defense

June 30, 2017

1 Kids Have Skills?

2 Is that really a thing?

3 History

- The 50's
- ok

4 Math

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.decomposition import SparsePCA, PCA
from sklearn.preprocessing import scale, normalize
from sklearn.metrics.pairwise import pairwise_distances
from sklearn.model_selection import GridSearchCV, KFold, cross_val_score
```

In [183]:
```
# %load_ext autoreload
# %autoreload 2
%pylab inline
pylab.rcParams['figure.figsize'] = (12, 8)

# Run this notebook outside of main module tree
import os
import sys
nb_dir = os.path.split(os.getcwd())[0]
if nb_dir not in sys.path:
    sys.path.append(nb_dir)

# Import our libs
from surveys.personality import *
from utils.factors import *
from factor_rotation._analytic_rotation import target_rotation
```

```
        from factor_rotation._gpa_rotation import orthomax_objective, GPA, rotateA as rotate
        from factor_rotation._wrappers import rotate_factors
```

```
Populating the interactive namespace from numpy and matplotlib
```

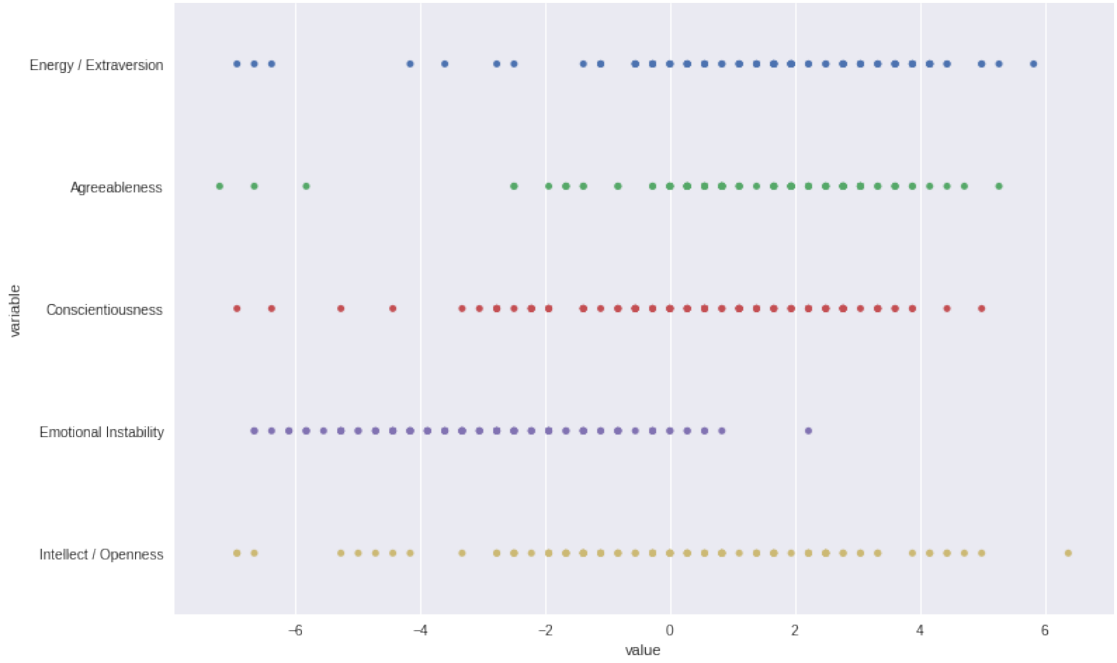
```
In [3]: X = prep_X(read_surveys("../data"))
        bf_survey = X.iloc[:, 0:65]
        bf_survey_scaled = pd.DataFrame(scale(bf_survey))
        others = X.iloc[:, 65:]
        others_scaled = pd.DataFrame(scale(others))
        ids = read_surveys("../data").user_id

        # Get the big five components, sparse positie loadings for the questions that
        # refer to personality traits.
        bigfive_key = pd.read_csv("../data/educatalyst/Auxil/q1_key_bigfive.csv")
        bf_comps = get_big_five_comps(bigfive_key)

