Assignment 4

May 7, 2019

---

*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 [ ]: 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 [ ]: # Use this dictionary to map state names to two letter acronyms
        states = {'OH': 'Ohio', 'KY': 'Kentucky', 'AS': 'American Samoa', 'NV': 'Nevada', 'WY':
```

```
In [ ]: import pandas as pd
        import numpy as py

        def get_list_of_university_towns():
            with open('university_towns.txt') as univ_town:
                state=[]
                for line in univ_town:
                    state.append(line[:-1])
                #print(state)

                state_town=[]
                for line in state:
                    if line[-6:] == '[edit]':
                        states = line[:-6]           # omits the last 6 chacters: [edit]
                        #print(states)
                    elif '(' in line:
                        towns=line[:(line.index('(')-1)]
                        state_town.append([states,towns])
                    else:
                        towns=line.rstrip()
                        state_town.append([states,towns])
                ans = pd.DataFrame(state_town, columns = ['State','RegionName'])
                return ans
        get_list_of_university_towns()
```

```
In [ ]: def get_recession_start():
            recess = pd.read_excel('gdplev.xls')
            recess.columns
            recess = recess[['Unnamed: 4','Unnamed: 5']]
            recess.columns = ['Quarter','GDP']
            recess = recess.iloc[219:]                   # filter on data from Q1 of 2000 onwards
            recess = recess.reset_index()
            recess = recess[['Quarter','GDP']]
            #recess
            #len(recess)


            recession_start = []
            for i in range(len(recess) - 4): #looking at the 4 quarters on a given year:
                if ((recess.iloc[i][1] > recess.iloc[i+1][1]) & (recess.iloc[i+1][1] > recess.il
                    recession_start.append(recess.iloc[i][0])

            ans = recession_start[0]
            return ans
        get_recession_start()
```

```
In [ ]: def get_recession_end():
            recess = pd.read_excel('gdplev.xls')
            recess.columns
            recess = recess[['Unnamed: 4','Unnamed: 5']]
            recess.columns = ['Quarter','GDP']
            recess = recess.iloc[219:]                  # filter on data from Q1 of 2000 onwards
            recess = recess.reset_index()
            recess = recess[['Quarter','GDP']]
            #recess
            #len(recess)

            recession_end = []
            for i in range(len(recess) - 4): #looking at the 4 quarters on a given year:
                if ((recess.iloc[i][1] > recess.iloc[i+1][1]) & (recess.iloc[i+1][1] > recess.il
                & (recess.iloc[i+2][1] < recess.iloc[i+3][1]) & (recess.iloc[i+3][1] < recess.il
                    recession_end.append([recess.iloc[i][0],recess.iloc[i+1][0],recess.iloc[i+2]

            ans = recession_end[0][4]
            return ans
        get_recession_end()

In [ ]: def get_recession_bottom():
            recess = pd.read_excel('gdplev.xls')
            recess.columns
            recess = recess[['Unnamed: 4','Unnamed: 5']]
            recess.columns = ['Quarter','GDP']
            recess = recess.iloc[219:]                  # filter on data from Q1 of 2000 onwards
            recess = recess.reset_index()
            recess = recess[['Quarter','GDP']]
            #recess
            #len(recess)

            recession_end = []
            for i in range(len(recess) - 4): #looking at the 4 quarters on a given year:
                if ((recess.iloc[i][1] > recess.iloc[i+1][1]) & (recess.iloc[i+1][1] > recess.il
                & (recess.iloc[i+2][1] < recess.iloc[i+3][1]) & (recess.iloc[i+3][1] < recess.il
                    recession_end.append([recess.iloc[i][0],recess.iloc[i+1][0],recess.iloc[i+2]

            ans = recession_end[0][2]
            return ans
        get_recession_bottom()

In [ ]: def convert_housing_data_to_quarters():

            housing_data = pd.read_csv('City_Zhvi_AllHomes.csv')
            housing_data = housing_data.drop(housing_data.columns[[0]+list(range(3,51))],axis=1)
            housing_data_2 = pd.DataFrame(housing_data[['State','RegionName']])
        states = {'OH': 'Ohio', 'KY': 'Kentucky', 'AS': 'American Samoa', 'NV': 'Nevada', 'W

        for year in range(2000,2016):
            housing_data_2[str(year)+'q1'] = housing_data[[str(year)+'-01',str(year)+'-02',s
            housing_data_2[str(year)+'q2'] = housing_data[[str(year)+'-04',str(year)+'-05',s
            housing_data_2[str(year)+'q3'] = housing_data[[str(year)+'-07',str(year)+'-08',s
            housing_data_2[str(year)+'q4'] = housing_data[[str(year)+'-10',str(year)+'-11',s

        year = 2016
        housing_data_2[str(year)+'q1'] = housing_data[[str(year)+'-01',str(year)+'-02',str(y
        housing_data_2[str(year)+'q2'] = housing_data[[str(year)+'-04',str(year)+'-05',str(y
        housing_data_2[str(year)+'q3'] = housing_data[[str(year)+'-07',str(year)+'-08']].mea
        housing_data_2 = housing_data_2.replace({'State':states})
        housing_data_2 = housing_data_2.set_index(['State','RegionName'])
        return housing_data_2

    convert_housing_data_to_quarters()
```

```
In [ ]: def run_ttest():
            from scipy.stats import ttest_ind

            univ_town = get_list_of_university_towns()
            recess_bottom = get_recession_bottom()
            recess_start = get_recession_start()
            housing_data = convert_housing_data_to_quarters()
            bstart = housing_data.columns[housing_data.columns.get_loc(recess_start)-1] #auarter
            #bstart

            # create the ratio variable
            housing_data['ratio'] =  housing_data[bstart] / housing_data[recess_bottom]
            housing_data = housing_data[[recess_bottom, bstart, 'ratio']]
            housing_data = housing_data.reset_index()
            housing_data


            univtowns_hdata = pd.merge(housing_data,univ_town,how='inner',on=['State','RegionNam
            univtowns_hdata['uni'] = True  # nice way to flag univ_town

            hdata2 = pd.merge(housing_data, univtowns_hdata, how='outer', on=['State','RegionNam
            hdata2['uni'] = hdata2['uni'].fillna(False)

            university_town = hdata2[hdata2['uni'] == True]
            non_university_town = hdata2[hdata2['uni'] == False]

            # run t-test
            t,p = ttest_ind(university_town['ratio'].dropna(), non_university_town['ratio'].drop

            different = True if p<0.01 else False
```

```
    better = "university town" if university_town['ratio'].mean() < non_university_town[
    return(different, p, better)

run_ttest()
```