range(0,68):
        writer.writerow({'name' : str(name[x]), 'total_steps' : str(step[x])})
```

=> list에 저장한 user의 name과 총 걸음수를 csv파일에 DictWriter와 writerow를 이용해서 한 줄 한 줄 저장 시킨다

가공결과

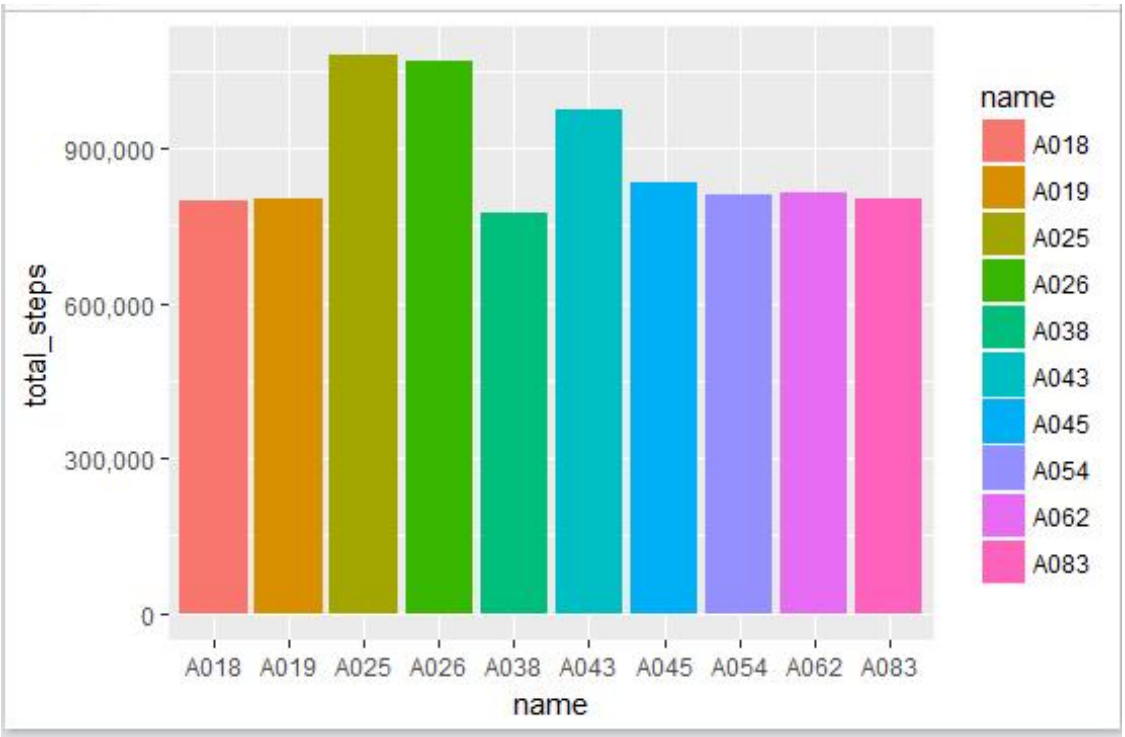
```
name,total_steps
A01,351518
A02,317873
A03,391302
A04,667043
A05,205815
A06,688864
A07,584462
A08,616758
A010,266725
A016,535526
A017,697669
A018,800509
A019,802587
A020,676258
```

2)시각화

```
1  library("ggplot2")
2  library("scales")
3
4  setwd('C:/Users/heebin/Documents/R/fitbit')
5  data <- read.csv('step.csv')
6
7  result <- data[order(-data$total_steps),]
8
9  top_user <- result[1:10,]
10
11 bar <- ggplot(top_user, aes(x=name, y=total_steps, fill=name)) + geom_bar(stat="identity") + scale_y_continuous(labels = comma)
12 print(bar)
13
```

=> 가공시킨 step.csv 파일을 읽은 후 총 발걸음 수를 중심으로 큰 순서로 sort시킨다. 그리고 top10의 정보만 top_user에 저장 시키고 geom_bar 형식으로 시각화를 시킨다.

3)시각화 그래프



2.가설 => 총 발걸음 수가 많을수록 운동을 많이 했을 것이다.

1)가공과정

```
import pandas as pd
import json
import os
import csv


dates = pd.date_range('20160401', '20160520')
dates05 = pd.date_range('20160501','20160520')
heart = []
name = []
step = []
user_name = 'A0'
num = 1
```

=> 유저의 name과 총 발걸음 수 그리고 운동 시간을 저장하기 위한 list를 만든다. 1번 문제와 마찬가지로 5월부터 있는 유저를 위해 date범위를 둘로 만들었다.

```
for x in range(1,99):
    i = 0
    if os.path.exists('sokulee/A0'+str(x)) == True:
        if os.path.exists('sokulee/A0'+ str(x) + '/A0' + str(x) + '_' + '20160401' + '_steps.json') == True:
            for date in dates:
                fname = 'sokulee/A0'+ str(x) + '/A0' + str(x) + '_' + date.strftime('%Y%m%d') + '_steps.json'
                try: f = open(fname)
                except IOError as e:
                    print(str(e))
                else:
                    date_data = json.loads(f.read())
                    if (list(date_data.keys()) == ['errors', 'success'] or list(date_data.keys()) == ['success','errors']) == False :
                        i = i + int(date_data['activities-steps'][0]['value'])
        else :
            for date in dates05:
                fname = 'sokulee/A0'+ str(x) + '/A0' + str(x) + '_' + date.strftime('%Y%m%d') + '_steps.json'
                try: f = open(fname)
                except IOError as e:
                    print(str(e))
                else:
                    date_data = json.loads(f.read())
                    if (list(date_data.keys()) == ['errors', 'success'] or list(date_data.keys()) == ['success','errors']) == False :
                        i = i + int(date_data['activities-steps'][0]['value'])
        name.append(user_name + str(x))
        step.append(i)
```

=> 1번 문제와 마찬가지로 총 발걸음 수를 가져오기 위한 알고리즘. 유저의 파일 존재 여부를 확인하고 4월부터 있는지에 대한 존재 여부를 확인 하고 그 파일에 에러가 존재하는지를 확인한다. 그리고 그 유저의 하루 발걸음 수를 모두 더해서 list에 저장시킨다.

```
for x in range(1,99):
    i = 0
    if os.path.exists('sokulee/A0'+str(x)) == True:
        if os.path.exists('sokulee/A0'+ str(x) + '/A0' + str(x) + '_' + '20160401' + '_heart.json') == True:
            for date in dates:
                fname = 'sokulee/A0'+ str(x) + '/A0' + str(x) + '_' + date.strftime('%Y%m%d') + '_heart.json'
                try: f = open(fname)
                except IOError as e:
                    print(str(e))
                else:
                    date_data = json.loads(f.read())
                    if (list(date_data.keys()) == ['errors', 'success'] or list(date_data.keys()) == ['success','errors']) == False :
                        if(len(list(date_data['activities-heart'][0]['value']['heartRateZones'][2].keys())) == 3):
                            i = i + 0
                        else:
                            i = i + int(date_data['activities-heart'][0]['value']['heartRateZones'][2]['minutes'])
                        print(fname)
```

```
        else :
            for date in dates05:
                fname = 'sokulee/A0'+ str(x) + '/A0' + str(x) + '_' + date.strftime('%Y%m%d') + '_heart.json'
                try: f = open(fname)
                except IOError as e:
                    print(str(e))
                else:
                    date_data = json.loads(f.read())
                    if (list(date_data.keys()) == ['errors', 'success'] or list(date_data.keys()) == ['success','errors']) == False :
                        if(len(list(date_data['activities-heart'][0]['value']['heartRateZones'][2].keys())) == 3):
                            i = i + 0
                        else:
                            i = i + int(date_data['activities-heart'][0]['value']['heartRateZones'][2]['minutes'])
        name.append(user_name + str(x))
        heart.append(i)
```

=> 유저의 운동량을 체크하기 위한 알고리즘 heart에 존재하는 heartRateZones의 cario([2])의 시간을 통해서 운동을 한 시간을 체크 했다. 이를 찾기위해 date_data['activities-heart'][0]['value']['heartRateZones'][2]['minutes']을 이용해서 그 유저의 하루하루 운동 시간을 저장해서 list에 저장했다. 그런데 총 발걸음 수와 마찬가지로 error가 있는 파일과 5월부터 있는 유저가 있어 if문을 통해 존재 여부를 확인하고 date_data['activities-heart'][0]['value']['heartRateZones'][2]에 존재하는 key값중 minutes이 없고 max, min, name만 존재하는 파일이 있었기에 그 킷값이 3개이면 운동시간을 0으로 저장하게 만들었다.

```
with open('heart.csv','w',newline='₩n') as csvfile:
    fieldnames = ['name','time','step']
    writer = csv.DictWriter(csvfile,fieldnames=fieldnames)
    writer.writeheader()
    for x in range(0,68):
        writer.writerow({'name' : str(name[x]), 'time' : str(heart[x]), 'step' : str(step[x])})
```

=> 마지막으로 csv 파일에 user의 name과 운동시간인 time, 총 발걸음 수인 step을 DictWriter와 writerow를 이용해서 한 줄 한 줄 저장시킨다.

가공 결과

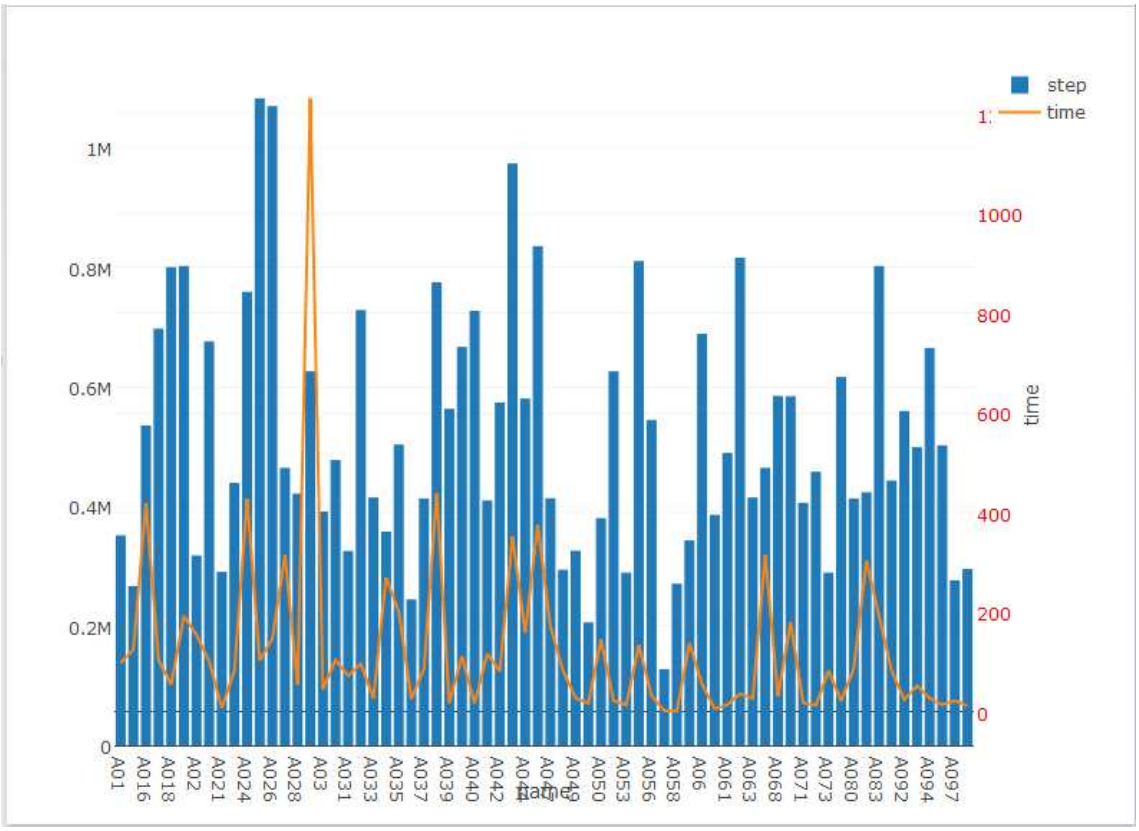
```
name,time,step
A01,99,351518
A02,155,317873
A03,46,391302
A04,112,667043
A05,18,205815
A06,56,688864
A07,180,584462
A08,23,616758
A010,125,266725
A016,419,535526
A017,104,697669
A018,56,800509
A019,193,802587
A020,102,676258
A021,9,290324
A022,83,439406
A024,428,759414
```

2) 시각화

```
library("ggplot2")
library("scales")
library("plotly")
setwd('C:/Users/heebin/Documents/R/fitbit')
result <- read.csv('heart.csv')
ay <- list(tickfont = list(color = "red"), overlaying = "y", side = "right", title = "time")
p <- plot_ly() %>%
  add_bars(x = result$name, y = result$step, name="step") %>%
  add_lines(x = result$name, y = result$time, name="time", yaxis="y2")%>%
  layout(yaxis2 = ay, xaxis=list(title="name"))
print(p)
```

=> plotly를 이용해 총 발걸음수와 운동시간을 나타내는 그래프를 결합시켰다.
plot_ly()함수를 이용해서 add_bars()함수를 이용해 총 발걸음 수를 막대그래프로 나타내고 add_lines()함수를 이용해 운동시간을 선 그래프로 한 그래프 안에 나타내게 하였는데 좌측 y축이 총 발걸음 수 우측 y축이 운동시간이며 x축이 유저의 name이다

3)운동시간과 발걸음 수의 관계



=> 그래프를 보면 발걸음 수가 많지만 그에 비해 운동 시간이 더 작고 발걸음 수가 적지만 그에 비해서 운동 시간이 굉장히 많은 것을 확인 할 수 있다. 비교적 발걸음이 많은 유저들의 운동 시간이 더 작고 발걸음 수는 많지 않지만 운동 시간이 탁월하게 높은 사람들이 많이 있었다. 이를 통해 가설이었던 발걸음 수가 많으면 운동시간도 많을 것이다 라는 가설은 거짓이라는 것을 알게 되었다.