ESC21SUMMER_HW5_woohyunchoi

August 16, 2021

0.1 Factor Analysis on Stock data

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
import warnings
warnings.filterwarnings('ignore')
```

[4]:
```
# Data Import
import ssl
import pandas as pd
ssl._create_default_https_context = ssl._create_unverified_context #Github
↪            .                    !
stock = pd.read_csv('https://github.com/YonseiESC/ESC-21SUMMER/blob/main/week5/
↪stock.DAT?raw=True', header=None, delim_whitespace=True)
stock.columns = ['X1', 'X2', 'X3', 'X4', 'X5']
stock.corr()
```

[4]:

	X1	X2	X3	X4	X5
X1	1.000000	0.576924	0.508656	0.386721	0.462178
X2	0.576924	1.000000	0.598384	0.389519	0.321953
X3	0.508656	0.598384	1.000000	0.436101	0.425627
X4	0.386721	0.389519	0.436101	1.000000	0.523529
X5	0.462178	0.321953	0.425627	0.523529	1.000000

[5]:
```
eigenValues, eigenVectors = np.linalg.eig(stock.corr())
idx = eigenValues.argsort()[::-1]
eigenValues = eigenValues[idx]
eigenVectors = eigenVectors[:,idx]
```

[6]:
```
# Case 0: p factors
loadings_p = pd.DataFrame()
loadings_p['Factor1'] = np.sqrt(eigenValues[0])*eigenVectors[:,0]
loadings_p['Factor2'] = np.sqrt(eigenValues[1])*eigenVectors[:,1]
loadings_p['Factor3'] = np.sqrt(eigenValues[2])*eigenVectors[:,2]
loadings_p['Factor4'] = np.sqrt(eigenValues[3])*eigenVectors[:,3]
loadings_p['Factor5'] = np.sqrt(eigenValues[4])*eigenVectors[:,4]
```

```python
loadings_p.index = stock.columns
np.round(loadings_p, 2)
```

[6]:

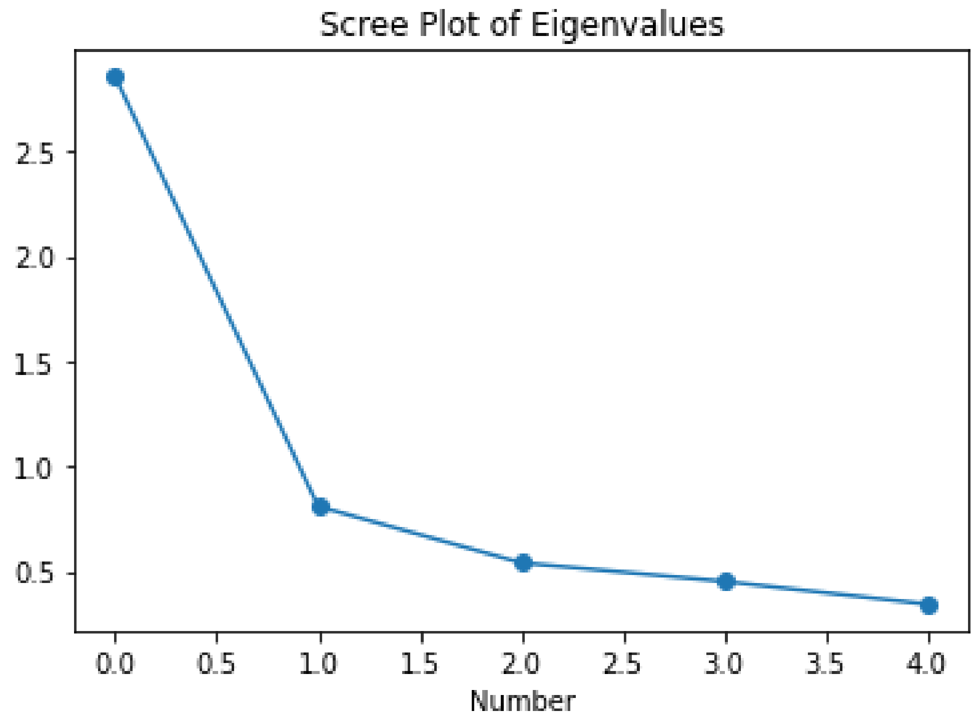
	Factor1	Factor2	Factor3	Factor4	Factor5
X1	0.78	0.22	0.45	-0.26	0.27
X2	0.77	0.46	-0.13	-0.14	-0.40
X3	0.79	0.23	-0.25	0.45	0.23
X4	0.71	-0.47	-0.40	-0.32	0.11
X5	0.71	-0.52	0.32	0.26	-0.23

[7]:
```python
eigval = pd.DataFrame()
eigval['Eigenvalue'] = eigenValues
eigval['Proportion'] = eigval['Eigenvalue'] / len(eigval)
eigval['Cumulative'] = eigval['Proportion'].cumsum(axis=0)
eigval
```

[7]:

	Eigenvalue	Proportion	Cumulative
0	2.856487	0.571297	0.571297
1	0.809118	0.161824	0.733121
2	0.540044	0.108009	0.841130
3	0.451347	0.090269	0.931399
4	0.343004	0.068601	1.000000

[8]:
```python
plt.title('Scree Plot of Eigenvalues')
plt.xlabel('Number')
plt.plot(eigenValues, 'o-')
plt.show()
```



