[Year]

Assignment 3

DATA, INFERENCE & APPLIED MACHINE LEARNING (COURSE 18-785)

YASSER CORZO

CARNEGIE MELLON UNIVERSITY

LIBRARIES USED

- Matplotlib
- Numpy
- Pandas
- Scipy
- Statsmodels
- Jupyter Notebook

REPORT

Q1

My null hypothesis is women's energy intake in 1991 is around the recommended value of 7725 kJ. My alternate hypothesis is that women's energy intake in 1991 deviates from the recommended amount. As a result, a two-tailed test would be appropriate since my alternative hypothesis states that the average energy intake from the data collected from 1991 would deviate from 7725 KJ, hence the actual average can be less than or greater than the recommended amount (μ != μ_0).

Below is the data requested from Q1:

```
sample mean: 6753.636363636364
sample standard deviation: 1142.1232221373727
SEM: 344.3631083801271
t-statistic = -2.8207540608310193
degrees of freedom: 10
p-value = 0.018137235176105812
```

Since the p-value < α, the null hypothesis is **rejected**.

How I computed this data was using numpy methods (np.std() [1], np.mean() [2], & np.sqrt() [3]) to calculate the sample mean, standard deviation, and SEM. Calculating the degrees of freedom involved taking the length of sample data and subtracting by 1. Calculating t statistic and p value involved using the stats.ttest_1samp method from scipy [4].

