# Assignment 4

January 27, 2017

---

*You are currently looking at **version 1.0** 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 [14]: 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 [15]: # Use this dictionary to map state names to two letter acronyms
         states = {'OH': 'Ohio', 'KY': 'Kentucky', 'AS': 'American Samoa', 'NV': 'N
```

```
In [27]: 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 "
             3. Depending on how you read the data, you may need to remove newline
             fp = open('university_towns.txt', 'r')
             lines = fp.readlines()
             fp.close()

             state = ''
             stateCity = []
             for i in range(len(lines)):
                 if '[edit]' in lines[i]:
                     state = lines[i].split('[edit]')[0]
                     townCity = []
                     for j in range(i+1, len(lines)):
                         i += 1
                         if '[edit]' in lines[j]:
                             break
                         townCity = [state, lines[j].split(' (')[0].strip()]
                         stateCity.append(townCity)
             ut = pd.DataFrame(stateCity, columns=['State', 'RegionName'])
             return ut

         get_list_of_university_towns().head()
```

```
Out[27]:      State    RegionName
         0  Alabama        Auburn
         1  Alabama      Florence
         2  Alabama  Jacksonville
         3  Alabama    Livingston
         4  Alabama    Montevallo
```

```
In [17]: def get_recession_start():
             '''Returns the year and quarter of the recession start time as a
             string value in a format such as 2005q3'''
             gdp = pd.read_excel('gdplev.xls', skiprows = 7)
             gdp = pd.DataFrame(gdp)#, columns=['1', '2', '3', '4', 'time', 'gdpCu
             gdp = gdp[gdp.columns[4:7]]
```

```
    gdp = gdp.rename(columns = {'Unnamed: 4':'time', 'Unnamed: 5':'gdpCurr
    gdp['year'] = gdp['time'].apply(lambda x: x.split('q')[0])
    gdp = gdp[gdp['year'].apply(lambda x: int(x) >= 2000)]

    index = gdp.index
    Start = 0
    for i in range(len(index)-2):
        if (gdp.loc[index[i], 'gdpChained2009'] > gdp.loc[index[i+1], 'gdp
            #print (index[i], index[i+1], index[i+2], gdp.loc[index[i], 'g
            Start = index[i+1]
            break

    return gdp.loc[Start,'time']

get_recession_start()
```

Out[17]: '2008q3'

In [18]:
```
def get_recession_end():
    '''Returns the year and quarter of the recession end time as a
    string value in a format such as 2005q3'''
    gdp = pd.read_excel('gdplev.xls', skiprows = 7)
    gdp = pd.DataFrame(gdp)#, columns=['1', '2', '3', '4', 'time', 'gdpCu
    gdp = gdp[gdp.columns[4:7]]
    gdp = gdp.rename(columns = {'Unnamed: 4':'time', 'Unnamed: 5':'gdpCurr
    gdp['year'] = gdp['time'].apply(lambda x: x.split('q')[0])
    gdp = gdp[gdp['year'].apply(lambda x: int(x) >= 2000)]

    index = gdp.index
    Start = 0
    for i in range(len(index)-2):
        if (gdp.loc[index[i], 'gdpChained2009'] > gdp.loc[index[i+1], 'gdp
            #print (index[i], index[i+1], index[i+2], gdp.loc[index[i], 'g
            Start = i
            break

    End = 0
    for i in range(Start, len(index)-2):
        if (gdp.loc[index[i], 'gdpChained2009'] < gdp.loc[index[i+1], 'gdp
            #print (index[i], index[i+1], index[i+2], gdp.loc[index[i], 'g
            End = index[i+2]
            break

    return gdp.loc[End, 'time']
get_recession_end()
```

Out[18]: '2009q4'

In [19]:
```
def get_recession_bottom():
    '''Returns the year and quarter of the recession bottom time as a
```

```
             string value in a format such as 2005q3'''
             gdp = pd.read_excel('gdplev.xls', skiprows = 7)
             gdp = pd.DataFrame(gdp)#, columns=['1', '2', '3', '4', 'time', 'gdpCur
             gdp = gdp[gdp.columns[4:7]]
             gdp = gdp.rename(columns = {'Unnamed: 4':'time', 'Unnamed: 5':'gdpCurr
             gdp['year'] = gdp['time'].apply(lambda x: x.split('q')[0])
             gdp = gdp[gdp['year'].apply(lambda x: int(x) >= 2000)]

             index = gdp.index
             Start = 0
             for i in range(len(index)-2):
                 if (gdp.loc[index[i], 'gdpChained2009'] > gdp.loc[index[i+1], 'gdp
                     #print (index[i], index[i+1], index[i+2], gdp.loc[index[i], 'g
                     Start = i+1
                     break

             End = 0
             for i in range(Start, len(index)-2):
                 if (gdp.loc[index[i], 'gdpChained2009'] < gdp.loc[index[i+1], 'gdp
                     #print (index[i], index[i+1], index[i+2], gdp.loc[index[i], 'g
                     End = index[i+2]
                     break

             Start = index[Start]

             return gdp.loc[gdp.loc[Start:End, 'gdpChained2009'].argmin(), 'time']

         get_recession_bottom()

Out[19]: '2009q2'

In [20]: 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.
             '''

             house = pd.read_csv('City_Zhvi_AllHomes.csv')
             house['State'] = house['State'].map(states)
             houseT = house.T.reset_index(drop=False)

             houseT['year'] = houseT['index'].apply(lambda x: x.split('-')[0] if '-
             houseT['month'] = houseT['index'].apply(lambda x: x.split('-')[1] if '

             houseT = houseT[(houseT['year']>='2000') | (houseT['year'] == 'NaN')]
             houseT = houseT.set_index('index')
             houseT['quarter'] = houseT['month'].apply(lambda x: 'q1' if x in ['01'
                                                           else 'q3' if x in ['07',
                                                           else 'NaN')
             houseT['yearQ'] = houseT['year']+houseT['quarter']
             houseT.loc['RegionName', 'yearQ'] = 'YearQReg'
             houseT.loc['State', 'yearQ'] = 'YearQSta'
             houseT = houseT.drop(['year', 'month', 'quarter'], 1)
             houseT = houseT.drop(['RegionID', 'Metro', 'CountyName', 'SizeRank'],
             houseT = houseT.reset_index()
             houseT = houseT.set_index(['yearQ', 'index'], 1)
             Qhouse = houseT.T
             Qhouse = Qhouse.set_index([('YearQSta', 'State'), ('YearQReg', 'Region
             Qhouse[:] = Qhouse[:].astype(float)
             QhouseAvg = Qhouse.groupby(axis=1, level=0).mean()
             QhouseAvg.index.names = ['State','RegionName']

             return QhouseAvg

         convert_housing_data_to_quarters().ix[:, '2008q3':'2009q4'].head()

Out[20]: yearQ                            2008q3         2008q4         2009q1  \
         State        RegionName
         New York     New York     499766.666667  487933.333333  477733.333333
         California   Los Angeles  469500.000000  443966.666667  426266.666667
         Illinois     Chicago      232000.000000  227033.333333  223766.666667
         Pennsylvania Philadelphia 116933.333333  115866.666667  116200.000000
         Arizona      Phoenix      193766.666667  183333.333333  177566.666667

         yearQ                            2009q2         2009q3         2009q4
         State        RegionName
         New York     New York     465833.333333  455933.333333  458366.666667
         California   Los Angeles  413900.000000  406366.666667  404333.333333
         Illinois     Chicago      219700.000000  214100.000000  211666.666667
         Pennsylvania Philadelphia 116166.666667  116733.333333  118566.666667
         Arizona      Phoenix      168233.333333  155933.333333  143466.666667

In [30]: def run_ttest():
             '''First creates new data showing the decline or growth of housing pri
             between the recession start and the recession bottom. Then runs a ttes
             comparing the university town values to the non-university towns value
             return whether the alternative hypothesis (that the two groups are the
             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-
```

```
    True at a p<0.01 (we reject the null hypothesis), or different=False i
    otherwise (we cannot reject the null hypothesis). The variable p shoul
    be equal to the exact p value returned from scipy.stats.ttest_ind(). T
    value for better should be either "university town" or "non-university
    depending on which has a lower mean price ratio (which is equivilent t
    reduced market loss).'''

    recStart = get_recession_start()
    recBottom = get_recession_bottom()
    houseData = convert_housing_data_to_quarters()

    houseData['quarter_before_recession'] = houseData.ix[:, houseData.colu

    houseData['recession_bottom'] = houseData.ix[:, recBottom]
    houseData['price_ratio'] = houseData['quarter_before_recession'].div(h

    uniTowns = get_list_of_university_towns()


    UniTownsHouse = pd.merge(houseData, uniTowns, left_index=True, right_i
    houseData = houseData.reset_index()

    houseData['key'] = houseData['State'] + houseData['RegionName'].astype
    uniTowns['key'] = uniTowns['State'] + uniTowns['RegionName'].astype(st
    NonUniTownHouse = houseData[~houseData.key.isin(uniTowns.key)]
    NonUniTownHouse = NonUniTownHouse.set_index(['State', 'RegionName']).i

    #print ("non university town shape:", NonUniTownHouse.shape)
    maskUT = UniTownsHouse['price_ratio'][np.isfinite(UniTownsHouse['price
    maskNUT = NonUniTownHouse['price_ratio'][np.isfinite(NonUniTownHouse['
    t_result = ttest_ind(maskUT, maskNUT)
    meanUT = maskUT.mean()
    meanNUT = maskNUT.mean()
    #print ('uni price:', UniTownsHouse['price_ratio'].mean(), 'NonUni pri

    return (True if t_result[1]<0.01 else False , t_result[1], 'university
    #return NonUniTownHouse

run_ttest()
```

```
Out[30]: (True, 0.0027240637047614541, 'university town')
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```