In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
#中文编码
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus']=False
```

In [ ]:
```
data_1=pd.read_excel('D:\M\MATLAB Driver/forward\eleven\有效问卷 （描述）.xlsx',s
data_2=pd.read_excel('D:\M\MATLAB Driver/forward\eleven\有效问卷 （描述）.xlsx',s
```

In [ ]:
```
#采用随机数替换data_2年龄里面的数字
#1替换为60-65之间的不同的随机数
for i in range(0,len(data_2['年龄'])):
    if data_2['年龄'][i]==1:
        data_2['年龄'][i]=np.random.randint(60,65)
        #2替换为66-75之间的不同的随机数
    if data_2['年龄'][i]==1:
        data_2['年龄'][i]=np.random.randint(61,65)
        #2替换为66-75之间的不同的随机数
    elif data_2['年龄'][i]==2:
        data_2['年龄'][i] = np.random.randint(66, 75)
        #3替换为76-80之间的不同的随机数
    elif data_2['年龄'][i]==3:
        data_2['年龄'][i]=np.random.randint(76, 80)
        #4替换为81-85之间的不同的随机数
    elif data_2['年龄'][i]==4:
        data_2['年龄'][i]=np.random.randint(81, 85)
        #5替换为86-90之间的不同的随机数
    else :
        data_2['年龄'][i]=np.random.randint(86, 90)
```

In [ ]:
```
#同样对月收入状况（退休金、养老金等总和）进行随机数替换处理
#1替换为2000元及以下之间的不同的随机数
for i in range(0,len(data_2['月收入状况（退休金、养老金等总和）'])):
    if data_2['月收入状况（退休金、养老金等总和）'][i]==1:
        data_2['月收入状况（退休金、养老金等总和）'][i]=np.random.randint(0,2000)
        #2替换为2001-4000之间的不同的随机数
    elif data_2['月收入状况（退休金、养老金等总和）'][i]==2:
        data_2['月收入状况（退休金、养老金等总和）'][i] = np.random.randint(2001, 4
        #3替换为4001-6000之间的不同的随机数
    elif data_2['月收入状况（退休金、养老金等总和）'][i]==3:
        data_2['月收入状况（退休金、养老金等总和）'][i]=np.random.randint(4001, 600
        #4替换为6001-8000之间的不同的随机数
    elif data_2['月收入状况（退休金、养老金等总和）'][i]==4:
        data_2['月收入状况（退休金、养老金等总和）'][i]=np.random.randint(6001, 800
        #5替换为8001-10000之间的不同的随机数
    elif data_2['月收入状况（退休金、养老金等总和）'][i]==5:
        data_2['月收入状况（退休金、养老金等总和）'][i]=np.random.randint(8001, 100
        #6替换为10001-12000之间的不同的随机数
    else :
        data_2['月收入状况（退休金、养老金等总和）'][i]=np.random.randint(10001, 12
```

In [ ]:
```
#对“每次旅居养老的预算范围”进行相同处理
#1替换为1000元及以下之间的不同的随机数
for i in range(0,len(data_2['每次旅居养老的预算范围'])):
```

```
    if data_2['每次旅居养老的预算范围'][i]==1:
        data_2['每次旅居养老的预算范围'][i]=np.random.randint(0,1000)
        #2替换为1001-2000之间的不同的随机数
    elif data_2['每次旅居养老的预算范围'][i]==2:
        data_2['每次旅居养老的预算范围'][i] = np.random.randint(1001, 2000)
        #3替换为2001-3000之间的不同的随机数
    elif data_2['每次旅居养老的预算范围'][i]==3:
        data_2['每次旅居养老的预算范围'][i]=np.random.randint(2001, 3000)
        #4替换为3001-4000之间的不同的随机数
    elif data_2['每次旅居养老的预算范围'][i]==4:
        data_2['每次旅居养老的预算范围'][i]=np.random.randint(3001, 4000)
        #5替换为4001-5000之间的不同的随机数
    elif data_2['每次旅居养老的预算范围'][i]==5:
        data_2['每次旅居养老的预算范围'][i]=np.random.randint(4001, 5000)
        #6替换为5001-6000之间的不同的随机数
    else :
        data_2['每次旅居养老的预算范围'][i]=np.random.randint(5001, 7000)
```

In [ ]:
```
#对“年出游次数(次)”进行同样处理
#1替换为0,2替换为1-3之间的不同的随机数
for i in range(0,len(data_2['年出游次数(次)'])):
    if data_2['年出游次数(次)'][i]==1:
        data_2['年出游次数(次)'][i]=0
        #2替换为1-3之间的不同的随机数
    elif data_2['年出游次数(次)'][i]==2:
        data_2['年出游次数(次)'][i] = np.random.randint(1, 3)
        #3替换为4-6之间的不同的随机数
    elif data_2['年出游次数(次)'][i]==3:
        data_2['年出游次数(次)'][i]=np.random.randint(4, 6)
        #4替换为7-9之间的不同的随机数
    elif data_2['年出游次数(次)'][i]==4:
        data_2['年出游次数(次)'][i]=np.random.randint(7, 9)
        #5替换为10-12之间的不同的随机数
    else :
        data_2['年出游次数(次)'][i]=np.random.randint(10,15)
```

In [ ]:
```
#对“次出游时长(天)”进行同样处理
#1替换为替换为1-3之间的不同的随机数
for i in range(0,len(data_2['次出游时长(天)'])):
    if data_2['次出游时长(天)'][i]==1:
        data_2['次出游时长(天)'][i]=np.random.randint(1, 3)
        #2替换为4-6之间的不同的随机数
    elif data_2['次出游时长(天)'][i]==2:
        data_2['次出游时长(天)'][i] = np.random.randint(4, 7)
        #3替换为7-9之间的不同的随机数
    elif data_2['次出游时长(天)'][i]==3:
        data_2['次出游时长(天)'][i]=np.random.randint(8, 15)
        #4替换为10-12之间的不同的随机数
    elif data_2['次出游时长(天)'][i]==4:
        data_2['次出游时长(天)'][i]=np.random.randint(16, 30)
        #5替换为13-15之间的不同的随机数
    else :
        data_2['次出游时长(天)'][i]=np.random.randint(30, 50)
```

In [ ]:
```
data_2.describe()
```

Out[ ]:

	序号	性别	年龄	教育程度	帮忙照看第三代(孙子，孙女)	退休前工作的单位性质	婚姻状况	(老
count	274.000000	274.000000	274.000000	274.000000	274.000000	274.000000	274.000000	2
mean	137.500000	1.375912	79.051095	2.635036	1.664234	2.857664	2.463504	48
std	79.241193	0.485244	8.399652	0.955998	0.473122	1.511190	1.037987	25
min	1.000000	1.000000	66.000000	1.000000	1.000000	1.000000	1.000000	
25%	69.250000	1.000000	71.000000	2.000000	1.000000	2.000000	2.000000	2
50%	137.500000	1.000000	78.500000	3.000000	2.000000	3.000000	2.000000	48
75%	205.750000	2.000000	87.000000	3.000000	2.000000	4.000000	2.000000	64
max	274.000000	2.000000	89.000000	4.000000	2.000000	7.000000	5.000000	11

8 rows × 51 columns

一 差异性分析

基本情况分析

In [ ]:
```
#将'年龄','性别','婚姻状况','教育程度','月收入状况（退休金、养老金等总和）','每次旅居
data_base=data_2[['年龄','性别','婚姻状况','教育程度','月收入状况（退休金、养老金等总
data_base.head()
#将数据集保存为csv文件
data_base.to_csv('data_base.csv',index=False,sep=',',encoding='utf-8')
```

In [ ]:
```
#对数据进行标准化处理
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(data_base)
data_base = scaler.transform(data_base)
data_base
```

Out[ ]:
```
array([[-0.60244608, -0.77610514, -0.44735785, ..., -1.53922562,
        -1.52845417,  0.40737875],
       [-0.00609411, -0.77610514, -0.44735785, ..., -1.53922562,
        -1.52845417,  0.40737875],
       [-1.19879805,  1.28848522, -0.44735785, ..., -1.53922562,
        -1.52845417,  0.40737875],
       ...,
       [ 0.82879866, -0.77610514, -0.44735785, ..., -1.53922562,
        -1.52845417, -2.45471812],
       [ 0.82879866, -0.77610514, -0.44735785, ...,  0.29048293,
         1.47008837,  0.40737875],
       [-0.96025727, -0.77610514, -0.44735785, ...,  0.29048293,
         1.47008837,  0.40737875]])
```

主成分分析，数据降维

Bartlett's球状检验

In [ ]:
```
from factor_analyzer.factor_analyzer import calculate_bartlett_sphericity
chi_square_value, p_value = calculate_bartlett_sphericity(data_base)
print(chi_square_value, p_value)
```

```
1071.7093081866808 1.598009223611259e-152
```

p值<0.05时，说明各变量间具有相关性，因子分析有效。

4.KMO检验

检查变量间的相关性和偏相关性，取值在0-1之间；KOM统计量越接近1，变量间的相关性越强，偏相关性越弱，因子分析的效果越好。

In [ ]:
```
# 通常取值从0.6开始进行因子分析
from factor_analyzer.factor_analyzer import calculate_kmo
kmo_all, kmo_model = calculate_kmo(data_base)
print(kmo_all)
```

```
[0.47255535 0.48255416 0.63048479 0.82178256 0.8125049  0.81898499
 0.77119596 0.81478759 0.79961415 0.57283559 0.50296886 0.60469466
 0.56499821 0.73439983 0.75768378 0.75981297]
```

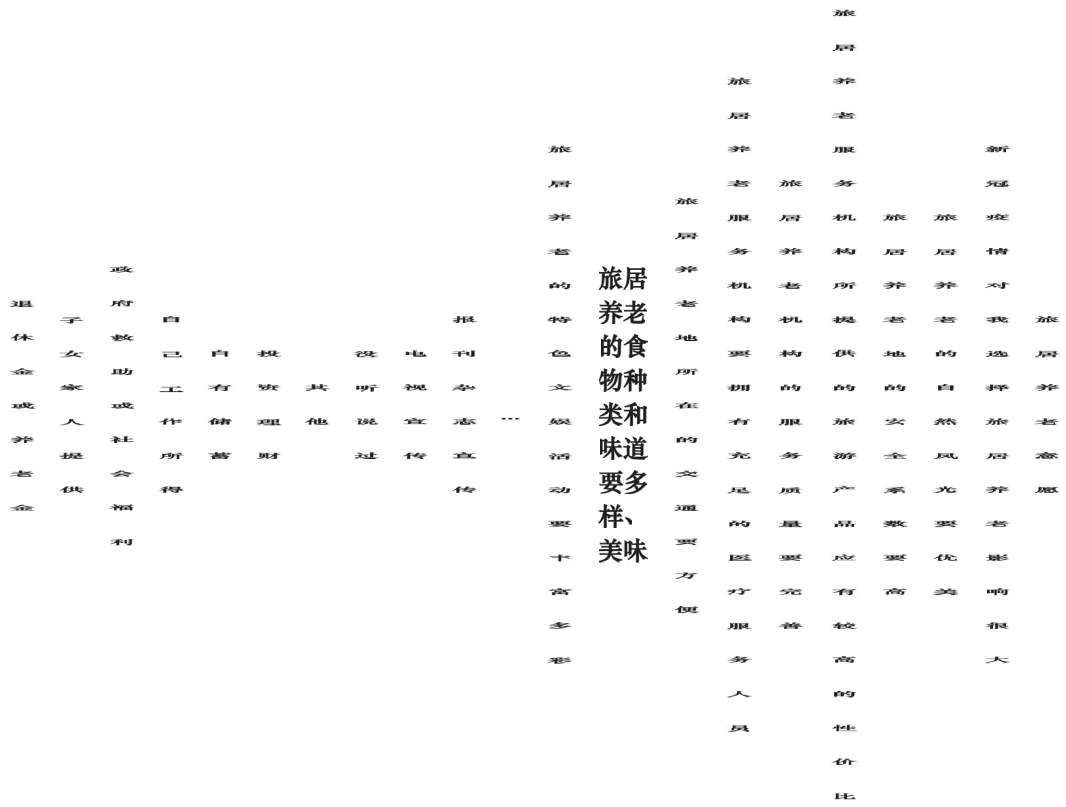
在主成分分析_选择.ipynb文件中运行，得到主成分分析的结果 'x1', 'x2', 'x3', 'x4', 'x5', 'x6', 'x7', 'x8', 'x13', 'x14'

In [ ]:
```
data_base_new=data_2[['年龄','性别','婚姻状况','教育程度','月收入状况（退休金、养老金
#将数据集保存为csv文件
data_base_new.to_csv('data_base_new.csv',index=False,sep=',',encoding='utf-8')
```

二 旅居养老意愿分析

In [ ]:
```
#第16列到最后为data_hope
data_hope=data_2.iloc[:,16:]
data_hope.head()
```

Out[ ]:



0	1	0	0	0	0	0	0	1	0	0	...	5	5	4	3	3	4	4	5	4	2
1	1	1	0	0	1	0	0	0	0	0	...	4	4	4	4	4	4	4	4	4	2
2	1	1	0	0	0	0	0	1	0	0	...	2	2	2	2	2	3	3	3	4	2
3	1	1	0	1	1	0	0	0	0	0	...	5	5	5	5	5	5	5	5	3	2
4	1	1	0	0	1	0	0	0	0	0	...	4	4	4	3	4	4	4	4	3	2

5 rows × 35 columns

In [ ]:
```
#数据标准化
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(data_hope)
data_hope = scaler.transform(data_hope)
data_hope
```

Out[ ]:
```
array([[ 0.52384979, -1.20267559, -0.50683403, ...,  0.87570082,
        -0.17653237,  0.40737875],
       [ 0.52384979,  0.83147942, -0.50683403, ..., -0.330038  ,
        -0.17653237,  0.40737875],
       [ 0.52384979,  0.83147942, -0.50683403, ..., -1.53577681,
        -0.17653237,  0.40737875],
       ...,
       [ 0.52384979, -1.20267559, -0.50683403, ..., -1.53577681,
        -0.17653237, -2.45471812],
       [-1.90894416, -1.20267559, -0.50683403, ...,  0.87570082,
        -0.17653237,  0.40737875],
       [ 0.52384979,  0.83147942,  1.97303247, ...,  0.87570082,
        -0.17653237,  0.40737875]])
```

In [ ]:
```
#转换为dataframe格式
data_hope=pd.DataFrame(data_hope)
```

```
#将data_hope保存为csv文件
data_hope.to_csv('data_hope.csv',index=False,sep=',',encoding='utf-8')
```

In [ ]:
```
from factor_analyzer.factor_analyzer import calculate_bartlett_sphericity
chi_square_value, p_value = calculate_bartlett_sphericity(data_hope[:,17:])
print(chi_square_value, p_value)
```

```
2840.189291633255 0.0
```

In [ ]:
```
# 通常取值从0.6开始进行因子分析
from factor_analyzer.factor_analyzer import calculate_kmo
kmo_all, kmo_model = calculate_kmo(data_hope[:,17:])
print(kmo_all)
```

```
[0.96753167 0.86218597 0.86471652 0.93682121 0.93317002 0.94469456
 0.94851532 0.93752486 0.91869193 0.94184338 0.94280484 0.92629545
 0.93174315 0.91557218 0.92755823 0.92358633 0.95510786 0.93736329]
```

最后选择"我曾经在旅居地投资买房 我的年出游花费消费数额较大 我的子女都支持我选择旅居养老 我曾经有过旅居养老的经历 通过宣传，我对旅居养老有了充分的了解 旅居养老地所在的交通要方便 旅居养老机构的服务质量要完善 旅居养老服务机构所提供的旅游产品应有较高的性价比 旅居养老地的安全系数要高 旅居养老的自然风光要优美"

In [ ]:
```
#data_hope_new为上面所选择的部分
data_hope_new=data_2[['我曾经在旅居地投资买房','我的年出游花费消费数额较大','我的子女
data_hope_new.head()
```

Out[ ]:

	我曾经在旅居地投资买房	我的年出游花费消费数额较大	我的子女都支持我选择旅居养老	我曾经有过旅居养老的经历	通过宣传，我对旅居养老有了充分的了解	旅居养老地所在的交通要方便	旅居养老机构的服务质量要完善	旅居养老服务机构所提供的旅游产品应有较高的性价比	旅居养老地的安全系数要高	旅居养老的自然风光要优美
0	1	2	4	2	4	4	3	4	4	5
1	1	2	4	2	3	4	4	4	4	4
2	1	2	5	2	2	2	2	3	3	3
3	1	4	4	5	4	5	5	5	5	5
4	1	2	3	1	3	4	4	4	4	4

In [ ]:
```
#写入csv文件
data_hope_new.to_csv('data_hope_new.csv',index=False,sep=',',encoding='utf-8')
```

In [ ]:
```
```