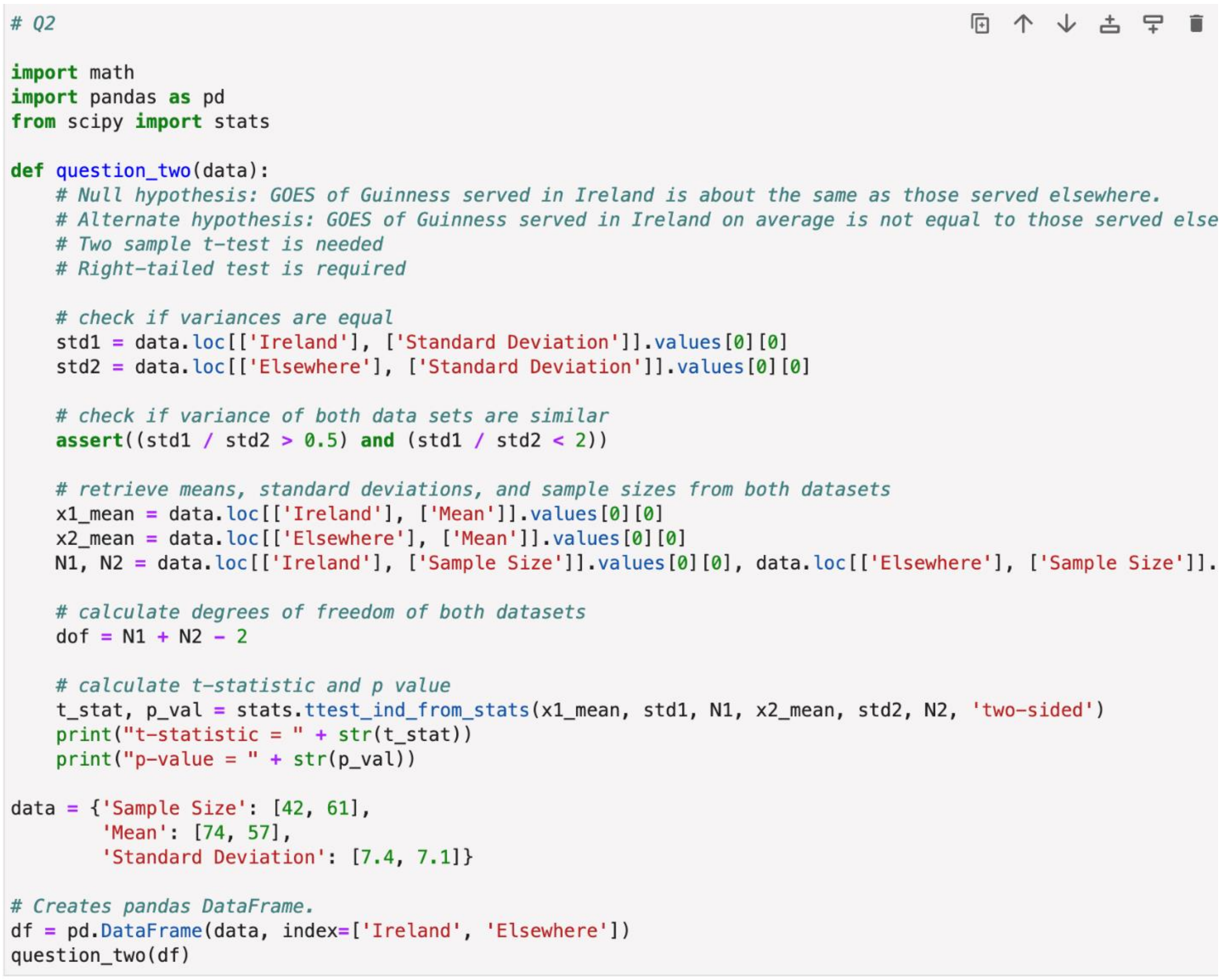
Q2

The difference in means is significant as taking the sample size into context of data regarding Ireland and Elsewhere is important. Ireland's higher mean GOES score can be a result of having a lower sample size than data from Elsewhere, as increasing the sample size can yield data with lower GOES scores, thus decreasing the mean GOES score of Ireland. Thus, using a t-test to compare GOES score data from Ireland and elsewhere would be beneficial.

Using a two-sample test is appropriate since two separate data sets are involved with equal variances [5] and we're trying to determine if the mean GOES scores from Ireland and elsewhere are comparable.

The null hypothesis for this task is the average GOES score of Guinness served in Ireland is about the same as the average GOES score of Guinness served elsewhere. The alternate hypothesis for this task is the average GOES of Guinness served is Ireland is not equal to the average GOES score of Guinness served elsewhere.

The steps involved in calculating the t statistic value were first checking if the variances of both datasets were equal (i.e. checking if the standard deviation ratio of data from Ireland and Elsewhere is between 0.5 and 2 [6]). The next step involved retrieving the mean, standard deviations, and sample sizes from both datasets (already presnt in the table). Then, using the stats.ttest_ind_from_stats method [7] from scipy as well as the means, standard devations, and sample sizes, we can calculate the t statistic and p value. A two-tailed test is required since my alternate hypothesis sets the averages of both scores from Ireland and elsewhere not being equal (μ != μ_0).



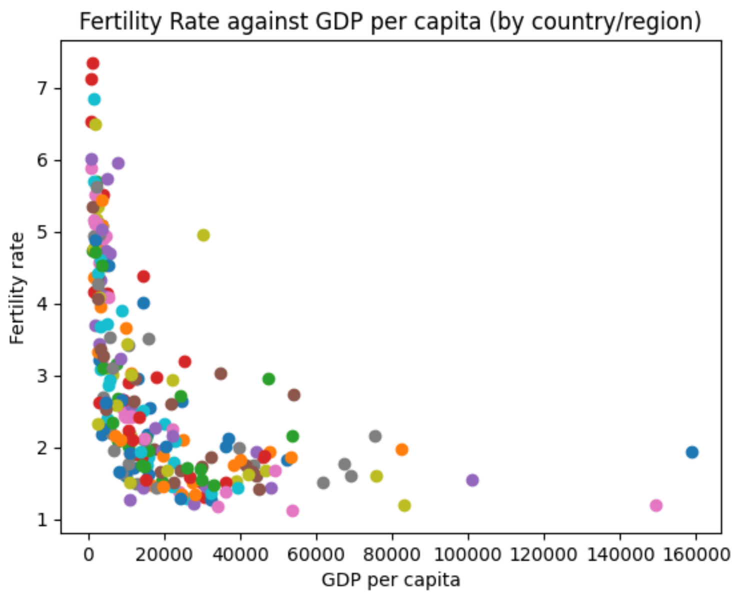
```python
# Q2

import math
import pandas as pd
from scipy import stats

def question_two(data):
    # Null hypothesis: GOES of Guinness served in Ireland is about the same as those served elsewhere.
    # Alternate hypothesis: GOES of Guinness served in Ireland on average is not equal to those served else
    # Two sample t-test is needed
    # Right-tailed test is required

    # check if variances are equal
    std1 = data.loc[['Ireland'], ['Standard Deviation']].values[0][0]
    std2 = data.loc[['Elsewhere'], ['Standard Deviation']].values[0][0]

    # check if variance of both data sets are similar
    assert((std1 / std2 > 0.5) and (std1 / std2 < 2))

    # retrieve means, standard deviations, and sample sizes from both datasets
    x1_mean = data.loc[['Ireland'], ['Mean']].values[0][0]
    x2_mean = data.loc[['Elsewhere'], ['Mean']].values[0][0]
    N1, N2 = data.loc[['Ireland'], ['Sample Size']].values[0][0], data.loc[['Elsewhere'], ['Sample Size']].

    # calculate degrees of freedom of both datasets
    dof = N1 + N2 - 2

    # calculate t-statistic and p value
    t_stat, p_val = stats.ttest_ind_from_stats(x1_mean, std1, N1, x2_mean, std2, N2, 'two-sided')
    print("t-statistic = " + str(t_stat))
    print("p-value = " + str(p_val))

data = {'Sample Size': [42, 61],
        'Mean': [74, 57],
        'Standard Deviation': [7.4, 7.1]}

# Creates pandas DataFrame.
df = pd.DataFrame(data, index=['Ireland', 'Elsewhere'])
question_two(df)
```

Below is the t statistic and p value:

```
t-statistic = 11.73775770205081
p-value = 1.3959536155161474e-20
```

Since the p-value < α, the null hypothesis is **rejected.**

Q3

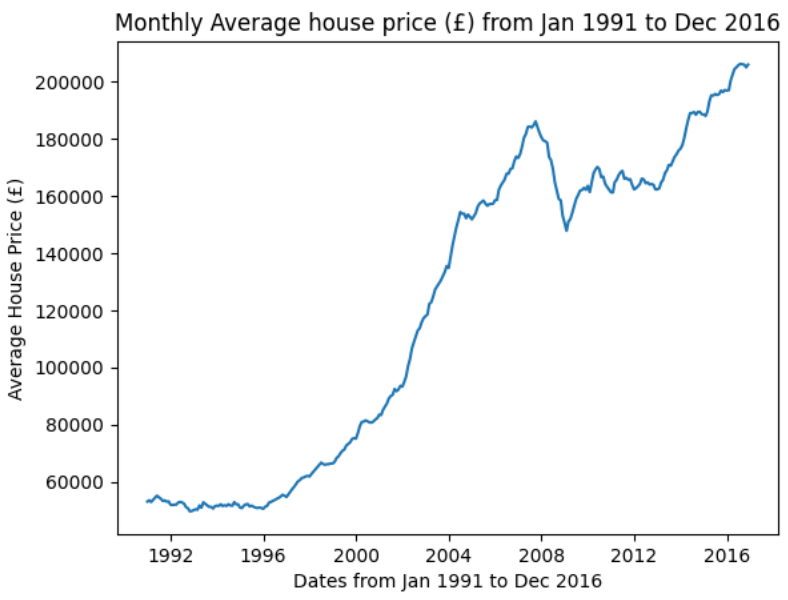


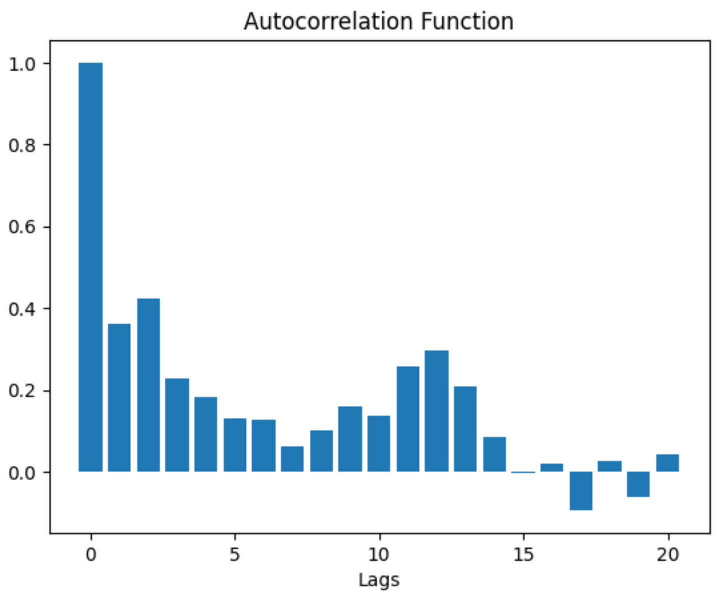
The correlation coefficient is -0.5150224715836512. This means that as one factor changes in one direction, the other factor changes in a different direction, by about half the magnitude. This can be seen in the graph as well as countries that have a higher fertility rate have a lower GDP per capita, and vice versa.

This graph was created by retrieving fertility rates and GDP per capita in 2013 from World Bank Indicators by country/region, matching them up by countries and using matplotlib to plot these data points.

Calculating the correlation coefficient involved using the corr panda method [8].

Q4





Of the ACF monthly data, there could be evidence of seasonality with the spike in the first monthly return, which would correspond to January 1991. The first lag has the highest spike compared to other spikes of that year (spikes 1-11). Spike 12 has a higher spike than ones after it (spikes 13-20) which could signal that house prices increase at the start of the year, although the spike at lag 12 isn't as significant as lag 1. There is a trend in the time series in the **Monthly Average house price** graph in which monthly average house prices increased from 1992 until 2008, after which the price dipped. This dip occurs at the same time as the financial crisis of 2008 (the worst economic crisis since 1929). After markets recovered in 2009 the average house price increases again.

Annualized return is around 5.35%.

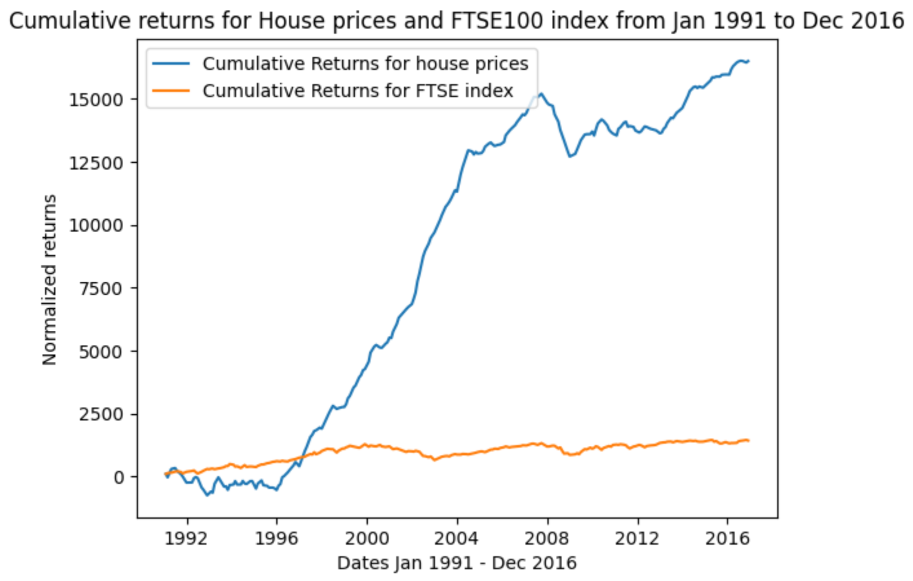
The Monthly Average house price graph was calculated by retrieving data regarding monthly average house prices from January 1991 to December 2016 as well as the dates and using matplotlib to plot the data. Creating the **Autocorrelation Function** graph involved calculating the monthly returns of the average house price via the pandas method pct_change() [9]. Afterwards, using the statsmodel method tsa.acf() [10] calculated the values of the autocorrelation function of the monthly returns. Displaying the ACF function in a bar graph involved the use of matplotlib's bar() method [11].

Calculating the annualized return involved using this formula [12]:

$$\text{Annualized Rate of Return} = \left(\left(\frac{\text{Final Value of Investment}}{\text{Initial Value of Investment}} \right)^{[1/n]} - 1 \right) \text{x } 100$$

Where the initial value of investment would be the average house price value in January 1991, the final value of investment would be the average house price value in December 2016, and n would be the number of years of investment (26 years).

Q5



The average annualized return from the FTSE100 index was around 4.46%. During this period, it would have been better to invest in a UK house as the annualized return over this period (Jan 1991 – Dec 2016) for a UK house was around 5.35%, compared to that from the FTSE100 index at around 4.46%.

How this graph was computed was to retrieve the average house price and adjusted close data from January 1991 to December 2016. Afterwards, I created a merged dataframe with the average house price and adjusted close data from the FTSE100 index by their respective dates. Using the pct_change() [9] pandas method, I calculated the monthly returns of both the FTSE100 index and average house price data, then used the cumsum() [13] pandas method to calculate the cumulative sums. Normalization involved dividing the whole array of cumulative sums of the FTSE100 index and average house price returns by the first values in each respective array and multiplying by 100, thus initializing both datasets with 100 on January 1991. Plotting with plt.plot() [14] from matplotlib displayed the data, as shown in the graph.

Calculating the annualized return for FTSE100 index involved retrieving the adjusted close value from January 1991, representing the initial value of investment, as well as the adjusted close value from December 2016, representing the final value of investment. I used this formula [12]:

$$\text{Annualized Rate of Return} = \left(\left(\frac{\text{Final Value of Investment}}{\text{Initial Value of Investment}} \right)^{1/n} - 1 \right) \times 100$$

Where n represents the number of years of investment, which would be 26 years.

REFERENCES

[1] “Numpy.std,” numpy.std - NumPy v1.26 Manual, https://numpy.org/doc/stable/reference/generated/numpy.std.html (accessed Oct. 2, 2023).

[2] “Numpy.mean#,” numpy.mean - NumPy v1.26 Manual, https://numpy.org/doc/stable/reference/generated/numpy.mean.html (accessed Oct. 2, 2023).

[3] “Numpy.sqrt#,” numpy.sqrt - NumPy v1.26 Manual, https://numpy.org/doc/stable/reference/generated/numpy.sqrt.html (accessed Oct. 2, 2023).

[4] “Scipy.stats.ttest_1samp#,” scipy.stats.ttest_1samp - SciPy v1.11.3 Manual, https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_1samp.html (accessed Oct. 2, 2023).

[5] Stephanie, “Two-sample T-test: When to use it,” Statistics How To, https://www.statisticshowto.com/two-sample-t-test-difference-means/ (accessed Oct. 2, 2023).

[6] “Student’s t-test,” Wikipedia, https://en.wikipedia.org/wiki/Student%27s_t-test#Equal_or_unequal_sample_sizes,_similar_variances_(1/2_%3C_sX1/sX2_%3C_2) (accessed Oct. 2, 2023).

[7] “Scipy.stats.ttest_ind_from_stats#,” scipy.stats.ttest_ind_from_stats - SciPy v1.11.3 Manual, https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_ind_from_stats.html (accessed Oct. 2, 2023).

[8] “Pandas DataFrame Corr() method,” GeeksforGeeks, https://www.geeksforgeeks.org/python-pandas-dataframe-corr/ (accessed Oct. 2, 2023).

[9] “Pandas.DataFrame.pct_change¶,” pandas.DataFrame.pct_change - pandas 0.23.1 documentation, https://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.DataFrame.pct_change.html (accessed Oct. 2, 2023).

[10] “Statsmodels.tsa.stattools.acf¶,” statsmodels.tsa.stattools.acf - statsmodels 0.14.0, https://www.statsmodels.org/stable/generated/statsmodels.tsa.stattools.acf.html (accessed Oct. 2, 2023).

[11] “Bar plot in Matplotlib,” GeeksforGeeks, https://www.geeksforgeeks.org/bar-plot-in-matplotlib/ (accessed Oct. 2, 2023).

[12] “Annual return,” Corporate Finance Institute, https://corporatefinanceinstitute.com/resources/career-map/sell-side/capital-markets/annual-return/ (accessed Oct. 2, 2023).

[13] Pandas dataframe cumsum() method, https://www.w3schools.com/python/pandas/ref_df_cumsum.asp (accessed Oct. 2, 2023).

[14] “Matplotlib.pyplot.plot#,” matplotlib.pyplot.plot - Matplotlib 3.8.0 documentation, https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.plot.html (accessed Oct. 2, 2023).