데이터과학 실습 보고서

-5주차 Sokulee 데이터 분석-

2017.4.5

201201185 장진우

1.전체 기간 동안의 걸음수 TOP 10

1)데이터 가공 과정

```
global json_file
global result

username=[]
filename=[]
for root,dirs,files, in os.walk("./") :
    username.append(dirs)
    filename.append(files)
```

먼저 현재 디렉토리의 폴더명과 파일 이름들을 리스트에 저장합니다.

```
for i in range(length-1) :
        for j in range(len(filename[i+1])) :
                try :
                        (names,date,state)=filename[i+1][j].split("_")
                except ValueError :
                        pass
                if(state=="steps.json") :
                        sr="./"+names+"/"+names+"_"+date+"_"+state
                        json_file = sr
                        result = readConfig(json_file)
                        try :
                                if count == 0 :
                                        steps.append(int(result['activities-steps'][0]['value']))
                                        count=1
                                else :
                                        steps[i]=int(steps[i])+int(result['activities-steps'][0]['value'])
                        except KeyError :
                                pass
        count=0
        dicsteps[username[0][i]]=steps[i]
print dicsteps
```

그 후, 파일명을 name,date,state 3가지 상태로 "_"를 기준으로 분리시킵니다. 여기서 try except를 이용하여 파일명이 다른 형태일 때에 대한 예외처리를 해줍니다. 그리고 현재 state가 steps.json의 값을 가지고 있다면 파일이 있는 주소를 만들어 json file을 parsing합니다. 그리고 그 결과를 result에 넣어 주고, steps 리스트에 사람마다 일별로 걸음 수를 누적하게 됩니다. 여기서도 try except를 이용하여 json 파일안의 내용이 다른 것들에 대한 예외처리를 하게 됩니다. 그 후, username과 steps 수를 dictionary로 합치게 됩니다.

2)데이터 분석 과정

```
        topten=[]
        convert=[]
        keys=sorted(dicsteps,key=dicsteps.__getitem__,reverse=True)
        values=sorted(dicsteps.values(),reverse=True)
        for i in range(len(keys)) :

                topten.append([keys[i],values[i]])

        print topten[0:10]

        y_pos=np.arange(10)

        plt.bar(y_pos,values[0:10],align='center')
        plt.xticks(y_pos,keys[0:10])
        plt.show()


if __name__=="__main__" :
        main()
```

데이터 가공 과정에서 dictionary로 합쳐진 데이터들을 이용하여
먼저 keys라는 리스트에는 dictionary의 key값을 value값의 내림차순으로 저장하게 됩니다. 그리고, values라는 리스트에는 현재 dictionary의 value값의 내림차순으로 저장을 합니다. 그렇게 되면 key값과 value값이 일치하게 내림차순으로 정렬됨을 알 수 있습니다. 그 후, topten이라는 리스트에 key값과 value값을 묶어 append 시킨 후 0:10 까지 출력하여 top10이 완성됩니다.
그 후, plt.bar를 이용하여 막대그래프도 그려지게 됩니다.

3)코드설명 (가공, 분석과정 참조)

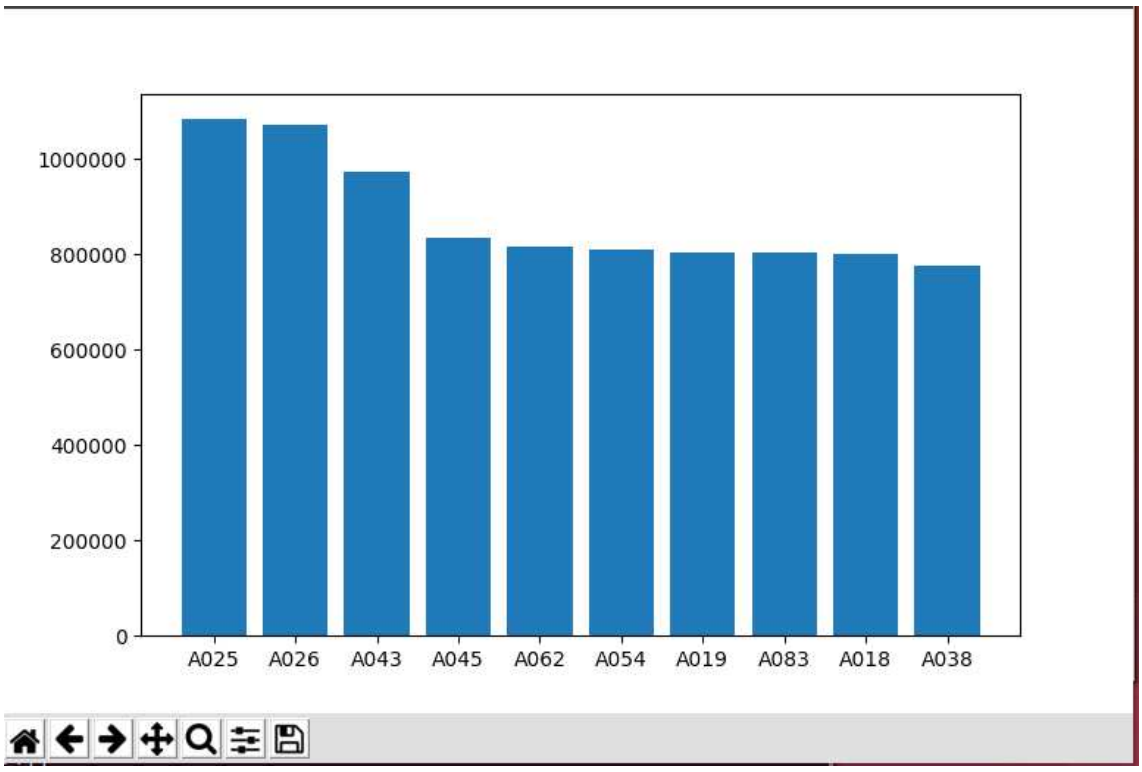
```
import json
import os
import datetime
import matplotlib.pyplot as plt
import numpy as np


result={}
def readConfig(filename):

        f = open(filename,'r')
        js = json.loads(f.read())
        f.close()
        return js

def main() :
        global json_file
        global result

        username=[]
        filename=[]
        for root,dirs,files, in os.walk("./") :
           username.append(dirs)
           filename.append(files)
        steps=[]
        length = len(username)
        values=[]
        dicsteps={}
        count=0
        for i in range(length-1) :
                for j in range(len(filename[i+1])) :
                        try :
                                (names,date,state)=filename[i+1][j].split("_")
                        except ValueError :
                                pass
                        if(state=="steps.json") :
                                sr="./"+names+"/"+names+"_"+date+"_"+state
                                json_file = sr
-- 끼워넣기 --
```

```
                        result = readConfig(json_file)
                        try :
                                if count == 0 :
                                        steps.append(int(result['activities-steps'][0]['value']))
                                        count=1
                                else :
                                        steps[i]=int(steps[i])+int(result['activities-steps'][0]['value'])
                        except KeyError :
                                pass
                count=0
                dicsteps[username[0][i]]=steps[i]
        print dicsteps

        topten=[]
        convert=[]
        keys=sorted(dicsteps,key=dicsteps.__getitem__,reverse=True)
        values=sorted(dicsteps.values(),reverse=True)
        for i in range(len(keys)) :

                topten.append([keys[i],values[i]])

        print topten[0:10]

        y_pos=np.arange(10)

        plt.bar(y_pos,values[0:10],align='center')
        plt.xticks(y_pos,keys[0:10])
        plt.show()


if __name__=="__main__" :
        main()
```

4)결과 그래프

```
jinwoo@jinwoo:~/sokulee$ python topten.py
{'A038': 775284, 'A039': 563457, 'A018': 800509, 'A019': 802587, 'A032': 728916,
 'A033': 414915, 'A030': 477748, 'A017': 697669, 'A010': 266725, 'A037': 413353,
 'A034': 357887, 'A035': 503583, 'A094': 665433, 'A096': 502283, 'A097': 276376,
 'A092': 559912, 'A093': 499193, 'A098': 300247, 'A043': 974336, 'A042': 574088,
 'A027': 464660, 'A025': 1083320, 'A068': 584932, 'A041': 409936, 'A040': 727438
, 'A021': 290324, 'A020': 676258, 'A045': 835907, 'A022': 439406, 'A061': 489414
, 'A060': 385730, 'A049': 325833, 'A062': 816567, 'A065': 464660, 'A016': 535526
, 'A029': 626306, 'A031': 325301, 'A036': 244243, 'A044': 580578, 'A024': 759414
, 'A084': 443310, 'A083': 802587, 'A028': 421421, 'A081': 423731, 'A080': 413353
, 'A063': 414915, 'A08': 616758, 'A048': 294072, 'A047': 413737, 'A02': 317873,
'A03': 391302, 'A01': 351518, 'A06': 688864, 'A07': 584462, 'A04': 667043, 'A05'
: 205815, 'A050': 380282, 'A052': 626213, 'A053': 289223, 'A054': 810861, 'A026'
: 1070291, 'A056': 544578, 'A057': 127678, 'A058': 270693, 'A059': 343124, 'A072
': 458095, 'A073': 289141, 'A071': 406062}
[['A025', 1083320], ['A026', 1070291], ['A043', 974336], ['A045', 835907], ['A06
2', 816567], ['A054', 810861], ['A019', 802587], ['A083', 802587], ['A018', 8005
09], ['A038', 775284]]
jinwoo@jinwoo:~/sokulee$
```

