HO-1

October 26, 2018

1 HO-1 Report

1.0.1 Objective

Get acquainted with the data science ecosystem by exploring tabular data and using functions for sorting, ranking and plotting its content and thereby understanding a data collection content.

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

Reading data

The way to read CSV (or any other separated value, providing the separator character) files in Pandas is by calling the method read_csv. Besides the name of the file, we add the key argument na_valuesto this method along with the character that represents "non available data" in the file.

```
In [2]: edu = pd.read_csv('files/ch02/educ_figdp_1_Data.csv',
                          na_values=':', usecols=['TIME', 'GEO', 'Value'])
        edu
```

```
Out[2]:      TIME                           GEO  Value
        0    2000  European Union (28 countries)    NaN
        1    2001  European Union (28 countries)    NaN
        2    2002  European Union (28 countries)   5.00
        3    2003  European Union (28 countries)   5.03
        4    2004  European Union (28 countries)   4.95
        5    2005  European Union (28 countries)   4.92
        6    2006  European Union (28 countries)   4.91
        7    2007  European Union (28 countries)   4.92
        8    2008  European Union (28 countries)   5.04
        9    2009  European Union (28 countries)   5.38
        10   2010  European Union (28 countries)   5.41
        11   2011  European Union (28 countries)   5.25
        12   2000  European Union (27 countries)   4.91
        13   2001  European Union (27 countries)   4.99
```

```
14   2002  European Union (27 countries)   5.00
15   2003  European Union (27 countries)   5.04
16   2004  European Union (27 countries)   4.95
17   2005  European Union (27 countries)   4.92
18   2006  European Union (27 countries)   4.91
19   2007  European Union (27 countries)   4.93
20   2008  European Union (27 countries)   5.04
21   2009  European Union (27 countries)   5.38
22   2010  European Union (27 countries)   5.41
23   2011  European Union (27 countries)   5.25
24   2000  European Union (25 countries)   4.94
25   2001  European Union (25 countries)   5.02
26   2002  European Union (25 countries)   5.03
27   2003  European Union (25 countries)   5.06
28   2004  European Union (25 countries)   4.98
29   2005  European Union (25 countries)   4.95
..    ...                            ...    ...
354  2006                       Slovenia   5.72
355  2007                       Slovenia   5.15
356  2008                       Slovenia   5.20
357  2009                       Slovenia   5.69
358  2010                       Slovenia   5.68
359  2011                       Slovenia   5.68
360  2000                       Slovakia   3.92
361  2001                       Slovakia   3.99
362  2002                       Slovakia   4.31
363  2003                       Slovakia   4.30
364  2004                       Slovakia   4.19
365  2005                       Slovakia   3.85
366  2006                       Slovakia   3.80
367  2007                       Slovakia   3.62
368  2008                       Slovakia   3.61
369  2009                       Slovakia   4.09
370  2010                       Slovakia   4.22
371  2011                       Slovakia   4.06
372  2000                        Finland   5.89
373  2001                        Finland   6.06
374  2002                        Finland   6.22
375  2003                        Finland   6.43
376  2004                        Finland   6.42
377  2005                        Finland   6.30
378  2006                        Finland   6.18
379  2007                        Finland   5.90
380  2008                        Finland   6.10
381  2009                        Finland   6.81
382  2010                        Finland   6.85
383  2011                        Finland   6.76
```

```
        [384 rows x 3 columns]
```

Q1. Which is the size of the edu DataFrame (rows x columns)?

```
In [3]: total_rows=edu.shape[0] # gets number of rows in dataframe
        print ("Number of rows:    ", total_rows)
        total_cols=edu.shape[1] # gets number of cols in dataframe
        print ("Number of columns: ", total_cols)
```

```
Number of rows:     384
Number of columns:  3
```

Q2. What happens if we give a number as argument to the method head()?

Argument value means the number of rows to be displayed.

```
In [4]: edu.head(2)
```

```
Out[4]:    TIME                            GEO  Value
        0  2000  European Union (28 countries)    NaN
        1  2001  European Union (28 countries)    NaN
```

Q3. What does the method tail()return?

