

Data Preparation

March 17, 2022

```
[1]: import sys
# !{sys.executable} -m pip install -U pandas-profiling[notebook]
# !jupyter nbextension enable --py widgetsnbextension
# !pip install matplotlib
# !pip install graphviz
```

```
[2]: import os
import numpy as np
import pandas as pd
```

```
[5]: df= pd.read_csv(r'C:\Users\zdehg\Downloads\archive\DASS_data_21.02.19\data.
→csv', error_bad_lines=False, warn_bad_lines=False, sep=r'\t' )
```

```
C:\ProgramData\Anaconda3\lib\site-packages\pandas\util\_decorators.py:311:
ParserWarning: Falling back to the 'python' engine because the 'c' engine does
not support regex separators (separators > 1 char and different from '\s+' are
interpreted as regex); you can avoid this warning by specifying engine='python'.
    return func(*args, **kwargs)
```

```
C:\ProgramData\Anaconda3\lib\site-
packages\IPython\core\interactiveshell.py:3444: FutureWarning: The
warn_bad_lines argument has been deprecated and will be removed in a future
version.
```

```
    exec(code_obj, self.user_global_ns, self.user_ns)
C:\ProgramData\Anaconda3\lib\site-
packages\IPython\core\interactiveshell.py:3444: FutureWarning: The
error_bad_lines argument has been deprecated and will be removed in a future
version.
```

```
    exec(code_obj, self.user_global_ns, self.user_ns)
```

```
[4]: df.head(10)
```

```
[4]:   Q1A  Q1I  Q1E  Q2A  Q2I  Q2E  Q3A  Q3I  Q3E  Q4A  ...  screensize  \
0     4    28  3890     4    25  2122     2    16  1944     4  ...           1
1     4     2  8118     1    36  2890     2    35  4777     3  ...           2
```

2	3	7	5784	1	33	4373	4	41	3242	1	...	2
3	2	23	5081	3	11	6837	2	37	5521	1	...	2
4	2	36	3215	2	13	7731	3	5	4156	4	...	2
5	1	18	6116	1	28	3193	2	2	12542	1	...	2
6	1	20	4325	1	34	4009	2	38	3604	3	...	2
7	1	34	4796	1	9	2618	1	39	5823	1	...	2
8	4	4	3470	4	14	2139	3	1	11043	4	...	1
9	3	38	5187	2	28	2600	4	9	2015	1	...	2

	uniquenetworklocation	hand	religion	orientation	race	voted	married	\
0		1	1	12	1	10	2	1
1		1	2	7	0	70	2	1
2		1	1	4	3	60	1	1
3		1	2	4	5	70	2	1
4		2	3	10	1	10	2	1
5		1	1	4	1	70	2	1
6		1	1	7	2	60	2	1
7		1	1	2	2	60	1	1
8		1	1	12	2	70	2	1
9		1	1	2	2	60	2	1

	familysize	major
0	2	None
1	4	None
2	3	None
3	5	biology
4	4	Psychology
5	4	None
6	4	Mechatronics engeneerieng
7	2	Music
8	4	Psychology
9	3	computer programming

[10 rows x 172 columns]

```
[244]: df.describe()
```

```
[244]:
```

	Q1A	Q1I	Q1E	Q2A	Q2I	\
count	39775.000000	39775.000000	3.977500e+04	39775.000000	39775.000000	
mean	2.619485	21.555977	6.970591e+03	2.172269	21.248070	
std	1.032117	12.133621	8.670513e+04	1.111563	12.125288	
min	1.000000	1.000000	1.800000e+02	1.000000	1.000000	
25%	2.000000	11.000000	2.664000e+03	1.000000	11.000000	
50%	3.000000	22.000000	3.609000e+03	2.000000	21.000000	
75%	4.000000	32.000000	5.358000e+03	3.000000	32.000000	
max	4.000000	42.000000	1.210228e+07	4.000000	42.000000	

	Q2E	Q3A	Q3I	Q3E	Q4A \
count	3.977500e+04	39775.000000	39775.000000	3.977500e+04	39775.000000
mean	5.332376e+03	2.226097	21.583004	7.426446e+03	1.950170
std	2.651361e+04	1.038526	12.115637	1.587024e+05	1.042218
min	1.760000e+02	1.000000	1.000000	-1.081400e+04	1.000000
25%	2.477000e+03	1.000000	11.000000	2.857000e+03	1.000000
50%	3.511000e+03	2.000000	22.000000	3.898000e+03	2.000000
75%	5.216000e+03	3.000000	32.000000	5.766000e+03	3.000000
max	2.161057e+06	4.000000	42.000000	2.858269e+07	4.000000

	...	age	screen size	uniquenetworklocation	hand \
count	...	39775.000000	39775.000000	39775.000000	39775.000000
mean	...	23.612168	1.274519	1.200025	1.13516
std	...	21.581722	0.446277	0.400024	0.40030
min	...	13.000000	1.000000	1.000000	0.00000
25%	...	18.000000	1.000000	1.000000	1.00000
50%	...	21.000000	1.000000	1.000000	1.00000
75%	...	25.000000	2.000000	1.000000	1.00000
max	...	1998.000000	2.000000	2.000000	3.00000

	religion	orientation	race	voted	married \
count	39775.000000	39775.000000	39775.000000	39775.000000	39775.000000
mean	7.555852	1.642992	31.312885	1.705795	1.159547
std	3.554395	1.351362	25.871272	0.473388	0.445882
min	0.000000	0.000000	10.000000	0.000000	0.000000
25%	4.000000	1.000000	10.000000	1.000000	1.000000
50%	10.000000	1.000000	10.000000	2.000000	1.000000
75%	10.000000	2.000000	60.000000	2.000000	1.000000
max	12.000000	5.000000	70.000000	2.000000	3.000000

	family size
count	39775.000000
mean	3.510270
std	2.141518
min	0.000000
25%	2.000000
50%	3.000000
75%	4.000000
max	133.000000

[8 rows x 170 columns]

[6]:

```
df.drop(['Q1I', 'Q1E', 'Q2I', 'Q2E', 'Q3I', 'Q3E', 'Q4I', 'Q4E', 'Q5I',
↳ 'Q5E', 'Q6I', 'Q6E', 'Q7I', 'Q7E', 'Q8I', 'Q8E', 'Q9I', 'Q9E', 'Q10I', 'Q10E',
↳ 'Q11I', 'Q11E', 'Q12I', 'Q12E', 'Q13I', 'Q13E', 'Q14I', 'Q14E', 'Q15I',
↳ 'Q15E', 'Q16I', 'Q16E', 'Q17I', 'Q17E', 'Q18I', 'Q18E', 'Q19I', 'Q19E', 'Q20I',
↳ 'Q20E', 'Q21I', 'Q21E', 'Q22I', 'Q22E', 'Q23I', 'Q23E', 'Q24I', 'Q24E', 'Q25I',
↳ 'Q25E', 'Q26I', 'Q26E', 'Q27I', 'Q27E', 'Q28I', 'Q28E', 'Q29I', 'Q29E', 'Q30I',
↳ 'Q30E', 'Q31I', 'Q31E', 'Q32I', 'Q32E', 'Q33I', 'Q33E', 'Q34I', 'Q34E', 'Q35I',
↳ 'Q35E', 'Q36I', 'Q36E', 'Q37I', 'Q37E', 'Q38I', 'Q38E', 'Q39I', 'Q39E', 'Q40I',
↳ 'Q40E', 'Q41I', 'Q41E', 'Q42I', 'Q42E'], axis=1)
```

```
[6]:      Q1A  Q2A  Q3A  Q4A  Q5A  Q6A  Q7A  Q8A  Q9A  Q10A  ...  screensize  \
0         4   4   2   4   4   4   4   4   2   1  ...           1
1         4   1   2   3   4   4   3   4   3   2  ...           2
2         3   1   4   1   4   3   1   3   2   4  ...           2
3         2   3   2   1   3   3   4   2   3   3  ...           2
4         2   2   3   4   4   2   4   4   4   3  ...           2
...
39770      2   1   3   2   3   2   1   3   1   4  ...           2
39771      3   4   3   4   3   4   4   4   3   4  ...           1
39772      2   1   2   1   1   1   1   1   2   1  ...           2
39773      3   1   2   2   3   3   3   4   3   1  ...           2
39774      2   1   2   1   4   2   1   1   1   1  ...           1
```

```
      uniquenetworklocation  hand  religion  orientation  race  voted  \
0                          1     1        12           1    10     2
1                          1     2         7           0    70     2
2                          1     1         4           3    60     1
3                          1     2         4           5    70     2
4                          2     3        10           1    10     2
...
39770                      1     1         2           4    60     2
39771                      1     1        10           0    10     2
39772                      1     1         7           1    30     1
39773                      1     1         6           1    60     1
39774                      1     1        10           1    10     1
```

```
      married  familysize      major
0           1           2         None
1           1           4         None
2           1           3         None
3           1           5      biology
4           1           4    Psychology
...
39770      1           2         None
39771      1           4    Mathematic
39772      2           3  Computer Science
39773      1           2        History
```

[39775 rows x 88 columns]

```
[7]: DASS_keys = {'Depression': [3, 5, 10, 13, 16, 17, 21, 24, 26, 31, 34, 37, 38, 42],
               'Anxiety': [2, 4, 7, 9, 15, 19, 20, 23, 25, 28, 30, 36, 40, 41],
               'Stress': [1, 6, 8, 11, 12, 14, 18, 22, 27, 29, 32, 33, 35, 39]}

DASS_bins = {'Depression': [(0, 10), (10, 14), (14, 21), (21, 28)],
             'Anxiety': [(0, 8), (8, 10), (10, 15), (15, 20)],
             'Stress': [(0, 15), (15, 19), (19, 26), (26, 34)]}

for name, keys in DASS_keys.items():
    # Subtract one to match definition of DASS score in source
    df[name] = (df.filter(regex='Q(%s)A' % '|'.join(map(str, keys))) - 1).
    sum(axis=1)

    bins = DASS_bins[name]
    bins.append( (DASS_bins[name][-1][-1], df[name].max() + 1) )
    bins = pd.IntervalIndex.from_tuples(bins, closed='left')
    df[name + '_cat'] = np.arange(len(bins))[pd.cut(df[name], bins=bins).cat.
    codes]

dass = df[DASS_keys.keys()]
dass_cat = df[[k + '_cat' for k in DASS_keys.keys()]]
```

```
[12]: # Add personality types to data
personality_types = ['Extraversion', 'Agreeableness', 'Conscientiousness',
                    'EmotionalStability', 'Openness']

# Invert some entries
tipi = df.filter(regex='TIPI\d+').copy()
tipi_inv = tipi.filter(regex='TIPI(2|4|6|8|10)').apply(lambda d: 7 - d)
tipi[tipi.columns.intersection(tipi_inv.columns)] = tipi_inv

# Calculate scores
for idx, pt in enumerate( personality_types ):
    df[pt] = tipi[['TIPI{}'.format(idx + 1), 'TIPI{}'.format(6 + idx)]]
    mean(axis=1)

personalities = df[personality_types]
personalities[['Extraversion', 'Agreeableness', 'Conscientiousness',
                'EmotionalStability', 'Openness']].describe()
```

```
[12]:      Extraversion  Agreeableness  Conscientiousness  EmotionalStability  \
count  39775.000000    39775.000000    39775.000000    39775.000000
mean    2.967278        4.040377        3.730660        2.738290
std     1.555143        1.229776        1.491769        1.527743
min     0.000000        0.000000        0.000000        0.000000
25%     1.500000        3.500000        2.500000        1.500000
50%     3.000000        4.000000        3.500000        2.500000
75%     4.000000        5.000000        5.000000        3.500000
max     7.000000        7.000000        7.000000        7.000000
```

```
      Openness
count  39775.000000
mean    4.101634
std     1.335035
min     0.000000
25%     3.500000
50%     4.000000
75%     5.000000
max     7.000000
```

```
[13]: #extracting the questions of DASS out of dataset
only_q = df.filter(regex='Q\d{1,2}A')
only_q.head(10)
```

```
[13]:      Q1A  Q2A  Q3A  Q4A  Q5A  Q6A  Q7A  Q8A  Q9A  Q10A  ...  Q33A  Q34A  Q35A  \
0      4    4    2    4    4    4    4    4    2    1  ...    2    3    4
1      4    1    2    3    4    4    3    4    3    2  ...    3    2    2
2      3    1    4    1    4    3    1    3    2    4  ...    1    4    3
3      2    3    2    1    3    3    4    2    3    3  ...    2    4    1
4      2    2    3    4    4    2    4    4    4    3  ...    4    4    3
5      1    1    2    1    3    1    1    3    3    2  ...    4    1    3
6      1    1    2    3    4    1    3    3    3    4  ...    4    3    2
7      1    1    1    1    3    2    2    1    1    1  ...    1    1    1
8      4    4    3    4    3    4    4    4    4    3  ...    4    4    2
9      3    2    4    1    4    4    3    4    4    4  ...    4    4    3
```

```
      Q36A  Q37A  Q38A  Q39A  Q40A  Q41A  Q42A
0      4      1      2      4      3      4      4
1      3      4      2      2      1      2      2
2      4      4      4      2      2      1      4
3      1      2      1      3      4      4      2
4      4      3      3      3      4      4      3
5      2      2      2      1      1      1      2
6      2      4      4      4      2      2      3
7      1      1      1      2      1      1      2
8      4      4      3      2      4      4      4
9      4      4      4      4      4      3      4
```

