Homework 5

PJ Grant

November 24th, 2024

Answer each question by writing the Python code needed to perform the task. Please only use the libraries requested in each problem.

Problem 1

Load the interest_inflation data from the statsmodels library as a pandas data frame assigned to `df`. Use the function `df.head()` to view the first 5 rows of the data. Notice the first observation is indexed at 0. Unlike R, Python is a 0 based index language which means when you iterate or wish to view the first observation of a data object it will be at the index 0.

What do the columns `Dp` and `R` represent? (You can find this using the documentation)

In [1]:

```python
from statsmodels.datasets.interest_inflation.data import load_pandas
import numpy as np
df = load_pandas().data
df.columns

## Dp is the Delta log gdp deflator and R is the nominal long term interest rate
```

Out[1]:

```
Index(['year', 'quarter', 'Dp', 'R'], dtype='object')
```

In [2]:

```python
df.head()
```

Out[2]:

	year	quarter	Dp	R
0	1972.0	2.0	-0.003133	0.083
1	1972.0	3.0	0.018871	0.083
2	1972.0	4.0	0.024804	0.087
3	1973.0	1.0	0.016278	0.087
4	1973.0	2.0	0.000290	0.102

Problem 2

Import scipy as sp and numpy as np. Using the `mean()` and `var()` function from scipy, validate that both functions equate to their numpy counterparts against the column `Dp`.

By using the scipy library you should receive a warning message. What does the warning message indicate? Which function should you use going forward?

In [3]:
```
#Importing scipy and numpy
import scipy as sp
import numpy as np
```

In [4]:
```
#validate that both functions equate to their numpy counterparts against the column
sp.mean(df['Dp']) == np.mean(df['Dp'])
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File ~\anaconda3\envs\DSE5002\Lib\site-packages\scipy\__init__.py:137, in __getattr_
_(name)
    136 try:
--> 137     return globals()[name]
    138 except KeyError:

KeyError: 'mean'

During handling of the above exception, another exception occurred:

AttributeError                            Traceback (most recent call last)
Cell In[4], line 2
      1 #validate that both functions equate to their numpy counterparts against the
column - mean
----> 2 sp.mean(df['Dp']) == np.mean(df['Dp'])

File ~\anaconda3\envs\DSE5002\Lib\site-packages\scipy\__init__.py:139, in __getattr_
_(name)
    137     return globals()[name]
    138 except KeyError:
--> 139     raise AttributeError(
    140         f"Module 'scipy' has no attribute '{name}'"
    141     )

AttributeError: Module 'scipy' has no attribute 'mean'
```

In [5]:
```
np.mean(df['Dp'])
#We get the error message that scipy has no mean function, but Numpy does so we wil
```

Out[5]:
```
0.008397309906542055
```

In [ ]:
```
#validate that both functions equate to their numpy counterparts against the column
```

In [6]:
```
sp.var(df['Dp']) == np.var(df['Dp'])
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File ~\anaconda3\envs\DSE5002\Lib\site-packages\scipy\__init__.py:137, in __getattr_
_(name)
    136 try:
--> 137     return globals()[name]
    138 except KeyError:

KeyError: 'var'

During handling of the above exception, another exception occurred:

AttributeError                            Traceback (most recent call last)
Cell In[6], line 1
----> 1 sp.var(df['Dp']) == np.var(df['Dp'])

File ~\anaconda3\envs\DSE5002\Lib\site-packages\scipy\__init__.py:139, in __getattr_
_(name)
    137     return globals()[name]
    138 except KeyError:
--> 139     raise AttributeError(
    140         f"Module 'scipy' has no attribute '{name}'"
    141     )

AttributeError: Module 'scipy' has no attribute 'var'
```

In [7]:

```python
np.var(df['Dp'])
##We get the error message that scipy has no var function, but Numpy does so we wil
```

Out[7]:

```
0.00035296754186450404
```

Problem 3

Fit an OLS regression (linear regression) using the statsmodels api where `y = df['Dp']` and `x = df['R']`. By default OLS estimates the theoretical mean of the dependent variable y. Statsmodels.ols does not fit a constant value by default so be sure to add a constant to `x`. Extract the coefficients into a variable named `res1_coefs`. See the documentation for `params`. Finally print the `summary()` of the model.

Documentation: https://www.statsmodels.org/dev/generated/statsmodels.regression.linear_model.OLS.html

In [9]:

```python
import statsmodels.api as sm
y = df['Dp']
x = df['R']
x = sm.add_constant(x)
model = sm.OLS(y,x)
results = model.fit()
res1_coefs = results.params
print(results.summary())
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                     Dp   R-squared:                       0.018
Model:                            OLS   Adj. R-squared:                  0.009
Method:                 Least Squares   F-statistic:                     1.954
Date:                Sun, 24 Nov 2024   Prob (F-statistic):              0.165
Time:                        17:00:33   Log-Likelihood:                 274.44
No. Observations:                 107   AIC:                            -544.9
Df Residuals:                     105   BIC:                            -539.5
Df Model:                           1                                         
Covariance Type:            nonrobust                                         
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0031      0.008     -0.370      0.712      -0.020       0.014
R              0.1545      0.111      1.398      0.165      -0.065       0.374
==============================================================================
Omnibus:                       11.018   Durbin-Watson:                   2.552
Prob(Omnibus):                  0.004   Jarque-Bera (JB):                3.844
Skew:                          -0.050   Prob(JB):                        0.146
Kurtosis:                       2.077   Cond. No.                         61.2
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
cified.
```

Probelm 4

Fit a quantile regression model using the statsmodels api using the formula `Dp ~ R`. By default quantreg creates a constant so there is no need to add one to this model. In your `fit()` method be sure to set `q = 0.5` so that we are estimating the theoritical median. Extract the coefficients into a variable named `res2_coefs`. Finally print the `summary()` of the model.

Documentation:

https://www.statsmodels.org/dev/generated/statsmodels.regression.quantile_regression.QuantR

In [10]:

```python
import statsmodels.formula.api as smf
mod = smf.quantreg("Dp~R", data = df)
res = mod.fit(q=0.5)
res2_coefs = res.params
print(res.summary())
```

```
                         QuantReg Regression Results                          
==============================================================================
Dep. Variable:                     Dp   Pseudo R-squared:              0.02100
Model:                       QuantReg   Bandwidth:                     0.02021
Method:                 Least Squares   Sparsity:                      0.05748
Date:                Sun, 24 Nov 2024   No. Observations:                  107
Time:                        17:00:35   Df Residuals:                      105
                                        Df Model:                            1
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -0.0054      0.013     -0.417      0.677      -0.031       0.020
R              0.1818      0.169      1.075      0.285      -0.153       0.517
==============================================================================
```

Problem 5

Part 1: Use the `type()` method to determine the type of `res1_coefs` and `res2_coefs`. Print the type in a Jupyter cell.

Part 2: In the next Jupyter cell show that `res1_coefs > res2_coefs`. What does the error mean? To resolve this error we must convert the data to an unnamed object or change the names of the objects. Since we are not focusing on pandas this week we will simply convert to a different data type.

Part 3: Now, do the same comparision using the `tolist()` function at the end of each object name.

Part 4: We performed two types of linear regression and compared their coefficients. Coefficients are essentially the rate at which x changes the values of y. Do some research on what OLS estimates versus what quantreg estimates and explain why we have two different coefficient estimates. In which cases do you think quantile regression will be useful? What about ordinary least squares regression?

In [11]:
```
#Part 1
print(type(res1_coefs))
print(type(res2_coefs))
```

```
<class 'pandas.core.series.Series'>
<class 'pandas.core.series.Series'>
```

In [ ]:
```
#Part 2a
res1_coefs > res2_coefs
```

In [12]:
```
#Part 2b - convery to a numpy array
coef1 = np.array(res1_coefs)
coef2 = np.array(res2_coefs)
```

In [ ]:
```
#Part 2c - compare numpy arrays
coef1 > coef2
```

In [13]:

```
#Part 3
res1_coefs.tolist() > res2_coefs.tolist()
```

Out[13]:

```
True
```

Part 4

OLS is best suited when trying to calculate the mean outcome for a particular input value, with residuals that are normally distributed around the line of best fit. Quantile regression is better suited for determining output values aside from the mean (median, percentile, etc), and can perform well on skewed data sets. In our example above, the coefficients are different because the OLS model represents median values and the Quant Reg model represens the median.