It returns the last rows in dataframe according to the argument given, if argument is blank it will show last 5 rows

```
In [5]: edu.tail(3)
```

```
Out[5]:      TIME      GEO  Value
        381  2009  Finland   6.81
        382  2010  Finland   6.85
        383  2011  Finland   6.76
```

```
In [6]: edu.tail()
```

```
Out[6]:      TIME      GEO  Value
        379  2007  Finland   5.90
        380  2008  Finland   6.10
        381  2009  Finland   6.81
        382  2010  Finland   6.85
        383  2011  Finland   6.76
```

Q4. Using describe() Which measures does the result show? It seems that it shows some default values, can you guess which ones? It shows some statistic values describing the dataframe as show in the output below

```
In [7]: edu.describe()
```

```
Out[7]:                TIME       Value
        count   384.000000  361.000000
        mean   2005.500000    5.203989
        std       3.456556    1.021694
        min    2000.000000    2.880000
        25%    2002.750000    4.620000
        50%    2005.500000    5.060000
        75%    2008.250000    5.660000
        max    2011.000000    8.810000
```

If we want to select a subset of columns and rows using the labels as our references instead of the positions, we can use ilocindexing:

Q5. What does this index return? What does the first index represent? And the second one?

First index range represents the position of rows to be displayed, whereas second index are the column names selected

```
In [8]: edu.iloc[90:94][['TIME','GEO']]
```

```
Out[8]:     TIME      GEO
        90  2006  Belgium
        91  2007  Belgium
        92  2008  Belgium
        93  2009  Belgium
```

Q6.1 What does the operation edu['Value'] > 6.5 produce? It evaluates every item in Value column if value is greater than 6.5 then it returns a pandas series with the boolean result

```
In [9]: edu['Value'] > 6.5
```

```
Out[9]: 0      False
        1      False
        2      False
        3      False
        4      False
        5      False
        6      False
        7      False
        8      False
        9      False
        10     False
```

```
11     False
12     False
13     False
14     False
15     False
16     False
17     False
18     False
19     False
20     False
21     False
22     False
23     False
24     False
25     False
26     False
27     False
28     False
29     False
       ...
354    False
355    False
356    False
357    False
358    False
359    False
360    False
361    False
362    False
363    False
364    False
365    False
366    False
367    False
368    False
369    False
370    False
371    False
372    False
373    False
374    False
375    False
376    False
377    False
378    False
379    False
380    False
381     True
```

```
382     True
383     True
Name: Value, Length: 384, dtype: bool
```

Q6.2 And if we apply the index edu[edu['Value'] > 6.5]? Is this aSeries or aDataFrame?

```
In [37]: type(edu[edu['Value'] > 6.5])
```

```
Out[37]: pandas.core.frame.DataFrame
```

It returns a dataFrame displaying only items with values greater than 6.5

```
In [10]: edu[edu['Value'] > 6.5]
```

```
Out[10]:      TIME      GEO  Value
         93   2009  Belgium   6.57
         94   2010  Belgium   6.58
         95   2011  Belgium   6.55
         120  2000  Denmark   8.28
         121  2001  Denmark   8.44
         122  2002  Denmark   8.44
         123  2003  Denmark   8.33
         124  2004  Denmark   8.43
         125  2005  Denmark   8.30
         126  2006  Denmark   7.97
         127  2007  Denmark   7.81
         128  2008  Denmark   7.68
         129  2009  Denmark   8.74
         130  2010  Denmark   8.81
         131  2011  Denmark   8.75
         218  2002   Cyprus   6.60
         219  2003   Cyprus   7.37
         220  2004   Cyprus   6.77
         221  2005   Cyprus   6.95
         222  2006   Cyprus   7.02
         223  2007   Cyprus   6.95
         224  2008   Cyprus   7.45
         225  2009   Cyprus   7.98
         226  2010   Cyprus   7.92
         227  2011   Cyprus   7.87
         229  2001   Latvia   7.22
         230  2002   Latvia   6.60
         281  2005    Malta   6.58
         286  2010    Malta   6.74
         287  2011    Malta   7.96
         381  2009  Finland   6.81
         382  2010  Finland   6.85
         383  2011  Finland   6.76
```

