5 Factor Analysis project

Daniel Heinsch

In the Field of Psychology, there is a popular dimension reduction method called Factor Analysis. This is a practice that is used to cluster a set of observed variables down into few hidden variables called Factors. These Factors' cumulative variance explains a large proportion of the original variables, allowing researchers and psychologists to detect relationships between variables.

Factor Analysis can be compared to PCA, but there are some notable differences between the two. For starters, PCA is made up of Principal Components, where the cumulative score explains the total variance of the data. Factor Analysis on the other hand explain covariance between variables.

Each principal component contains the linear relationship between the variables in a dataset, while factors show collinearity between the variables with some unobserved variable. This is very helpful when trying to analyze surveys that may contain hundreds of results, and this is why it is heavily used in the field of psychology.

I have always been very interested in certain topics of psychology, and personality types is one of them. I want to build a factor analysis model that will recreate the 5 factors that largely encapsulate and categorizes human personality. In this reconstruction, I will be following an article posted on DataCamp.org: https://www.datacamp.com/community/tutorials/introduction-factor-analysis.

Let's Start off by importing the packages needed to analyze this BFI data.

In [1]:
```
!pip3 install factor_analyzer
```

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

We will import the BFI dataset from https://vincentarelbundock.github.io/Rdatasets/datasets.html.

In [3]:
```
from google.colab import drive
drive.mount("/content/gdrive")
```

```
Mounted at /content/gdrive
```

In [4]:
```
df_dirty = pd.read_csv('/content/gdrive/My Drive/data/bfi.csv')
```

Now that we have it, lets get an understanding of the data that we are working with. We can see that we have 25 columns for each question that holds a value that ranges 1-6, 1 meaning that the surveyor completely disagrees with the question and 6 meaning they complete agree. These columns are pre-labeled with a given letter, meaning that the questions are already ordered under a given factor. This does not interfere with this reconstruction of the analysis and it purely orders the questions under their real factors. However, if a real factor analysis study were to be conducted on data, then we would have no idea which questions would cluster with however many factors. Keep this in mind, I will come back to this.

Along with the question columns, we see that there are additional columns. We have Unamed: 0, gender, age, and education.

When we look at the entries value, we can see that there are 2800 observations / people who took the survey. The Non-Null column shows that some people did not answer every question.

In [5]:
```
df_dirty.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2800 entries, 0 to 2799
Data columns (total 29 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Unnamed: 0  2800 non-null   int64  
 1   A1          2784 non-null   float64
 2   A2          2773 non-null   float64
 3   A3          2774 non-null   float64
 4   A4          2781 non-null   float64
 5   A5          2784 non-null   float64
 6   C1          2779 non-null   float64
 7   C2          2776 non-null   float64
 8   C3          2780 non-null   float64
 9   C4          2774 non-null   float64
 10  C5          2784 non-null   float64
 11  E1          2777 non-null   float64
 12  E2          2784 non-null   float64
 13  E3          2775 non-null   float64
 14  E4          2791 non-null   float64
 15  E5          2779 non-null   float64
 16  N1          2778 non-null   float64
 17  N2          2779 non-null   float64
 18  N3          2789 non-null   float64
 19  N4          2764 non-null   float64
 20  N5          2771 non-null   float64
 21  O1          2778 non-null   float64
 22  O2          2800 non-null   int64  
 23  O3          2772 non-null   float64
 24  O4          2786 non-null   float64
 25  O5          2780 non-null   float64
 26  gender      2800 non-null   int64  
 27  education   2577 non-null   float64
 28  age         2800 non-null   int64  
dtypes: float64(25), int64(4)
memory usage: 634.5 KB
```

Now let's start cleaning the data. We are only worried about the question scores for each instance, so I am going to delete any column that isn't just that. Additionally we have to delete any observations that have NA's. Below the charts show that 364 observations were dropped and there are now 2436 observations.

After this, we have a clean dataset for Factor Analysis!

