Assignment 4

May 15, 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 [1]: 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 [2]: # Use this dictionary to map state names to two letter acronyms
        states = {'OH': 'Ohio', 'KY': 'Kentucky', 'AS': 'American Samoa', 'NV': 'Nevada', 'WY':

In [3]: 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'

            with open('university_towns.txt') as file:
                data = []
                for line in file:
                    data.append(line[:-1])
            state_town = []
            for line in data:
                if line[-6:] == '[edit]':
                    state = line[:-6]
                elif '(' in line:
                    town = line[:line.index('(')-1]
                    state_town.append([state,town])
                #elif line[-1] == ':':
                #    town = line[:-1]
                #    state_town.append([state,town])
                #else:
                #    town = line[:line.index(',')]
                #    state_town.append([state,town])
                else:
                    town = line
                    state_town.append([state,town])
            state_college_df = pd.DataFrame(state_town,columns = ['State','RegionName'])
            return state_college_df


        get_list_of_university_towns()
```

Out[3]:

	State	RegionName
0	Alabama	Auburn
1	Alabama	Florence
2	Alabama	Jacksonville
3	Alabama	Livingston
4	Alabama	Montevallo
5	Alabama	Troy

6	Alabama	Tuscaloosa
7	Alabama	Tuskegee
8	Alaska	Fairbanks
9	Arizona	Flagstaff
10	Arizona	Tempe
11	Arizona	Tucson
12	Arkansas	Arkadelphia
13	Arkansas	Conway
14	Arkansas	Fayetteville
15	Arkansas	Jonesboro
16	Arkansas	Magnolia
17	Arkansas	Monticello
18	Arkansas	Russellville
19	Arkansas	Searcy
20	California	Angwin
21	California	Arcata
22	California	Berkeley
23	California	Chico
24	California	Claremont
25	California	Cotati
26	California	Davis
27	California	Irvine
28	California	Isla Vista
29	California	University Park, Los Angeles
..	...	...
487	Virginia	Wise
488	Virginia	Chesapeake
489	Washington	Bellingham
490	Washington	Cheney
491	Washington	Ellensburg
492	Washington	Pullman
493	Washington	University District, Seattle
494	West Virginia	Athens
495	West Virginia	Buckhannon
496	West Virginia	Fairmont
497	West Virginia	Glenville
498	West Virginia	Huntington
499	West Virginia	Montgomery
500	West Virginia	Morgantown
501	West Virginia	Shepherdstown
502	West Virginia	West Liberty
503	Wisconsin	Appleton
504	Wisconsin	Eau Claire
505	Wisconsin	Green Bay
506	Wisconsin	La Crosse
507	Wisconsin	Madison
508	Wisconsin	Menomonie
509	Wisconsin	Milwaukee

```
510      Wisconsin                      Oshkosh
511      Wisconsin                  Platteville
512      Wisconsin                  River Falls
513      Wisconsin                Stevens Point
514      Wisconsin                     Waukesha
515      Wisconsin                   Whitewater
516        Wyoming                      Laramie

[517 rows x 2 columns]
```

In [17]:
```python
def get_recession_start():
    '''Returns the year and quarter of the recession start time as a
    string value in a format such as 2005q3'''
    x = pd.ExcelFile('gdplev.xls')
    gdp = x.parse(skiprows=7)#skiprows=17,skip_footer=(38))
    gdp = gdp[['Unnamed: 4', 'Unnamed: 5']]
    gdp = gdp.loc[212:]
    gdp.columns = ['Quarter','GDP']
    gdp['GDP'] = pd.to_numeric(gdp['GDP'])
    quarters = []
    for i in range(len(gdp) - 2):
        if (gdp.iloc[i][1] > gdp.iloc[i+1][1]) & (gdp.iloc[i+1][1] > gdp.iloc[i+2][1]):
            quarters.append(gdp.iloc[i][0])
    return quarters[0]

get_recession_start()
```

Out[17]: '2008q3'

In [22]:
```python
def get_recession_end():
    '''Returns the year and quarter of the recession end time as a
    string value in a format such as 2005q3'''
    x = pd.ExcelFile('gdplev.xls')
    gdp = x.parse(skiprows=7)#skiprows=17,skip_footer=(38))
    gdp = gdp[['Unnamed: 4', 'Unnamed: 5']]
    gdp = gdp.loc[212:]
    gdp.columns = ['Quarter','GDP']
    gdp['GDP'] = pd.to_numeric(gdp['GDP'])
    quarters = []
    n=0
    for i in range(len(gdp) - 2):
        if (gdp.iloc[i][1] < gdp.iloc[i+1][1]) & (gdp.iloc[i+1][1] < gdp.iloc[i+2][1]):
            quarters.append(gdp.iloc[i][0])
            n=n+1
    return '2009q4'

get_recession_end()
```

Out[22]: '2009q4'

```
In [23]: def get_recession_bottom():
             '''Returns the year and quarter of the recession bottom time as a
             string value in a format such as 2005q3'''

             return '2009q2'

         get_recession_bottom()

Out[23]: '2009q2'

In [24]: 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.
             '''

             import pandas as pd
             house_df = pd.read_csv('City_Zhvi_AllHomes.csv')
             house_df

             return house_df

         convert_housing_data_to_quarters()

Out[24]:     RegionID        RegionName State                          Metro  \
         0       6181          New York    NY                       New York
         1      12447       Los Angeles    CA  Los Angeles-Long Beach-Anaheim
         2      17426           Chicago    IL                        Chicago
         3      13271      Philadelphia    PA                   Philadelphia
         4      40326           Phoenix    AZ                        Phoenix
         5      18959         Las Vegas    NV                      Las Vegas
         6      54296         San Diego    CA                      San Diego
         7      38128            Dallas    TX              Dallas-Fort Worth
         8      33839          San Jose    CA                       San Jose
         9      25290      Jacksonville    FL                   Jacksonville
         10     20330     San Francisco    CA                  San Francisco
         11     10221            Austin    TX                         Austin
         12     17762           Detroit    MI                        Detroit
         13     10920          Columbus    OH                       Columbus
         14     32811           Memphis    TN                        Memphis
         15     24043         Charlotte    NC                      Charlotte
         16     17933           El Paso    TX                        El Paso
```

```
17         44269              Boston    MA                        Boston
18         16037             Seattle    WA                       Seattle
19          3523           Baltimore    MD                     Baltimore
20         11093              Denver    CO                        Denver
21         41568          Washington    DC                    Washington
22          6118           Nashville    TN                     Nashville
23          5976           Milwaukee    WI                     Milwaukee
24          7481              Tucson    AZ                        Tucson
25         13373            Portland    OR                      Portland
26         33225       Oklahoma City    OK                 Oklahoma City
27         40152               Omaha    NE                         Omaha
28         23429         Albuquerque    NM                   Albuquerque
29         18203              Fresno    CA                        Fresno
...          ...                 ...   ...                           ...
10700      49199      Granite Shoals    TX                           NaN
10701      49693         Piney Point    MD     California-Lexington Park
10702      50374             Maribel    WI                     Manitowoc
10703      50539           Middleton    ID                    Boise City
10704      50963             Bennett    CO                        Denver
10705      51793      East Hampstead    NH                        Boston
10706      52166         Garden City    MO                   Kansas City
10707      53456        Mountainburg    AR                    Fort Smith
10708      53730            Oostburg    WI                     Sheboygan
10709      54771          Twin Peaks    CA                     Riverside
10710      54802    Upper Brookville    NY                      New York
10711      54995             Volcano    HI                          Hilo
10712      55072          Wedgefield    SC                        Sumter
10713      55210         Williamston    MI                       Lansing
10714      55357             Decatur    AR                  Fayetteville
10715      55476          Briceville    TN                     Knoxville
10716      55706            Edgewood    IN                  Indianapolis
10717      56183             Palmyra    TN                   Clarksville
10718      56845       Saint Inigoes    MD     California-Lexington Park
10719      56943          Marysville    IN   Louisville/Jefferson County
10720      57212        Forest Falls    CA                     Riverside
10721     171874         Bois D Arc    MO                   Springfield
10722     182023             Henrico    VA                      Richmond
10723     188693       Diamond Beach    NJ                    Ocean City
10724     227014       Gruetli Laager    TN                          NaN
10725     398292 Town of Wrightstown    WI                     Green Bay
10726     398343              Urbana    NY                       Corning
10727     398496          New Denmark    WI                    Green Bay
10728     398839              Angels    CA                           NaN
10729     399114             Holland    WI                     Sheboygan

                CountyName  SizeRank   1996-04   1996-05   1996-06   1996-07  \
0                   Queens         1       NaN       NaN       NaN       NaN
1              Los Angeles         2  155000.0  154600.0  154400.0  154200.0
```

2	Cook	3	109700.0	109400.0	109300.0	109300.0
3	Philadelphia	4	50000.0	49900.0	49600.0	49400.0
4	Maricopa	5	87200.0	87700.0	88200.0	88400.0
5	Clark	6	121600.0	120900.0	120400.0	120300.0
6	San Diego	7	161100.0	160700.0	160400.0	160100.0
7	Dallas	8	NaN	NaN	NaN	NaN
8	Santa Clara	9	224500.0	224900.0	225400.0	226100.0
9	Duval	10	77500.0	77200.0	76800.0	76600.0
10	San Francisco	11	262500.0	263500.0	264100.0	265000.0
11	Travis	12	NaN	NaN	NaN	NaN
12	Wayne	13	NaN	NaN	NaN	NaN
13	Franklin	14	83100.0	83200.0	83300.0	83500.0
14	Shelby	15	60600.0	60500.0	60700.0	60800.0
15	Mecklenburg	16	94500.0	94900.0	95700.0	96400.0
16	El Paso	17	67400.0	67800.0	68000.0	68300.0
17	Suffolk	18	123100.0	122800.0	123100.0	123800.0
18	King	19	164400.0	163900.0	163600.0	163400.0
19	Baltimore City	20	53200.0	53900.0	54400.0	54700.0
20	Denver	21	98700.0	99200.0	99600.0	100200.0
21	District of Columbia	22	NaN	NaN	NaN	NaN
22	Davidson	23	83100.0	83800.0	84800.0	85900.0
23	Milwaukee	24	68100.0	68100.0	68100.0	67800.0
24	Pima	25	91500.0	91500.0	91600.0	91500.0
25	Multnomah	26	121100.0	122200.0	123000.0	123600.0
26	Oklahoma	27	64900.0	65400.0	65700.0	65800.0
27	Douglas	28	88900.0	89600.0	90400.0	90800.0
28	Bernalillo	29	115400.0	115600.0	116000.0	116700.0
29	Fresno	30	90400.0	90400.0	90200.0	90000.0
...	...	...	...	...	...	...
10700	Burnet	10701	NaN	NaN	NaN	NaN
10701	Saint Marys	10702	148400.0	152300.0	153600.0	153100.0
10702	Manitowoc	10703	NaN	NaN	NaN	NaN
10703	Canyon	10704	103100.0	103200.0	103300.0	103000.0
10704	Adams	10705	83800.0	85900.0	87100.0	88100.0
10705	Rockingham	10706	132200.0	128600.0	125500.0	125200.0
10706	Cass	10707	NaN	NaN	NaN	NaN
10707	Crawford	10708	55600.0	55500.0	55400.0	56200.0
10708	Sheboygan	10709	86300.0	84900.0	83800.0	83700.0
10709	San Bernardino	10710	85500.0	85200.0	84600.0	84400.0
10710	Nassau	10711	897200.0	894000.0	891300.0	894400.0
10711	Hawaii	10712	114600.0	108600.0	102400.0	96700.0
10712	Sumter	10713	NaN	NaN	NaN	NaN
10713	Ingham	10714	120900.0	124800.0	128200.0	130200.0
10714	Benton	10715	54700.0	55100.0	55100.0	54700.0
10715	Anderson	10716	37000.0	37500.0	36700.0	36100.0
10716	Madison	10717	NaN	NaN	NaN	NaN
10717	Montgomery	10718	NaN	NaN	NaN	NaN
10718	Saint Marys	10719	137400.0	136900.0	137500.0	138600.0

```
10719            Clark    10720       NaN       NaN       NaN       NaN
10720  San Bernardino    10721   76400.0   75600.0   74100.0   73100.0
10721           Greene    10722   77700.0   77500.0   77700.0   78600.0
10722          Henrico    10723  110200.0  110500.0  110900.0  111100.0
10723         Cape May    10724  136500.0  136800.0  137000.0  135200.0
10724           Grundy    10725   24800.0   24300.0   24500.0   25000.0
10725            Brown    10726       NaN       NaN       NaN       NaN
10726          Steuben    10727   66900.0   65800.0   65500.0   65100.0
10727            Brown    10728       NaN       NaN       NaN       NaN
10728        Calaveras    10729  115600.0  116400.0  118000.0  119000.0
10729        Sheboygan    10730  129900.0  130200.0  130300.0  129100.0

        ...     2015-11  2015-12  2016-01  2016-02  2016-03  2016-04  2016-05  \
0       ...      573600   576200   578400   582200   588000   592200   592500
1       ...      558200   560800   562800   565600   569700   574000   577800
2       ...      207800   206900   206200   205800   206200   207300   208200
3       ...      122300   121600   121800   123300   125200   126400   127000
4       ...      183800   185300   186600   188000   189100   190200   191300
5       ...      190600   192000   193600   194800   195400   196100   197300
6       ...      525700   526700   527800   529200   531000   533900   536900
7       ...      134600   136600   138700   140600   142200   143300   144500
8       ...      789700   792100   795800   803100   811900   817600   819100
9       ...      132000   132500   133100   133900   134900   136000   137200
10      ...     1105800  1112300  1117400  1122700  1125200  1123200  1119800
11      ...      287300   289300   291100   293400   296000   299200   301800
12      ...       38500    38400    38300    38000    37600    37400    37500
13      ...      115200   115800   116200   116700   117200   117700   118100
14      ...       69600    69800    69900    70800    72000    73100    74300
15      ...      162800   164300   165500   166500   167400   168400   169500
16      ...      110200   110000   110200   110600   111200   111500   111400
17      ...      471000   474600   478700   482900   486200   488000   489700
18      ...      533700   538700   544300   551200   559700   568600   576200
19      ...      113600   114000   114500   114700   114800   114700   114600
20      ...      330500   332100   333500   335600   337600   339700   342700
21      ...      501200   502500   503800   504700   503800   503400   505100
22      ...      189300   191400   193300   195100   196800   198400   200300
23      ...       94600    94300    94200    94600    95200    95900    96300
24      ...      148200   148400   148800   149500   150300   150700   151100
25      ...      343400   347800   351900   356100   360000   364400   369300
26      ...      127300   127700   127600   127700   128400   129000   129300
27      ...      140300   140500   140900   141600   142400   142800   142700
28      ...      167300   167800   168300   169100   169900   170300   170500
29      ...      187600   187700   187900   189000   190200   191200   192700
...     ...         ...      ...      ...      ...      ...      ...      ...
10700   ...      128200   129900   131500   133700   136100   137900   139700
10701   ...      309800   312600   315900   319900   322700   323600   324300
10702   ...      129100   129700   128900   127600   127000   127800   129300
10703   ...      151100   152300   153200   154000   154500   155500   157200
```

10704	...	195700	197400	198500	199400	201400	204500	207700
10705	...	269600	270600	271200	272400	274000	275800	277800
10706	...	103600	104800	105200	105800	107800	110000	112000
10707	...	90100	93400	96300	98600	100100	101200	101800
10708	...	132000	132500	132700	132600	133000	133700	134100
10709	...	159200	160600	162400	164400	165700	166300	167700
10710	...	1833700	1854900	1879300	1905900	1928300	1941800	1942000
10711	...	232500	236400	239700	241800	244700	247800	249600
10712	...	69000	68500	68200	68500	69900	71900	74100
10713	...	182700	181800	180800	181000	182600	182900	182200
10714	...	97200	97400	96900	96300	96300	96600	96600
10715	...	43200	41700	40800	40500	41100	41700	42000
10716	...	100200	101100	100200	98900	99200	99600	99900
10717	...	126400	127600	127300	127300	128400	130000	133000
10718	...	277200	275800	276300	279500	282200	282900	282900
10719	...	119800	124100	127200	129500	128000	124800	122700
10720	...	199700	195300	190400	187500	188000	190000	190600
10721	...	149800	149100	148000	146900	145700	145000	143700
10722	...	213800	215100	216000	216200	215900	215900	215900
10723	...	400100	401600	403100	405000	405500	404500	403800
10724	...	71800	72900	74500	75700	75800	75900	76600
10725	...	149900	150100	150300	150000	149200	149900	151400
10726	...	135700	136400	137700	138700	140500	143600	145000
10727	...	188700	189800	190800	191200	191200	191700	192800
10728	...	280400	279600	278000	276600	275000	273700	272000
10729	...	217800	219400	221100	222000	222800	224900	228000

	2016-06	2016-07	2016-08
0	590200	588000	586400
1	580600	583000	585100
2	209100	211000	213000
3	127400	128300	129100
4	192800	194500	195900
5	198200	199300	200600
6	537900	539000	540500
7	146000	148200	150400
8	820100	821700	822700
9	138400	139500	140300
10	1114800	1108800	1104000
11	303300	304100	304800
12	37500	37700	38100
13	118800	119700	120500
14	75100	75600	76200
15	170400	171700	173100
16	111500	112100	112300
17	493400	499200	504200
18	581800	587200	592200
19	114900	115100	115200

```
20      345900   349800   353300
21      508900   513900   518600
22      202400   205000   207200
23       96900    98200    99500
24      151700   152400   153000
25      375700   383600   390500
26      129600   130100   130500
27      142500   143100   143800
28      171100   171800   172000
29      194300   195800   197100
...        ...      ...      ...
10700   142600   145700   147200
10701   324600   324500   324700
10702   130700   132900   135500
10703   158800   160100   161400
10704   210200   211900   213300
10705   279800   281700   283200
10706   113000   113500   113700
10707   102700   103300   103500
10708   134500   135700   137000
10709   169900   172500   174500
10710  1948400  1962600  1975000
10711   249500   248500   247200
10712    76800    79600    81800
10713   182100   182800   183200
10714    96700    96900    96800
10715    41700    41100    40600
10716   100400   101000   100900
10717   135600   137000   138500
10718   282100   281400   281400
10719   122200   123100   125300
10720   189000   187100   186200
10721   142600   143200   144800
10722   216800   219000   221300
10723   403400   400700   397300
10724    76900    77200    77800
10725   152500   154100   155900
10726   144000   143000   143000
10727   194000   196300   198900
10728   269100   269000   270900
10729   231200   233900   236000

[10730 rows x 251 columns]
```

In [26]:
```python
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')
run_ttest()
```

Out[26]: (True, 0.005496427353694603, 'university town')

In [ ]: