New Economic School, program MAF/MSF

Econometrics 2

Home assignment 3

To be handed in at 23:00 on Friday, February 18, 2022

Your homework should consist of two parts: 1) a report on all exercises; 2) a thoroughly commented script which loads the data, makes all the required computations, and produces all the required tables and/or graphs. The script could be in any language you want (Python, R, EViews, Stata, Gretl etc.). The report can be either handwritten or in PDF format.

The homework is to be handed in into a Git repository, created for you at Bamboo.nes.ru. Since my.NES is expected to be closed for maintenance next week, both the code and the report should be committed to the repository.

The repository for this particular homework is

`https://bamboo.nes.ru/git/<username>/metrix2-2022-hw3.git`,

where `<username>` denotes your NES username (e.g. my username is `sgolovan`).

In order to hand your work in do the following:

- Clone your Git repository (use your usual NES username and password when asked):

```
git clone https://bamboo.nes.ru/git/<username>/metrix2-2022-hw3.git
cd metrix2-2022-hw3
```

- Create, modify, commit your code. Use the default `master` branch for simplicity.
- Push it to the remote repository:

```
git push
```

- Do the same for your report.

Problem 1. In your repository you will find a data file `daily_MSFT.csv` that contains daily data on the Microsoft stock prices.

Variables in the data set are:

`timestamp`	date in YYYY-MM-DD format
`open`	opening stock price in US \$ per share
`high`	the highest stock price in US \$ per share
`low`	the lowest stock price in US \$ per share
`close`	closing stock price in US \$ per share
`volume`	volume traded in US \$

The goal of the exercise is to estimate and to predict the returns volatility using EWMA and the intraday data (the high and low prices).

Use these data to perform the exercises below.

(1) Read the dataset into computer's memory. Convert all the variables to time series.
(2) Calculate the logarithmic returns based on the closing price. Plot it and comment on the plot.
(3) Calculate the series of volatilities $\hat{\sigma}_t$ using the EWMA technique discussed in class (use $\lambda = 0.94$). Plot the estimated series together with the returns series. What can you conclude from the plot?
(4) Calculate the series $s_t^2 = \log(\mathtt{high}_t) - \log(\mathtt{low}_t)$, estimate an ARMA$(p, q)$ model for it (explain why did you choose your particular values of p and q).
(5) Forecast the returns variance for two weeks (starting from February 8, 2019) using the models estimated in (3) and (4). Plot them onto a single graph and comment on the differences between the forecasts.

Problem 2. In your repository you will find a data file `bondreturns.txt` that contains monthly data on bond returns from January 1952 till December 2003.

Variables in the data set are:

`year`	year
`month`	month
`m1_12`	return for bonds with 1–12 month maturity
`m24_36`	return for bonds with 24–36 month maturity
`m48_60`	return for bonds with 48–60 month maturity
`m61_120`	return for bonds with 61–120 month maturity

The goal of the exercise is to test whether the bond returns exhibit autoregressive conditional heteroskedasticity, and build a model which capture it.

Use these data to perform the exercises below.

(1) Read the dataset into computer's memory. Convert it to time series. Show the plot and the histogram of the `m24_36` variable. Does it have the distribution features discussed in class?
(2) Report the skewness and kurtosis of the bonds returns series. Do they confirm your findings from the previous exercise?
(3) Use the method of your choice to test whether the return series is a white noise. Depending on your conclusion, build a simple ARMA model for the series and collect its residuals.
(4) Test whether there is an ARCH effect in the residuals from the model estimated in (3).
(5) Estimate the ARCH(4) model for the bonds returns series. Predict the series itself and its volatility for a few periods ahead (say, 10).
(6) Estimate the GARCH(1, 1) model for the bonds returns series. Predict the series itself and its volatility for the same 10 periods ahead. Compare this model with the one from (5) using some information criterion. Are there any differences in predictions for these two models?
(7) Construct the standardised residual series for the two estimated models (they are the residuals divided by the estimated conditional standard errors, find out how to get them from the fitted model object). Look at their summary statistics. Have you mitigated the distributional issues which the original series show?

Hint: If you're using Python, then there's a good `arch` package which helps in estimating ARCH–GARCH models including threshold ones. There's a notebook with some examples: `http://nbviewer.jupyter.org/github/bashtage/arch/blob/master/examples/univariate_volatility_modeling.ipynb`.

To plot a histogram one can use `matplotlib.pyplot.hist()`: `https://matplotlib.org/devdocs/api/_as_gen/matplotlib.pyplot.hist.html`.

`scipy.stats.describe()` (`https://docs.scipy.org/doc/scipy-0.19.1/reference/generated/scipy.stats.describe.html`) returns descriptive statistics including skewness and kurtosis.

`statsmodels.stats.diagnostic.het_arch()` (`http://www.statsmodels.org/dev/generated/statsmodels.stats.diagnostic.het_arch.html`) can be used to perform ARCH LM test as discussed in class.

Hint: If you're using R, then it's very helpful to install the `FinTS` package to your R distribution. It's a companion to "Analysis of Financial Time Series" by R. Tsay, and it contains quite a few time series data examples and implements many of the tests and model estimators which we've discussed in class:

- The `FinTS.stats()` command reports a few statistics including skewness and kurtosis.
- The `ArchTest()` command the test statistic for testing ARCH effect and its p-value.
- The `garchFit()` function from the `fGarch` package estimates ARCH or GARCH models. It its notation ARCH(p) means GARCH(p, 0), so the white noise model for the series x with ARCH(4) errors can be fitted using `garchFit(~garch(4,0), data=x)`. And as usual, `predict()` makes the predictions.