Mid-Term Project

Jonathan Henin

Dataset is from NCES.ED.GOV

Graduation Rates 2017

[https://nces.ed.gov/ipeds/datacenter/DataFiles.aspx (https://nces.ed.gov/ipeds/datacenter/DataFiles.aspx)](https://nces.ed.gov/ipeds/datacenter/DataFiles.aspx)

The goal of this project is to clean up the data to do an analysis of a comparison of 4-year degree graduation rates versus 2-year gradutation rates.

In [1]:
```
import pandas as pd
import string
import re
import math
import numpy as np

from fuzzywuzzy import fuzz, process
from scipy import stats
from collections import OrderedDict, defaultdict
from pathlib import Path
```

```
C:\Users\jonat\Anaconda3\lib\site-packages\fuzzywuzzy\fuzz.py:11: UserWarning: Using slow pure-python Se
quenceMatcher. Install python-Levenshtein to remove this warning
  warnings.warn('Using slow pure-python SequenceMatcher. Install python-Levenshtein to remove this warni
ng')
```

There are two main files, `gr2017.csv` which has all the data, and `gr2017_Dict.xlsx` which is the data dictionary containing all the lookups for column and header values.

In [2]:
```
path_data = Path('Data/')
file_dict = path_data / 'gr2017_Dict.xlsx'
file_data = path_data / 'gr2017.csv'
```

We'll import all the individual sheets in the dictionary file into individual datasets. The `varlist` tab contains all the header values so we'll create a dictionary object to store the `varname` and `varTitle` .

In [3]:
```
df_data =  pd.read_csv(file_data)
df_varlist = pd.read_excel(open(file_dict, 'rb'), sheet_name='varlist', index_col=0)
df_freq = pd.read_excel(open(file_dict, 'rb'), sheet_name='Frequencies', index_col=0)
df_stats = pd.read_excel(open(file_dict, 'rb'), sheet_name='Statistics', index_col=0)
df_imp = pd.read_excel(open(file_dict, 'rb'), sheet_name='imputation values', skiprows=1, index_col=0)
headers_dict = dict(zip(df_varlist.varname, df_varlist.varTitle))
```

Taking a look at the shape of the dataframe we can see it has 66 columns and 54714 rows.

In [4]:
```
df_data.shape
```

Out[4]:
```
(54714, 66)
```

Taking a look at this data we can see that the column names are not very friendly to read. On top of that, it seems a lot of the row values also have lookup values. So, before we do our data checks, lets change the column names. This will make it a little easier to do the analysis.

In [5]:
```
df_data.head(5)
```

Out[5]:

	UNITID	GRTYPE	CHRTSTAT	SECTION	COHORT	LINE	XGRTOTLT	GRTOTLT	XGRTOTLM	GRTOTLM	...	XGRUNKNM	GR
0	100654	2	12	1	1	999	R	839	R	414.0	...	R	
1	100654	3	13	1	1	999	R	201	R	83.0	...	R	
2	100654	4	20	1	1	999	R	336	R	152.0	...	R	
3	100654	6	10	2	2	10	R	839	R	414.0	...	R	
4	100654	8	12	2	2	50	R	839	R	414.0	...	R	

5 rows × 66 columns

In [6]:
```
# Every dictionary in data_records represents an individual row.
# First we loop through each row, then we loop through the columns
# If the column is found in our varname list then we find it's
# corresponding lookup value in df_freq as well as lookup the column
# name.  If it is not a lookup value then we just lookup the column
# name and place in the original value.

# Columns that have lookup values
varname = ['GRTYPE', 'CHRTSTAT', 'SECTION', 'COHORT', 'LINE']

def readable(df):
    # Create a dictionary of the pandas dataframe which preserves the column order.
    dd = defaultdict(list)
    data_records = df.to_dict('records', into=dd)

    new_rows = []
    for data_dict in data_records:
        new_row = {}
        for dk, dv in data_dict.items():
            for hk, hv in headers_dict.items():
                if dk in hk:
                    # If column is a lookup column
                    if hk in varname:
                        # Lookup the varname and codevalue, if value is found, use description
                        try:
                            new_row[headers_dict.get(hk)] = df_freq.loc[(df_freq['varname']==hk) &
                                                                        (df_freq['codevalue']==str(dv).st
rip()), 'valuelabel'].values[0]
                        # If value is not there, use the original value
                        except:
                            new_row[headers_dict.get(hk)] = dv
                    # For non-lookup columns, use the original value
                    else:
                        new_row[headers_dict.get(hk)] = dv
        new_rows.append(new_row)
    return new_rows

new_rows = readable(df_data)
```

These values are much easier to read!

In [27]:

```
new_rows[0]
```

Out[27]:

```
{'Unique identification number of the institution': 100654,
 'Cohort data': '4-year institutions, Adjusted cohort (revised cohort minus exclusions)',
 'Graduation rate status in cohort': 'Adjusted cohort (revised cohort minus exclusions)',
 'Section of survey form': "Bachelor's/ equiv +  other degree/certif-seeking 2011 subcohort (4-yr instit
ution)",
 'Cohort': "Bachelor's/ equiv +  other degree/certif-seeking 2011 subcohorts (4-yr institution)",
 'Original line number of survey form': 'Generated record not on original survey form',
 'Grand total': 839,
 'Total men': 414.0,
 'Total women': 425.0,
 'American Indian or Alaska Native total': 1.0,
 'American Indian or Alaska Native men': 1.0,
 'American Indian or Alaska Native women': 0.0,
 'Asian total': 2.0,
 'Asian men': 2.0,
 'Asian women': 0.0,
 'Black or African American total': 807.0,
 'Black or African American men': 390.0,
 'Black or African American women': 417.0,
 'Hispanic total': 6.0,
 'Hispanic men': 3.0,
 'Hispanic women': 3.0,
 'Native Hawaiian or Other Pacific Islander total': 0.0,
 'Native Hawaiian or Other Pacific Islander men': 0.0,
 'Native Hawaiian or Other Pacific Islander women': 0.0,
 'White total': 16.0,
 'White men': 14.0,
 'White women': 2.0,
 'Two or more races total': 6.0,
 'Two or more races men': 3.0,
 'Two or more races women': 3.0,
 'Race/ethnicity unknown total': 1.0,
 'Race/ethnicity unknown men': 1.0,
 'Race/ethnicity unknown women': 0.0,
 'Nonresident alien total': 0.0,
 'Nonresident alien men': 0.0}
```

Now let's put it back into a DataFrame for easier reading as well as change the long name for the ID field.

In [8]:

```
df_new = pd.DataFrame(new_rows, columns=new_rows[0].keys())
df_new.rename(index=str, columns={'Unique identification number of the institution': 'ID'}, inplace=True)
```

In [9]:
```
df_new.head(5)
```

Out[9]:

	ID	Cohort data	Graduation rate status in cohort	Section of survey form	Cohort	Original line number of survey form	Grand total	Total men	Total women	American Indian or Alaska Native total	...	White men	White women	Tw mo race tot
0	100654	4-year institutions, Adjusted cohort (revised ...	Adjusted cohort (revised cohort minus exclusions)	Bachelor's/ equiv + other degree/certif-seeki...	Bachelor's/ equiv + other degree/certif-seeki...	Generated record not on original survey form	839	414.0	425.0	1.0	...	14.0	2.0	6
1	100654	4-year institutions, Completers within 150% of...	Completers within 150% of normal time	Bachelor's/ equiv + other degree/certif-seeki...	Bachelor's/ equiv + other degree/certif-seeki...	Generated record not on original survey form	201	83.0	118.0	0.0	...	2.0	1.0	2
2	100654	4-year institutions, Transfer-out students	Transfer-out students	Bachelor's/ equiv + other degree/certif-seeki...	Bachelor's/ equiv + other degree/certif-seeki...	Generated record not on original survey form	336	152.0	184.0	0.0	...	11.0	1.0	3
3	100654	Bachelor's or equiv subcohort (4-yr institution)	Revised cohort	Bachelor's or equiv 2011 subcohort (4-yr insti...	Bachelor's or equiv 2011 subcohort (4-yr inst...	Revised cohort	839	414.0	425.0	1.0	...	14.0	2.0	6
4	100654	Bachelor's or equiv subcohort (4-yr institutio...	Adjusted cohort (revised cohort minus exclusions)	Bachelor's or equiv 2011 subcohort (4-yr insti...	Bachelor's or equiv 2011 subcohort (4-yr inst...	Adjusted cohort (revised cohort minus exclusions)	839	414.0	425.0	1.0	...	14.0	2.0	6

5 rows × 35 columns

While the number of rows remained unchanged, it seems we did lose some columns. If we were to look back, it seems that headers that start with X are not represented. While we could go and change our code to bring them in, these particular rows represent information about the data, aka meta information. At the moment these aren't important but in the future might want to bring them back if it's deemed necessary.

We can now start to do some row analysis.

In [10]:
```
df_new.shape
```

Out[10]:
```
(54714, 35)
```

Doing a null analysis using the `isnull()` method we can see that there are 4733 nulls in every survey. This falls in line with the data dictionary as the total n = 54714 but the n of any total line is 49981 which is a difference of 4733.

In [11]:

```
df_new.isnull().sum()
```

Out[11]:

```
ID                                                    0
Cohort data                                           0
Graduation rate status in cohort                      0
Section of survey form                                0
Cohort                                                0
Original line number of survey form                   0
Grand total                                           0
Total men                                          4733
Total women                                        4733
American Indian or Alaska Native total             4733
American Indian or Alaska Native men               4733
American Indian or Alaska Native women             4733
Asian total                                        4733
Asian men                                          4733
Asian women                                        4733
Black or African American total                    4733
Black or African American men                      4733
Black or African American women                    4733
Hispanic total                                     4733
Hispanic men                                       4733
Hispanic women                                     4733
Native Hawaiian or Other Pacific Islander total    4733
Native Hawaiian or Other Pacific Islander men      4733
Native Hawaiian or Other Pacific Islander women    4733
White total                                        4733
White men                                          4733
White women                                        4733
Two or more races total                            4733
Two or more races men                              4733
Two or more races women                            4733
Race/ethnicity unknown total                       4733
Race/ethnicity unknown men                         4733
Race/ethnicity unknown women                       4733
Nonresident alien total                            4733
Nonresident alien men                              4733
dtype: int64
```

The goal of this analysis is to compare 4-year schools to 2-year programs, so we'll want to add a new column that has this column. It looks like `Cohort` gives us the fewest elements that have information about whether it's a 4-year or 2-year program.

In [12]:

```
cohorts = df_new['Cohort'].unique()
cohorts
```

Out[12]:

```
array(["Bachelor's/ equiv +  other degree/certif-seeking 2011 subcohorts (4-yr institution)",
       "Bachelor's or equiv 2011  subcohort (4-yr institution)",
       'Other degree/certif-seeking 2011 subcohort (4-yr institution)',
       'Degree/certif-seeking students 2014 cohort ( 2-yr institution)'],
      dtype=object)
```

Doing some fuzzy logic on finding strings that match `4-yr institution` doesn't really produce great results as the difference we are looking for is based off one character. It seems safer to just search the whole string for `4-yr`.

In [13]:

```
process.extract('4-yr institution', cohorts)
```

Out[13]:

```
[("Bachelor's/ equiv +  other degree/certif-seeking 2011 subcohorts (4-yr institution)",
  90),
 ("Bachelor's or equiv 2011  subcohort (4-yr institution)", 90),
 ('Other degree/certif-seeking 2011 subcohort (4-yr institution)', 90),
 ('Degree/certif-seeking students 2014 cohort ( 2-yr institution)', 86)]
```

In [14]:
```python
word = '4-yr'
for index, row in df_new.iterrows():
    if re.search(word, row['Cohort']):
        df_new.at[index, 'Degree'] = '4 Year'
    else:
        df_new.at[index, 'Degree'] = '2 Year'
```

In [15]:
```python
df_new.head(5)
```

Out[15]:

	ID	Cohort data	Graduation rate status in cohort	Section of survey form	Cohort	Original line number of survey form	Grand total	Total men	Total women	American Indian or Alaska Native total	...	White women	Two or more races total	Tw (mor race me
0	100654	4-year institutions, Adjusted cohort (revised ...	Adjusted cohort (revised cohort minus exclusions)	Bachelor's/ equiv + other degree/certif-seeki...	Bachelor's/ equiv + other degree/certif-seeki...	Generated record not on original survey form	839	414.0	425.0	1.0	...	2.0	6.0	3.
1	100654	4-year institutions, Completers within 150% of...	Completers within 150% of normal time	Bachelor's/ equiv + other degree/certif-seeki...	Bachelor's/ equiv + other degree/certif-seeki...	Generated record not on original survey form	201	83.0	118.0	0.0	...	1.0	2.0	1.
2	100654	4-year institutions, Transfer-out students	Transfer-out students	Bachelor's/ equiv + other degree/certif-seeki...	Bachelor's/ equiv + other degree/certif-seeki...	Generated record not on original survey form	336	152.0	184.0	0.0	...	1.0	3.0	2.
3	100654	Bachelor's or equiv subcohort (4-yr institution)	Revised cohort	Bachelor's or equiv 2011 subcohort (4-yr insti...	Bachelor's or equiv 2011 subcohort (4-yr inst...	Revised cohort	839	414.0	425.0	1.0	...	2.0	6.0	3.
4	100654	Bachelor's or equiv subcohort (4-yr institutio...	Adjusted cohort (revised cohort minus exclusions)	Bachelor's or equiv 2011 subcohort (4-yr insti...	Bachelor's or equiv 2011 subcohort (4-yr inst...	Adjusted cohort (revised cohort minus exclusions)	839	414.0	425.0	1.0	...	2.0	6.0	3.

5 rows × 36 columns

Another analysis we can do is looking for any potential outliers. Just doing a rough check of the data using the `describe()` method, none of the data looks out of reason for how many students might be attending any particular school. It's probably safe to say that we don't want to exclude any data.

```
In [16]: df_new.describe()
```

Out[16]:

	ID	Grand total	Total men	Total women	American Indian or Alaska Native total	American Indian or Alaska Native men	American Indian or Alaska Native women	Asian total	Asia
count	54714.000000	54714.000000	49981.000000	49981.000000	49981.000000	49981.000000	49981.000000	49981.000000	49981.0
mean	232152.602424	255.522389	122.746324	150.603429	1.949641	0.845341	1.104300	15.060483	7.1
std	110895.437727	751.107353	317.155525	495.144238	10.131617	4.311014	6.312001	78.071327	37.3
min	100654.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
25%	156860.000000	11.000000	4.000000	5.000000	0.000000	0.000000	0.000000	0.000000	0.0
50%	199476.000000	60.000000	27.000000	34.000000	0.000000	0.000000	0.000000	1.000000	0.0
75%	237385.000000	223.000000	109.000000	134.000000	1.000000	0.000000	1.000000	5.000000	2.0
max	490975.000000	50695.000000	13847.000000	36848.000000	486.000000	165.000000	366.000000	2359.000000	1013.0

8 rows × 30 columns

Now we want to check for any duplicated rows using a DataFrames `duplicated()` method. We can see that there are no duplicated rows.

While the above satisfies the MidTerm requirements, below is continuing on with my analysis.

The below analysis will require us to look at a subset of the data because the data lines represent individual slices of the data. We only need to look at the total number of cohorts and the number of completers. To do this it is actually easier to work with the original dataset values.

```
In [18]: df_filtered = df_data[(df_data['SECTION'].isin([1,4])) & (df_data['CHRTSTAT'].isin([12,13]))].reset_index
         ()
```

```
In [19]: df = pd.DataFrame(readable(df_filtered), columns=new_rows[0].keys())
```

```
In [20]: df.rename(index=str, columns={'Unique identification number of the institution': 'ID'}, inplace=True)
         df_c = df.copy()
```

```
In [22]: # Use the index from df but use the column names from df_filtered as they are simpler

         for i in df.index:
             val = df_filtered.at[int(i), 'SECTION']
             if val == 1:
                 df.at[i, 'Degree'] = '4 Year'
             else:
                 df.at[i, 'Degree'] = '2 Year'

         for i in df.index:
             val = df_filtered.at[int(i), 'CHRTSTAT']
             if val == 12:
                 df.at[i, 'Population'] = 'Total'
             else:
                 df.at[i, 'Population'] = 'Completers'
```

In [23]:

```
df.head(5)
```

Out[23]:

	ID	Cohort data	Graduation rate status in cohort	Section of survey form	Cohort	Original line number of survey form	Grand total	Total men	Total women	American Indian or Alaska Native total	...	Two or more races total	Two or more races men	Two o mor race wome
0	100654	4-year institutions, Adjusted cohort (revised ...	Adjusted cohort (revised cohort minus exclusions)	Bachelor's/ equiv + other degree/certif-seeki...	Bachelor's/ equiv + other degree/certif-seeki...	Generated record not on original survey form	839	414.0	425.0	1.0	...	6.0	3.0	3.
1	100654	4-year institutions, Completers within 150% of...	Completers within 150% of normal time	Bachelor's/ equiv + other degree/certif-seeki...	Bachelor's/ equiv + other degree/certif-seeki...	Generated record not on original survey form	201	83.0	118.0	0.0	...	2.0	1.0	1.
2	100663	4-year institutions, Adjusted cohort (revised ...	Adjusted cohort (revised cohort minus exclusions)	Bachelor's/ equiv + other degree/certif-seeki...	Bachelor's/ equiv + other degree/certif-seeki...	Generated record not on original survey form	1576	686.0	890.0	4.0	...	72.0	26.0	46.
3	100663	4-year institutions, Completers within 150% of...	Completers within 150% of normal time	Bachelor's/ equiv + other degree/certif-seeki...	Bachelor's/ equiv + other degree/certif-seeki...	Generated record not on original survey form	834	306.0	528.0	2.0	...	32.0	9.0	23.
4	100690	4-year institutions, Adjusted cohort (revised ...	Adjusted cohort (revised cohort minus exclusions)	Bachelor's/ equiv + other degree/certif-seeki...	Bachelor's/ equiv + other degree/certif-seeki...	Generated record not on original survey form	11	6.0	5.0	0.0	...	0.0	0.0	0.

5 rows × 37 columns

In [24]:

```
column_names = df.iloc[:,6:-2].columns.values.tolist()

# Create a new dataframe which sums data and groups on Degree and Population
df_degree = df.groupby(['Degree', 'Population'])[column_names].agg('sum').reset_index()
```

In [25]:

```
df_degree
```

Out[25]:

	Degree	Population	Grand total	Total men	Total women	American Indian or Alaska Native total	American Indian or Alaska Native men	American Indian or Alaska Native women	Asian total	Asian men	...	White men	White women	Tw r ra
0	2 Year	Completers	277161	119280.0	157881.0	2673.0	1037.0	1636.0	13297.0	5898.0	...	61782.0	71382.0	70
1	2 Year	Total	808338	377225.0	431113.0	8759.0	3955.0	4804.0	33514.0	17056.0	...	179044.0	191015.0	251
2	4 Year	Completers	997251	426934.0	570317.0	3987.0	1636.0	2351.0	71655.0	32547.0	...	272200.0	348385.0	304
3	4 Year	Total	1807156	809811.0	997345.0	12258.0	5269.0	6989.0	99765.0	47695.0	...	467486.0	555226.0	574

4 rows × 31 columns

In [26]:

```
print('2 Year Degree Graduation Rate:', '{:.2%}'.format(df_degree.at[0, 'Grand total'] / df_degree.at[1,
'Grand total']))
print('4 Year Degree Graduation Rate:', '{:.2%}'.format(df_degree.at[2, 'Grand total'] / df_degree.at[3,
'Grand total']))
```

```
2 Year Degree Graduation Rate: 34.29%
4 Year Degree Graduation Rate: 55.18%
```