Rearranging data

```
In [58]: filtered_data = edu[edu["TIME"] > 2005]
         pivedu = pd.pivot_table(filtered_data, values = 'Value',
                                 index = ['GEO'], columns = ['TIME'])
         pivedu.head()
```

```
Out[58]: TIME            2006  2007  2008  2009  2010  2011
         GEO
         Austria         5.40  5.33  5.47  5.98  5.91  5.80
         Belgium         5.98  6.00  6.43  6.57  6.58  6.55
         Bulgaria        4.04  3.88  4.44  4.58  4.10  3.82
         Cyprus          7.02  6.95  7.45  7.98  7.92  7.87
         Czech Republic  4.42  4.05  3.92  4.36  4.25  4.51
```

```
In [59]: pivedu.loc[['Spain','Portugal'], [2006,2011]]
```

```
Out[59]: TIME      2006  2011
         GEO
         Spain     4.26  4.82
         Portugal  5.07  5.27
```

Q7. What do you observe regarding the parameter ascending=False?

```
In [60]: pivedu = pivedu.drop(['Euro area (13 countries)',
                               'Euro area (15 countries)',
                               'Euro area (17 countries)',
                               'Euro area (18 countries)',
                               'European Union (25 countries)',
                               'European Union (27 countries)',
                               'European Union (28 countries)'
                               ], axis=0)
         pivedu = pivedu.rename(
             index={'Germany (until 1990 former territory of the FRG)': 'Germany'})
         pivedu = pivedu.dropna()
         #pivedu.rank(ascending=False, method='first')
         pivedu
```

```
Out[60]: TIME            2006  2007  2008  2009  2010  2011
         GEO
         Austria         5.40  5.33  5.47  5.98  5.91  5.80
         Belgium         5.98  6.00  6.43  6.57  6.58  6.55
         Bulgaria        4.04  3.88  4.44  4.58  4.10  3.82
         Cyprus          7.02  6.95  7.45  7.98  7.92  7.87
         Czech Republic  4.42  4.05  3.92  4.36  4.25  4.51
         Denmark         7.97  7.81  7.68  8.74  8.81  8.75
         Estonia         4.70  4.72  5.61  6.03  5.66  5.16
         Finland         6.18  5.90  6.10  6.81  6.85  6.76
         France          5.61  5.62  5.62  5.90  5.86  5.68
         Germany         4.43  4.49  4.57  5.06  5.08  4.98
         Hungary         5.44  5.29  5.10  5.12  4.90  4.71
         Ireland         4.73  4.92  5.67  6.43  6.41  6.15
         Italy           4.67  4.27  4.56  4.70  4.50  4.29
         Latvia          5.13  5.07  5.71  5.59  4.96  4.96
         Lithuania       4.82  4.64  4.88  5.64  5.36  5.17
         Malta           6.45  6.18  5.72  5.32  6.74  7.96
         Netherlands     5.50  5.32  5.50  5.95  5.98  5.93
         Poland          5.25  4.91  5.08  5.09  5.17  4.94
         Portugal        5.07  5.10  4.89  5.79  5.62  5.27
         Slovakia        3.80  3.62  3.61  4.09  4.22  4.06
         Slovenia        5.72  5.15  5.20  5.69  5.68  5.68
         Spain           4.26  4.34  4.62  5.02  4.98  4.82
```

```
In [57]: pivedu.rank(ascending=False, method='first')
```

```
Out[57]: TIME            2006  2007  2008  2009  2010  2011
         GEO
         Austria         10.0   7.0  11.0   7.0   8.0   8.0
         Belgium          5.0   4.0   3.0   4.0   5.0   5.0
         Bulgaria        21.0  21.0  20.0  20.0  22.0  22.0
         Cyprus           2.0   2.0   2.0   2.0   2.0   3.0
         Czech Republic  19.0  20.0  21.0  21.0  20.0  19.0
         Denmark          1.0   1.0   1.0   1.0   1.0   1.0
         Estonia         16.0  15.0   9.0   6.0  11.0  13.0
         Finland          4.0   5.0   4.0   3.0   3.0   4.0
         France           7.0   6.0   8.0   9.0   9.0   9.0
         Germany         18.0  17.0  18.0  17.0  15.0  14.0
         Hungary          9.0   9.0  13.0  15.0  18.0  18.0
         Ireland         15.0  13.0   7.0   5.0   6.0   6.0
         Italy           17.0  19.0  19.0  19.0  19.0  20.0
         Latvia          12.0  12.0   6.0  13.0  17.0  15.0
         Lithuania       14.0  16.0  16.0  12.0  13.0  12.0
         Malta            3.0   3.0   5.0  14.0   4.0   2.0
         Netherlands      8.0   8.0  10.0   8.0   7.0   7.0
         Poland          11.0  14.0  14.0  16.0  14.0  16.0
         Portugal        13.0  11.0  15.0  10.0  12.0  11.0
         Slovakia        22.0  22.0  22.0  22.0  21.0  21.0
         Slovenia         6.0  10.0  12.0  11.0  10.0  10.0
         Spain           20.0  18.0  17.0  18.0  16.0  17.0
```

```
In [51]: pivedu.rank(ascending=True, method='first')
```

```
Out[51]: TIME            2006  2007  2008  2009  2010  2011
         GEO
         Austria         13.0  16.0  12.0  16.0  15.0  15.0
         Belgium         18.0  19.0  20.0  19.0  18.0  18.0
         Bulgaria         2.0   2.0   3.0   3.0   1.0   1.0
         Cyprus          21.0  21.0  21.0  21.0  21.0  20.0
```

```
        Czech Republic   4.0   3.0   2.0   2.0   3.0   4.0
        Denmark         22.0  22.0  22.0  22.0  22.0  22.0
        Estonia          7.0   8.0  14.0  17.0  12.0  10.0
        Finland         19.0  18.0  19.0  20.0  20.0  19.0
        France          16.0  17.0  15.0  14.0  14.0  13.0
        Germany          5.0   6.0   5.0   6.0   8.0   9.0
        Hungary         14.0  14.0  10.0   8.0   5.0   5.0
        Ireland          8.0  10.0  16.0  18.0  17.0  17.0
        Italy            6.0   4.0   4.0   4.0   4.0   3.0
        Latvia          11.0  11.0  17.0  10.0   6.0   8.0
        Lithuania        9.0   7.0   7.0  11.0  10.0  11.0
        Malta           20.0  20.0  18.0   9.0  19.0  21.0
        Netherlands     15.0  15.0  13.0  15.0  16.0  16.0
        Poland          12.0   9.0   9.0   7.0   9.0   7.0
        Portugal        10.0  12.0   8.0  13.0  11.0  12.0
        Slovakia         1.0   1.0   1.0   1.0   2.0   2.0
        Slovenia        17.0  13.0  11.0  12.0  13.0  14.0
        Spain            3.0   5.0   6.0   5.0   7.0   6.0
```

