# Annand Homework 5

August 6, 2023

## 1 Homework 5

### 1.0.1 Joseph Annand

### 1.0.2 8/6/2023

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]:
```
# your code here
import pandas as pd
import statsmodels.api as sm
from statsmodels.datasets.interest_inflation.data import load_pandas

df = load_pandas().data

df.head()

# Documentation for interest_inflation dataset: https://www.statsmodels.org/dev/
 ↪datasets/generated/interest_inflation.html
# Dp = Delta log GDP deflator
# R = nominal long term interest rate
```

[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?

[3]:
```
# your code here
import scipy as sp
import numpy as np

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

'''
Warning message indicates that scipy.mean is deprecated and will be removed in␣
 ↪SciPy 2.0.0.
Numpy.mean should be used going forward instead
'''
```

```
C:\Users\janna\AppData\Local\Temp\ipykernel_31600\1182379785.py:5:
DeprecationWarning: scipy.mean is deprecated and will be removed in SciPy 2.0.0,
use numpy.mean instead
  sp.mean(df['Dp']) == np.mean(df['Dp'])
C:\Users\janna\AppData\Local\Temp\ipykernel_31600\1182379785.py:6:
DeprecationWarning: scipy.var is deprecated and will be removed in SciPy 2.0.0,
use numpy.var instead
  sp.var(df['Dp']) == np.var(df['Dp'])
```

[3]:
```
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

[4]:
```
# your code here
y = df['Dp']
x = df['R']
x = sm.add_constant(x)
ols_model = sm.OLS(y,x)
results = ols_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:                Thu, 03 Aug 2023   Prob (F-statistic):              0.165
Time:                        21:57:18   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.QuantR

[5]:
```
# your code here
import statsmodels.formula.api as smf

quantile_model = smf.quantreg("Dp ~ R", df)
res = quantile_model.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:                Thu, 03 Aug 2023   No. Observations:                  107
Time:                        21:57:18   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?

[8]:
```
# your code here

# Part 1
type(res1_coefs) == type(res2_coefs)
print(type(res1_coefs))

# Part 2
res1_coefs > res2_coefs # raises ValueError: "Can only compare␣
 ↪identically-labelled Series objects"

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

# Part 4
'''
OLS estimates the conditional mean of the response variable across the predictor␣
 ↪variable
where quantile regression estimates the median of the response variable across␣
 ↪the predictor variable. Quantile regression can be useful
in situations where the conditions for OLS are not met. Quantile regression is␣
 ↪also more robust to outliers. It is also able to detect
```

```
heteroskedasticity, thus checking if the condition of constant variance for OLS␣
 ↪is met. In cases where the conditions for OLS
are met and a linear relationship between a response variable and one or more␣
 ↪explanatory variables is being evaluated. OLS would be useful.

https://lost-stats.github.io/Model_Estimation/OLS/simple_linear_regression.html
https://soc.utah.edu/sociology3112/regression.php
https://www.xlstat.com/en/solutions/features/
 ↪ordinary-least-squares-regression-ols
https://towardsdatascience.com/quantile-regression-ff2343c4a03
'''
```

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

[8]:
```
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.

The advantages of using Python as a general purpose programming language is that is may be implemented in a variety of workflows and applications. Python being a general purpose programming language makes it easier to colloborate with software developers and other engineers, who may also be working with Python. The deployment and reproducibility of Python applications are easy for programmers thanks to the wide variety of free technologies for Python use and development. Python is easier to read, faster, and has more functionalities compared to statistically focused languages. On top of text, csv, and Excel files; Python accepts many other data formats, like SQL tables and JSON.

The disadvantges of using Python mainly stem from it not being speficially designed for data analysis. Python has weak perfomance when dealing with large amounts of data and has poor memory efficiency. While great for debugging code, errors showing up in runtime means Python code requires vigorous testing. Additionally, data visualization is more difficult in Python than a statisitically focused language, like R, as the matplotlib and seaborn libraries do not produce as beautiful and informative graphics.

I think that Python is more popular for data scientists and machine learning engineers because its advantages outweigh its disadvantages. Python's simplicity and speed make it the better choice for cross-team workflows and large-scale applications, respectively. Python has multiple machine learning libraries (sckit-learn, TensorFlow, Keras) to build simple models from scratch. Above all, Python is one of the most popular open-source programming languages in the world right now, meaning there is a large community developing free-to-use packages and technologies for Python development and implementation.

https://www.datacamp.com/blog/python-vs-r-for-data-science-whats-the-difference#
https://www.geeksforgeeks.org/python-features/ https://www.ibm.com/blog/python-

vs-r/ https://www.projectpro.io/article/data-science-programming-python-vs-r/128#mcetoc_1ght2bqoe8p