        # Projec the survey data onto those big five personality components
        bigfive = big_five_projection(bigfive_key, bf_survey)
        bigfive_scaled = big_five_projection(bigfive_key, bf_survey_scaled)
```

```
/opt/conda/envs/python2/lib/python2.7/site-packages/sklearn/utils/validation.py:429: DataConvers
  warnings.warn(msg, _DataConversionWarning)
```

```
In [296]: def get_exlained_variance(df, L, PCA = True, p = 5):
              # sklearn mangles the PCA loadings, so we treat it differently
              if PCA:
                  F = df.dot(normalize(L).T)
                  v =  np.var(F, axis=0)/np.var(df).sum() * 100
              else:
                  v = np.sum(L ** 2, axis=1)/np.var(df).sum() * 100
              print 'Summed variance of first %s components: %s' % (p,v[0:p].sum())
              return v

          def plot_corr(A, B):
              corr = np.corrcoef(A, B, rowvar = False)
              p = sns.heatmap(pd.DataFrame(corr), center = 0)
              plt.show()
              print max_corr(A, bigfive)

          def rotated_fa(X, rotator, rotate_factors=True):
              """ rotator gets called with F factor matrix """
              fa = RotatableFA(5).fit(X)
              F = fa.transform(X)
              T = rotator(F)
              fa.rotate_components(rotator, rotate_factors)
              F = fa.transform(X)
```

```
        return fa.components_,F

    def varimax_gpa(F, Y, rotation_method = 'orthogonal', ff = None):
        T = target_rotation(F, Y)
        ff = VarimaxCorrelationObjective(Y) if ff is None else ff
        _,_,T,_ = GPA(F, ff = ff, T = T, rotation_method=rotation_method)
        return T
```

In [196]:
```
sns.stripplot(x = 'value', y = 'variable', data = pd.melt(bigfive), orient = 'h')
plt.show()
```



In [316]:
```
bf_pca = PCA(5).fit(bf_survey_scaled)
A = bf_pca.components_.T
V,T = rotate_factors(A, 'varimax')
n = np.stack([
        get_exlained_variance(bf_survey_scaled, A.T),
        get_exlained_variance(bf_survey_scaled, V.T),
        get_exlained_variance(bf_survey_scaled, bf_comps.T)]).T

from IPython.display import display, Math, Latex
print pd.DataFrame(n, columns=["Components", "Varimax", "Big Five"]).to_latex()
```

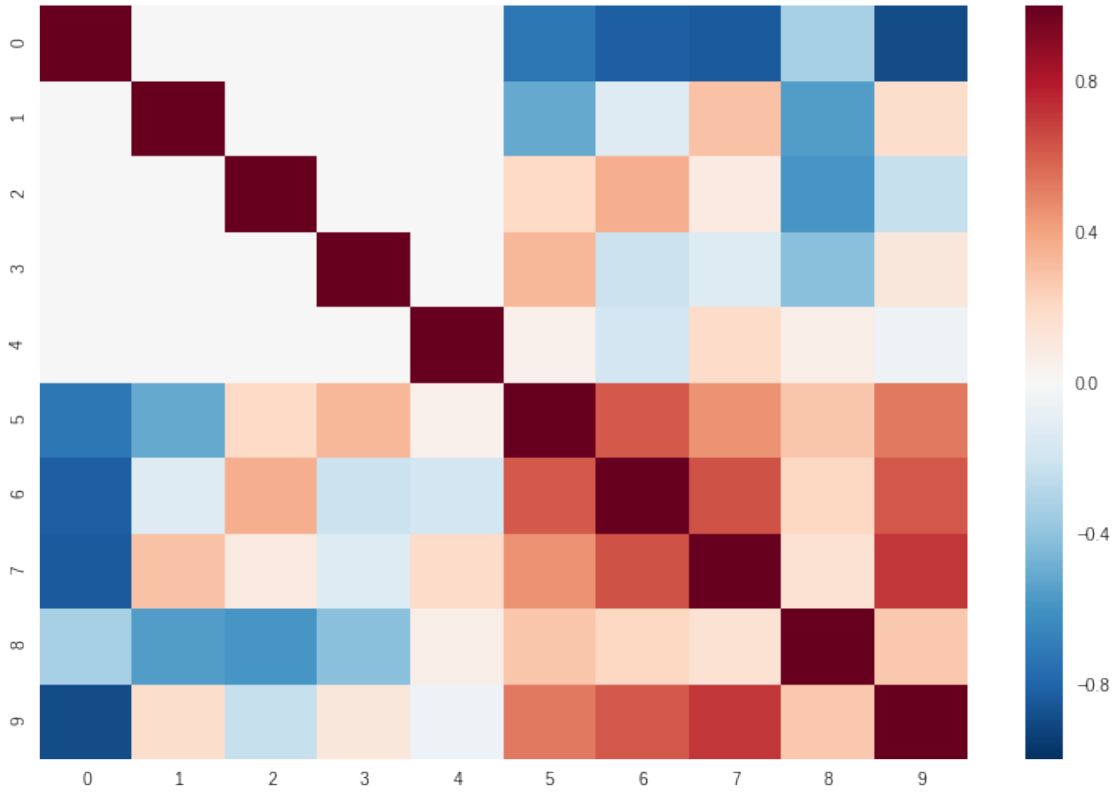
```
Summed variance of first 5 components: 46.3517700504
Summed variance of first 5 components: 46.3517700504
Summed variance of first 5 components: 37.2189799093
\begin{tabular}{lrrr}
```