[10 rows x 42 columns]

```
[14]: only_q.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39775 entries, 0 to 39774
Data columns (total 42 columns):
#   Column      Non-Null Count  Dtype
---  -
0   Q1A          39775 non-null  int64
1   Q2A          39775 non-null  int64
2   Q3A          39775 non-null  int64
3   Q4A          39775 non-null  int64
4   Q5A          39775 non-null  int64
5   Q6A          39775 non-null  int64
6   Q7A          39775 non-null  int64
7   Q8A          39775 non-null  int64
8   Q9A          39775 non-null  int64
9   Q10A         39775 non-null  int64
10  Q11A         39775 non-null  int64
11  Q12A         39775 non-null  int64
12  Q13A         39775 non-null  int64
13  Q14A         39775 non-null  int64
14  Q15A         39775 non-null  int64
15  Q16A         39775 non-null  int64
16  Q17A         39775 non-null  int64
17  Q18A         39775 non-null  int64
18  Q19A         39775 non-null  int64
19  Q20A         39775 non-null  int64
20  Q21A         39775 non-null  int64
21  Q22A         39775 non-null  int64
22  Q23A         39775 non-null  int64
23  Q24A         39775 non-null  int64
24  Q25A         39775 non-null  int64
25  Q26A         39775 non-null  int64
26  Q27A         39775 non-null  int64
27  Q28A         39775 non-null  int64
28  Q29A         39775 non-null  int64
29  Q30A         39775 non-null  int64
30  Q31A         39775 non-null  int64
31  Q32A         39775 non-null  int64
32  Q33A         39775 non-null  int64
33  Q34A         39775 non-null  int64
34  Q35A         39775 non-null  int64
35  Q36A         39775 non-null  int64
36  Q37A         39775 non-null  int64
37  Q38A         39775 non-null  int64
```

```
38 Q39A      39775 non-null int64
39 Q40A      39775 non-null int64
40 Q41A      39775 non-null int64
41 Q42A      39775 non-null int64
dtypes: int64(42)
memory usage: 12.7 MB
```

```
[15]: only_q.isnull().sum()
```

```
[15]: Q1A      0
      Q2A      0
      Q3A      0
      Q4A      0
      Q5A      0
      Q6A      0
      Q7A      0
      Q8A      0
      Q9A      0
      Q10A     0
      Q11A     0
      Q12A     0
      Q13A     0
      Q14A     0
      Q15A     0
      Q16A     0
      Q17A     0
      Q18A     0
      Q19A     0
      Q20A     0
      Q21A     0
      Q22A     0
      Q23A     0
      Q24A     0
      Q25A     0
      Q26A     0
      Q27A     0
      Q28A     0
      Q29A     0
      Q30A     0
      Q31A     0
      Q32A     0
      Q33A     0
      Q34A     0
      Q35A     0
      Q36A     0
      Q37A     0
      Q38A     0
```



```
Q39A    0
Q40A    0
Q41A    0
Q42A    0
dtype: int64
```

```
[16]: only_t = df.filter(regex='TIPI\d+')
only_t
```

```
[16]:
```

	TIPI1	TIPI2	TIPI3	TIPI4	TIPI5	TIPI6	TIPI7	TIPI8	TIPI9	TIPI10
0	1	5	7	7	7	7	7	5	1	1
1	6	5	4	7	5	4	7	7	1	5
2	2	5	2	2	5	6	5	5	3	2
3	1	1	7	4	6	4	6	1	6	1
4	2	5	3	6	5	5	5	6	3	3
...
39770	2	2	3	5	6	5	5	3	3	6
39771	4	5	5	7	4	6	4	7	4	4
39772	6	6	7	5	6	3	6	1	5	4
39773	1	6	5	7	3	5	3	5	3	4
39774	6	2	3	5	6	3	5	5	1	2

[39775 rows x 10 columns]

```
[17]: only_t.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39775 entries, 0 to 39774
Data columns (total 10 columns):
#   Column   Non-Null Count  Dtype
---  -
0   TIPI1    39775 non-null  int64
1   TIPI2    39775 non-null  int64
2   TIPI3    39775 non-null  int64
3   TIPI4    39775 non-null  int64
4   TIPI5    39775 non-null  int64
5   TIPI6    39775 non-null  int64
6   TIPI7    39775 non-null  int64
7   TIPI8    39775 non-null  int64
8   TIPI9    39775 non-null  int64
9   TIPI10   39775 non-null  int64
dtypes: int64(10)
memory usage: 3.0 MB
```

```
[18]: only_t.isnull().sum()
```

```
[18]: TIPI1    0
      TIPI2    0
```

TUPI3	0
TUPI4	0
TUPI5	0
TUPI6	0
TUPI7	0
TUPI8	0
TUPI9	0
TUPI10	0

dtype: int64