In [6]:
```
df_dirty.drop([
               'gender',
               'education',
               'age',
               'Unnamed: 0'],
              axis=1,
              inplace=True)

df = df_dirty.dropna()

print(df.info())

print(df.head())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2436 entries, 0 to 2799
Data columns (total 25 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   A1      2436 non-null   float64
 1   A2      2436 non-null   float64
 2   A3      2436 non-null   float64
 3   A4      2436 non-null   float64
 4   A5      2436 non-null   float64
 5   C1      2436 non-null   float64
 6   C2      2436 non-null   float64
 7   C3      2436 non-null   float64
 8   C4      2436 non-null   float64
 9   C5      2436 non-null   float64
 10  E1      2436 non-null   float64
 11  E2      2436 non-null   float64
 12  E3      2436 non-null   float64
 13  E4      2436 non-null   float64
 14  E5      2436 non-null   float64
 15  N1      2436 non-null   float64
 16  N2      2436 non-null   float64
 17  N3      2436 non-null   float64
 18  N4      2436 non-null   float64
 19  N5      2436 non-null   float64
 20  O1      2436 non-null   float64
 21  O2      2436 non-null   int64  
 22  O3      2436 non-null   float64
 23  O4      2436 non-null   float64
 24  O5      2436 non-null   float64
dtypes: float64(24), int64(1)
memory usage: 494.8 KB
None
    A1   A2   A3   A4   A5   C1   C2  ...   N4   N5   O1  O2   O3   O4   O5
0  2.0  4.0  3.0  4.0  4.0  2.0  3.0  ...  2.0  3.0  3.0   6  3.0  4.0  3.0
1  2.0  4.0  5.0  2.0  5.0  5.0  4.0  ...  5.0  5.0  4.0   2  4.0  3.0  3.0
2  5.0  4.0  5.0  4.0  4.0  4.0  5.0  ...  2.0  3.0  4.0   2  5.0  5.0  2.0
3  4.0  4.0  6.0  5.0  5.0  4.0  4.0  ...  4.0  1.0  3.0   3  4.0  3.0  5.0
4  2.0  3.0  3.0  4.0  5.0  4.0  4.0  ...  4.0  3.0  3.0   3  4.0  3.0  3.0

[5 rows x 25 columns]
```

We now need to measure the factorability of the dataset. Meaning we need some metric to decide whether the observed variables have enough collinearity to validate clustering the observed variables into these hidden / factor variables. The Kaiser-Meyer-Oklin(KMO) test is just the thing we need.

The KMO Test measures each observed variable's adequacy and compiles it into a model variable -ranging from 0 to 1- that will determine whether or not to continue with the current data. The higher the model score the better, with the standard practice for the cutoff being .6 or lower.

We will use the calculate_kmo() method from the factor_analyzer package and print the kmo_model score.

In [7]:
```
from factor_analyzer.factor_analyzer import calculate_kmo
kmo_all,kmo_model=calculate_kmo(df)
print(f'The KMO Model variable : {kmo_model:6.3f}',)
```

```
The KMO Model variable :  0.849
```

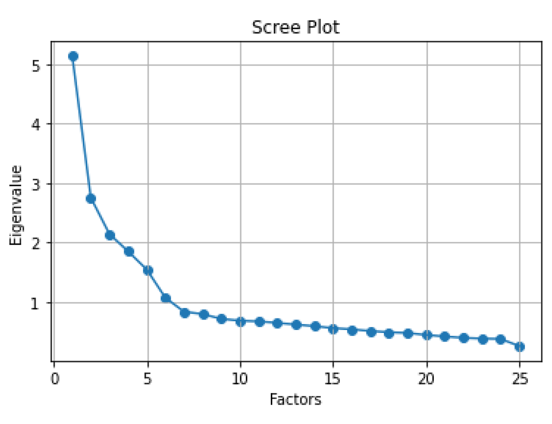
With a score of .849, that should be more than enough to pass the adequacy test, and we can confidently move forward into factor analysis.

We will now import the FactorAnalyzer class that will be used to fit the whole dataset. After fitting it we can use get_eigenvalues() to return an array of the eigenvalues. This array has ranked the eigenvalues from greatest to least. An eigenvalue over one tells us that is an additional hidden / factor variable. To restate, we essentially need to count the number of eigenvalues over 1 and that will tell us the total amount of factors to choose. We can either count the eigenvalues in the array or use a scree plot to visualize it.

