CHALMERS | UNIVERSITY OF GOTHENBURG

Financial Time Series TMS088/MSA410 – LP4 2020/21

Project 2: Calculating Values at Risk

The file *dat_intel.csv* contains daily closing stock prices of the Intel Corporation [1] from March 1st 2010 to February 9th 2017, in the form of the time series

$$v = (v_t, t = 1, \dots, 1751).$$

Let us interpret v as observations from a stochastic process $V = (V_t, t \in \mathbb{Z})$ modeling the daily stock prices of Intel. The goal of this project is to see how well we can calculate the $100p\%$ VaR (value at risk) for this asset for a 1 day horizon, using the Econometric Toolbox in MATLAB (or equivalent packages in R). This means that we should calculate the value VaR_t, defined by

$$P\left(V_t - V_{t-1} \le \mathrm{VaR}_t | (V_s, s < t)\right) = p$$

for $p \in [0, 1]$.

For this project you should make sure you are familiar with the lecture notes covering ARMA processes and GARCH processes. You should implement your code in MATLAB or R. if you use the former, you should use the functions of the Econometric Toolbox[2], and we list a few useful ones below. If you use the latter, please refer to the dedicated document containing some tips. As usual you need to make a detailed description of your implementation along with a discussion of the result.

1. (5 points) Let the log returns X be given by $X_t = \log(V_t) - \log(V_{t-1})$. Let $F_{X_t} = P(X_t \le \cdot | (X_s, V_s, s < t))$ be the cumulative distribution function of X_t given all information up to time $t - 1$. Show that

$$\mathrm{VaR}_t = V_{t-1}\left(\exp(F_{X_t}^{-1}(p)) - 1\right) \tag{1}$$

for $t \ge 2$.

[1]Source: `https://finance.yahoo.com/`

[2]See https://se.mathworks.com/help/econ/

Please turn!

2. (3 points) Plot the (sample) autocorrelation and partial autocorrelation functions of X for lags $h = 0, \dots, 50$. Then, do a Ljung–Box test with $h = 50$. Repeat these two steps with the squared returns X^2. Discuss the results: Do you think the returns and/or squared-returns should be modeled as white noise?

3. (7 points) Now split the time series X into a *training set* $(X_t, t = 2, \dots, 1526)$ and a *test set* $(X_t, t = 1527, \dots, 1751)$.
For all values of p and q in the range $0, 1, \dots, 15$ and such that $p + q > 0$ and $p \geq q$, fit an $\mathrm{ARMA}(p, q)$ process with Gaussian noise to the *square* returns obtained from the training data and compute the BIC and the AICC (this will take a couple of minutes). Which models minimize the two information criteria?
Repeat this operation assuming now that the noise follows a t-distribution. Write these candidate models down. Perform Ljung–Box tests on the standardized residuals of each model and interpret the results.
Finally, deduce from the obtained ARMA models the orders of GARCH models that could be suitable to model the returns.

Hints and notes:

- *Include a non-zero offset term in your estimation (this corresponds to including the sample mean).*
- *Compute the BIC and AICC using the formulas given in Methods 3.2.22 and 3.2.23 of the lecture notes. Think of replacing the quantity $p + q + 1$ by the number M of (non-zero) parameters actually estimated by the algorithm.*
- *It may be the case that not all calls to estimate yield a valid model fit, especially for big p and q. If so, ignore these failed models in the minimization. The functions try and catch can be useful here.*
- *You may encounter the message "Warning: Lower bound constraints are active; standard errors may be inaccurate.". It can be an indication that the model includes too many coefficients and/or that the likelihood maximization is stuck at a local maximum. For this project, simply ignore it.*
- *In* MATLAB, *start with the following optimization parameters: Algorithm = 'interior-point'; ConstraintTolerance = 1e-7. Change these parameters if you feel like the optimization is taking too long or too many errors are thrown out.*

4. (4 points) Let K be the maximal order of the ARMA models with Gaussian noise fitted in the previous task. For all values of p and q such that $p < 0 \leq K$ and $0 \leq q \leq K$, fit a $\mathrm{GARCH}(p, q)$ process driven by Gaussian noise to the training data and compute the BIC and the AICC (this will take a couple of minutes). Which models minimize the two information criteria? Write these candidate models down including the assumptions on the noise.

See next page!

Hints and notes:

- *When writing your code, keep in mind that* $\mathrm{GARCH}(p,q)$ *as defined in the lecture notes corresponds to a* $\mathrm{GARCH}(q,p)$ *in* MATLAB.
- *Include once again a non-zero offset term in your estimation (this corresponds to including the sample mean) and do not (explicitly) use presample data.*
- *Use the definition* $\mathrm{BIC} = -2\,\mathrm{logL} + M \cdot \log(n)$, *where* logL *is the log-likelihood of the model fit,* M *is the number of parameters estimated, and* n *is the sample size. As for the AICC, use the definition given in Method 4.2.3 of the lecture notes.*
- *If you encounter a invalid fit or a warning message, proceed as in Task 3.*

5. (4 points) For the two identified candidate models that minimized the information criteria, perform a diagnostic check, i.e., compute the residuals and analyze these: What should the distribution and covariance structure of the residuals be if the model is correct? How does the autocorrelation plot of the residuals look like? How does the autocorrelation plot of the squared residuals look like? What about a qq-plot of the residuals? Are there problems? Finally, are the orders of your GARCH models coherent with the orders of the ARMA models obtained in Task 3?

6. (6 points) Repeat the previous two tasks, assuming now that the noise follows a t-distribution (and taking K as the maximal order of the ARMA models with t-distributed noise fitted in Task 3), so that you end up with four candidate models.

7. (6 points) Estimate VaR_t and evaluate the estimates for $t = 1527, \ldots, 1751$ by following these steps for each of the four candidate models obtained in Tasks 4 and 6, as well as the four GARCH models obtained in Task 3:

- Report the distribution of X_t given $(X_s, V_s, s < t)$ including its parameters (this will be either a Gaussian or a (generalized) t-distribution).
- For each $t = 1527, \ldots, 1751$ of the test data set, compute VaR_t given $(X_s, V_s, s < t)$ using (1) for $p = 0.1, 0.05$ and 0.01. Do not reestimate the parameters of the models at each time t, but use the observations $(X_2, \ldots, X_{t-1})$ to update the conditional variances as needed.
- Count the number of VaR breaches, i.e., for $t = 1577, \ldots, 1751$, count the number of times that $V_t - V_{t-1} \leq \mathrm{VaR}_t$.

After doing this, also compute VaR_t under the simple model that $X_t \sim \mathcal{N}(0, \hat{\sigma}^2_{t-1})$, where $\hat{\sigma}^2_{t-1}$ is the sample variance of $X_2, X_3, \ldots, X_{t-1}$ and count the number of breaches as above.

Please turn!

Report the number of breaches for each p and each of the candidate models. Based on the outcome of this experiment, what model for the estimation of VaR_t would you recommend for this stock and why? (You are very welcome to implement alternative models for the returns if you can find one that appears to fit the data better than the ones above. If so, you need to make a careful description of it and report your results in detail.)

Deadline: May 26th, 2021.
Requirement: You must do this project in MATLAB or R. For this project there are 35 points available. To qualify for bonus points you need to score at least 10 points. After that, every 2.5 points you score on this project will translate into 0.5 bonus points on the exam.
Formalities: You may work alone but you are strongly encouraged to work in pairs. Write your project report as a single pdf preferably in LaTeX. You can also use MATLAB's LiveEditor or R's Rmarkdown and export your report as a single pdf or HTML document. It should include all plots, explanations, and answers to the questions as well as your implemented code in an appendix. If you do not write your report in LaTeX, it is acceptable to scan your handwritten solutions to the theoretical parts of the project and include them in the pdf file. Upload this report in Canvas. Reports without code will not be graded and the code should be structured and include comments that make it readable. Your report will be subject to a plagiarism review with Urkund.

Appendix: Some useful MATLAB[3] functions and commands in no particular order:

- *readtable* - Import csv file as table.
- *arima* - Used to specify an ARMA model. Note that you should set the differencing degree D to 0.
- *garch* - Used to specify a GARCH model.
- *estimate* - Used for the model fitting.
- *aicbic* - Calculate the AIC and BIC of a given fit.
- *infer* - Computes the residuals of an ARMA model or the conditional variances of a GARCH model.
- *forecast* - Computes a forecast of the conditional variance of a GARCH model.

[3]Version R2020b.

See next page!

- *makedist* - Used to specify a probability distribution object. For a t-distribution with x degrees of freedom, write `tdist =makedist('tLocationScale','nu',x);`.
- *cell2mat* - Converts cell arrays into matrices/vectors.
- *X.Distribution* - Returns the noise distribution of the GARCH model *X* as a structure array.
- *norminv* - Used to find critical values of the normal distribution.
- *tinv* - Used to find critical values of the t-distribution.
- *qqplot* - Creates a qqplot. Note that you can specify the target distribution yourself.
- *[i,j] = find(M == min(min(M)))* - Returns the indices *[i,j]* of the minimal value of the matrix *M*.
- *autocorr* - Computes the sample ACF.
- *parcorr* - Computes the sample PACF.
- *try* and *catch* - Useful for catching errors thrown by estimate in loops.
- *optimoptions* - Set optimization parameters.