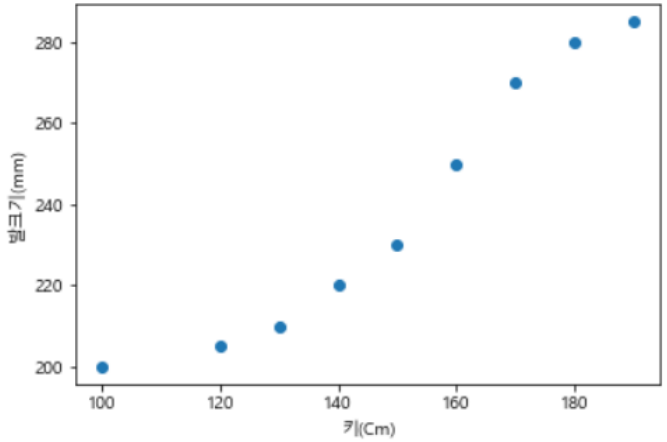
Python

상관관계

참고문헌 : 파이썬을 이용한 빅데이터 수집, 분석과 시각화 - 비팬북스, 이원화

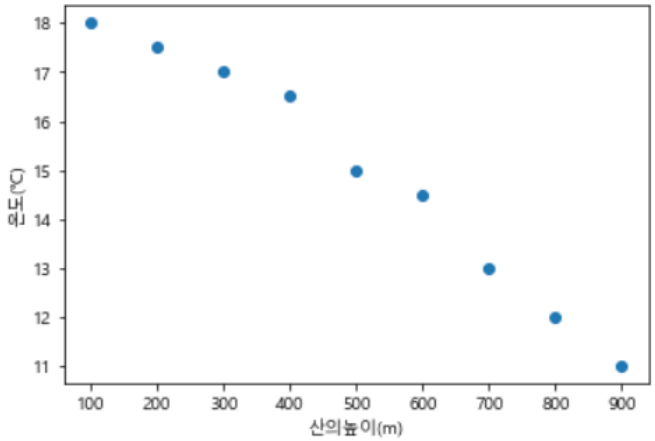
양의 상관관계

```
1  import matplotlib.pyplot as plt
2  from matplotlib import font_manager, rc
3  import matplotlib
4  font_location = "c:/Windows/fonts/malgun.ttf"
5  font_name = font_manager.FontProperties(fname=font_location).get_name()
6
7  matplotlib.rc('font', family=font_name)
8  height = [100, 120, 130, 140, 150, 160, 170, 180, 190]
9  foot_size = [200, 205, 210, 220, 230, 250, 270, 280, 285]
10 plt.scatter(height, foot_size)
11 plt.xlabel('키(Cm)')
12 plt.ylabel('발크기(mm)')
13 plt.show()
```



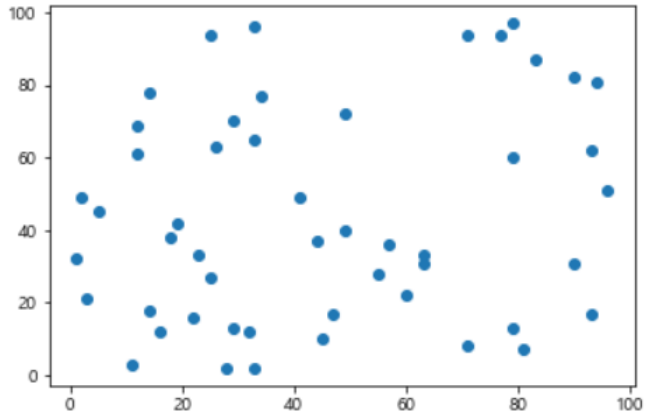
음의 상관관계

```
height = [100, 200, 300, 400, 500, 600, 700, 800, 900]
temperature = [18.0, 17.5, 17, 16.5, 15, 14.5, 13, 12, 11]
plt.scatter(height, temperature)
plt.xlabel('산의높이(m)')
plt.ylabel('온도(℃)')
plt.show()
```