In [8]:
```
from factor_analyzer import FactorAnalyzer
fa = FactorAnalyzer()
fa.fit(df)
ev, v = fa.get_eigenvalues()
print(ev)

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()
```

```
[5.13431118 2.75188667 2.14270195 1.85232761 1.54816285 1.07358247
 0.83953893 0.79920618 0.71898919 0.68808879 0.67637336 0.65179984
 0.62325295 0.59656284 0.56309083 0.54330533 0.51451752 0.49450315
 0.48263952 0.448921   0.42336611 0.40067145 0.38780448 0.38185679
 0.26253902]
```



Based on this, we will use 6 factors for further analysis. We will create another FactorAnalyzer object and fit the highest scoring eigenvalues to the dataframe.

In [9]:
```
fa_6 = FactorAnalyzer(6,rotation="varimax")
fa_6.fit(df)
index = df_columns = df.columns.to_numpy()
columns = np.array(['Factor 1','Factor 2','Factor 3','Factor 4','Factor 5','Factor 6'])
loadings = pd.DataFrame(fa_6.loadings_,index=index,columns=columns)
loadings.head()
```

Out[9]:

	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5	Factor 6
A1	0.095220	0.040783	0.048734	-0.530987	-0.113057	0.161216
A2	0.033131	0.235538	0.133714	0.661141	0.063734	-0.006244
A3	-0.009621	0.343008	0.121353	0.605933	0.033990	0.160106
A4	-0.081518	0.219717	0.235140	0.404594	-0.125338	0.086356
A5	-0.149616	0.414458	0.106382	0.469698	0.030977	0.236519

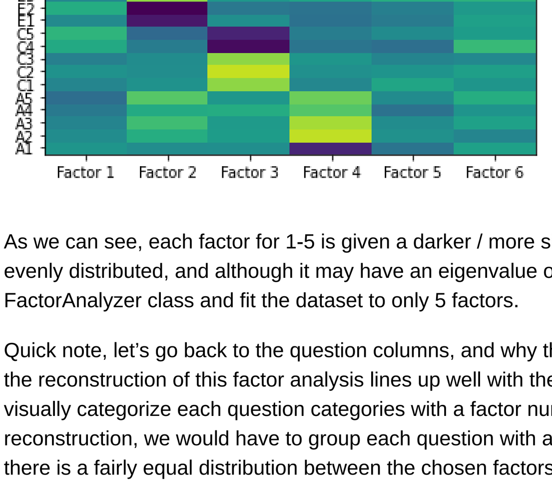
We can now plot the factor variance scores for the 6 factor object. This will return 3 arrays: **SS Loadings[ ]**, **Proportion Var[ ]**, and **Cumlative Var[ ]**. We can look at the Cumlative Variance score to see if the factors that we have chosen explain a large enough variance of the dataset.

In [10]:
```
fa6_index = ['SS Loadings','Proportion Var','Cumlative Var']
fa6_columns = columns
fa6_fv = pd.DataFrame(fa_6.get_factor_variance(),index=fa6_index,columns=fa6_columns)
print(fa6_fv)
```

```
                Factor 1  Factor 2  Factor 3  Factor 4  Factor 5  Factor 6
SS Loadings     2.726989  2.602239  2.073471  1.713499  1.504831  0.630297
Proportion Var  0.109080  0.104090  0.082939  0.068540  0.060193  0.025212
Cumlative Var   0.109080  0.213169  0.296108  0.364648  0.424841  0.450053
```

However, If we look at the Proportion Variance for each factor, we can see that factor 6 only explains around 2% of the data. Let's look at a heatmap to see if there are any significant questions explained by Factor 6.

In [11]:
```
plt.pcolor(loadings)
plt.yticks(np.arange(0.5, len(loadings.index), 1), loadings.index)
plt.xticks(np.arange(0.5, len(loadings.columns), 1), loadings.columns)
plt.show()
```



As we can see, each factor for 1-5 is given a darker / more significant chunk for each question type. However, Factor 6 is evenly distributed, and although it may have an eigenvalue of over 1, we do not need to include it. Let's create a new FactorAnalyzer class and fit the dataset to only 5 factors.

Quick note, let's go back to the question columns, and why they are pre-labeled. We can see that when we plot a heat map, the reconstruction of this factor analysis lines up well with the factor variance scores. This creates nice blocks where we can visually categorize each question categories with a factor number. But when we are doing a new study that isn't merely a reconstruction, we would have to group each question with a little extra code, and then maybe add/delete some questions so there is a fairly equal distribution between the chosen factors.

Now let's do the same thing with 5 factors instead of 6

In [12]:
```
fa_5 = FactorAnalyzer(5,rotation="varimax")
fa_5.fit(df)
index = df_columns = df.columns.to_numpy()
columns = np.array(['Factor 1','Factor 2','Factor 3','Factor 4','Factor 5'])
loadings = pd.DataFrame(fa_5.loadings_,index=index,columns=columns)
loadings.head()
```

Out[12]:

	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5
A1	0.111126	0.040465	0.022798	-0.428166	-0.077931
A2	0.029588	0.213716	0.139037	0.626946	0.062139
A3	0.009357	0.317848	0.109331	0.650743	0.056196
A4	-0.066476	0.204566	0.230584	0.435624	-0.112700
A5	-0.122113	0.393034	0.087869	0.537087	0.066708

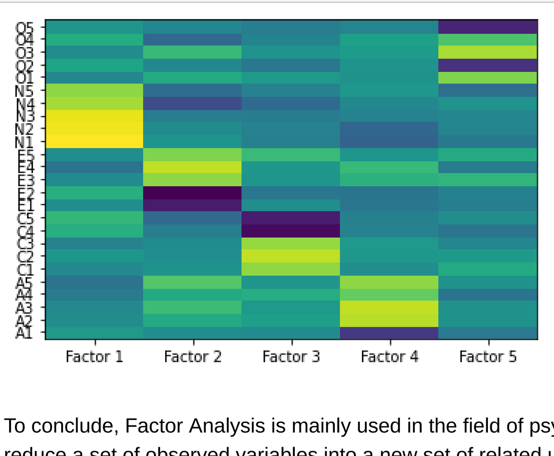
When we look at the cumulative factor variance for the 5 factors, we can see that these 5 factors explain 42% of the data, which is only slightly less to not include the 6th factor. 42% does not explain a lot of variance, but we need to consider the subject that is being analyzed, the human brain. Psychology is a very complex field where the studies often do not end with satisfying results. 42%, in actuallity, is a fairly high score, and these 5 factors are integral to modern psychology.

In [13]:
```
fa5_index = ['SS Loadings','Proportion Var','Cumlative Var']
fa5_columns = columns
fa5_fv = pd.DataFrame(fa_5.get_factor_variance(),index=fa5_index,columns=fa5_columns)
print(fa5_fv)
```

```
                Factor 1  Factor 2  Factor 3  Factor 4  Factor 5
SS Loadings     2.709633  2.473090  2.041106  1.844498  1.522153
Proportion Var  0.108385  0.098924  0.081644  0.073780  0.060886
Cumlative Var   0.108385  0.207309  0.288953  0.362733  0.423619
```

Let's show the Heat map for the 5 Factors, We can see that the prelabled data shows that Factor 1 is neuroticism, Factor 2 is Extroversion, Factor 3 is Conscientiousness, Factor 4 is Agreeableness, and Factor 5 is Openness.

In [14]:
```
plt.pcolor(loadings)
plt.yticks(np.arange(0.5, len(loadings.index), 1), loadings.index)
plt.xticks(np.arange(0.5, len(loadings.columns), 1), loadings.columns)
plt.show()
```



To conclude, Factor Analysis is mainly used in the field of psychology and it is a form of dimension reduction that aims to reduce a set of observed variables into a new set of related unobserved factor variables. These Factor can then explain a sizeable proportion of the original variables' variance.