```
\toprule
{} &  Components &    Varimax &  Big Five \\
\midrule
0 &   23.543123 &  14.691594 &  7.569611 \\
1 &    7.580135 &   5.573418 &  7.590264 \\
2 &    6.321525 &  12.284804 &  7.664056 \\
3 &    5.111549 &   7.432257 &  6.073631 \\
4 &    3.795437 &   6.369697 &  8.321417 \\
\bottomrule
\end{tabular}
```

In [308]:

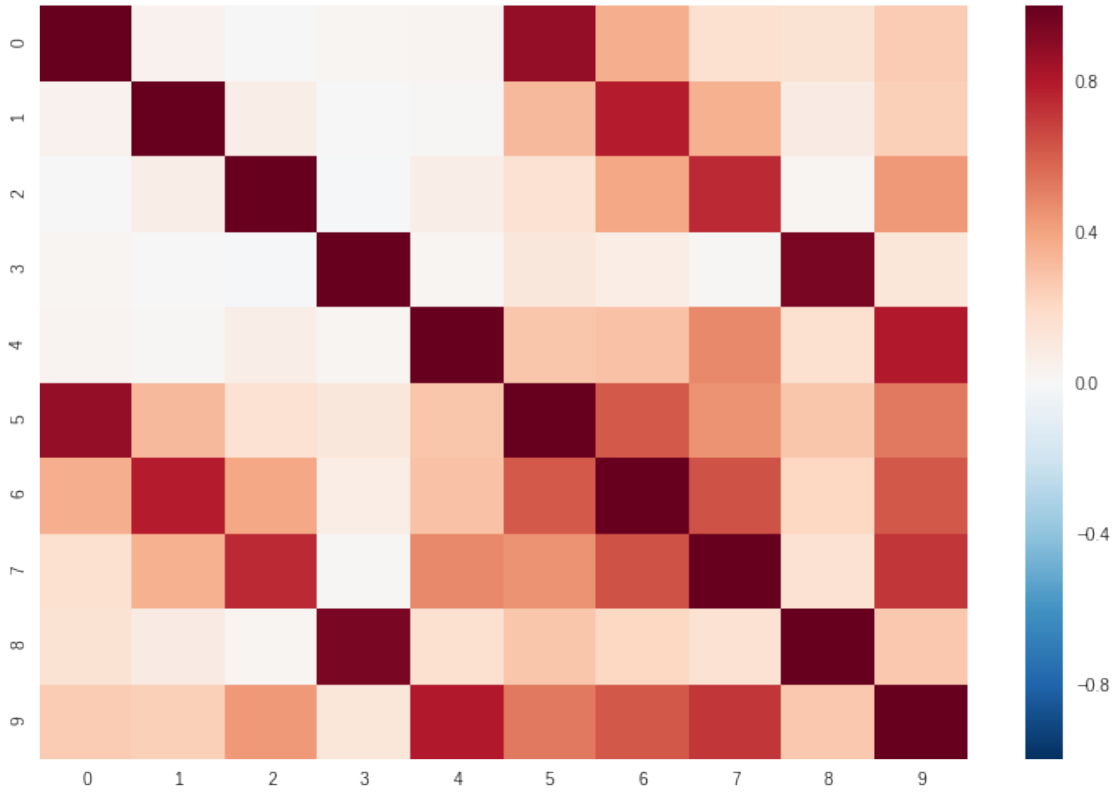
pd.DataFrame(n).to_latex()