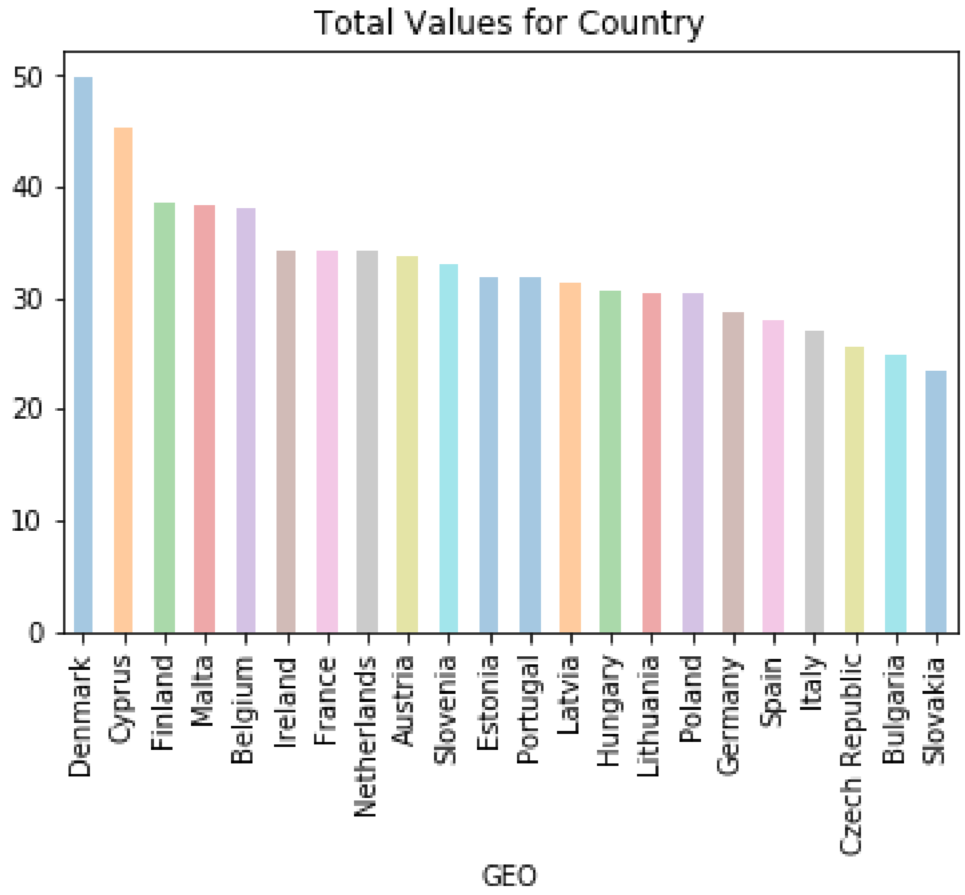
```
In [33]: totalSum = pivedu.sum(axis = 1)

         totalSum.rank(ascending = False, method = 'dense').sort_values().head()
```

```
Out[33]: GEO
         Denmark    1.0
         Cyprus     2.0
         Finland    3.0
         Malta      4.0
         Belgium    5.0
         dtype: float64
```

