benjaminmichaels module05lab01

August 5, 2023

1 Homework 5

1.0.1 Benjamin Michaels

$1.0.2 \quad 8/3/23$

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)

```
[7]: # your code here

from statsmodels.datasets.interest_inflation.data import load_pandas
import numpy as np
df = load_pandas().data
print(df.head())

#Dp is the dependent variable while R is the independent variable
```

	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?

```
[14]: # your code here
      import scipy as sp
      import numpy as np
      np.mean(df['Dp'])
      np.var(df['Dp'])
      np.mean(df['Dp']) == np.mean(df['Dp'])
      np.var(df['Dp']) == np.var(df['Dp'])
     C:\Users\ben98\AppData\Local\Temp\ipykernel_10712\3741978279.py:8:
     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'])
     C:\Users\ben98\AppData\Local\Temp\ipykernel_10712\3741978279.py:9:
     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'])
[14]: True
```

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.

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

```
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
results.summary()
```

[15]: <class 'statsmodels.iolib.summary.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:	Thu, 03 Aug 2023	Prob (F-statistic):	0.165
Time:	21:49:12	Log-Likelihood:	274.44
No. Observations:	107	AIC:	-544.9
Df Residuals:	105	BIC:	-539.5
Df Model:	1		

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 Durb	in-Watson:		2.552
Prob(Omnib	us):	0	.004 Jarq	ue-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 11 11

1.0.6 Probelm 4

Fit a quantile regression model using the statsmodels api using the formula $Dp \sim R$. 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.

 $Documentation: \ https://www.statsmodels.org/dev/generated/statsmodels.regression.quantile_regression.QuantRegression.quantile_regression.QuantRegression.quantile_regression.QuantRegression.quantile_regression.QuantRegression.quantile_regression.QuantRegression.quantile_regression.QuantRegression.QuantRegression.quantile_regression.QuantRegression.quantregressio$

```
import statsmodels.formula.api as smf
# your code here
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:	${\tt QuantReg}$	Bandwidth:	0.02021
Method:	Least Squares	Sparsity:	0.05748
Date:	Thu, 03 Aug 2023	No. Observations:	107
Time:	21:51:43	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

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?

```
[21]: # your code here
      print(type(res1_coefs))
      print(type(res2_coefs))
     <class 'pandas.core.series.Series'>
     <class 'pandas.core.series.Series'>
[25]: res1_coefs > res2_coefs
[25]: Intercept
                   False
      R
                   False
      dtype: bool
[26]: #it means that we compared two dataframes that have different labels or indexes,
       →when this error can be thrown.
     res1 coefs.tolist() > res2 coefs.tolist()
[24]: res1_coefs.tolist() > res2_coefs.tolist()
[24]: False
 []:
```