Notebook

April 29, 2025

1 Problem statement:

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.

The goal of this project: First of all generate a structured dataset from the unstructured dataset `university_towns.txt`. Secondly convert data from the file `City_Zhvi_AllHomes.csv` which is currently displayed in a monthly basis to a quarterly basis. The transformed dataframe should be a dataframe with columns going from 2000q1 to 2016q3, and should have a multi-index in the shape of [“State”,“RegionName”]. And finally I will run a t test to check whether the hypothesis according to which University towns have their mean housing prices less effected by recessions is true.

[39]:
```
import pandas as pd
import numpy as np
from scipy.stats import ttest_ind
```

[40]:
```
# 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'}
```

[41]:
```
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'. '''



    # Open the file with read only permit
    f = open('university_towns.txt')
    # use readline() to read the first line
    line = f.readline()
    # use the read line to read further.
    # If the file is not empty keep reading one line at a time, till the file␣
 ↪is empty

    states=[]
    regions=[]
    state=''
    region=''
    while line:
        line=line.replace('\n','')
```

```
        if (len(line.split('[ed']))>1):

             state=line.split('[ed')[0]

        else:

            region=line.split(' (')[0]
            regions.append(region)
            states.append(state)

        line = f.readline()
    f.close()
    s=pd.Series(states)
    r=pd.Series(regions)

    df=pd.DataFrame(np.column_stack([s,r]),columns=["State", "RegionName"])

    return df
```

[42]:
```
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.
    '''
    df=pd.read_csv('City_Zhvi_AllHomes.csv')
    fin_date=len(df.columns)
    df1=df[df.columns[51:fin_date]]

    res = (df1.groupby(pd.PeriodIndex(df1.columns, freq='Q'), axis=1)
                  .mean()
                  .rename(columns=lambda c: str(c).lower()))
    res[["State1","RegionName"]]=df[["State","RegionName"]]

    #print(df.columns[51:len(df.columns)]
```

```
    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'}

    res["State"]=res['State1'].map(states)
    res=res.set_index(["State","RegionName"])
    res=res.drop('State1',axis=1)

    return res
```

[43]:
```
get_list_of_university_towns().head()
```

[43]:
```
      State    RegionName
0  Alabama        Auburn
1  Alabama      Florence
2  Alabama  Jacksonville
3  Alabama    Livingston
4  Alabama    Montevallo
```

[44]:
```
convert_housing_data_to_quarters().head()
```

[44]:
```
                                  2000q1         2000q2         2000q3  \
State        RegionName
New York     New York                NaN            NaN            NaN
California   Los Angeles    207066.666667  214466.666667  220966.666667
Illinois     Chicago        138400.000000  143633.333333  147866.666667
Pennsylvania Philadelphia    53000.000000   53633.333333   54133.333333
Arizona      Phoenix        111833.333333  114366.666667  116000.000000

                                  2000q4         2001q1         2001q2  \
State        RegionName
New York     New York                NaN            NaN            NaN
California   Los Angeles    226166.666667  233000.000000  239100.000000
Illinois     Chicago        152133.333333  156933.333333  161800.000000
```

```
Pennsylvania Philadelphia   54700.000000   55333.333333   55533.333333
Arizona      Phoenix       117400.000000  119600.000000  121566.666667

                                 2001q3         2001q4         2002q1  \
State        RegionName
New York     New York               NaN            NaN            NaN
California   Los Angeles   245066.666667  253033.333333  261966.666667
Illinois     Chicago       166400.000000  170433.333333  175500.000000
Pennsylvania Philadelphia   56266.666667   57533.333333   59133.333333
Arizona      Phoenix       122700.000000  124300.000000  126533.333333

                                 2002q2  …         2014q2         2014q3  \
State        RegionName                 …
New York     New York               NaN  …  515466.666667  522800.000000
California   Los Angeles   272700.000000  …  498033.333333  509066.666667
Illinois     Chicago       177566.666667  …  192633.333333  195766.666667
Pennsylvania Philadelphia   60733.333333  …  113733.333333  115300.000000
Arizona      Phoenix       128366.666667  …  164266.666667  165366.666667

                                 2014q4         2015q1         2015q2  \
State        RegionName
New York     New York      528066.666667  532266.666667  540800.000000
California   Los Angeles   518866.666667  528800.000000  538166.666667
Illinois     Chicago       201266.666667  201066.666667  206033.333333
Pennsylvania Philadelphia  115666.666667  116200.000000  117966.666667
Arizona      Phoenix       168500.000000  171533.333333  174166.666667

                                 2015q3         2015q4         2016q1  \
State        RegionName
New York     New York      557200.000000  572833.333333  582866.666667
California   Los Angeles   547266.666667  557733.333333  566033.333333
Illinois     Chicago       208300.000000  207900.000000  206066.666667
Pennsylvania Philadelphia  121233.333333  122200.000000  123433.333333
Arizona      Phoenix       179066.666667  183833.333333  187900.000000

                                 2016q2    2016q3
State        RegionName
New York     New York      591633.333333  587200.0
California   Los Angeles   577466.666667  584050.0
Illinois     Chicago       208200.000000  212000.0
Pennsylvania Philadelphia  126933.333333  128700.0
Arizona      Phoenix       191433.333333  195200.0

[5 rows x 67 columns]
```

[45]:
```
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',header=None,names=['quarter','gdp'],usecols=[4,6],skiprows=220)
    #print(gdp.head())
    gdp=gdp.set_index('quarter')

    #diff = gdp['gdp'].diff()
    #print(diff.head())
    gdp['gdp_change']=gdp['gdp'].diff()
    negativ=gdp[gdp['gdp_change']<0]
    negativ=negativ.reset_index()
    #print( negativ)
    taille=negativ.shape[0]
    index=1
    while(index<=taille):
        index1=int(negativ['quarter'][index].split('q')[1])
        index2=int(negativ['quarter'][index-1].split('q')[1])
        if((index1-index2)==1):
            recession=negativ['quarter'][index2]
            index=taille+1
        else:
            index+=1
    #for index in gdp.shape[0]:
    #print(recession)



    return recession
get_recession_start()
```

[45]:
```
'2008q3'
```

[46]:
```
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',header=None,names=['quarter','gdp'],usecols=[4,6],skiprows=220)
    #print(gdp.head())

    gdp['gdp_change']=gdp['gdp'].diff()
    start_recession=get_recession_start()
    #print(start_recession)
    taille= gdp['gdp_change'].shape[0]

    for row in gdp.itertuples():
        if (row[1]== start_recession):
```

```
            index=row[0]

    while (index<taille):

        if((gdp['gdp_change'][index]>0) & (gdp['gdp_change'][index+1]>0) ):
            end_recession=gdp['quarter'][index+1]
            index=taille+1
        else:
            index+=1


    return end_recession
    #return "answer"
#get_recession_end()
```

[47]:
```
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',header=None,names=['quarter','gdp'],usecols=[4,6],skiprows=220)
    start=get_recession_start()
    end=get_recession_end()
    gdp=gdp.set_index('quarter')
    index=list(gdp.index.values)
    #print(index)
    index_start=index.index(start)

    index_end=index.index(end)
    #print(index_start)
    #print(index_end)
    #print(gdp['gdp'][index_start:index_end])
    bottom=gdp['gdp'][index_start:index_end].argmin()
    recession_bottom=list(gdp['gdp'][index_start:index_end].index.values)[␣
 ↪bottom]

    return recession_bottom

get_recession_bottom()
```

[47]:
```
'2009q2'
```

[48]:
```
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).'''

   #gdp=pd.read_excel('gdplev.
↪xls',header=None,names=['quarter','gdp'],usecols=[4,6],skiprows=220)
    start=get_recession_start()
    bottom=get_recession_bottom()
    print(bottom)
    print(start)
    data=convert_housing_data_to_quarters()
    columns=data.columns.tolist()
    index_start=columns.index(start)-1
    print(index_start)
    quarter=columns[index_start]
    index_end=columns.index(bottom)+1
    print(index_end)
    plage=columns[index_start:index_end]
    uni=get_list_of_university_towns()
    uni_town=list(uni['RegionName'])
    print(len(uni_town))
    #print(uni_town)
    df=data.copy()
    df=data.iloc[:,index_start:index_end]
    df=df.reset_index()
    df=df.set_index('RegionName')
    df['ratio']=df[quarter]/df[bottom]
    #data=data[plage]
    uni_data=df.loc[df.index.isin(uni_town)]
    non_uni_data=df.loc[~df.index.isin(uni_town)]

    print(df.shape[0])
    print(uni_data.shape[0])
    print(non_uni_data.shape[0])
    print(uni_data.head())
    t,p=ttest_ind(np.array(uni_data['ratio']),np.
↪array(non_uni_data['ratio']),equal_var = False,nan_policy='omit')
    if (p<0.01):
        different=True
    else:
```

```
        different=False
    #print(test)
    if (uni_data['ratio'].mean()< non_uni_data['ratio'].mean()):
        better="university town"
    else:
        better="non-university town"
    '''for row in data.itertuples():
        try:
            ind=uni_town.index(row[2])
            data['town'][row[0]]='university town'
            break

        except:
            data['town'][row[0]]='non-university town'

    '''
    #print(columns)

    #gdp=gdp.set_index('quarter')
    #index=list(gdp.index.values)
    #print(index)

    res=(different,p,better)
    #print(data.head())
    return res

run_ttest()
```

```
2009q2
2008q3
33
38
517
10730
883
9847
                   State         2008q2         2008q3         2008q4  \
RegionName
Las Vegas         Nevada  232300.000000  213366.666667  194933.333333
San Diego     California  441400.000000  424666.666667  407633.333333
Dallas             Texas  115366.666667  112166.666667  109900.000000
Jacksonville     Florida  160433.333333  154733.333333  149166.666667
Austin             Texas  213733.333333  211033.333333  207466.666667

                    2009q1         2009q2     ratio
RegionName
Las Vegas     181200.000000  164333.333333  1.413590
San Diego     395700.000000  389500.000000  1.133248
```

```
Dallas         107666.666667  105100.000000  1.097685
Jacksonville   145266.666667  140833.333333  1.139172
Austin         207000.000000  204000.000000  1.047712
```

[48]:
```
(True, 1.0258636101150937e-05, 'university town')
```

2 CONCLUSION:

I have cleaned the data and prepared in such a way to be exploitable. At the end I run the ttest and the conclusion is: It is true to say University towns have their mean housing prices less effected by recessions.

[ ]:

[ ]:

This notebook was converted with convert.ploomber.io