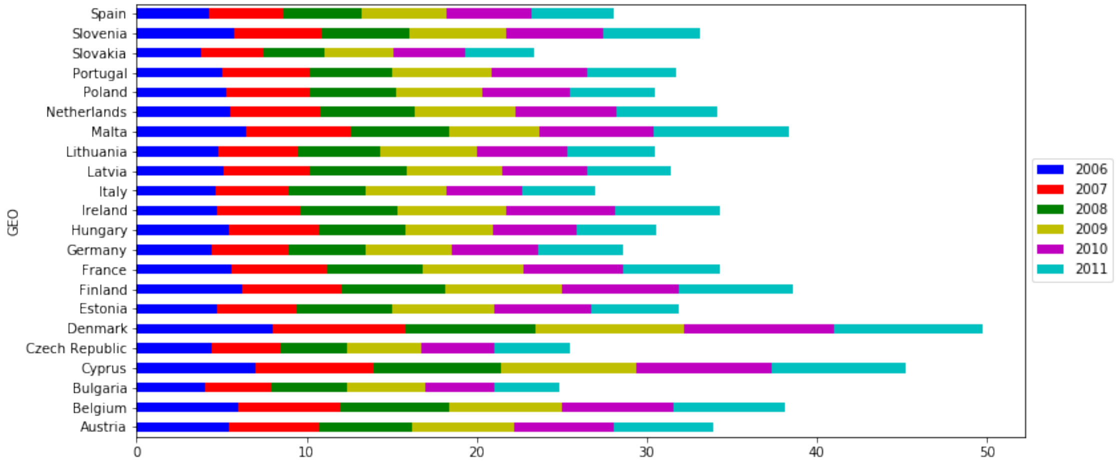
Plotting data

```
In [34]: totalSum = pivedu.sum(axis = 1).sort_values(ascending = False)
         totalSum.plot(kind = 'bar', style = 'b', alpha = 0.4,
                       title = "Total Values for Country")
```

```
Out[34]: <matplotlib.axes._subplots.AxesSubplot at 0x7f960da7b7b8>
```



```
In [35]: my_colors = ['b', 'r', 'g', 'y', 'm', 'c']
         ax = pivedu.plot(kind='barh', stacked=True, color=my_colors, figsize=(12, 6))
         ax.legend(loc='center left', bbox_to_anchor=(1, 0.5))
         plt.savefig('Value_Time_Country.png', dpi=300, bbox_inches='tight')
```


Spain
Slovenia
Slovakia
Portugal
Poland
Netherlands
Malta
Lithuania
Latvia
Italy
Ireland
Hungary
Germany
France
Finland
Estonia
Denmark
Czech Republic
Cyprus
Bulgaria
Belgium
Austria
GEO