<텍스트 TOP10 출력>



<그래프 TOP10 출력>

2.전체 기간 걸음 수에 따른 지방연소 시간

1)데이터 가공 과정

```
global json_file
global result

username=[]
filename=[]
for root,dirs,files, in os.walk("./") :
    username.append(dirs)
    filename.append(files)
```

먼저 현재 디렉토리의 폴더명과 파일 이름들을 리스트에 저장합니다.

```
for i in range(length-1) :
        for j in range(len(filename[i+1])) :
                try :
                        (names,date,state)=filename[i+1][j].split("_")
                except ValueError :
                        pass
                if(state=="steps.json") :
                        sr="./"+names+"/"+names+"_"+date+"_"+state
                        json_file = sr
                        result = readConfig(json_file)
                        try :
                                if count == 0 :
                                        steps.append(int(result['activities-steps'][0]['value']))
                                        count=1
                                else :
                                        steps[i]=int(steps[i])+int(result['activities-steps'][0]['value'])
                        except KeyError :
                                pass
                if(state=="heart.json") :
                        sr="./"+names+"/"+names+"_"+date+"_"+state
                        json_file = sr
                        result = readConfig(json_file)
                        try :
                                if count2 == 0 :
                                        burn.append(int(result['activities-heart'][0]['value']['heartRateZones'][1]['minutes'])
)
                                        count2=1
                                else :
                                        burn[i]=int(burn[i])+int(result['activities-heart'][0]['value']['heartRateZones'][1]['m
inutes'])
                        except KeyError :
                                pass
        count2=0
        count=0
        dicsteps[username[0][i]]=steps[i]
        dicheart[username[0][i]]=burn[i]
print dicheart
```

걸음 수의 경우 실습과 방법이 동일하며 심박수중 지방연소의 누적시간은 걸음수와 같이 사람마다 일별 지방연소 시간을 누적시켜 줍니다.

2)데이터 분석 과정

```
heartkeys = stepkeys
heartvalues=[]
for i in range(len(stepkeys)) :
        heartvalues.append(dicheart[heartkeys[i]])
        topten.append([stepkeys[i],stepvalues[i]])
```

우선 지방연소 시간의 key값은 걸음 수의 key값과 동일해야 합니다.(x축이 같으므로) 그 후, 지방연소 시간의 value값은 순차적으로 key값에 있는 value

값을 누적시켜 key값을 기준으로 value값을 정렬 시킵니다. 그렇게 되면 최종적으로 많이 걸은 순으로 지방연소 시간의 데이터가 나타나게 됩니다.

3)코드설명 (가공, 분석과정 참조)

```
import json
import os
import datetime
import matplotlib.pyplot as plt
import numpy as np
import plotly.plotly as py
import plotly.graph_objs as go


result={}
def readConfig(filename):

        f = open(filename,'r')
        js = json.loads(f.read())
        f.close()
        return js

def main() :
        global json_file
        global result

        username=[]
        filename=[]
        for root,dirs,files, in os.walk("./") :
           username.append(dirs)
           filename.append(files)
        steps=[]
        length = len(username)
        stepvalues=[]
        dicsteps={}
        dicheart={}
        count=0
        count2=0
        burn=[]
        for i in range(length-1) :
                for j in range(len(filename[i+1])) :
                        try :
                                (names,date,state)=filename[i+1][j].split("_")
```

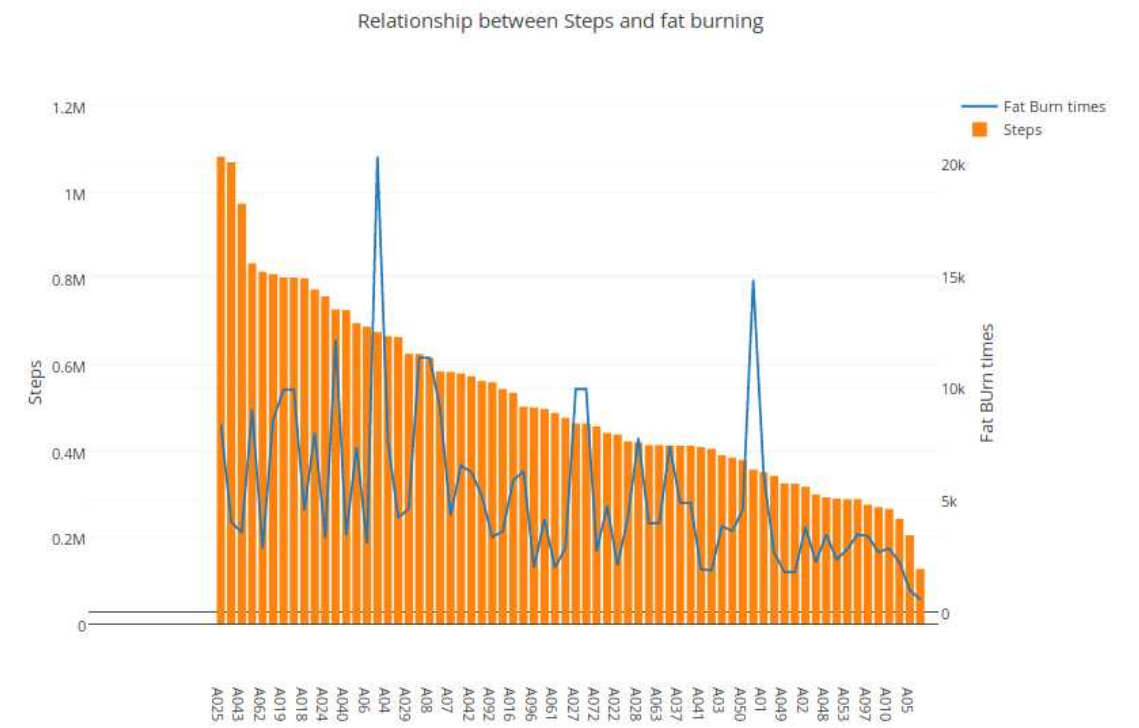
```
                except ValueError :
                        pass
                if(state=="steps.json") :
                        sr="./"+names+"/"+names+"_"+date+"_"+state
                        json_file = sr
                        result = readConfig(json_file)
                        try :
                                if count == 0 :
                                        steps.append(int(result['activities-steps'][0]['value']))
                                        count=1
                                else :
                                        steps[i]=int(steps[i])+int(result['activities-steps'][0]['value'])
                        except KeyError :
                                pass
                if(state=="heart.json") :
                        sr="./"+names+"/"+names+"_"+date+"_"+state
                        json_file = sr
                        result = readConfig(json_file)
                        try :
                                if count2 == 0 :
                                        burn.append(int(result['activities-heart'][0]['value']['heartRateZones'][1]['minutes'])
)
                                        count2=1
                                else :
                                        burn[i]=int(burn[i])+int(result['activities-heart'][0]['value']['heartRateZones'][1]['m
inutes'])
                        except KeyError :
                                pass
                count2=0
                count=0
                dicsteps[username[0][i]]=steps[i]
                dicheart[username[0][i]]=burn[i]
        print dicheart

        topten=[]
        convert=[]
        stepkeys=sorted(dicsteps,key=dicsteps.__getitem__,reverse=True)
        stepvalues=sorted(dicsteps.values(),reverse=True)
```

Dubeolsik 한자 전용

74,1-8 61%

```
        heartkeys = stepkeys
        heartvalues=[]
        for i in range(len(stepkeys)) :
                heartvalues.append(dicheart[heartkeys[i]])
                topten.append([stepkeys[i],stepvalues[i]])

        print topten[0:10]


        trace1= go.Scatter(x=heartkeys,y=heartvalues,name="Fat Burn times",yaxis="y2")
        trace2= go.Bar (x=stepkeys,y=stepvalues,name="Steps")
        data = [trace1,trace2]
        layout = go.Layout(title='Relationship between Steps and fat burning',yaxis=dict(title='Steps'), yaxis2=dict(title='Fat BUrn ti
mes', overlaying='y', side='right'))
        fig = go.Figure(data=data,layout=layout)


        py.plot(fig,filename='bar-line')


if __name__=="__main__" :
        main()
```

4)결과 그래프



<걸음수와 지방연소 시간의 상관관계 그래프>

분석결과 - 걸음수가 많다고 해서 지방연소가 잘 되는 것이 아니며 지방 연소 시간을 통해 사람마다의 운동량도 체크 할 수 있으며, 행동패턴(움직임은 적지만 꾸준한 운동을 하는 사람, 여기저기 많이 돌아다니는 사람 등)을 파악 할 수 있습니다.