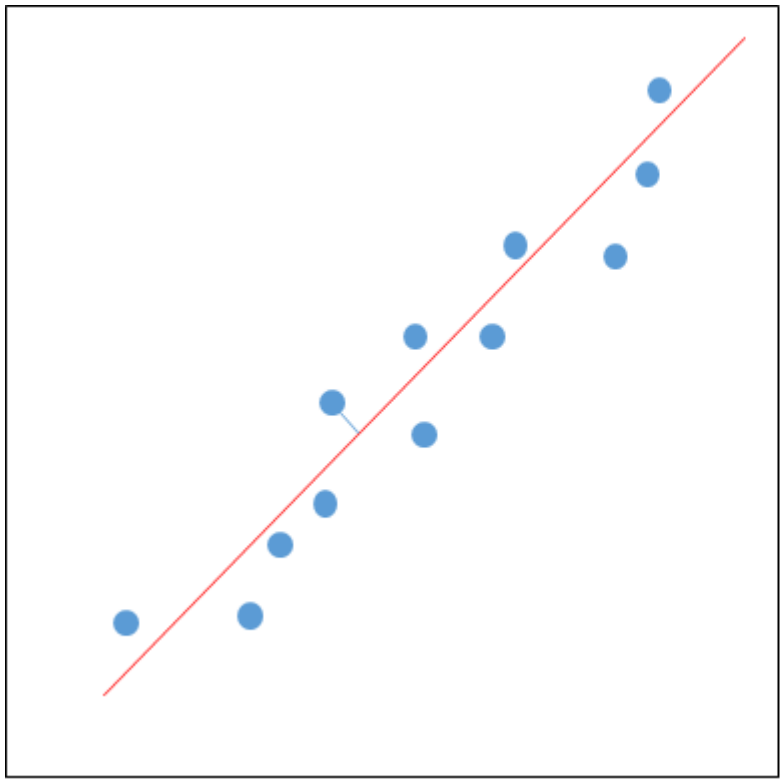
상관관계

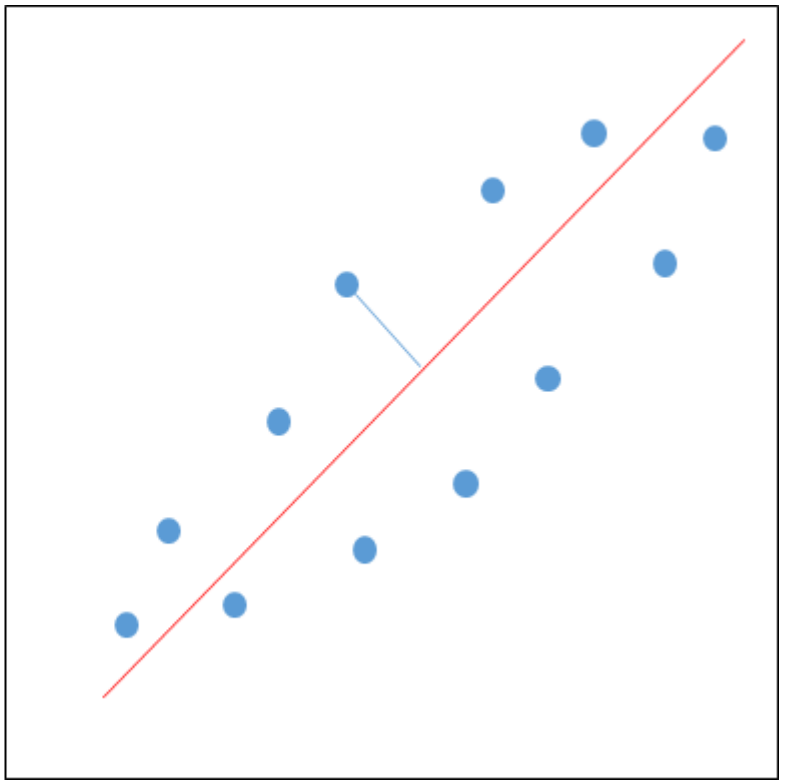
```
1  import numpy as np
2  random_x = np.random.randint(0, 100, 50) # 1~100 사이의 난수를 50개 생성
3  random_y = np.random.randint(0, 100, 50) # 1~100 사이의 난수를 50개 생성
4  plt.scatter(random_x, random_y)
5  plt.show()
```



상관계수

- 상관계수는 $-1 \le r \le 1$ 의 값을 가지며, 수치가 0에 근접할 수록 두 변수간에 상관관계가 없음을 의미
- -1의 경우에는 음(-)의 상관 관계가 강하며, +1에 근접할수록 양(+)의 상관관계가 크다.





상관계수

0.0 ~ 0.2	상관 관계가 거의 없다
0.2 ~ 0.4	약한 상관 관계
0.4 ~ 0.6	상관 관계가 있다
0.6 ~ 0.8	강한 상관 관계
0.8 ~ 1.0	매우 강한 상관 관계

$$r = \frac{\sum_{i=1}^{n}(x-\overline{x})(y-\overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i-\overline{y})^2}}$$

데이터 테이블 생성

- pip install pandas
- 데이터를 엑셀 스프레드시트(excel spreadsheet)나 SQL 테이블과 같이 서로 다른 속성을 가지는 컬럼들로 구성된 데이터 테이블 형태(DataFrame)이나 시계열 데이터와 같이 연속된 데이터 셋(Series)으로 구성하고, 이를 분석하기 위한 도구로 사용

Series

```
1 import numpy as np
2 import pandas as pd
```

```
1 s = pd.Series(np.random.randn(5))
2 s
```

```
0    0.949444
1   -0.696565
2    0.273719
3    0.146401
4    2.284608
dtype: float64
```

```
1 s = s = pd.Series(np.random.randn(5), index=['A','B','C','D','E'])
2 s
```

```
A    0.315061
B   -0.877433
C   -2.025374
D   -0.973703
E   -1.476231
dtype: float64
```

Series

```
# dictionary
d = {'a' : 0., 'b' : 1., 'c' : 2.}
pd.Series(d)
```

```
a    0.0
b    1.0
c    2.0
dtype: float64
```

```
pd.Series(d, index=['a', 'b', 'B', 'c'])
```

```
a    0.0
b    1.0
B    NaN
c    2.0
dtype: float64
```

```
# 스칼라값
pd.Series(7, index=['a', 'b', 'c', 'd', 'e'])
```

```
a    7
b    7
c    7
d    7
e    7
dtype: int64
```

Series

```
1  s = pd.Series([1,2,3,4,5], index=['a', 'b', 'c', 'd', 'e'])
2  s[0]
```

```
1
```

```
1  s[:3]
```

```
a    1
b    2
c    3
dtype: int64
```

```
1  s[[4,1]]
```

```
e    5
b    2
dtype: int64
```

```
1  np.power(s, 2)
```

```
a     1
b     4
c     9
d    16
e    25
dtype: int64
```

DataFrame

```
# Series/Dict 데이터의 활용
d = {'one' : pd.Series([1., 2., 3.], index=['a', 'b', 'c']),'two' : pd.Series([1., 2., 3., 4.], index=['a', 'b', 'c', 'd'])}
d
```

```
{'one': a    1.0
 b    2.0
 c    3.0
 dtype: float64, 'two': a    1.0
 b    2.0
 c    3.0
 d    4.0
 dtype: float64}
```

```
pd.DataFrame(d)
```

	one	two
a	1.0	1.0
b	2.0	2.0
c	3.0	3.0
d	NaN	4.0

```
# 인덱스 값을 부여하지 않으면 자동으로 0부터 두개의 데이터 중 큰 배열의 길이 - 1 만큼이 부여
d = {'one' : pd.Series([1., 2., 3.]), 'two' : pd.Series([1., 2., 3., 4.])}
pd.DataFrame(d)
```

