DEGREE OF MASTER OF SCIENCE IN FINANCIAL ECONOMICS

FINANCIAL ECONOMETRICS

HILARY TERM 2020 COMPUTATIONAL ASSIGNMENT 2
PRACTICAL WORK 4

February 2020.

This assignment must be submitted before
noon (12:00) Friday 5th Week (19 February 2020)
by uploading to SAMS**.**

This is individual work.
All solutions must be submitted by the due date and time.
Do not write your name on your submission.

Submission limit: the minimum of 5 pages or 1,250 words
Material on pages 6+ will not be assessed.
The limit does not include the cover sheet, academic honesty declaration or submitted code files.
All material, including figures, equations, and explanatory text must fit within 5 pages.

Assessment

This assignment is assessed in 2 parts:

- 67% - Report. This report should focus on analysis and synthesis and not code or the numerical values of the problems.
- 33% - Autograder. 2 functions must be submitted to compute the required outputs using the inputs. The signature of each function is provided as part of the problem. You must ***exactly*** match the function name. Submissions must be in Python and must be in a single Python file (*some_filename*.py) containing all functions. IPython notebooks are **not** accepted. Please pay close attention to the dimensions and types of each input or output. All data inputs will be pandas DataFrames or Series, as indicated in the program description.
  Note: *Please* run your code in the function you submit to ensure that it does not produce an error. A function that errors when run is given a mark of 0. The autograder uses loose criteria when judging correctness, and so any value within about 1% of the reference will be marked as correct. See the example file solutions-pw3.py for the structure of the file expected by the autograder.

Tips for Autograded Code

- Your submissions **MUST** be a Python file (.py). IPython notebooks are not accepted, and submitting your solution in a notebook will result in a mark of 0.
- Your submission will not have access to any data files. You must not read or write anything from your program.
- Your submission will be run in a random directory. You cannot assume anything about data files existing in any particular location. Your code does not need to access data. All required data is passed through the inputs.
- You can use demo-autograder-pw3.py to check that your solution accepts the required arguments and returns values with the correct type and shape. You should run the program along with your code *in an empty directory*, which is how the autograder runs it. Code similar to

  ```
  mkdir empty
  cd empty
  <copy demo-autograder-pw3.py to empty along with your submission>
  <edit demo-autograder-pw3.py to give it the name of your submission>
  python demo-autograder-pw3.py
  ```

  can be used to check that your code will run correctly.
- The autograder submission should not normally have code that you wrote as part of the analysis. It should only contain the required functions, imports, and code necessary to run the functions. In *ideal* submission would not execute any code when imported, and instead would only contain import statements and functions.

```
import numpy as np
import pandas as pd

def first_function(a, b):
    # Do something
    return "something"

def used_by_second_function(x, y):
    # Do something
    return "something", "else"

def used_by_second_function(x, y, z):
    w = used_by_second_function(x, y)
    return w[0], w[1], z
```

Problem 1

Your ID determines which data series you should analyze. The formula to determine your data set ID is

$$\text{Data Set ID} = ID \mod 4 + 1.$$

Your Data Set ID should be in 1, 2, 3 or 4. The data sets are:

Data Set ID	FRED Code
1	UNRATENSA
2	HOUSTNSA
3	UMCSENT
4	TOTALNSA

1. Report the Data Set ID and the name Code of the Data Set you are analyzing. [5%]
2. Download the full sample of data for your assigned Data Set from FRED and describe the data (quantitative, graphical, qualitative as you find useful). [5%]
3. Decide whether you should log your data. Explain your choice. [5%]
4. Formally test the data for a unit root, and if your find one, remove it using the difference (not seasonal difference) operator Δ. Describe the test you ran and any choices you made. [10%]
5. Using the transformed data (if any was needed), build two distinct ARMA models for the series using the first 50% of the data. Describe how you arrived at your preferred models. [15%]
6. Evaluate the objective performance of your models at horizons 1, 3 and 12 using the data in the second half of the sample. You should re-estimate the model parameters each month, but do not need to re-select the model specification. [20%]
7. Compare your models at horizons 1, 3 and 12 against random walk forecasts. [20%]

Code Problems

Mincer-Zarnowitz

Implement Mincer-Zarnowitz regression.

```
[parameters, indiv_stats, joint_stat] = mincer_zarnowitz(realization, forecast)
```

Outputs

- `parameters` - pandas Series containing two values, $\hat{\alpha}$ and $\hat{\beta}$ from MZ regression. The index must be `["alpha","beta"]`.
- `indiv_stats` - pandas Series containing the test statistics based on the t-ratio for, $H_0 : \alpha = 0$ and $H_0 : \beta = 1$ (**Note**: the values on the right side of the equals signs differ). The index must be `["alpha","beta"]`.

- `joint_stat` - scalar, Wald statistic value testing the joint null of correct specification, $H_0 : \alpha = 0 \cap \beta = 1$.

Inputs

- `realization` - R-element pandas Series of realizations of variable being forecast. Will have a `DatetimeIndex`.
- `forecast` - R-element pandas Series of 1-step ahead forecasts of the variable being forecast. Will have a `DatetimeIndex` where the forecast is already aligned with the realization so that the forecast errors are `realization - forecast`.

Diebold-Mariano

Implement a Diebold-Mariano test.

```
[avg_diff, std_err, dm_stat, concl] = diebold_mariano(loss_a, loss_b, nw_bandwidth)
```

Outputs

- `avg_diff` - Mean loss difference where $\delta_t = L_t^A - L_t^B$ (float).
- `std_err` - Estimated standard error of the difference (float).
- `dm_stat` - The Diebold-Mariano test statistic (float).
- `concl` - Conclusion. Should be -1 if null is rejected in favor of model A, 0 if null is not rejected and 1 if null is rejected in favor of model B. You should use a 5% size for the test (2.5% in each tail) (int).

Inputs

- `loss_a` - R-element pandas Series containing the losses from model A. Will have a `DatetimeIndex` that matches `loss_b`.
- `loss_b` - R-element pandas Series containing the losses from model B. Will have a `DatetimeIndex` that matches `loss_a`.
- `nw_bandwidth` - Bandwidth (number of lags) to use in the Newey-West estimator (integer).