EFA analysis

Made in Python

December 20, 2020

```
import pandas as pd
```

```
from sklearn.datasets import load_iris
from factor_analyzer import FactorAnalyzer
import matplotlib.pyplot as plt
```

```
df = pd.read_csv("data_efa.csv")
```

```
df
```

```
df.drop(['cinsiyet','yasaraligi',␣
↪'egitim','pozisyon','calismayili','isletmekusak'],axis=1,inplace=True)
```

```
df
```

```
df.dropna(inplace=True)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 193 entries, 0 to 192
Data columns (total 18 columns):
dtypes: int64(18)
memory usage: 28.6 KB
```

```
df.head()
```

Adequacy Test

Before you perform factor analysis, you need to evaluate the "factorability" of our dataset. Factorability means "can we found the factors in the dataset?". There are two methods to check the factorability or sampling adequacy: Bartlett's Test Kaiser-Meyer-Olkin Test Bartlett's test of sphericity checks whether or not the observed variables intercorrelate at all using the observed correlation matrix against the identity matrix. If the test found statistically insignificant, you should not employ a factor analysis.

```
from factor_analyzer.factor_analyzer import calculate_bartlett_sphericity
chi_square_value,p_value=calculate_bartlett_sphericity(df)
print((round(chi_square_value,None), round(p_value,None)))
```

```
(1456, 0)
```

In this Bartlett 's test, the p-value is 0. The test was statistically significant, indicating that the observed correlation matrix is not an identity matrix.

Kaiser-Meyer-Olkin (KMO) Test measures the suitability of data for factor analysis. It determines the adequacy for each observed variable and for the complete model. KMO estimates the proportion of variance among all the observed variable. Lower proportion id more suitable for factor analysis. KMO values range between 0 and 1. Value of KMO less than 0.6 is considered inadequate.

```
from factor_analyzer.factor_analyzer import calculate_kmo
kmo_all, kmo_model = calculate_kmo(df)
```

```
kmo_model
```

```
0.8511180227872862
```

The overall KMO for our data is 0.85, which is excellent. This value indicates that you can proceed with your planned factor analysis.

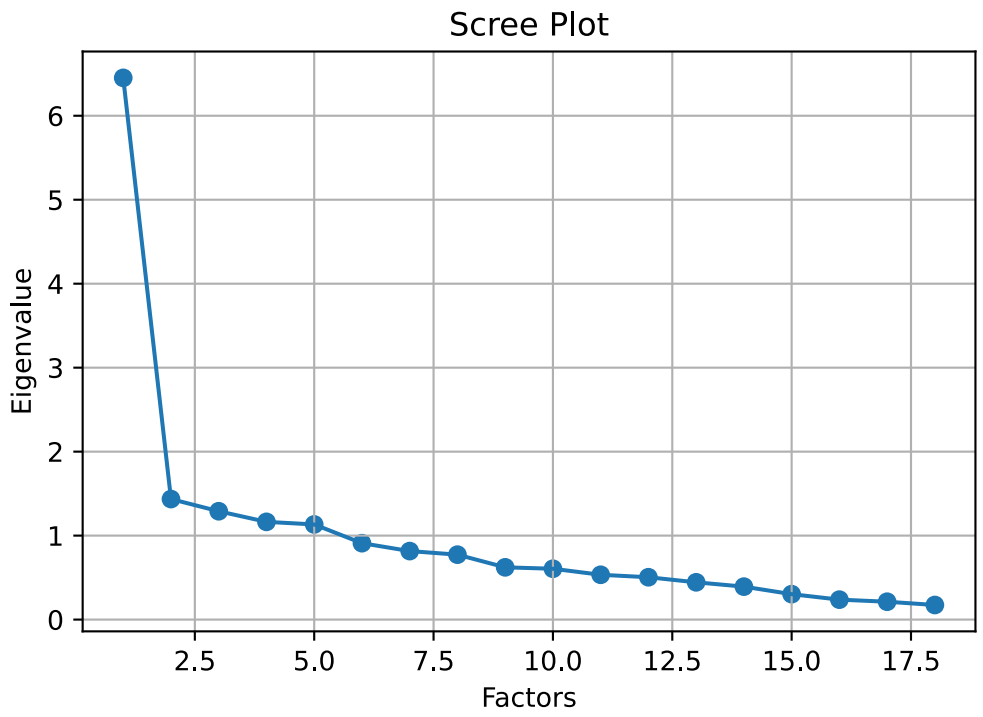
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 193 entries, 0 to 192
Data columns (total 18 columns):
```

```
fa = FactorAnalyzer(18, rotation="varimax")
fa.fit(df)
fa.loadings_
ev, v = fa.get_eigenvalues()
ev
```

```
array([6.45111848, 1.43532773, 1.29014865, 1.1636872 , 1.13292152,
       0.909978  , 0.81572196, 0.77307327, 0.62222918, 0.60532309,
       0.53354971, 0.50426572, 0.44390692, 0.39222225, 0.30275576,
       0.23739791, 0.21266934, 0.1737033 ])
```

```
plt.scatter(range(1,df.shape[1]+1),ev)
plt.plot(range(1,df.shape[1]+1),ev)
plt.title('Scree Plot')
plt.xlabel('Factors')
plt.ylabel('Eigenvalue')
plt.grid()
plt.show()
```




```
fa_1 = FactorAnalyzer(5, rotation="varimax")
fa_1.fit(df)
```

```
FactorAnalyzer(n_factors=5, rotation='varimax', rotation_kwargs={})
```

```
fa_1.loadings_
```

```
array([[-0.08715612,  0.00076917, -0.02127592, -0.43946876, -0.09425799],
       [ 0.49458846,  0.36081168,  0.27777283,  0.08404885,  0.07411627],
       [ 0.42490219,  0.37367676,  0.16795095,  0.15492366, -0.02090526],
       [ 0.73129401,  0.17664615,  0.15845688,  0.02694815,  0.08735393],
       [ 0.60347123,  0.18527698,  0.14916255,  0.43485745,  0.24313767],
       [ 0.62627345,  0.24706141,  0.12894906,  0.51937843,  0.11444781],
       [ 0.34624218,  0.51462186,  0.21426834,  0.41142763, -0.06760151],
       [ 0.05946643,  0.18768145,  0.27497555,  0.65097479,  0.05595513],
       [ 0.53268341,  0.1177069 ,  0.19008847,  0.0862303 ,  0.15617828],
       [ 0.16985577,  0.04322926,  0.34112115,  0.02075241,  0.28618944],
       [-0.01817355,  0.00106299,  0.11809976, -0.07804024, -0.28913172],
       [ 0.17927569,  0.15007302,  0.75950225,  0.26196857, -0.07845234],
       [ 0.34863969,  0.26242692,  0.7654684 ,  0.18030032, -0.06085245],
       [ 0.19517361,  0.71947921,  0.1010424 ,  0.01075777,  0.20867813],
       [ 0.20201731,  0.75561   ,  0.1266592 ,  0.16567236,  0.11824766],
       [ 0.14076991,  0.44277189,  0.25740653,  0.06921739,  0.55655685],
```

```
       [ 0.18261632,  0.29480699,  0.16069125,  0.06241127,  0.42135441],
       [ 0.26496873,  0.36532364,  0.38878073,  0.07989293,  0.18135526]])
```

```
factor_variance = fa_1.get_factor_variance()
```

```
for line in fa_1.loadings_:
    print(*line)
```

```
for line in factor_variance:
    print(*line)
```

```

```