	one	two
0	1.0	1.0
1	2.0	2.0
2	3.0	3.0
3	NaN	4.0

```
d = {'one' : [1., 2., 3., 4.], 'two' : [4., 3., 2., 1.]}
pd.DataFrame(d)
```

	one	two
0	1.0	4.0
1	2.0	3.0
2	3.0	2.0
3	4.0	1.0

DataFrame

```
# Dict 리스트 데이터의 활용
data2 = [{'a': 1, 'b': 2}, {'a': 5, 'b': 10, 'c': 20}]
pd.DataFrame(data2)
```

	a	b	c
0	1	2	NaN
1	5	10	20.0

```
pd.DataFrame(data2, index=['first', 'second'])
```

	a	b	c
first	1	2	NaN
second	5	10	20.0

```
pd.DataFrame(data2, columns=['a', 'b'])
```

	a	b
0	1	2
1	5	10

```
df = pd.DataFrame(data2, columns=['a', 'b'])
df.rename(columns={'a':'COL1'})
```

	COL1	b
0	1	2
1	5	10

```
# 데이터 추가 및 합치기(merge)
data1 = [{'name':'Mark'},{'name':'Eric'},{'name':'Jennifer'}]
df = pd.DataFrame(data1)
df
```

	name
0	Mark
1	Eric
2	Jennifer

```
df['age'] = [10, 11, 12]
pd.DataFrame(data1)df
```

	name	age
0	Mark	10
1	Eric	11
2	Jennifer	12

```
data2 = [{'sido':'서울'}, {'sido':'경기'}, {'sido':'인천'}]
df2 = pd.DataFrame(data2)
df2
```

	sido
0	서울
1	경기
2	인천

```
pd.merge(df, df2, left_index=True, right_index=True)
```

	name	age	sido
0	Mark	10	서울
1	Eric	11	경기
2	Jennifer	12	인천

난수(random) 데이터를 얻는 함수

함수	설명
rand(d0, d1, ..., dn)	N차원 배열의 난수 발생
randn(d0, d1, ..., dn)	표준 정규분포에 따른 N차원 난수 발생
randint(low[, high, size])	low 이상 high 미만의 정수형 난수 발생
random_sample([size])	0.0이상 1.0미만의 실수형 난수 발생
random([size])	
ranf([size])	
sample([size])	
choice(a[, size, replace, p])	주어진 1차원 배열을 기반으로 무작위 샘플 추출
bytes(length)	바이트형 난수 발생

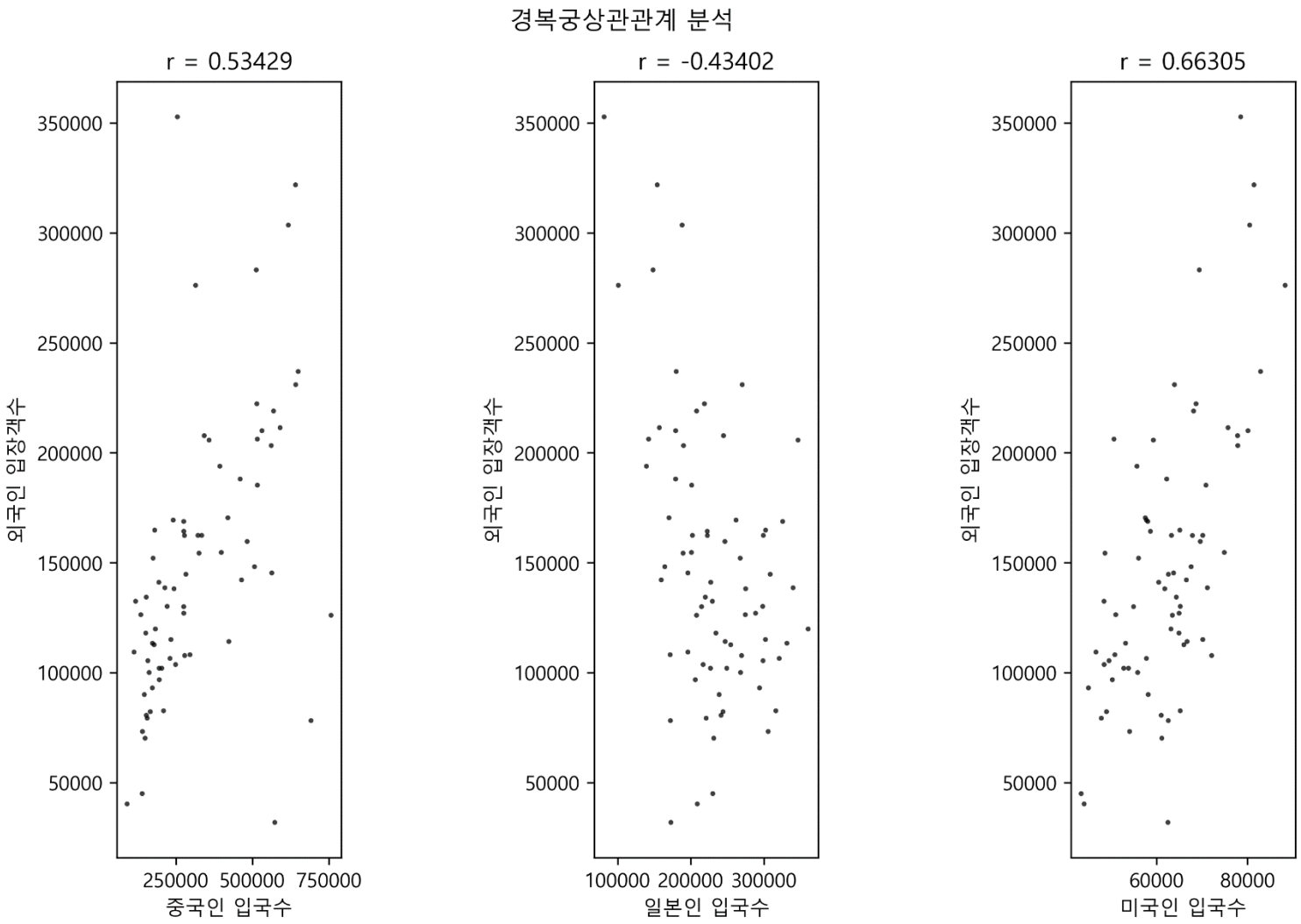
서울시 입장객과 각국 입국수에 따른 상관분석 자료

```
import json
import math
import numpy as np

import matplotlib.pyplot as plt
import matplotlib
from matplotlib import font_manager, rc

import pandas as pd

#[CODE 1]

def correlation(x, y):
    n = len(x)
    vals = range(n)

    x_sum = 0.0
    y_sum = 0.0
    x_sum_pow = 0.0
    y_sum_pow = 0.0
    mul_xy_sum = 0.0

    for i in vals:
        mul_xy_sum = mul_xy_sum + float(x[i]) * float(y[i])
        x_sum = x_sum + float(x[i])
        y_sum = y_sum + float(y[i])
        x_sum_pow = x_sum_pow + pow(float(x[i]), 2)
        y_sum_pow = y_sum_pow + pow(float(y[i]), 2)

    try:
        r = ((n * mul_xy_sum) - (x_sum * y_sum)) / math.sqrt( ((n*x_sum_pow) - pow(x_sum, 2)) * ((n*y_sum_pow) - pow(y_sum, 2)) )
    except:
        r = 0.0

    return r

```

```python
#[CODE 2]

def setScatterGraph(tour_table, visit_table, tourpoint):

    tour = tour_table[tour_table['resNm'] == tourpoint]
    merge_table = pd.merge(tour, visit_table, left_index=True, right_index=True)

    fig = plt.figure()

    fig.suptitle(tourpoint + '상관관계 분석')

    plt.subplot(1, 3, 1)
    plt.xlabel('중국인 입국수')
    plt.ylabel('외국인 입장객수')
    r = correlation(list(merge_table['china']), list(merge_table['ForNum']))
    plt.title('r = {:.5f}'.format(r))
    plt.scatter(list(merge_table['china']), list(merge_table['ForNum']), edgecolor='none', alpha=0.75, s=6, c='black')

    plt.subplot(1, 3, 2)
    plt.xlabel('일본인 입국수')
    plt.ylabel('외국인 입장객수')
    r = correlation(list(merge_table['japan']), list(merge_table['ForNum']))
    plt.title('r = {:.5f}'.format(r))
    plt.scatter(list(merge_table['japan']), list(merge_table['ForNum']), edgecolor='none', alpha=0.75, s=6, c='black')

    plt.subplot(1, 3, 3)
    plt.xlabel('미국인 입국수')
    plt.ylabel('외국인 입장객수')
    r = correlation(list(merge_table['usa']), list(merge_table['ForNum']))
    plt.title('r = {:.5f}'.format(r))
    plt.scatter(list(merge_table['usa']), list(merge_table['ForNum']), edgecolor='none', alpha=0.75, s=6, c='black')

    plt.tight_layout()

    # 이미지 저장
    #fig = matplotlib.pyplot.gcf()

    #fig.set_size_inches(10, 7)

    #fig.savefig(tourpoint+'.png', dpi=300)

    plt.show()
```

```python

def main():

    font_location = "c:/Windows/fonts/malgun.ttf"
    font_name = font_manager.FontProperties(fname=font_location).get_name()
    matplotlib.rc('font', family=font_name)

    tpFileName = '서울특별시_관광지입장정보_2011_2016.json'
    jsonTP = json.loads(open(tpFileName, 'r', encoding='utf-8').read())
    tour_table = pd.DataFrame(jsonTP, columns=('yyyymm', 'resNm', 'ForNum'))  # 필요한 데이터만 추출
    tour_table = tour_table.set_index('yyyymm')

    '''['창덕궁', '운현궁', '경복궁', '창경궁', '종묘', '국립중앙박물관', '서울역사박물관', '덕수궁',
    '서울시립미술관 본관', '태릉·강릉·조선왕릉전시관', '서대문형무소역사관', '서대문자연사박물관',
    '트릭아이미술관', '헌릉·인릉', '선릉·정릉', '롯데월드']'''

    resNm = tour_table.resNm.unique()

    fv_CFileName = '중국(112)_해외방문객정보_2011_2016.json'
    jsonFV = json.loads(open(fv_CFileName, 'r', encoding='utf-8').read())
    china_table = pd.DataFrame(jsonFV, columns=('yyyymm', 'visit_cnt'))
    china_table = china_table.rename(columns={'visit_cnt': 'china'})
    china_table = china_table.set_index('yyyymm')

    fv_JFileName = '일본(130)_해외방문객정보_2011_2016.json'
    jsonFV = json.loads(open(fv_JFileName, 'r', encoding='utf-8').read())
    japan_table = pd.DataFrame(jsonFV, columns=('yyyymm', 'visit_cnt'))
    japan_table = japan_table.rename(columns={'visit_cnt': 'japan'})
    japan_table = japan_table.set_index('yyyymm')

    fv_UFileName = '미국(275)_해외방문객정보_2011_2016.json'
    jsonFV = json.loads(open(fv_UFileName, 'r', encoding='utf-8').read())
    usa_table = pd.DataFrame(jsonFV, columns=('yyyymm', 'visit_cnt'))
    usa_table = usa_table.rename(columns={"visit_cnt": "usa"})
    usa_table = usa_table.set_index('yyyymm')

    fv_table = pd.merge(china_table, japan_table, left_index=True, right_index=True)
    fv_table = pd.merge(fv_table, usa_table, left_index=True, right_index=True)

    for tourpoint in resNm:
        setScatterGraph(tour_table, fv_table, tourpoint)

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

서울시 입장객과 각국 입국수에 따른 상관분석 자료



상관계수 분석

```
import json
import math
import numpy as np

import matplotlib.pyplot as plt
import matplotlib
from matplotlib import font_manager, rc
import pandas as pd

'''[['창덕궁', -0.058791104060063125, 0.27744435701410114, 0.40281606330501574], ['운현궁', 0.44594488384450376, 0.3026152182879861,
0.2812576500158649], ['경복궁', 0.5256734293511214, -0.4352281861341233, 0.42513726387044926], ['창경궁', 0.4512325398089607,
-0.16458589402253013, 0.6245403780269381], ['종묘', -0.5834218986767474, 0.5298702802205212, 0.12112666820204959], ['국립중앙박물관'
, 0.39663594900292837, -0.06923889417914424, 0.3789788348060077], ['서울역사박물관', 0.
0.2411976107709704], ['덕수궁', 0.4332132943587757, -0.4326719125679966, 0.480858846954
['태릉 · 강릉 · 조선왕릉전시관', -0.08179909096513825, 0.0634032985209752, -0.068840
0.47262271531670347, 0.006098570233700235, 0.22900879409607508], ['서대문자연사박물관',
0.340084882575556, -0.15036015533747007, 0.18094502388483083], ['헌릉·인릉', -0.581325
-0.1853887818740637], ['선릉·정릉', -0.5715258789199192, 0.38806730592260075, -0.12494
0.23511773800458452, -0.12673869767365747]]'''

f = open("rlist.txt", "r")
r_table = f
f.close()


r_table = pd.DataFrame(r_list, columns=('tourpoint', 'china', 'japan', 'usa'))
r_table = r_table.set_index('tourpoint')
r_table
```

tourpoint	china	japan	usa
창덕궁	-0.058791	0.277444	0.402816
운현궁	0.445945	0.302615	0.281258
경복궁	0.525673	-0.435228	0.425137
창경궁	0.451233	-0.164586	0.624540
종묘	-0.583422	0.529870	-0.121127
국립중앙박물관	0.396636	-0.069239	0.378979
서울역사박물관	0.416999	0.492978	0.241198
덕수궁	0.433213	-0.432672	0.480859
서울시립미술관 본관	0.000000	0.000000	0.000000
태릉 · 강릉 · 조선왕릉전시관	-0.081799	0.063403	-0.068840
서대문형무소역사관	0.472623	0.006099	0.229009
서대문자연사박물관	0.000000	0.000000	0.000000
트릭아이미술관	0.340085	-0.150360	0.180945
헌릉 · 인릉	-0.581325	0.464530	-0.185389
선릉·정릉	-0.571526	0.388067	-0.124945
롯데월드	0.510559	0.235118	-0.126739

상관계수 분석

```
1  # 상관관계값이 없는 경우에는 삭제
2  r_table.drop('서울시립미술관 본관')
3  r_table.drop('서대문자연사박물관')
4  r_table = r_table.sort_values('china', ascending=False)
5  r_table.head() # default 가 5개
```

tourpoint	china	japan	usa
경복궁	0.525673	-0.435228	0.425137
롯데월드	0.510559	0.235118	-0.126739
서대문형무소역사관	0.472623	0.006099	0.229009
창경궁	0.451233	-0.164586	0.624540
운현궁	0.445945	0.302615	0.281258

상관계수 분석

```
# 중국인 입국수 대비 관광객 입장객수의 상관계수가 높은 순서대로 3개국의 비교
font_location = "c:/Windows/fonts/malgun.ttf"
font_name = font_manager.FontProperties(fname=font_location).get_name()
matplotlib.rc('font', family=font_name)
r_table.plot(kind='bar', rot=70)

plt.show()
```