R_Hawthorne_Homework 5

August 5, 2023

1 Homework 5

1.0.1 Rebecca Hawthorne

1.0.2 8/1/2023

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

1.0.3 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)

[2]:
```
from statsmodels.datasets.interest_inflation.data import load_pandas
df = load_pandas().data
df.head()
```

[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

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

1.0.4 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?

[3]:
```
import numpy as np
import scipy as sp
```

```
np.mean(df['Dp']) == sp.mean(df['Dp'])
```

```
C:\Users\hawthorner\AppData\Local\Temp\ipykernel_10632\558539862.py:4:
DeprecationWarning: scipy.mean is deprecated and will be removed in SciPy 2.0.0,
use numpy.mean instead
  np.mean(df['Dp']) == sp.mean(df['Dp'])
```

[3]: True

[4]:
```
np.var(df['Dp']) == sp.var(df['Dp'])
```

```
C:\Users\hawthorner\AppData\Local\Temp\ipykernel_10632\3874408497.py:1:
DeprecationWarning: scipy.var is deprecated and will be removed in SciPy 2.0.0,
use numpy.var instead
  np.var(df['Dp']) == sp.var(df['Dp'])
```

[4]: True

The values calculated by both numpy and scipy are equal.

The warning is that scipy.var and scipy.mean have been depreciated and, instead, we should use the numpy versions.

1.0.5 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

[5]:
```
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(res1_coefs)
print(results.summary())
```

```
const   -0.003126
R        0.154512
dtype: float64
```

```
                            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:                Wed, 02 Aug 2023   Prob (F-statistic):              0.165
Time:                        20:14:33   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
specified.
```

1.0.6 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.QuantRe

[11]:
```
model_QR = sm.regression.quantile_regression.QuantReg(y, x)

results_QR = model_QR.fit(q=0.5)
res2_coefs = results_QR.params

print(res2_coefs)
print(results.summary())
```

```
const   -0.005388
R        0.181800
dtype: float64
                            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:                Wed, 02 Aug 2023   Prob (F-statistic):              0.165
Time:                        20:18:11   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
specified.
```

1.0.7 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?

[12]:
```
print(type(res1_coefs))
print(type(res2_coefs))
```

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

[13]:
```
res1_coefs > res2_coefs
```

[13]:
```
const     True
R        False
dtype: bool
```

[20]:
```
res1_coefs.tolist()

res2_coefs.tolist()

print(res1_coefs[0]>res2_coefs[0])

print(res1_coefs[1]>res2_coefs[1])
```

```
True
False
```

[ ]:

The main difference between OLS and Quantile regression is the dependent value they are approximating. OLS approximates the mean of the dependent values based on the independent value; whereas Quantile regression will approximate any quantile(often the median) of the dependent value based on the independent value. Mean and median are not necessarily equal, the mean is more heavily swayed by extreme values; therefore, it is often important to consider both measures of central tendency to create a more accurate portryal of the data.

The confidence interval surrounding an OLS approximation remains constant along the length of the line. Where as a the width of the confidence interval for QR can change as the spread in the dependent values changes. This can more accurately refelct the data in cases where the spread is not consistent. OLS models assume the errors in the data are normally distributed with constant variation, this is not the case in many data sets and is not required by QR.

QR is then useful in cases where the errors do not meet the requirements for OLS or where extreme values may be effecting the data. QR is also useful in that it does not have to approximate the mean. For example, instead QR can create a function that looks at predciting the 90 percentile in the dependent data based on the independent variable. This may be useful in certain applications such as salary, spending, or costs. It is often useful to consider the bottom 90% (or equaivalently the top 10%) in these applications.

For data that whose errors are normal and consistently spread in which the mean value is desired. Many data fall into the catergory of having normal errors and often the average value that coccurs is the one we are looking for. For example, height verses weight distribution tends to be normal when looking at segments of the population.

(source: https://support.sas.com/resources/papers/proceedings16/5620-2016.pdf)