```
In [300]: A,FA = rotated_fa(bf_survey_scaled, lambda F: np.eye(F.shape[1]))
          V,FV = rotated_fa(bf_survey_scaled, lambda L: rotate_factors(L.T, 'varimax')[1], False

          np.stack([
                  get_exlained_variance(bf_survey_scaled, A, False),
                  get_exlained_variance(bf_survey_scaled, V, False),
                  get_exlained_variance(bf_survey_scaled, bf_comps.T)]).T

Summed variance of first 5 components: 41.8972335086
Summed variance of first 5 components: 41.8972335086
Summed variance of first 5 components: 37.2189799093


Out[300]: array([[ 22.58694921,  13.64555959,   7.56961141],
                 [  6.47489471,  10.34216105,   7.59026414],
                 [  5.54495361,   7.85892645,   7.66405559],
                 [  4.3679396 ,   4.54343408,   6.07363134],
                 [  2.92249637,   5.50715234,   8.32141743]])

In [193]: # Basic Factor Analysis on Big Five Survey without rotation
          L,F = rotated_fa(bf_survey_scaled, lambda F: np.eye(F.shape[1]))
          plot_corr(F, bigfive)
```



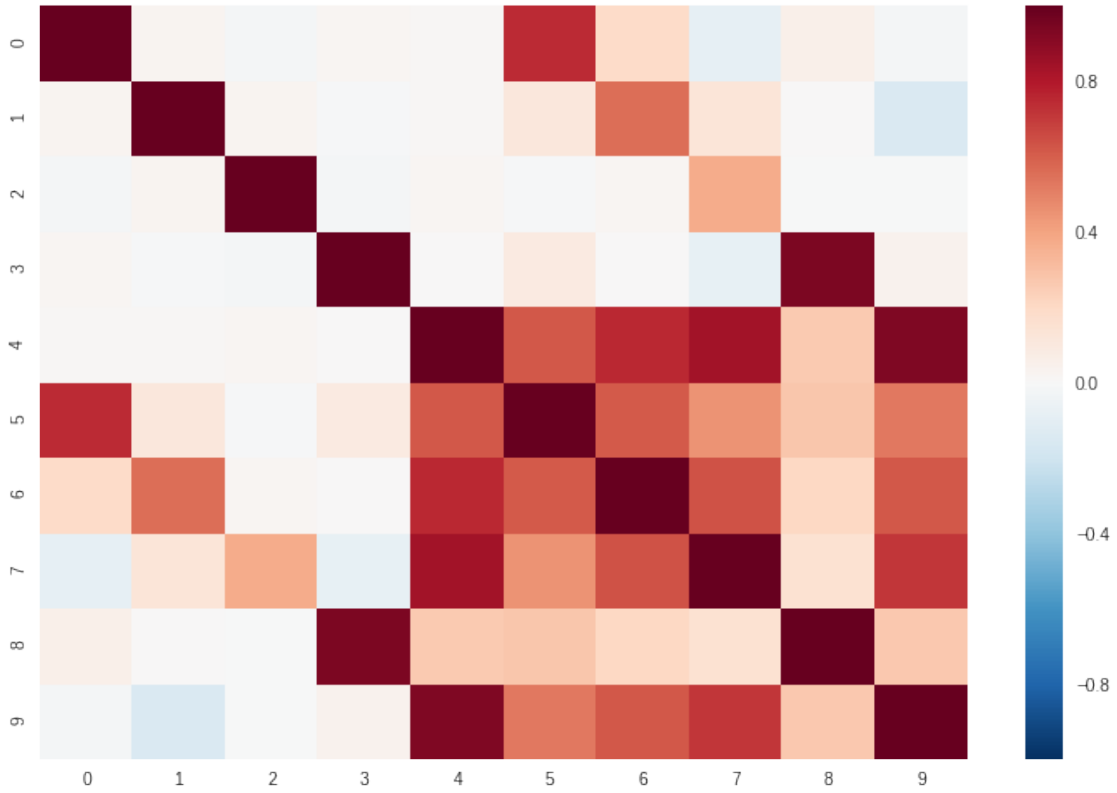

```
[ 0.33188668  0.29106531  0.35949404  0.33212472  0.19515571]
```

```
In [194]: L,F = rotated_fa(bf_survey_scaled, lambda F: target_rotation(F, bigfive))
          plot_corr(F, bigfive)
```




```
[ 0.88173985  0.78164744  0.74798731  0.95296956  0.80297988]
```

In [195]:
```
L,F = rotated_fa(bf_survey_scaled, lambda F: varimax_gpa(F, bigfive))
plot_corr(F, bigfive)
```




```
[ 0.74813693  0.55708017  0.36998887  0.93868981  0.93258694]
```

```
In [189]: L,F = rotated_fa(bf_survey_scaled, lambda F: varimax_gpa(F, bigfive, 'oblique'))
          plot_corr(F, bigfive)
```

```
        TypeErrorTraceback (most recent call last)

        <ipython-input-189-039d823b718d> in <module>()
    ----> 1 L,F = rotated_fa(bf_survey_scaled, lambda F: varimax_gpa(F, bigfive, 'oblique'))
          2 plot_corr(F, bigfive)


        <ipython-input-157-2e272f0667d4> in rotated_fa(X, rotator)
         15     fa = RotatableFA(5).fit(X)
         16     F = fa.transform(X)
    ---> 17     T = rotator(F)
         18     fa.rotate_components(rotator, True)
         19     F = fa.transform(X)
```

```
        <ipython-input-189-039d823b718d> in <lambda>(F)
    ----> 1 L,F = rotated_fa(bf_survey_scaled, lambda F: varimax_gpa(F, bigfive, 'oblique'))
          2 plot_corr(F, bigfive)


        TypeError: varimax_gpa() takes exactly 2 arguments (3 given)
```

```
In [ ]: L,F = rotated_fa(others_scaled, lambda F: target_rotation(F, bigfive))
        plot_corr(F, bigfive)
```

```
In [ ]: def norm(A=None, T=None, L=None):
            L = A.dot(T) if L is None else L
            return np.linalg.norm(L - bigfive, 1)

        L,F = rotated_fa(others_scaled, lambda F: varimax_gpa(F, bigfive, 'oblique', norm))
        plot_corr(F, bigfive)
```

```
In [135]: L,F = rotated_fa(others_scaled, lambda F: varimax_gpa(F, bigfive, 'oblique'))
          plot_corr(F, bigfive)
```

```
<matplotlib.figure.Figure at 0x7fa1e824e3d0>
```

```
[ 0.49359586  0.63905825  0.48940927  0.42176986  0.3127845 ]
```

```
In [ ]: others_pca = PCA(5).fit(others_scaled)
        A = others_pca.components_.T
        F = others_scaled.dot(A)
        T = target_rotation(F, bigfive)
        plot_corr(F.dot(T), bigfive)
```

```
In [ ]: others_pca = PCA(5).fit(others_scaled)
        A = others_pca.components_.T
        F = others_scaled.dot(A)
        T = varimax_gpa(F, bigfive, 'oblique', norm)
        plot_corr(F.dot(T), bigfive)
```

```
In [89]: from sklearn.cross_decomposition import CCA, PLSSVD
         plssvd = PLSSVD(5).fit(bf_survey_scaled, others_scaled)
         T = target_rotation(plssvd.x_scores_, bigfive)
         F = rotate(plssvd.x_scores_, T)
         plot_corr(F, bigfive)
         max_corr(F, bigfive)
```

```
<matplotlib.figure.Figure at 0x7fa1e8a78710>
```

```
[ 0.94824938  0.85805373  0.87439256  0.91508571  0.92328544]
```

```
Out[89]: array([ 0.94824938,  0.85805373,  0.87439256,  0.91508571,  0.92328544])
```

```
In [63]: F = rotate(plssvd.y_scores_, T)
         plot_corr(F, bigfive)
         print plssvd.y_scores_.shape
         get_exlained_variance(others, plssvd.y_weights_.T)
```

```
<matplotlib.figure.Figure at 0x7fa1e8b3d810>
```

```
[ 0.74175469  0.63407187  0.70374776  0.53098374  0.69675451]
(94, 5)
Summed variance of first 5 components: 41.1344585457
```

```
Out[63]: 0     3.518061
         1     5.381139
         2     3.692150
         3    11.627026
         4    16.916084
         dtype: float64
```

```
In [25]: F = others.dot(others_pca.components_.T)
         # F.dot(others.T)
         F.shape, others.shape
```

```
Out[25]: ((94, 5), (94, 38))
```

```
In [280]: # Variance described by Sparse PCA in others survey space
          others_sparse_pca = SparsePCA(8, .8).fit(others)
          get_exlained_variance(others, others_sparse_pca.components_)
```

```
Out[280]: 0    10.629493
          1    10.414766
          2     7.607404
          3     5.766579
          4     6.323208
          5     5.765240
          6     4.522215
          7     7.120568
          dtype: float64
```

```
Summed variance of first 5 components: 40.7414493917
```

4.0.1 In the space of the Big Five survey:

```
In [ ]: cv = KFold(3, random_state = 1, shuffle = True)
        enet = MultiElasticNet(alpha = 0.4, l1_ratio = .5)
        t = bigfive.as_matrix()
        fitted = enet.fit(others, t)
        reg_comps = normalize(enet.coef_).T

        # Take a look at how sparse our loadings are:
        cross_val_score(enet, others, t, cv = cv, scoring = 'neg_mean_squared_error').mean(), en
```

```
In [122]: # Variance described by Big Five Regression from others survey space
          get_exlained_variance(others, enet.coef_)
          L = np.array(enet.coef_).T
          plot_corr(others_scaled.dot(L), bigfive)
```

```
Summed variance of first 5 components: 39.3411583718
```

```
<matplotlib.figure.Figure at 0x7fa1e86c49d0>
```

```
[ 0.80027118  0.80359245  0.84047172  0.59949781  0.8253834 ]
```

```
In [91]: # Variance explained by pure PCA
         bf_survey_pca = PCA(5).fit(bf_survey)
         # get_exlained_variance(bf_survey, bf_survey_pca.components_)
         A = bf_survey_pca.components_
         V, _ = rotate_factors(A, 'varimax')
         # get_exlained_variance(bf_survey, V)
         # np.round(V*10000)
```

```
In [283]: # Variance explained by Sparse PCA
          bf_survey_sparse_pca = SparsePCA(8, .8).fit(bf_survey)
          get_exlained_variance(bf_survey, bf_survey_sparse_pca.components_)
```

```
Out[283]: 0     6.954491
          1    12.506588
          2     5.411025
          3     9.582842
          4     7.341020
          5     6.396424
          6     4.112881
          7     7.628662
          dtype: float64
```

```
Summed variance of first 5 components: 41.7959663825
```

In [154]:
```
from factor_rotation._gpa_rotation import GPA
a = np.array([[0.01,1.0], [1,0.01]])
b = np.array([[-1.0,0.01], [0.01, 1.0]])

def ff(A, T, L=None):
    # print "translate"
    # print T
    L = A.dot(T)
    # print L
    p = (L).dot(b.T).sum()
    # print p
    return 1/p

a = GPA(a, ff, max_tries = 2001)
np.round(a[0])
```

Out[154]:
```
array([[-1.,  0.],
       [ 0.,  1.]])
```