Assignment 4

January 2, 2020

---

*You are currently looking at **version 1.1** of this notebook. To download notebooks and datafiles, as well as get help on Jupyter notebooks in the Coursera platform, visit the Jupyter Notebook FAQ course resource.*

```
In [4]: import pandas as pd
        import numpy as np
        from scipy.stats import ttest_ind
```

1 Assignment 4 - Hypothesis Testing

This assignment requires more individual learning than previous assignments - you are encouraged to check out the pandas documentation to find functions or methods you might not have used yet, or ask questions on Stack Overflow and tag them as pandas and python related. And of course, the discussion forums are open for interaction with your peers and the course staff.

Definitions: * A *quarter* is a specific three month period, Q1 is January through March, Q2 is April through June, Q3 is July through September, Q4 is October through December. * A *recession* is defined as starting with two consecutive quarters of GDP decline, and ending with two consecutive quarters of GDP growth. * A *recession bottom* is the quarter within a recession which had the lowest GDP. * A *university town* is a city which has a high percentage of university students compared to the total population of the city.

Hypothesis: University towns have their mean housing prices less effected by recessions. Run a t-test to compare the ratio of the mean price of houses in university towns the quarter before the recession starts compared to the recession bottom. (`price_ratio=quarter_before_recession/recession_bottom`)

The following data files are available for this assignment: * From the Zillow research data site there is housing data for the United States. In particular the datafile for all homes at a city level, `City_Zhvi_AllHomes.csv`, has median home sale prices at a fine grained level. * From the Wikipedia page on college towns is a list of university towns in the United States which has been copy and pasted into the file `university_towns.txt`. * From Bureau of Economic Analysis, US Department of Commerce, the GDP over time of the United States in current dollars (use the chained value in 2009 dollars), in quarterly intervals, in the file `gdplev.xls`. For this assignment, only look at GDP data from the first quarter of 2000 onward.

Each function in this assignment below is worth 10%, with the exception of `run_ttest()`, which is worth 50%.

```
In [27]: states = {'OH': 'Ohio', 'KY': 'Kentucky', 'AS': 'American Samoa', 'NV': 'Nevada', 'WY':

In [48]: def get_list_of_university_towns():
             '''Returns a DataFrame of towns and the states they are in from the
             university_towns.txt list. The format of the DataFrame should be:
             DataFrame( [ ["Michigan", "Ann Arbor"], ["Michigan", "Yipsilanti"] ],
             columns=["State", "RegionName"]  )

             The following cleaning needs to be done:
             1. For "State", removing characters from "[" to the end.
             2. For "RegionName", when applicable, removing every character from " (" to the end
             3. Depending on how you read the data, you may need to remove newline character '\n
             data = []
             state = None
             state_towns = []
             with open('university_towns.txt') as file:
                 for line in file:
                     thisLine = line[:-1]
                     if thisLine[-6:] == '[edit]':
                         state = thisLine[:-6]
                         continue
                     if '(' in line:
                         town = thisLine[:thisLine.index('(')-1]
                         state_towns.append([state,town])
                     else:
                         town = thisLine
                         state_towns.append([state,town])
                     data.append(thisLine)
             df = pd.DataFrame(state_towns,columns = ['State','RegionName'])
             return df

In [34]: def get_recession_start():
             '''Returns the year and quarter of the recession start time as a
             string value in a format such as 2005q3'''
             df2 = pd.read_excel('gdplev.xls', skiprows=6, header =1)
             df = df2[df2['Unnamed: 4'].str.contains('^20')][['Unnamed: 4','Unnamed: 6']]
             df = df.reset_index()
             df = df.drop('index',axis=1)
             df = df.rename(columns ={'Unnamed: 4':'Quarter', 'Unnamed: 6':'Chained value in 200
             i=0
             series_start = ''
             while (i+4 < df.shape[0]):
                 if((df['Chained value in 2009 dollars'].loc[i] > df['Chained value in 2009 doll
            (df['Chained value in 2009 dollars'].loc[i+1] > df['Chained value in 2009 dollars'].
             & (df['Chained value in 2009 dollars'].loc[i+2] < df['Chained value in 2009 dollars'
            & (df['Chained value in 2009 dollars'].loc[i+3] < df['Chained value in 2009 dollars']
                     series = df.loc[i-1][0]
                     series_start = series_start + series
```

```
                i = i+1
            return series_start

In [35]: get_recession_start()

Out[35]: '2008q3'

In [40]: def get_recession_end():
            '''Returns the year and quarter of the recession end time as a
            string value in a format such as 2005q3'''
            df2 = pd.read_excel('gdplev.xls', skiprows=6, header =1)
            df = df2[df2['Unnamed: 4'].str.contains('^20')][['Unnamed: 4','Unnamed: 6']]
            df = df.reset_index()
            df = df.drop('index',axis=1)
            df = df.rename(columns ={'Unnamed: 4':'Quarter', 'Unnamed: 6':'Chained value in 200
            i=0
            series_end = ''
            while (i+4 < df.shape[0]):
                if((df['Chained value in 2009 dollars'].loc[i] > df['Chained value in 2009 doll
                   & (df['Chained value in 2009 dollars'].loc[i+1] > df['Chained value in 2009
                   & (df['Chained value in 2009 dollars'].loc[i+2] < df['Chained value in 2009
                   & (df['Chained value in 2009 dollars'].loc[i+3] < df['Chained value in 2009
                    series = df.loc[i+4][0]
                    series_end = series_end+str(series)
                i = i+1
            return series_end

In [41]: get_recession_end()

Out[41]: '2009q4'

In [10]: def get_recession_bottom():
            '''Returns the year and quarter of the recession bottom time as a
            string value in a format such as 2005q3'''
            df2 = pd.read_excel('gdplev.xls', skiprows=6, header =1)
            df = df2[df2['Unnamed: 4'].str.contains('^20')][['Unnamed: 4','Unnamed: 6']]
            df = df.reset_index()
            df = df.drop('index',axis=1)
            df = df.rename(columns ={'Unnamed: 4':'Quarter', 'Unnamed: 6':'Chained value in 200
            i=0
            series_min = ''
            while (i+4 < df.shape[0]):
                if((df['Chained value in 2009 dollars'].loc[i] > df['Chained value in 2009 doll
                   (df['Chained value in 2009 dollars'].loc[i+1] > df['Chained value in 2009 do
                   & (df['Chained value in 2009 dollars'].loc[i+2] < df['Chained value in 2009
                   & (df['Chained value in 2009 dollars'].loc[i+3] < df['Chained value in 2009
                    series_min = series_min+str(df.loc[i+2][0])
                i = i+1
            return series_min
```

```
In [11]: get_recession_bottom()

Out[11]: '2009q2'

In [12]: def col_names():
             years = list(range(2000,2017))
             year_quart = []
             for i in years:
                 year_quart.append(str(i)+'q1')
                 year_quart.append(str(i)+'q2')
                 year_quart.append(str(i)+'q3')
                 year_quart.append(str(i)+'q4')
             return year_quart

In [28]: def convert_housing_data_to_quarters():
             '''Converts the housing data to quarters and returns it as mean
             values in a dataframe. This dataframe should be a dataframe with
             columns for 2000q1 through 2016q3, and should have a multi-index
             in the shape of ["State","RegionName"].

             Note: Quarters are defined in the assignment description, they are
             not arbitrary three month periods.

             The resulting dataframe should have 67 columns, and 10,730 rows.
             '''
             df2 = pd.read_csv("City_Zhvi_AllHomes.csv")
             df2['State'] = df2['State'].map(states)
             df2.set_index(['State','RegionName'],inplace=True)
             x = [list(df2.columns)[i*3+49:i*3+52] for i in range(0,67)]
             column_names  = col_names()
             for col,q in zip(column_names,x):
                 df2[col] = df2[q].mean(axis=1)
             df3 = pd.DataFrame(df2.iloc[:,249:317])
             return df3

In [14]: from scipy import stats

In [15]: def price_ratio(row):
             return (row['2008q3'] - row['2009q2'])/row['2008q3']

In [18]: xx = get_list_of_university_towns()['RegionName']
         xx = set(xx)

In [19]: def is_uni_town(row):
             if row['RegionName'] in xx:
                 return 1
             else:
                 return 0
```

In [20]:
```
new_data= convert_housing_data_to_quarters()
new_data = new_data.reset_index()
new_data['up&down'] = new_data.apply(price_ratio,axis=1)
new_data['is_uni'] = new_data.apply(is_uni_town,axis=1)
```

In [21]:
```
not_uni = new_data[new_data['is_uni']==0].loc[:,'up&down'].dropna()
is_uni  = new_data[new_data['is_uni']==1].loc[:,'up&down'].dropna()
```

In [22]:
```
def better():
    if not_uni.mean() < is_uni.mean():
        return 'non-university town'
    else:
        return 'university town'
```

In [23]:
```
def run_ttest():
    '''First creates new data showing the decline or growth of housing prices
    between the recession start and the recession bottom. Then runs a ttest
    comparing the university town values to the non-university towns values,
    return whether the alternative hypothesis (that the two groups are the same)
    is true or not as well as the p-value of the confidence.

    Return the tuple (different, p, better) where different=True if the t-test is
    True at a p<0.01 (we reject the null hypothesis), or different=False if
    otherwise (we cannot reject the null hypothesis). The variable p should
    be equal to the exact p value returned from scipy.stats.ttest_ind(). The
    value for better should be either "university town" or "non-university town"
    depending on which has a lower mean price ratio (which is equivilent to a
    reduced market loss).'''
    p_val = list(stats.ttest_ind(not_uni, is_uni))[1]
    result = (True,p_val,better())
    return result
```

In [24]:
```
run_ttest()
```

Out[24]:
```
(True, 0.00036641601595526971, 'university town')
```

In [ ]: