Giorgio Mendoza

CS539-F23-F02

Dr. J. Sethi

Lab: Week 8: Hypothesis Testing

Lab_3-3: 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 affected 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, there is a list of university towns in the United States which has been copied and pasted into the file `university_towns.txt`.
- From the 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 lab, only look at GDP data from the first quarter of 2000 onward.

```
import pandas as pd
import numpy as np
from scipy.stats import ttest_ind
```

Each function in this assignment below is worth 10%, with the exception of `run_ttest()`, which is worth 50%.

```
# 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': 'I
```

Question 0

Let's get the list of university towns first:

```
import pandas as pd
from google.colab import drive
drive.mount('/content/drive')

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", "Ypsilanti"]], 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'.
    '''

    university = []

    file_path = '/content/drive/MyDrive/Colab Notebooks/university_towns.txt'

    with open(file_path, 'r') as file:
        state = None
        region = None

        for line in file:
            index = line.find('[edit]')
            if index > 0:
                state = line[0:index].strip()  # Extract the state
            else:
                index = line.find('(')
                if index != -1:
                    region = line[0:index].strip()  # Extract the region
                else:
                    region = line.strip()

                if state is not None and region is not None:
                    new_row = {'State': state, 'RegionName': region}
```

```
                    university.append(new_row)

    university_df = pd.DataFrame(university, columns=["State", "RegionName"])
    return university_df

# Call the function to get the DataFrame
university_towns_df = get_list_of_university_towns()
print(university_towns_df)
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
         State     RegionName
0      Alabama         Auburn
1      Alabama       Florence
2      Alabama   Jacksonville
3      Alabama     Livingston
4      Alabama     Montevallo
..         ...            ...
512  Wisconsin    River Falls
513  Wisconsin  Stevens Point
514  Wisconsin       Waukesha
515  Wisconsin     Whitewater
516    Wyoming        Laramie

[517 rows x 2 columns]
```

Question 1

Lets' check the year and quarter of the recession start time next:

```
import pandas as pd
 #Returns the year and quarter of the recession start time as a string value in a format such as 2005q3
def get_recession_start():
    my_gdp = pd.read_excel('/content/drive/MyDrive/Colab Notebooks/gdplev.xls', skiprows=219)
    # get YearQuarter & GDP
    my_gdp = my_gdp.iloc[:, [4, 6]]
    my_gdp.columns = ['Quarter', 'GDP']
    for i in range(2, len(my_gdp)):
      # Check if the GDP in the current quarter is lower than the GDP in the previous quarter
      # and if the GDP in the previous quarter is lower than the GDP two quarters ago.
      # checks potential recession when both criteria are met.
        if (my_gdp.iloc[i - 2, 1] > my_gdp.iloc[i - 1, 1]) and (my_gdp.iloc[i - 1, 1] > my_gdp.iloc[i, 1]):
            startDate = my_gdp.iloc[i - 3, 0]
    return startDate

recession_start = get_recession_start()
print(recession_start)
```

```
2008q3
```

Question 2

Let's also get the year and quarter of the recession end time:

```
import pandas as pd
    #Returns the year and quarter of the recession end time as a string value in a format such as 2005q3
def get_recession_end():
    my_gdp = pd.read_excel('/content/drive/MyDrive/Colab Notebooks/gdplev.xls', skiprows=219)
    # get YearQuarter & GDP
    my_gdp = my_gdp.iloc[:, 4:6]
    my_gdp.columns = ['Quarter', 'GDP']
    recStart = get_recession_start()
    recStartIndex = my_gdp.index[my_gdp['Quarter'] == recStart].tolist()[0]
    for i in range(recStartIndex + 1, len(my_gdp)):
       # Check if GDP in current quarter is higher than GDP in previous quarter
       # and if GDP in previous quarter is higher than GDP two quarters ago.
        if (my_gdp.iloc[i - 2, 1] < my_gdp.iloc[i - 1, 1]) and (my_gdp.iloc[i - 1, 1] < my_gdp.iloc[i, 1]):
           # Return year and quarter of GDP data for current quarter.
         return my_gdp.iloc[i, 0]

recEnd = get_recession_end()
print(recEnd)
```

```
2009q4
```

Question 3

Then, let's get the year and quarter of the recession bottom time:

```
import pandas as pd
    #Returns the year and quarter of the recession bottom time as a string value in a format such as 2005q3
def get_recession_bottom():
    my_gdp = pd.read_excel('/content/drive/MyDrive/Colab Notebooks/gdplev.xls', skiprows=219)
    my_gdp = my_gdp.iloc[:, 4:6]
    my_gdp.columns = ['Quarter', 'GPD']

    recStart = get_recession_start()
    start_index = my_gdp[my_gdp['Quarter'] == recStart].index[0]
    recEnd = get_recession_end()
    end_index = my_gdp[my_gdp['Quarter'] == recEnd].index[0]
    #slice DataFrame to select rows from start_index to end_index (inclusive) & all columns.
    my_gdp = my_gdp.iloc[start_index:end_index + 1, :]
    #reset index of DataFrame, dropping old index and creating a new one.
```

```
    my_gdp = my_gdp.reset_index(drop=True)
    #get row in DataFrame where GDP (assuming it's a column) is at its minimum.
    #get YearQuarter value from that row to identify the bottom point of recession.
    bottom = my_gdp.loc[my_gdp['GPD'].idxmin(), 'Quarter']

    return bottom

recBottom = get_recession_bottom()
print(recBottom)
```

```
 2009q2
```

Question 4

And then we can convert the housing data to quarters (as defined above!) and return the mean values:

```
import pandas as pd
states = {'OH': 'Ohio', 'KY': 'Kentucky', 'AS': 'American Samoa', 'NV': 'Nevada', 'WY': 'Wyoming', 'NA': 'National', 'AL': 'Alabama', 'MD': 'M
def year_qtr(col):
    year_start = 0
    year_end = 4

    if col.endswith(("01", "02", "03")):
        return col[year_start:year_end] + "q1"
    elif col.endswith(("04", "05", "06")):
        return col[year_start:year_end] + "q2"
    elif col.endswith(("07", "08", "09")):
        return col[year_start:year_end] + "q3"
    else:
        return col[year_start:year_end] + "q4"

def convert_housing_data_to_quarters():
    houses_to_quarters = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/City_Zhvi_AllHomes.csv')
    #replace state abbreviations with full state names using 'states' dictionary
    houses_to_quarters['State'] = houses_to_quarters['State'].replace(states)
    #set index using 'State' & 'RegionName'
    houses_to_quarters.set_index(['State', 'RegionName'], inplace=True, drop=False)
    #keep columns from index 49 onwards
    houses_to_quarters = houses_to_quarters.iloc[:, 49:]
    houses_to_quarters = houses_to_quarters.groupby(year_qtr, axis=1).mean()

    #replace NaN values with 0
    #houses_to_quarters = houses_to_quarters.fillna(0)

    houses_to_quarters.sort_index()
    #pd.options.display.float_format = '{:.2f}'.format
    return houses_to_quarters

# Call the function to convert housing data to quarters with NaN values replaced by 0
result = convert_housing_data_to_quarters()
print(result)
```

```
                                    1999q4    2000q1    2000q2    2000q3  \
State        RegionName
New York     New York                  NaN       NaN       NaN       NaN
California   Los Angeles         201500.00 207066.67 214466.67 220966.67
Illinois     Chicago             135050.00 138400.00 143633.33 147866.67
Pennsylvania Philadelphia         52200.00  53000.00  53633.33  54133.33
Arizona      Phoenix             110050.00 111833.33 114366.67 116000.00
...                                    ...       ...       ...       ...
Wisconsin    Town of Wrightstown 103550.00 101766.67 105400.00 111366.67
New York     Urbana               77450.00  79200.00  81666.67  91700.00
Wisconsin    New Denmark         113900.00 114566.67 119266.67 126066.67
California   Angels              141850.00 151000.00 155900.00 158100.00
Wisconsin    Holland             149950.00 151033.33 150500.00 153233.33

                                    2000q4    2001q1    2001q2    2001q3  \
State        RegionName
New York     New York                  NaN       NaN       NaN       NaN
California   Los Angeles         226166.67 233000.00 239100.00 245066.67
Illinois     Chicago             152133.33 156933.33 161800.00 166400.00
Pennsylvania Philadelphia         54700.00  55333.33  55533.33  56266.67
Arizona      Phoenix             117400.00 119600.00 121566.67 122700.00
...                                    ...       ...       ...       ...
Wisconsin    Town of Wrightstown 114866.67 125966.67 129900.00 129900.00
New York     Urbana               98366.67  94866.67  98533.33 102966.67
Wisconsin    New Denmark         131966.67 143800.00 146966.67 148366.67
California   Angels              167466.67 176833.33 183766.67 190233.33
Wisconsin    Holland             155833.33 161866.67 165733.33 168033.33

                                    2001q4    2002q1  ...    2014q2    2014q3  \
State        RegionName                                ...
New York     New York                  NaN       NaN  ... 515466.67 522800.00
California   Los Angeles         253033.33 261966.67  ... 498033.33 509066.67
Illinois     Chicago             170433.33 175500.00  ... 192633.33 195766.67
Pennsylvania Philadelphia         57533.33  59133.33  ... 113733.33 115300.00
Arizona      Phoenix             124300.00 126533.33  ... 164266.67 165366.67
...                                    ...       ...  ...       ...       ...
Wisconsin    Town of Wrightstown 129433.33 131900.00  ... 144866.67 146866.67
New York     Urbana               98033.33  93966.67  ... 132133.33 137033.33
Wisconsin    New Denmark         149166.67 153133.33  ... 174566.67 181166.67
California   Angels              184566.67 184033.33  ... 244466.67 254066.67
Wisconsin    Holland             167400.00 165766.67  ... 201266.67 201566.67

                                    2014q4    2015q1    2015q2    2015q3  \
State        RegionName
New York     New York            528066.67 532266.67 540800.00 557200.00
California   Los Angeles         518866.67 528800.00 538166.67 547266.67
Illinois     Chicago             201266.67 201066.67 206033.33 208300.00
Pennsylvania Philadelphia        115666.67 116200.00 117966.67 121233.33
```

```
Arizona      Phoenix             168500.00 171533.33 174166.67 179066.67
...                                    ...       ...       ...       ...
Wisconsin    Town of Wrightstown 149233.33 148666.67 149333.33 149866.67
New York     Urbana              140066.67 141700.00 137866.67 136466.67
Wisconsin    New Denmark         186166.67 187600.00 188666.67 188433.33
California   Angels              259933.33 260100.00 250633.33 263500.00
Wisconsin    Holland             201266.67 206000.00 207600.00 212866.67

                                    2015q4    2016q1    2016q2    2016q3
State        RegionName
```

Note: What I noticed in this section 4, is that if I replaced the NaN values here with 0's I would get a different final result in the question 5. That's why I commented out #replace NaN values with 0

```
#houses_to_quarters = houses_to_quarters.fillna(0)
```

Question 5

Finally, let's run the actual t-test now:

```
import pandas as pd
import numpy as np
from scipy.stats import ttest_ind

'''First creates new data showing the decline or growth of housing prices between the recession start and the recession bottom.
Then runs a t-test comparing the university town values to the non-university towns values, return whether the alternative hypothesis (that th
Return the tuple (different, p, better) where different=True if the t-test is True at p < 0.01 (we reject the null hypothesis), or different=F
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 equ
'''

def run_ttest():
    # Retrieve the recession start and recession bottom dates
    recStart = get_recession_start()
    recBottom = get_recession_bottom()

    # Get the list of university towns
    university_towns = get_list_of_university_towns()

    # Get the housing price data and create a new column 'price_ratio' for price change between recStart and recBottom
    df = convert_housing_data_to_quarters()
    university_towns['University town'] = True

    df['price_ratio'] = df.loc[:, recStart] - df.loc[:, recBottom]

    # Merge, replace NaN, reset index, and select columns
    df = df.merge(university_towns, how='outer', left_index=True, right_on=['State', 'RegionName'])
    df['University town'].fillna(False, inplace=True)
    # Set index directly
    df.set_index(['State', 'RegionName'], inplace=True)
    # Select desired columns
    df = df[['price_ratio', 'University town']]
    # Separate data into university and non-university towns
    university = df.loc[df['University town'], 'price_ratio']
    non_university = df.loc[~df['University town'], 'price_ratio']

    # Perform a t-test to compare the price ratios of university and non-university towns
    t_test_result = ttest_ind(university, non_university, nan_policy='omit')

    # Access the p-value from the t-test result
    p = t_test_result.pvalue

    # Check if the t-test result is statistically significant
    different = p < 0.01

    # Determine which group has a lower mean price ratio
    if university.mean() < non_university.mean():
        better = "university town"
    else:
        better = "non-university town"

    return (t_test_result, p, better)

# Call the function to run the t-test
result = run_ttest()
print(result)
```

```
(TtestResult(statistic=-2.8540746960114087, pvalue=0.00432521485351121, df=9882.0), 0.00432521485351121, 'university town')
```