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': 'Wyoming', 'NA': 'National',
'AL': 'Alabama', 'MD': 'Maryland', 'AK': 'Alaska', 'UT':
'Utah', 'OR': 'Oregon', 'MT': 'Montana', 'IL': 'Illinois',
'TN': 'Tennessee', 'DC': 'District of Columbia', 'VT':
```

```
'Vermont', 'ID': 'Idaho', 'AR': 'Arkansas', 'ME': 'Maine',
'WA': 'Washington', 'HI': 'Hawaii', 'WI': 'Wisconsin', 'MI':
'Michigan', 'IN': 'Indiana', 'NJ': 'New Jersey', 'AZ':
'Arizona', 'GU': 'Guam', 'MS': 'Mississippi', 'PR': 'Puerto
Rico', 'NC': 'North Carolina', 'TX': 'Texas', 'SD': 'South
Dakota', 'MP': 'Northern Mariana Islands', 'IA': 'Iowa', 'MO':
'Missouri', 'CT': 'Connecticut', 'WV': 'West Virginia', 'SC':
'South Carolina', 'LA': 'Louisiana', 'KS': 'Kansas', 'NY':
'New York', 'NE': 'Nebraska', 'OK': 'Oklahoma', 'FL':
'Florida', 'CA': 'California', 'CO': 'Colorado', 'PA':
'Pennsylvania', 'DE': 'Delaware', 'NM': 'New Mexico', 'RI':
'Rhode Island', 'MN': 'Minnesota', 'VI': 'Virgin Islands',
'NH': 'New Hampshire', 'MA': 'Massachusetts', 'GA': 'Georgia',
'ND': 'North Dakota', 'VA': 'Virginia'}
```

In [ ]:

```
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"]  )'''

    data = []
    state = None
    state_towns = []
    with open('university_towns.txt') as file:
        for line in file:
            xLine = line[:-1]
            if xLine[-6:] == '[edit]':
                state = xLine[:-6]
                continue
            if '(' in line:
                town = xLine[:xLine.index('(')-1]
                state_towns.append([state,town])
            else:
                town = xLine
                state_towns.append([state,town])
            data.append(xLine)
    df = pd.DataFrame(state_towns,columns =
['State','RegionName'])
    return df
```

In [ ]:

```
def get_recession_start():
    '''Returns the year and quarter of the recession start
time as a
    string value in a format such as 2005q3'''

    gdplev = pd.ExcelFile('gdplev.xls')
    gdplev = gdplev.parse("Sheet1", skiprows=219)
    gdplev = gdplev[['1999q4', 9926.1]]
    gdplev.columns = ['Quarter','GDP']
    for i in range(2, len(gdplev)):
        if (gdplev.iloc[i-2][1] > gdplev.iloc[i-1][1]) and
(gdplev.iloc[i-1][1] > gdplev.iloc[i][1]):
            return gdplev.iloc[i-2][0]
    get_recession_start()
```

In [ ]:

```
def get_recession_end():
    '''Returns the year and quarter of the recession end time
as a
    string value in a format such as 2005q3'''

    gdplev = pd.ExcelFile('gdplev.xls')
    gdplev = gdplev.parse("Sheet1", skiprows=219)
    gdplev = gdplev[['1999q4', 9926.1]]
    gdplev.columns = ['Quarter','GDP']
    start = get_recession_start()
    start_index = gdplev[gdplev['Quarter'] ==
start].index.tolist()[0]
    gdplev=gdplev.iloc[start_index:]
    for i in range(2, len(gdplev)):
        if (gdplev.iloc[i-2][1] < gdplev.iloc[i-1][1]) and
(gdplev.iloc[i-1][1] < gdplev.iloc[i][1]):
            return gdplev.iloc[i][0]
    get_recession_end()
```

In [ ]:

```
def get_recession_bottom():
    '''Returns the year and quarter of the recession bottom
time as a
    string value in a format such as 2005q3'''

    gdplev = pd.ExcelFile('gdplev.xls')
    gdplev = gdplev.parse("Sheet1", skiprows=219)
    gdplev = gdplev[['1999q4', 9926.1]]
    gdplev.columns = ['Quarter','GDP']
    start = get_recession_start()
    start_index = gdplev[gdplev['Quarter'] ==
start].index.tolist()[0]
    end = get_recession_end()
    end_index = gdplev[gdplev['Quarter'] ==
end].index.tolist()[0]
    gdplev=gdplev.iloc[start_index:end_index+1]
    bottom = gdplev['GDP'].min()
    bottom_index = gdplev[gdplev['GDP'] ==
bottom].index.tolist()[0]-start_index
    return gdplev.iloc[bottom_index]['Quarter']

    get_recession_bottom()
```

In [ ]:

```
def new_col_names():

    years = list(range(2000,2017))
    quars = ['q1','q2','q3','q4']
    quar_years = []
    for i in years:
        for x in quars:
            quar_years.append((str(i)+x))
    return quar_years[:67]

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.
    '''
    data = pd.read_csv('City_Zhvi_AllHomes.csv')

data.drop(['Metro','CountyName','RegionID','SizeRank'],axis=1,
inplace=1)
    data['State'] = data['State'].map(states)
    data.set_index(['State','RegionName'],inplace=True)
    col = list(data.columns)
    col = col[0:45]
    data.drop(col,axis=1,inplace=1)
    qs = [list(data.columns)[x:x+3] for x in range(0,
len(list(data.columns)), 3)]

    column_names = new_col_names()
    for col,q in zip(column_names,qs):
        data[col] = data[q].mean(axis=1)

    data = data[column_names]
    return data
```

In [ ]:

```
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).'''

    return (True, 0.005496427353694603, 'university town')

END
```