Homework 5

November 24, 2023

1 Homework 5

1.0.1 Wanida Ruangsiriluk

1.0.2 November 21, 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)

Dp is Delta log gdp deflator (changes in log scale of GFP deflator), measure of price inflation/deflation with respect to a specific base year. R is nominal long term interest rate, an interest rate that is unadjusted for inflation

[2]:
```
from statsmodels.datasets.interest_inflation.data import load_pandas
import numpy as np

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
```

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?

#Answer:

The answers from both libraries are equivalent. The warning message from using scipy function refers to the function of the future release may not be available. It suggests to use numpy library instead to go forward.

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


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

[3]:
```
0.008397309906542055
```

[4]:
```
sp.mean(df['Dp'])
```

```
/var/folders/14/1r6k98j90wv9bfq7ymzly7c80000gn/T/ipykernel_60094/2735895050.py:1
: DeprecationWarning: scipy.mean is deprecated and will be removed in SciPy
2.0.0, use numpy.mean instead
  sp.mean(df['Dp'])
```

[4]:
```
0.008397309906542055
```

[5]:
```
np.var(df['Dp'])
```

[5]:
```
0.00035296754186450404
```

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

```
/var/folders/14/1r6k98j90wv9bfq7ymzly7c80000gn/T/ipykernel_60094/1383872187.py:1
: DeprecationWarning: scipy.var is deprecated and will be removed in SciPy
2.0.0, use numpy.var instead
  sp.var(df['Dp'])
```

[6]:
```
0.00035296754186450404
```

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

[10]:
```
import statsmodels.api as sm
import numpy as np
```

```python
import pandas as pd

y = df['Dp']
x = sm.add_constant(df['R'])

res1 = sm.OLS(y, x).fit()
res1_coefs = res1.params

res1.summary()
```

[10]:

Dep. Variable:	Dp	**R-squared:**	0.018
Model:	OLS	**Adj. R-squared:**	0.009
Method:	Least Squares	**F-statistic:**	1.954
Date:	Tue, 21 Nov 2023	**Prob (F-statistic):**	0.165
Time:	15:27:11	**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.

[11]:
```python
print(res1_coefs)
```

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

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

[14]:
```python
res2 = sm.QuantReg(y, x).fit(q = 0.5)
res2_coefs = res2.params
res2.summary()
```

[14]:

Dep. Variable:	Dp	**Pseudo R-squared:**	0.02100
Model:	QuantReg	**Bandwidth:**	0.02021
Method:	Least Squares	**Sparsity:**	0.05748
Date:	Tue, 21 Nov 2023	**No. Observations:**	107
Time:	15:48:09	**Df Residuals:**	105
		Df Model:	1

	coef	**std err**	**t**	**P> |t|**	**[0.025**	**0.975]**
const	-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

[15]:
```
print(res2_coefs)
```

```
const   -0.005388
R        0.181800
dtype: float64
```

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?

[20]:
```
#Part 1

type(res1_coefs)
```

[20]:
```
pandas.core.series.Series
```

[21]:
```
type(res2_coefs)
```

[21]:
```
pandas.core.series.Series
```

[22]:
```
#Part 2

res1_coefs > res2_coefs
```

[22]:
```
const     True
R        False
dtype: bool
```

#Note that the above syntax did not show error.

[29]:
```python
#Part 3
ols_coefs = res1_coefs
qreg_coefs = res2_coefs
ols_coefs.tolist() > qreg_coefs.tolist()
```

[29]:
```
True
```

1.0.8 Problem 6

What are the advantages of using Python as a general purpose programming language? What are the disadvantages? Why do you think data scientists and machine learning engineers prefer Python over other statistically focused languages like R? Your answer should a paragraph for: (1) advantages, (2) disadvantages, and (3) why its popular. Please cite each source used in your answer.

#Answer:

The advantage of using Python is that the code can be run across different platforms (Windows, macOS, Linux, etc.). The language is user-friendly and easily understandable or followed (1). It is also widely used globally, so there will be many libraries, framework resources, and tools, available to use in different applications from developers, and many are free under Open Source Initiative (OSI) license (2),(3). The code can be run right away after being written without the need for compiling steps (4). Because of these advantages of using Python, there is also robust and growing community support which is attractive to users and makes Python popular (5). Disadvantages of using Python include slow speed because the program is interpreted and dynamically typed and will use the computer RAM intensively (4). It is also prone to runtime errors as the data types of variables in Python can change suddenly, as it is a dynamically typed language (6). And because of its simplicity, Python can be misused for tasks where better alternatives are being ignored. It's also not commonly used in large enterprises because Python has limitations in database access for complex legacy data since it's still underdeveloped (7).

1. https://www.geeksforgeeks.org/python-for-data-science/
2. https://onlinedegrees.sandiego.edu/python-for-data-science/
3. https://opensource.org/osd/
4. https://www.linode.com/docs/guides/pros-and-cons-of-python/
5. https://www.analyticsinsight.net/10-reasons-why-python-is-one-of-the-best-programming-languages/
6. https://webandcrafts.com/blog/advantages-and-disadvantages-of-python
7. https://pythonistaplanet.com/disadvantages-of-python/