Project 2: Calculating Values at Risk

Financial Time Series
Course Code: TMS088/MSA410

Devosmita Chatterjee & Siddhant Som

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

$$VaR_t = V_{t-1}(\exp(F_{X_t}^{-1}(p)) - 1) \tag{1}$$

for $t \geq 2$.

Solution:

The cumulative distribution function of X is given by

$$F_X(x) = P(X \leq x) = q \ \ for \ \ q \in [0,1]. \tag{2}$$

The inverse of the cumulative distribution function of X is given by

$$F_X^{-1}(q) = x \ \ for \ \ q \in [0,1]. \tag{3}$$

Given that $V = (V_t,\, t \geq 1)$ is a stochastic process, the log returns of X is

$$X_t = \log(V_t) - \log(V_{t-1}) \ \ for \ \ t \geq 2$$

$$\Rightarrow X_t = \log(\frac{V_t}{V_{t-1}}) \ \ for \ \ t \geq 2. \tag{4}$$

Given that the cumulative distribution function of X_t given all information upto time $t-1$ is

$$F_{X_t} = P(X_t \leq .|(X_s, V_s, s < t)). \tag{5}$$

If $a > 1$, then the logarithmic functions are monotone increasing functions,

$$log_a(x) > log_a(z) \ \ for \ \ x > z. \tag{6}$$

Now,

$$P(V_t - V_{t-1} \leq VaR_t|(V_s, s < t)) = p \ \ for \ \ p \in [0,1] \ \ (\because \ Given)$$

$$\Rightarrow P(\frac{V_t}{V_{t-1}} \leq \frac{VaR_t + V_{t-1}}{V_{t-1}}|(V_s, s < t)) = p$$

$$\Rightarrow P(\log(\frac{V_t}{V_{t-1}}) \leq \log(\frac{VaR_t + V_{t-1}}{V_{t-1}})|(V_s, s < t)) = p \ \ (\because \ Using \ \ (6))$$

$$\Rightarrow P(X_t \leq \log(\frac{VaR_t + V_{t-1}}{V_{t-1}})|(X_s, V_s, s < t)) = p \ \ (\because \ Using \ \ (4))$$

$$\Rightarrow F_{X_t}(\log(\frac{VaR_t + V_{t-1}}{V_{t-1}})) = P(X_t \leq \log(\frac{VaR_t + V_{t-1}}{V_{t-1}})|(X_s, V_s, s < t)) = p \ \ (\because \ Using \ \ (2))$$

$$\Rightarrow F_{X_t}^{-1}(p) = \log(\frac{VaR_t + V_{t-1}}{V_{t-1}}) \ \ (\because \ Using \ \ (3))$$

$$\Rightarrow \exp(F_{X_t}^{-1}(p)) = \frac{VaR_t + V_{t-1}}{V_{t-1}}$$

$$\Rightarrow VaR_t = V_{t-1}(\exp(F_{X_t}^{-1}(p)) - 1) \ \ for \ \ t \geq 2.$$

Hence proved.

2. Plot the sample autocorrelation and partial autocorrelation functions of X for lags $h = 0, \cdots, 50$. 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 modelled as white noise?

Solution:

The time series V where $V = (V_t,\ t = 1, \cdots, N)$ are the data points in *Close* is plotted in figure 1. For the given data *dat_intel.csv*, $N = 1751$.

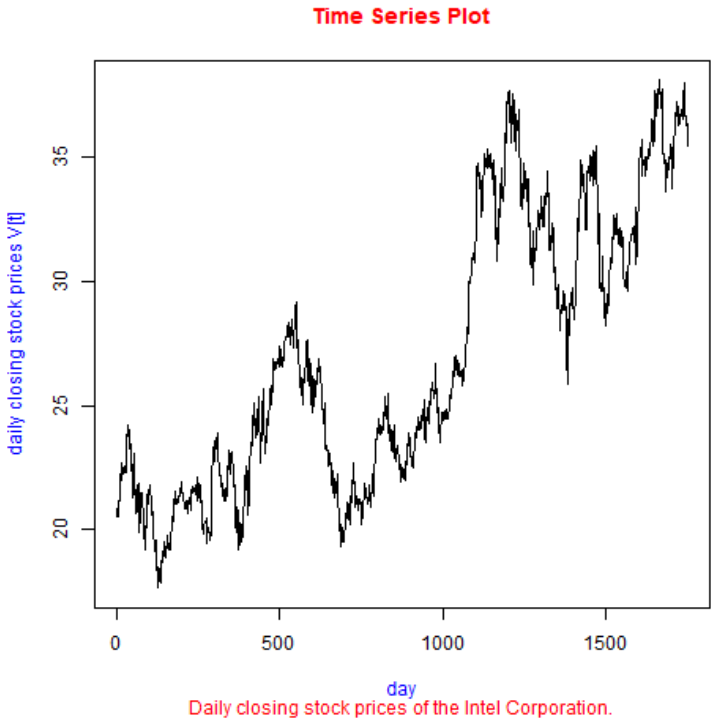


Figure. 1 Time series V where $V = (V_t,\ t = 1, \cdots, 1751)$ are the data points in *Close*.

The time series X where $X = (X_t,\ t = 2, \cdots, N)$ is plotted in figure 2. Here $X_t = \log V_t - \log V_{t-1}$ where $(V_t,\ t = 1, \cdots, N)$ are the data points in *Close*. For the given data *dat_intel.csv*, $N = 1751$. The log returns $X_t = \log(V_t) - \log(V_{t-1})$ for $t = 2, \ldots, 1751$ is computed and thus, the trend is removed.

↳ Not asked.

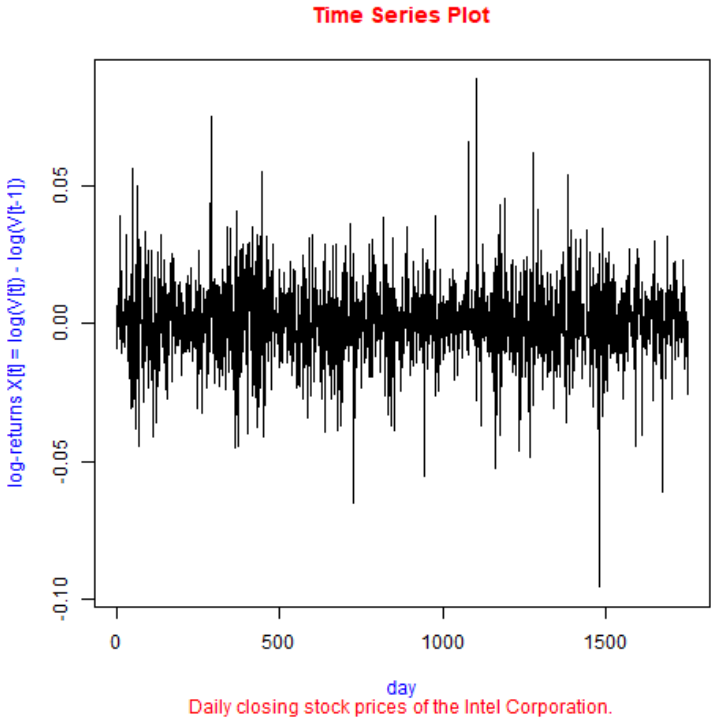


Figure. 2 Time series X where $X = (X_t,\, t = 2, \cdots, 1751)$. Here log returns $X_t = \log V_t - \log X_{t-1}$ where $(V_t,\, t = 1, \cdots, 1751)$ are the data points in *Close*.

The sample autocorrelation function (ACF) of the log returns $\rho_X(\mathrm{h})$ for $h = 1, \ldots, 50$ is plotted in figure 3. The partial autocorrelation function (PACF) of the log returns $\rho_X(\mathrm{h})$ for $h = 1, \ldots, 50$ is plotted in figure 4. In the ACF & PACF plots, the blue dashed lines or bounds represent an approximate 95% confidence interval which is given by C.I. $= \pm 1.96/\sqrt{N}$. Around 84% of the ACF values and around 86% of the PACF values are within these bounds $\pm 1.96/\sqrt{N}$ which is not consistent with white noise.

To check for IID, we use the Ljung-Box test:

$$H_0 : X \sim IID(\mu_1, \sigma_1^2)$$
$$H_1 : X \not\sim IID(\mu_1, \sigma_1^2)$$

Using the Ljung-Box test with lag $h = 50$, we get a p-value of 0.03068. We reject the null hypothesis at significance level $\alpha = 0.05$ and conclude that the log returns data is not consistent being IID.

Therefore, the log returns data should not be modeled as white noise.

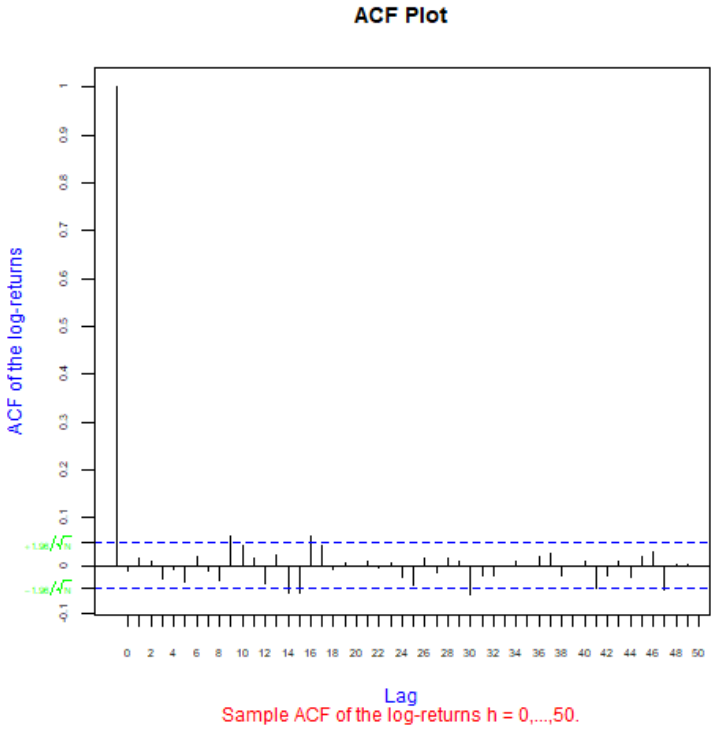


Figure. 3 Sample autocorrelation function (ACF) of the log returns $\rho_X(\mathrm{h})$ for $h = 0, \cdots, 50$.

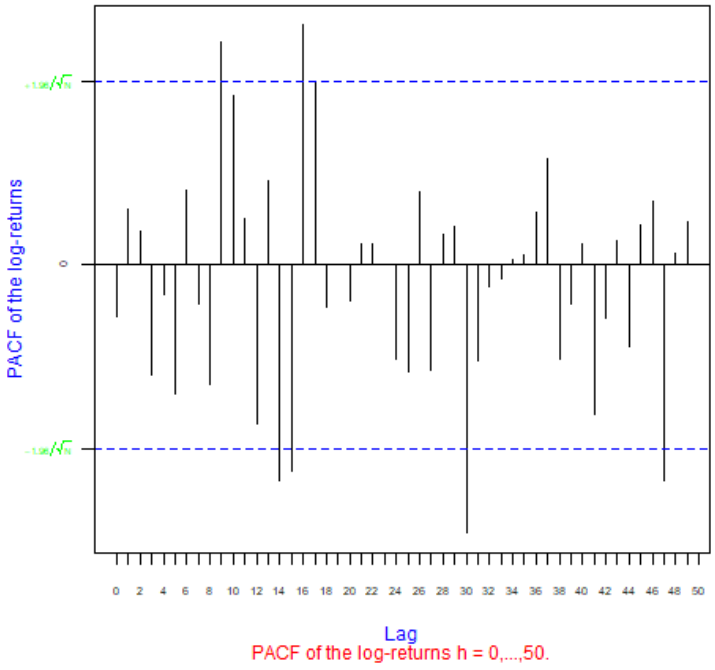


Figure. 4 Partial autocorrelation function (PACF) of the log returns $\rho_X(\mathrm{h})$ for $h = 0, \cdots, 50$.

The squared log returns X^2 for $t = 2, \ldots, 1751$ is computed. The sample autocorrelation function (ACF) of the log returns $\rho_{X^2}(\mathrm{h})$ for $h = 1, \ldots, 50$ is plotted in figure

5. The partial function (PACF) of the log returns $\rho_{X^2}(\mathrm{h})$ for $h = 1, \ldots, 50$ is plotted in figure 6.

In the ACF & PACF plots, the blue dashed lines or bounds represent an approximate 95% confidence interval which is given by C.I. $= \pm 1.96/\sqrt{N}$. Around 88% of the ACF values and 90% of the PACF values are within these bounds $\pm 1.96/\sqrt{N}$ which is fairly consistent with white noise. iid

To check for IID, we use the Ljung-Box test:

$$H_0 : X^2 \sim IID(\mu_2, \sigma_2^2)$$
$$H_1 : X^2 \not\sim IID(\mu_2, \sigma_2^2)$$

Using the Ljung-Box test with lag $h = 50$, we get a p-value of 0.02694. We reject the null hypothesis at significance level $\alpha = 0.05$ and conclude that the squared log returns data is not consistent being IID.

Therefore, the squared log returns data should be modeled as white noise. iid

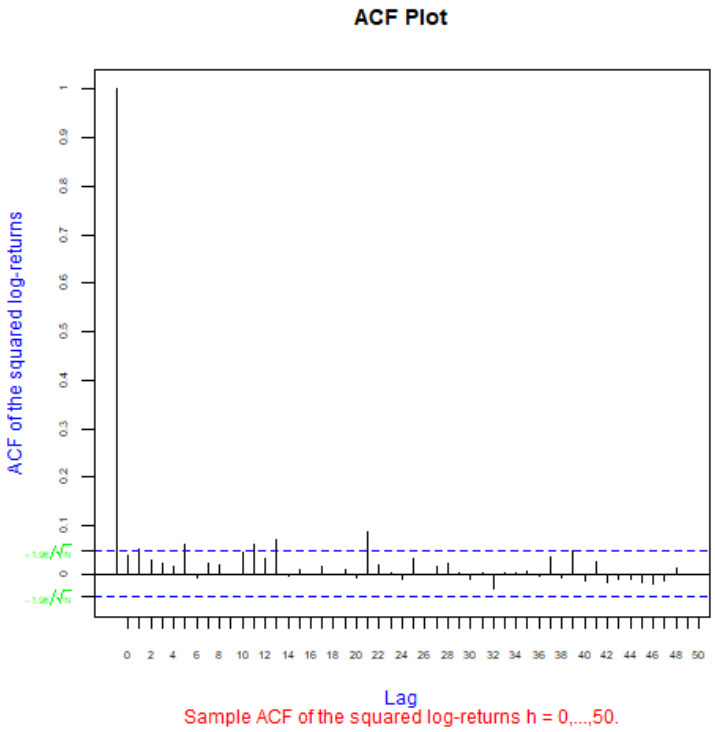


Figure. 5 Sample autocorrelation function (ACF) of the squared log returns $\rho_{X^2}(\mathrm{h})$ for $h = 0, \cdots, 50$.

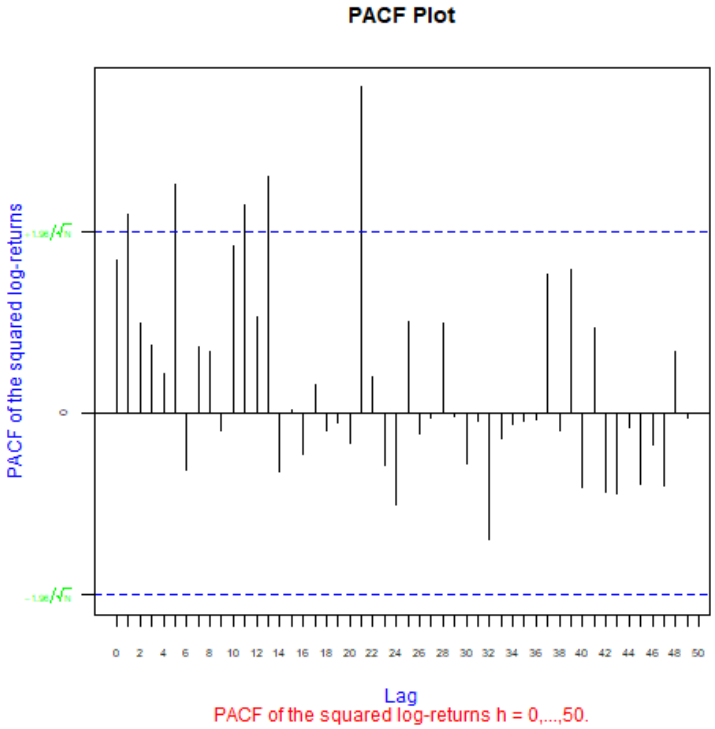


Figure. 6 Partial autocorrelation function (PACF) of the squared log returns $\rho_{X^2}(\mathrm{h})$ for $h = 0, \cdots, 50$.

Doesn't this contradicts your last conclusion.

Since the squared log returns is modeled as white noise which has constant variance, we first try to fit an ARMA model for the squared log returns.

Particularly, we see in the ACF & PACF plots for the log returns that the ACF values go up and down which suggests that it has a non-constant conditional variance. Therefore, it is a good idea to try a GARCH model for the log returns.

The full R code can be found in appendix A.

3. Now split the time series X into a *training set* (X_t, t = 2,.....1526) and a *test set* (X_t, t = 1527,$\cdots$,1751). For all values of p and q in the range 0,1,$\cdots$,15 and such that $p+q > 0$ and $p \geq q$, fit an ARMA(p, q) process with Gaussian noise to the *square* returns obtained from the training data and compute the BIC and the AICC. Which models minimise 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.

Solution:

We split the time series X into a *training set* (X_t, t = 2,.....1526) and a *test set* (X_t, t = 1527,$\cdots$,1751).

For all values of p and q in the range 0,1,$\cdots$,15 such that $p+q > 0$ and $p \geq q$, we fit an ARMA(p, q) process with Gaussian noise to the squared log returns obtained from the training data, resulting in 135 models, and compute the BIC and the AICC. Out of 135 models, ARMA(1,1) minimises BIC given by

p	q	AICC	BIC
1	1	-12.31068	-12.2967

and ARMA(10,8) minimises AICC given by

p	q	AICC	BIC
10	8	-12.31597	-12.24607

For all values of p and q in the range 0,1,$\cdots$,15 such that $p+q > 0$ and $p \geq q$, we fit an ARMA(p, q) process with t-distributed noise to the squared log returns obtained from the training data, and compute the BIC and the AICC. ARMA(2,2) minimises BIC given by

p	q	AICC	BIC
2	2	-13.8146	-13.79013

and ARMA(14,8) minimises AICC given by

p	q	AICC	BIC
14	8	-13.85261	-13.76524

We perform Ljung–Box test on the standardized residuals of ARMA(1,1) model with Gaussian noise:

$$H_0 : Z \sim IID(0, \sigma^2)$$
$$H_1 : Z \not\sim IID(0, \sigma^2)$$

Using the Ljung-Box test with lag $h = 50$, we get a p-value of 0.696. We fail to reject the null hypothesis at significance level $\alpha = 0.05$ and conclude that the standardized residuals is consistent being IID. Since an IID noise is a white noise, the standardized residuals should be modeled as white noise. The result is expected from an ARMA

process which is by definition 3.2.3 of the lecture notes:
A time series X is an ARMA(p, q) process if X is stationary and if for all $t \in \mathbb{Z}$,

$$X_t - \sum_{j=1}^{p} \phi_j X_{t-j} = Z_t + \sum_{j=1}^{q} \theta_j Z_{t-j}$$

where $Z \sim \text{WN}(0,\sigma^2)$.

We perform Ljung–Box test on the standardized residuals of ARMA(10,8) model with Gaussian noise:

$$H_0 : Z \sim IID(0, \sigma^2)$$
$$H_1 : Z \not\sim IID(0, \sigma^2)$$

Using the Ljung-Box test with lag $h = 50$, we get a p-value of 0.9938. We fail to reject the null hypothesis at significance level $\alpha = 0.05$ and conclude that the standardized residuals is consistent being IID. Since an IID noise is a white noise, the standardized residuals should be modeled as white noise. The result is expected from an ARMA process which is by definition 3.2.3 of the lecture notes.

We perform Ljung–Box test on the standardized residuals of ARMA(14,8) model with t-distributed noise:

$$H_0 : Z \sim IID(0, \sigma^2)$$
$$H_1 : Z \not\sim IID(0, \sigma^2)$$

Using the Ljung-Box test with lag $h = 50$, we get a p-value of 0.759. We fail to reject the null hypothesis at significance level $\alpha = 0.05$ and conclude that the standardized residuals is consistent being IID. Since an IID noise is a white noise, the standardized residuals should be modeled as white noise. The result is expected from an ARMA process which is by definition 3.2.3 of the lecture notes.

We perform Ljung–Box test on the standardized residuals of ARMA(2,2) model with t-distributed noise:

$$H_0 : Z \sim IID(0, \sigma^2)$$
$$H_1 : Z \not\sim IID(0, \sigma^2)$$

Using the Ljung-Box test with lag $h = 50$, we get a p-value of 0.5458. We fail to reject the null hypothesis at significance level $\alpha = 0.05$ and conclude that the standardized residuals is consistent being IID. Since an IID noise is a white noise, the standardized residuals should be modeled as white noise. The result is expected from an ARMA process which is by definition 3.2.3 of the lecture notes.

From the obtained ARMA models, we deduce the orders of GARCH models based on the proposition 4.1.5 of the lecture notes that if $(X_t^2,\ t \in \mathbb{Z})$ is an ARMA($max\{p, q\}$,q) process, then $(X_t,\ t \in \mathbb{Z})$ is a GARCH(p,q) process. Based on our obtained ARMA models, we derive the order for the following GARCH models:

Distribution of noise	ARMA($max\{p,q\}$,q)	GARCH(p,q)
Gaussian	ARMA(1,1)	GARCH(0,1) or GARCH(1,1)
Gaussian	ARMA(10,8)	GARCH(10,8)
t-distributed	ARMA(2,2)	GARCH(0,2) or GARCH(1,2) or GARCH(2,2)
t-distributed	ARMA(14,8)	GARCH(14,8)

OK

The full R code can be found in appendix B.

4. 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 $0 < p \leq$ K and $0 \leq q \leq K$, fit a GARCH (p, q) process driven by Gaussian noise to the training data and compute the BIC and AICC. Which models minimize the two information criteria? Write these candidate models down including the assumptions on the noise.

Solution:

The maximal order of the ARMA models with Gaussian noise fitted in the previous task is $K = \max(10,8) = 10$.

For all values of p in the range 0,1,$\cdots$,10 and q in the range 0,1,$\cdots$,10 such that $p+q > 0$ and $p \geq q$, we fit an GARCH(p, q) process with Gaussian noise to the log returns obtained from the training data, and compute the BIC and the AICC. GARCH(1,1) minimises both the information criteria - AICC and BIC given by

p	q	AICC	BIC
1	1	-5.579184	-5.565204

Based on the lecture notes (page 65), a GARCH(1, 1) process yields a sufficient model, without the need for higher orders. The minimum model is the GARCH(1,1) model

$$X_t = \sigma_t Z_t, \;\; Z_t \sim IID(0,1)$$

$$\sigma_t^2 = \alpha_0 + \alpha_1 X_{t-1}^2 + \beta_1 \sigma_{t-1}^2.$$

Even when the maximal order of the ARMA models with Gaussian noise is K = max(1,1) = 1, we get the same GARCH(1,1) model.

The full R code can be found in appendix C.

5. 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?

Solution:

Using the GARCH(1,1) model in the previous task 4, we compute the residuals as follows

$$\hat{R}_t = (X_t - \hat{X}_t)/\sqrt{\hat{\sigma}_t}.$$

If the GARCH(1,1) model is correct, we would expect that the distribution of the residuals are IID and Gaussian with mean 0 and variance 1 and the covariance structure of the residuals will be an identity matrix.

Based on page 19 of lecture notes, if the residuals are IID random variables with finite variance, then the autocorrelations $\hat{\rho}(h), 1, 2, \ldots$ for sufficiently large N will be approximately IID and have a normal distribution with N^{-1} variance. Thus, 95% of the ACF values should lie within the bounds $\pm 1.96/\sqrt{N}$ which is drawn in the ACF plots.

The sample autocorrelation function (ACF) of the residuals ρ_Z(h) for $h = 1, \ldots, 50$ is plotted in figure 7. The autocorrelation of the residuals has around 86% of the values within the bounds $\pm 1.96/\sqrt{N}$.

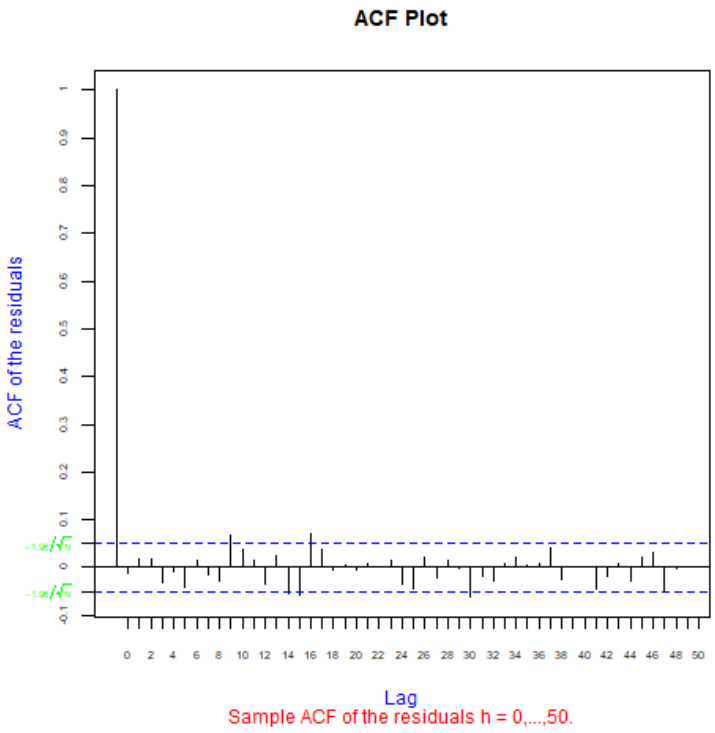


Figure. 7 Sample autocorrelation function (ACF) of the residuals ρ_Z(h) for $h = 0, \cdots, 50$.

The sample autocorrelation function (ACF) of the squared residuals ρ_{Z^2}(h) for $h =$

$1, \ldots, 50$ is plotted in figure 8. The autocorrelation of the squared residuals has around 88% of the values within $\pm 1.96/\sqrt{N}$. ← Interpretation?

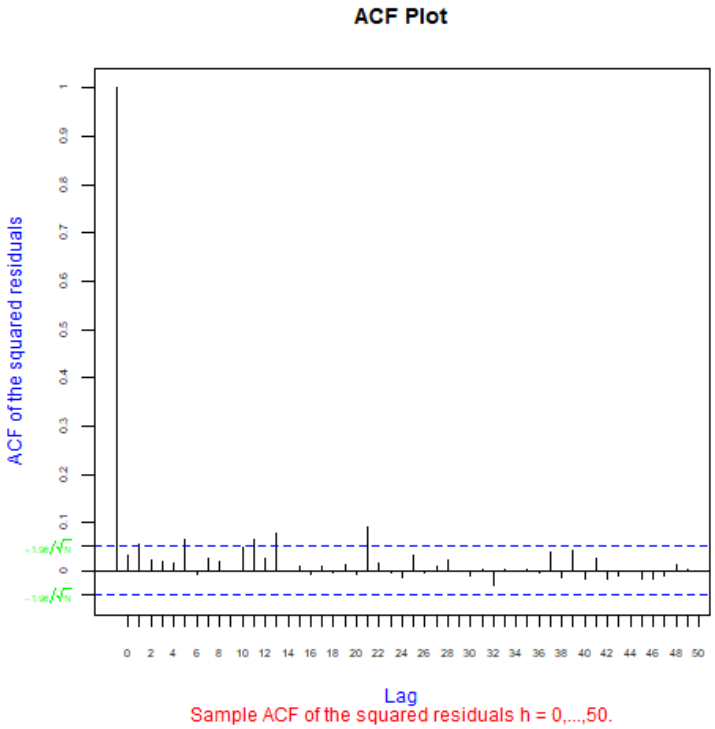


Figure. 8 Sample autocorrelation function (ACF) of the squared residuals $\rho_{Z^2}(\mathrm{h})$ for $h = 0, \cdots, 50$.

The QQ-plot of the residuals is plotted in figure 9.

In the QQ-plot of the residuals, we see that there is some deviation from the standard normal distribution. We have heavier tails than we would expect from a standard normal distribution. Based on the QQ-plot, it is a good idea to investigate whether a Student's t-distribution would fit the data better which allows for heavier tails. Ok

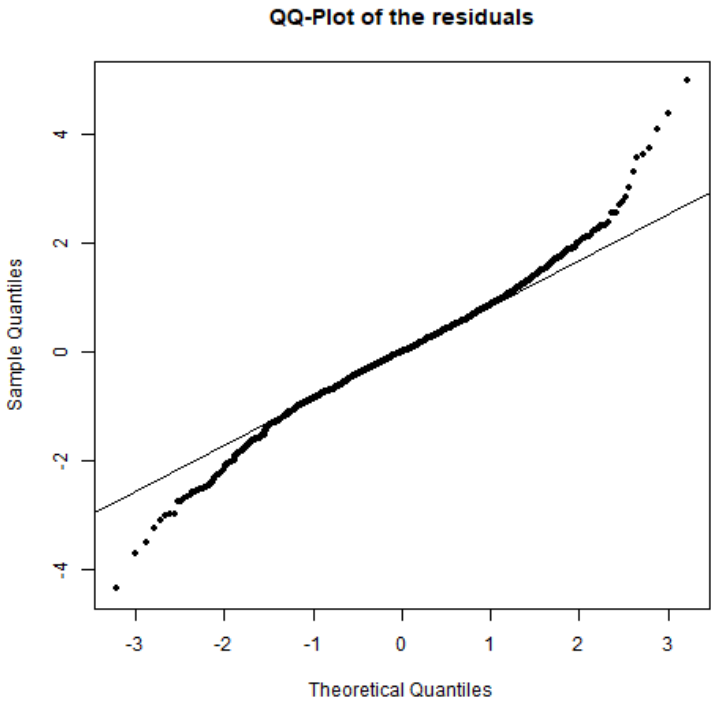


Figure. 9 QQ-plot of the residuals.

The orders of the GARCH models obtained are coherent with the ARMA models obtained in task 3 as we had already shown in task 3 that based on the orders of the ARMA models, GARCH (1,1) is a possible candidate model, which minimizes both the AICC and BIC scores.

The full R code can be found in appendix D.

6. 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.

Solution:

The maximal order of the ARMA models with Gaussian noise fitted in the previous task is $K = \max(14,8) = 14$.

For all values of p in the range 0,1,$\cdots$,14 and q in the range 0,1,$\cdots$,14 such that $p + q > 0$ and $p \geq q$, we fit an GARCH(p, q) process with t-distributed noise to the log returns obtained from the training data, and compute the BIC and the AICC. GARCH(1,1) minimises both the information criteria - AICC and BIC given by

p	q	AICC	BIC
1	1	-5.655583	-5.638108

Based on the lecture notes (page 65), a GARCH(1, 1) process yields a sufficient model, without the need for higher orders. The minimum model is the GARCH(1,1) model

$$X_t = \sigma_t Z_t, \;\; Z_t \sim IIDN(0,1)$$

$$\sigma_t^2 = \alpha_0 + \alpha_1 X_{t-1}^2 + \beta_1 \sigma_{t-1}^2.$$

Even when the maximal order of the ARMA models with Gaussian noise is $K = \max(2,2) = 2$, we get the same GARCH(1,1) model.

Using the GARCH(1,1) model in the task, we compute the residuals as follows

$$\hat{R}_t = (X_t - \hat{X}_t)/\sqrt{\hat{\sigma}_t}.$$

If the GARCH(1,1) model is correct, we would expect that the distribution of the residuals are IID and Gaussian with mean 0 and variance 1 and the covariance structure of the residuals will be an identity matrix.

Based on page 19 of lecture notes, if the residuals are IID random variables with finite variance, then the autocorrelations $\hat{\rho}(h), 1, 2, \ldots$ for sufficiently large N will be approximately IID and have a normal distribution with N^{-1} variance. Thus, 95% of the ACF values should lie within the bounds $\pm 1.96/\sqrt{N}$ which is drawn in the ACF plots.

The sample autocorrelation function (ACF) of the residuals ρ_Z(h) for $h = 1, \ldots, 50$ is plotted in figure 10. The autocorrelation of the residuals has around 86% of the values within the bounds $\pm 1.96/\sqrt{N}$.

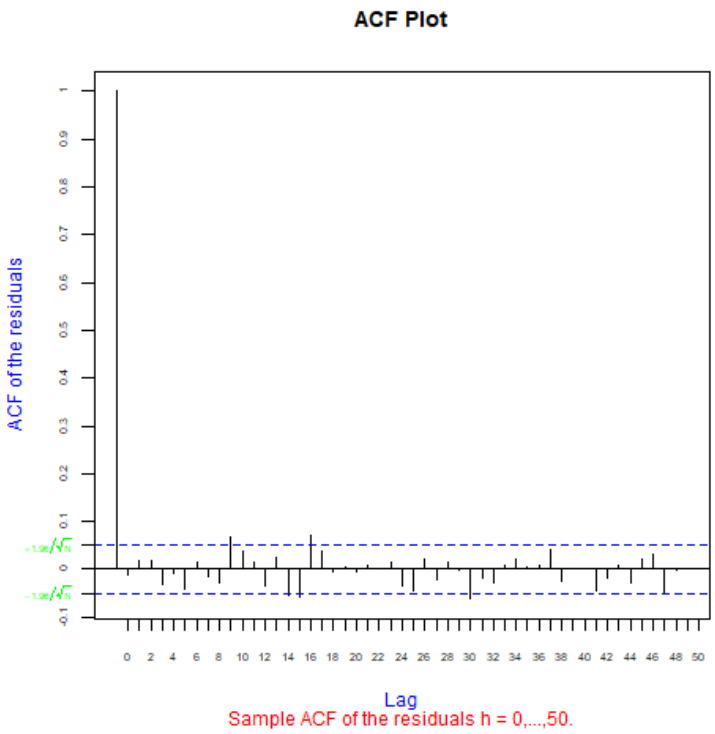


Figure. 10 Sample autocorrelation function (ACF) of the residuals ρ_Z(h) for $h = 0, \cdots, 50$.

The sample autocorrelation function (ACF) of the squared residuals ρ_{Z^2}(h) for $h = 1, \ldots, 50$ is plotted in figure 11. The autocorrelation of the squared residuals has around 88% of the values within $\pm 1.96/\sqrt{N}$.

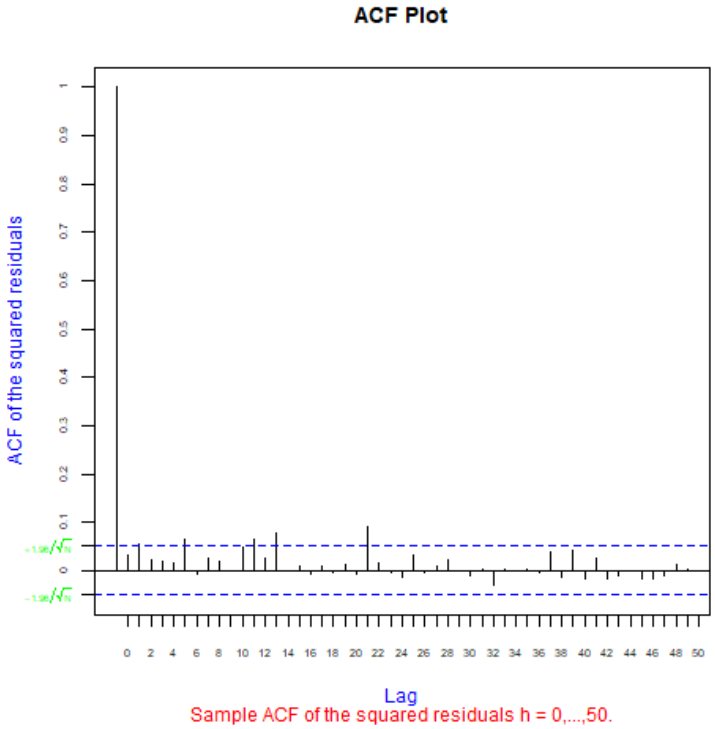
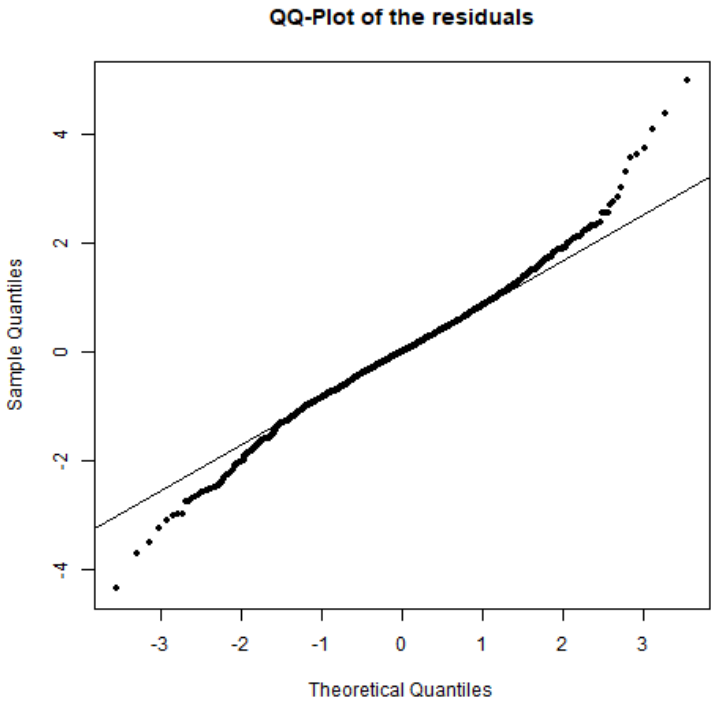


Figure. 11 Sample autocorrelation function (ACF) of the squared residuals $\rho_{Z^2}(\mathrm{h})$ for $h = 0, \cdots, 50$.

The QQ-plot of the residuals is plotted in figure 12.

In this QQ-plot of the residuals, we investigate a Student's t-distribution which give slightly better results. In what sense?



Wrong QQ plot
I presume.
The residual
should be
ε - distributed.

Figure. 12 QQ-plot of the residuals.

The full R code can be found in appendix E.

7. Estimate VaR_t and evaluate the estimates for t = 1527,.....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,....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,...X_{t-1}) to update the conditional variances as needed.
- Count the number of VaR breaches i.e for t = 1577,....1751, count the number of times that V_t - $V_{t-1} \leq \text{VaR}_t$.

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,.....X_{t-1} and count the number of breaches as above. 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?

Solution:

tab. 1 GARCH(p,q) with Gaussian noise obtained from task 3.

p	q	Number of breaches for different confidence levels		
		0.01	0.05	0.1
0	1	90	94	98
1	1	90	94	98
10	8	90	94	98

Wrong orders of magnitude → Code problems

tab. 2 GARCH(p,q) with t-distributed noise obtained from task 3.

p	q	Number of breaches for different confidence levels		
		0.01	0.05	0.1
0	2	90	94	98
1	2	90	94	98
2	2	90	94	98
14	8	90	94	98

tab. 3 GARCH(p,q) with Gaussian noise obtained from task 4.

p	q	Number of breaches for different confidence levels		
		0.01	0.05	0.1
1	1	90	94	98

tab. 4 GARCH(p,q) with t-distributed noise obtained from task 6.

p	q	Number of breaches for different confidence levels		
		0.01	0.05	0.1
1	1	98	98	98

Based on the outcome of this experiment, GARCH(1,1) model with t-distributed noise is recommended for the estimation of VaR_t for this stock because the number of breaches for different confidence levels for this model is maximum.

The full R code can be found in appendix F.

Not what we look for!

You should have compared the proportion of breaches to p.

Next Time, export the Rmd file directly! We only accept one submission file per group so I can't consider other files by fairness towards the other students.

Appendix A

Read the given data dat_intel.csv.

```
data = read.csv("intel.csv",header = TRUE)
```

Compute and plot the time series X where X[t] are the data points in Close.

```
V = data[,2]
plot(V, type = "l", main = "Time Series Plot", sub = expression("Daily closing stock prices of the Intel Corporation."), xlab = "day", ylab = "daily closing stock prices V[t]", col.main = "red", col.sub = "red", col.lab = "blue")
```

Compute and plot the differenced log time series X[t] = log(V[t]) - log(V[t-1]) where X[t] are the data points in Close.

```
V = data[,2]
X = matrix(, nrow = length(V)-1, ncol = 1)
for (i in 2:length(V))
(
X[i-1] = log(V[i]) - log(V[i-1])
)
plot(X, type = "l", main = "Time Series Plot", sub = expression("Daily closing stock prices of the Intel Corporation."), xlab = "day", ylab = "log-returns X[t] = log(V[t]) - log(V[t-1])", col.main = "red", col.sub = "red", col.lab = "blue")
```

OK

Compute the sample autocorrelation function (ACF) of the log-returns.

```
ACF = acf(X, lag = 50, type = "correlation", na.action = na.pass, plot = FALSE) n = length(X)
```

Plot the sample autocorrelation function (ACF) of the log-returns.

```
plot(ACF, xaxt = 'no', main = "ACF Plot", sub = expression("Sample ACF of the log-returns h = 0,...,50."), ylab = "ACF of the log-returns", xlab = "Lag", col.sub = "red", col.lab = "blue", xaxt = 'n', yaxt = 'n')
axis(1, at = seq(1,51,1), labels = seq(0,50,1), cex.axis = 0.6)
axis(2, at = seq(-1,1,0.1), labels = seq(-1,1,0.1), cex.axis = 0.6) par(las = 2)
axis(2, at = 1.96/sqrt(n), labels = expression(+1.96/sqrt(N)), cex.axis = 0.5, col.axis = "green")
axis(2, at = -1.96/sqrt(n), labels = expression(-1.96/sqrt(N)), cex.axis = 0.5, col.axis = "green")
```

Compute whether autocorrelations are significant.

```
sum = 0
for (i in 1:51)
(
if (ACFacf[i] > −1.96/sqrt(n)ACFacf[i] < 1.96/sqrt(n))
(
sum = sum + 1
)
end
)
end
print((sum/51)*100)
```

Compute the partial autocorrelation function (PACF) of the log-returns.
PACF = acf(X, lag = 50, type = "partial", na.action = na.pass, plot = FALSE)
Plot the partial autocorrelation function (PACF) of the log-returns.
plot(PACF, xaxt = 'no', main = "", sub = expression("PACF of the log-returns h = 0,...,50."), ylab = "PACF of the log-returns", xlab = "Lag", col.sub = "red", col.lab = "blue", xaxt = 'n', yaxt = 'n')
axis(1, at = seq(1,51,1), labels = seq(0,50,1), cex.axis = 0.6)
axis(2, at = seq(-1,1,0.1), labels = seq(-1,1,0.1), cex.axis = 0.6) par(las = 2)
axis(2, at = 1.96/sqrt(n), labels = expression(+1.96/sqrt(N)), cex.axis = 0.5, col.axis = "green")
axis(2, at = -1.96/sqrt(n), labels = expression(-1.96/sqrt(N)), cex.axis = 0.5, col.axis = "green")
Compute whether partial autocorrelations are significant.
sum2 = 0
for (i in 1:50)
(
if (PACFacf[i] > −1.96/sqrt(n)PACFacf[i] < 1.96/sqrt(n))
(
sum2 = sum2 + 1
)
end
)
end
*print((sum2/50)*100)*

LjungBoxTest ¡- Box.test(X, lag=50, type="Ljung")
LjungBoxTest
qchisq(.95, df=50)
qchisq(.96, df=50)
qchisq(.9694, df=50)
qchisq(.97, df=50)
qchisq(.98, df=50)
qchisq(.99, df=50)

Compute the sample autocorrelation function (ACF) of the squared log-returns.
ACF2 = acf(X2, lag = 50, type = "correlation", na.action = na.pass, plot = FALSE)
n2 = length(X2)
Plot the sample autocorrelation function (ACF) of the squared log-returns.

plot(ACF2, xaxt = 'no', main = "ACF Plot", sub = expression("Sample ACF of the squared log-returns h = 0,...,50."), ylab = "ACF of the squared log-returns", xlab = "Lag", col.sub = "red", col.lab = "blue", xaxt = 'n', yaxt = 'n')
axis(1, at = seq(1,51,1), labels = seq(0,50,1), cex.axis = 0.6)
axis(2, at = seq(-1,1,0.1), labels = seq(-1,1,0.1), cex.axis = 0.6) par(las = 2)
axis(2, at = 1.96/sqrt(n2), labels = expression(+1.96/sqrt(N)), cex.axis = 0.5, col.axis = "green")

axis(2, at = -1.96/sqrt(n2), labels = expression(-1.96/sqrt(N)), cex.axis = 0.5, col.axis = "green")

Compute whether autocorrelations are significant.

sum3 = 0
for (i in 1:51)
(
if (ACF2acf[i] > −1.96/sqrt(n2)ACF2acf[i] ¡ 1.96/sqrt(n2))
(sum3 = sum3 + 1
) end
)
end
*print((sum3/51)*100)*

Compute the partial autocorrelation function (PACF) of the squared log-returns.

PACF2 = acf(X2, lag = 50, type = "partial", na.action = na.pass, plot = FALSE)

Plot the partial autocorrelation function (PACF) of the squared log-returns.

plot(PACF2, xaxt = 'no', main = "PACF Plot", sub = expression("PACF of the squared log-returns h = 0,...,50."), ylab = "PACF of the squared log-returns", xlab = "Lag", col.sub = "red", col.lab = "blue", xaxt = 'n', yaxt = 'n')
axis(1, at = seq(1,51,1), labels = seq(0,50,1), cex.axis = 0.6)
axis(2, at = seq(-1,1,0.1), labels = seq(-1,1,0.1), cex.axis = 0.6) par(las = 2)
axis(2, at = 1.96/sqrt(n2), labels = expression(+1.96/sqrt(N)), cex.axis = 0.5, col.axis = "green")
axis(2, at = -1.96/sqrt(n2), labels = expression(-1.96/sqrt(N)), cex.axis = 0.5, col.axis = "green")

Compute whether partial autocorrelations are significant.

sum4 = 0
for (i in 1:50)
(
if (PACF2acf[i] > −1.96/sqrt(n2)PACF2acf[i] ¡ 1.96/sqrt(n2))
(sum4 = sum4 + 1
) end
)
end
*print((sum4/50)*100)*

$LjungBoxTest2 \text{ ¡- } Box.test(X^2, lag = 50, type = "Ljung")$
$LjungBoxTest2$
$qchisq(.95, df = 50)$
$qchisq(.96, df = 50)$
$qchisq(.97, df = 50)$
$qchisq(.98, df = 50)$
$qchisq(.99, df = 50)$

Appendix B

Split the time series X into a training set and a test set.
trainingSet = X[1:1525,1]
testSet = X[1526:1750,1]
install.packages("rugarch")
library("rugarch")
Fit an ARMA model with Gaussian noise to the squared log returns obtained from the training data.
df ¡- data.frame(p = integer(), q = integer(), AICC = double(), BIC = double())

Fit the model you just specified to the training dataset.
GF=arfimafit(spec=GFSpec,data = trainingSet^2, solver = "nloptr", solver.control = list($xtol_rel = 1e-6, ftol_rel = 1e-6$, solver = 3))
print(GF)

AICC = infocriteria(GF)[1]
BIC = infocriteria(GF)[2]
df[i,] ¡- data.frame(p, q, AICC, BIC)
i = i+1
)))i = 1
for (p in 0:15)
(
for (q in 0:15)
(
if (p+q ¿ 0 p ¿= q)
*(**Specify the ARMA(p,q) model you want to fit on your data using the function arfimaspec.***
GFSpec=arfimaspec(mean.model = list(armaOrder=c(p,q),include.mean=TRUE,arfima=FALSE))

Fit the model you just specified to the training dataset.
GF=arfimafit(spec=GFSpec,data = trainingSet^2, solver = "nloptr", solver.control = list($xtol_rel = 1e-6, ftol_rel = 1e-6$, solver = 3))
print(GF)

AICC = infocriteria(GF)[1]
BIC = infocriteria(GF)[2]
df[i,] ¡- data.frame(p, q, AICC, BIC)
i = i+1
))) df[which(df[,3] df[which(df[,4] **Perform Ljung–Box tests on the standardized residuals of ARMA(1,1) model with Gaussian noise.**

StandardizedResiduals = stdize(as.numeric(residuals(GF)[,1]))**use MuMIn package**
LjungBoxTest = Box.test(StandardizedResiduals, lag=50, type="Ljung")
LjungBoxTest
qchisq(.95, df=50)
qchisq(.96, df=50)

qchisq(.97, df=50)
qchisq(.98, df=50)
qchisq(.99, df=50) *GFSpec=arfimaspec(mean.model = list(armaOrder=c(1,1),include.mean=TRUE,arfima*
GF = arfimafit(spec=GFSpec,data = trainingSet^2*, solver = "nloptr", solver.control =*
list(xtol$_r$*el* $= 1e-6, ftol_r el = 1e-6, solver = 3$))

StandardizedResiduals = stdize(as.numeric(residuals(GF)[,1])) ***use MuMIn package***
LjungBoxTest = Box.test(StandardizedResiduals, lag=50, type="Ljung")
LjungBoxTest
qchisq(.95, df=50)
qchisq(.96, df=50)
qchisq(.97, df=50)
qchisq(.98, df=50)
qchisq(.99, df=50)

Perform Ljung–Box tests on the standardized residuals of ARMA(10,8) model with Gaussian noise.

StandardizedResiduals = stdize(as.numeric(residuals(GF)[,1])) ***use MuMIn package***
LjungBoxTest = Box.test(StandardizedResiduals, lag=50, type="Ljung")
LjungBoxTest
qchisq(.95, df=50)
qchisq(.96, df=50)
qchisq(.97, df=50)
qchisq(.98, df=50)
qchisq(.99, df=50) GFSpec=arfimaspec(mean.model = list(armaOrder=c(10,8),include.mean=TRUE,arfim
GF = arfimafit(spec=GFSpec,data = (trainingSet)^2*, solver = "nloptr", solver.control =*
list(xtol$_r$*el* $= 1e-6, ftol_r el = 1e-6, solver = 3$))

StandardizedResiduals = stdize(as.numeric(residuals(GF)[,1])) ***use MuMIn package***
LjungBoxTest = Box.test(StandardizedResiduals, lag=50, type="Ljung")
LjungBoxTest
qchisq(.95, df=50)
qchisq(.96, df=50)
qchisq(.97, df=50)
qchisq(.98, df=50)
qchisq(.99, df=50)

Fit an ARMA model with t-distribution noise to the squared log returns obtained from the training data.
df2 = data.frame(p2 = integer(), q2 = integer(), AICC2 = double(), BIC2 = double())
i2 = 1

tryCatch(

where is it defined ?

```
(
```
Fit the model you just specified to the training dataset.
```
GF2 = arfimafit(spec=GFSpec2,data = trainingSet2,solver="nloptr",solver.control =
list(xtolrel=1e-6,ftolrel=1e-6,solver=3))
AICC2 = infocriteria(GF2)[1]
BIC2 = infocriteria(GF2)[2]
df2[i2,] ¡- data.frame(p2, q2, AICC2, BIC2)
i2 = i2+1
),
    error=function(errormessage)
(
print('Model Not Valid')
return(NULL)
))
)
)
for (p2 in 0:15)
(
for (q2 in 0:15)
(
if (p2+q2 > 0 and p2 >= q2)
(
```
Specify the ARMA(p,q) model you want to fit on your data using the function arfimaspec.
```
GFSpec2=arfimaspec(mean.model = list(armaOrder=c(p2,q2),include.mean=TRUE,arfima=FALSE),
distribution.model = "std")

    tryCatch(

    (
```
Fit the model you just specified to the training dataset.
```
GF2 = arfimafit(spec=GFSpec2,data = trainingSet2,solver="nloptr",solver.control =
list(xtolrel=1e-6,ftolrel=1e-6,solver=3))
AICC2 = infocriteria(GF2)[1]
BIC2 = infocriteria(GF2)[2]
df2[i2,] ¡- data.frame(p2, q2, AICC2, BIC2)
i2 = i2+1
),
    error=function(errormessage)
(
print('Model Not Valid')
return(NULL)
))
)
)
df2[which(df2[,3] in min(df2[,3])),]  minimise AICC
df2[which(df2[,4] in min(df2[,4])),]  minimise BIC
```

Perform Ljung–Box tests on the standardized residuals of ARMA(14,8) model with t distributed noise.