0.1.1 Q. How many factors are required to describe adequately the space in which these data actually fall?

2 factors are required.

[9]:
```
loadings2 = pd.DataFrame()
loadings2['Factor1'] = np.sqrt(eigenValues[0])*eigenVectors[:,0]
loadings2['Factor2'] = np.sqrt(eigenValues[1])*eigenVectors[:,1]
loadings2.index = stock.columns
np.round(loadings2, 2)
```

[9]:

	Factor1	Factor2
X1	0.78	0.22
X2	0.77	0.46
X3	0.79	0.23
X4	0.71	-0.47
X5	0.71	-0.52

0.1.2 Q. Compute the communality of each variable.

[28]:
```
com = []
for i in range(5):
    com.append(loadings2['Factor1'][i]**2 + loadings2['Factor2'][i]**2)
```

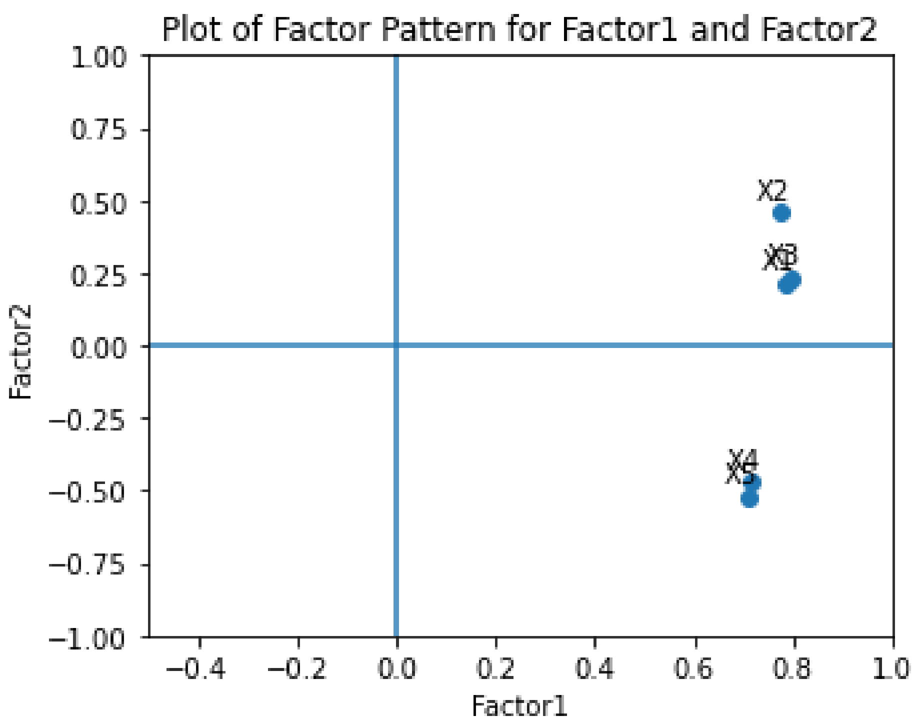
[29]:
```
loadings2['communality'] = com
```

[30]:
```
np.round(loadings2, 2)
```

[30]:

	Factor1	Factor2	communality
X1	0.78	0.22	0.66
X2	0.77	0.46	0.81
X3	0.79	0.23	0.69
X4	0.71	-0.47	0.73
X5	0.71	-0.52	0.78

[10]:
```
# Preplot(Before Rotation)
x = loadings2.Factor1 ; y = loadings2.Factor2
plt.figure(figsize = (5,4))
plt.title('Plot of Factor Pattern for Factor1 and Factor2')
plt.xlabel('Factor1') ; plt.ylabel('Factor2')
plt.scatter(x,y)
for i in range(len(loadings2)):
    plt.text(x[i]-0.05, y[i]+0.05, loadings2.index[i])
plt.axvline(x = 0) ; plt.axhline(y = 0)
plt.xlim(-0.5,1); plt.ylim(-1,1)
plt.show()
```



0.1.3 Q. Compute the 2 x 2 matrix to rotate the Xi's 45 degrees anti-clockwise. Fill in a11, a12, a21, a22 on the following code.

[19]:
```
A = pd.DataFrame([[1/np.sqrt(2), -1/np.sqrt(2)], [1/np.sqrt(2), 1/np.
 ↪sqrt(2)]],columns=('Factor1','Factor2'))
loadings_rotation = A.dot(loadings2.transpose()).transpose()
loadings_rotation.columns = ('Factor1','Factor2')
loadings_rotation
```

[19]:

	Factor1	Factor2
X1	0.400781	0.707166
X2	0.222435	0.870061
X3	0.395929	0.727410
X4	0.838037	0.169848
X5	0.873861	0.133192

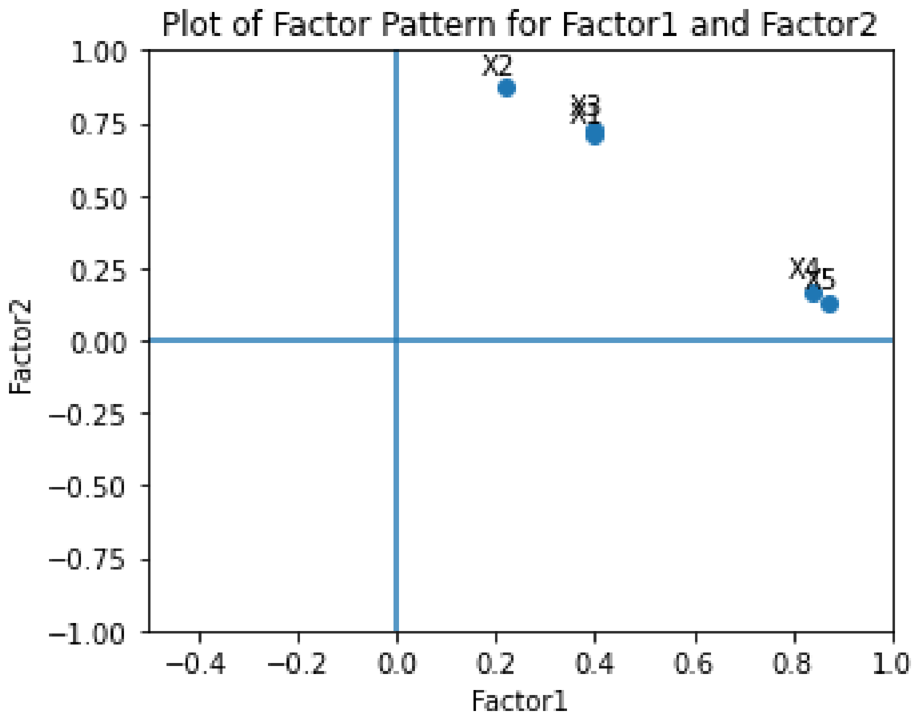
[31]:
```
com1 = []
for i in range(5):
    com1.append(loadings_rotation['Factor1'][i]**2 +␣
 ↪loadings_rotation['Factor2'][i]**2)
loadings_rotation['communality'] = com1
```

[32]:
```
np.round(loadings_rotation, 2)
```

[32]:

	Factor1	Factor2	communality
X1	0.40	0.71	0.66
X2	0.22	0.87	0.81
X3	0.40	0.73	0.69
X4	0.84	0.17	0.73
X5	0.87	0.13	0.78

[20]:
```
# Plot(After Rotation)
x = loadings_rotation.Factor1 ; y = loadings_rotation.Factor2
plt.figure(figsize = (5,4))
plt.title('Plot of Factor Pattern for Factor1 and Factor2')
plt.xlabel('Factor1') ; plt.ylabel('Factor2')
plt.scatter(x,y)
for i in range(len(loadings_rotation)):
    plt.text(x[i]-0.05, y[i]+0.05, loadings_rotation.index[i])
plt.axvline(x = 0) ; plt.axhline(y = 0)
plt.xlim(-0.5,1); plt.ylim(-1,1)
plt.show()
```


Plot of Factor Pattern for Factor1 and Factor2
X2
X3
X1
X4
X5
Factor1
Factor2
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8
1.0