GFSpec2=arfimaspec(mean.model = list(armaOrder=c(14,8),include.mean=TRUE,arfima=FALSE), distribution.model = "std")

GF2 = arfimafit(spec=GFSpec2,data = trainingSet^2*, solver = "nloptr", solver.control = list(*$xtol_rel = 1e-6, ftol_rel = 1e-6, solver = 3$*))*

StandardizedResiduals = stdize(as.numeric(residuals(GF2)[,1]))

use MuMIn package

LjungBoxTest3 = Box.test(StandardizedResiduals, lag=50, type="Ljung")

LjungBoxTest3

qchisq(.95, df=50)

qchisq(.96, df=50)

qchisq(.97, df=50)

qchisq(.98, df=50)

qchisq(.99, df=50)

Perform Ljung–Box tests on the standardized residuals of ARMA(2,2) model with t distributed noise.

GFSpec2=arfimaspec(mean.model = list(armaOrder=c(2,2),include.mean=TRUE,arfima=FALSE), distribution.model = "std")

GF2 = arfimafit(spec=GFSpec2,data = trainingSet^2*, solver = "nloptr", solver.control = list(*$xtol_rel = 1e-6, ftol_rel = 1e-6, solver = 3$*))*

StandardizedResiduals2 = stdize(as.numeric(residuals(GF2)[,1]))LjungBoxTest4 = Box.test(Standardized

I give up going too much into the details of this code, since it is very hard read...

Appendix C

Fit an GARCH model with Gaussian noise to the log returns.

Fit the model you just specified to the training dataset.
GF3 = ugarchfit(spec = GFSpec3, data = trainingSet, solver="nloptr", solver.control = list(xtol_rel = 1e − 6, ftol_rel = 1e − 6, solver = 3))
print(GF3)

AICC3 = infocriteria(GF3)[1]
BIC3 = infocriteria(GF3)[2]
df3[i3,] ¡- data.frame(p3, q3, AICC3, BIC3)
i3 = i3+1
)))df3 = data.frame(p3 = integer(), q3 = integer(), AICC3 = double(), BIC3 = double())
i3 = 1
K = 10
for (p3 in 0:K)
(
for (q3 in 0:K)
(
if (p3+q3 > 0 and p3 >= q3)
(
Specify the GARCH(p,q) model you want to fit on your data using the function ugarchspec.
GFSpec3 = ugarchspec(variance.model = list(model="sGARCH",garchOrder=c(p3,q3)), mean.model = list(armaOrder=c(0,0),include.mean=TRUE))

Fit the model you just specified to the training dataset.
GF3 = ugarchfit(spec = GFSpec3, data = trainingSet, solver="nloptr", solver.control = list(xtol_rel = 1e − 6, ftol_rel = 1e − 6, solver = 3))
print(GF3)

AICC3 = infocriteria(GF3)[1]
BIC3 = infocriteria(GF3)[2]
df3[i3,] ¡- data.frame(p3, q3, AICC3, BIC3)
i3 = i3+1
))) df3[which(df3[,3] in min(df3[,3])),] ***minimise AICC***
df3[which(df3[,4] in min(df3[,4])),] ***minimise BIC***
Fit an GARCH model with Gaussian noise to the log returns.

Fit the model you just specified to the training dataset.
GF31 = ugarchfit(spec = GFSpec31, data = trainingSet, solver="nloptr", solver.control = list(xtol_rel = 1e − 6, ftol_rel = 1e − 6, solver = 3))
print(GF31)

AICC31 = infocriteria(GF31)[1]
BIC31 = infocriteria(GF31)[2]

df31[i31,] ¡- data.frame(p31, q31, AICC31, BIC31)
i31 = i31+1
)
)
)df31 = data.frame(p31 = integer(), q31 = integer(), AICC31 = double(), BIC31 = double()) i31 = 1
K1 = 1
for (p31 in 0:K1)
(
for (q31 in 0:K1)
(
if (p31+q31 > 0 and p31 >= q31)
(
Specify the GARCH(p,q) model you want to fit on your data using the function ugarchspec.
GFSpec31 = ugarchspec(variance.model = list(model="sGARCH",garchOrder=c(p31,q31)), mean.model = list(armaOrder=c(0,0),include.mean=TRUE))

Fit the model you just specified to the training dataset.
GF31 = ugarchfit(spec = GFSpec31, data = trainingSet, solver="nloptr", solver.control = list(xtol$_r$*el* $= 1e-6, ftol_rel = 1e-6, solver = 3)$)
print$(GF31)$

AICC31 = infocriteria(GF31)[1]
BIC31 = infocriteria(GF31)[2]
df31[i31,] ¡- data.frame(p31, q31, AICC31, BIC31)
i31 = i31+1
)
)
)
df31[which(df31[,3] in min(df31[,3])),] ***minimise AICC***
df31[which(df31[,4] in min(df31[,4])),] ***minimise BIC***

‘‘‘

Appendix D

Residuals of GARCH(1,1) model with Gaussian noise.

GFSpec3 = ugarchspec(variance.model = list(model="sGARCH",garchOrder=c(1,1)), mean.model = list(armaOrder=c(0,0),include.mean=TRUE))

GF3 = ugarchfit(spec = GFSpec3, data = trainingSet, solver="nloptr", solver.control = list(xtol_rel = 1e − 6, ftol_rel = 1e − 6, solver = 3))

$StandardizedResiduals3 = stdize(as.numeric(residuals(GF3)[, 1]))$

Compute the sample autocorrelation function (ACF) of the standardized residuals.

ACF3 = acf(StandardizedResiduals3, lag = 50, type = "correlation", na.action = na.pass, plot = FALSE

Plot the sample autocorrelation function (ACF) of the standardized residuals.

plot(ACF3, xaxt = 'no', main = "ACF Plot", sub = expression("Sample ACF of the residuals h = 0,...,50."), ylab = "ACF of the residuals", xlab = "Lag", col.sub = "red", col.lab = "blue", xaxt = 'n', yaxt = 'n')

axis(1, at = seq(1,51,1), labels = seq(0,50,1), cex.axis = 0.6)

axis(2, at = seq(-1,1,0.1), labels = seq(-1,1,0.1), cex.axis = 0.6)

par(las = 2)

axis(2, at = 1.96/sqrt(n3), labels = expression(+1.96/sqrt(N)), cex.axis = 0.5, col.axis = "green")

axis(2, at = -1.96/sqrt(n3), labels = expression(-1.96/sqrt(N)), cex.axis = 0.5, col.axis = "green")

Compute whether autocorrelations are significant.

```
sum5 = 0
for (i in 1:51)
(
if (ACF3acf[i] > -1.96/sqrt(n3) and ACF3acf[i] < 1.96/sqrt(n3))
(
sum5 = sum5 + 1
)
end
)
end
print((sum5/51)*100)
```

Compute the sample autocorrelation function (ACF) of the squared standardized residuals.

$ACF4 = acf(StandardizedResiduals3^2, lag = 50, type = "correlation", na.action = na.pass, plot = FALSE)$

$n4 = length(StandardizedResiduals3^2)$

Plot the sample autocorrelation function (ACF) of the squared standardized residuals.

plot(ACF4, xaxt = 'no', main = "ACF Plot", sub = expression("Sample ACF of the squared residuals h

Compute whether autocorrelations are significant.

```
sum6 = 0
for (i in 1:51)
(if (ACF4acf[i] > -1.96/sqrt(n4) and ACF4acf[i] < 1.96/sqrt(n4))
(sum6 = sum6 + 1) end) endprint((sum6/51) * 100)
qqplotfunc=function(Xdat,distrQuant,...)npts=length(Xdat)
qvec=seq(from=0,to=1,length.out = npts)[-c(1,npts)]
```

Plot theoretical quantile VS sample quantiles

$plot(distrQuant(qvec, ...), quantile(Xdat, qvec),$
$main=paste0("QQ\text{-}Plot\ of\ the\ residuals"),\ xlab="Theoretical\ Quantiles",ylab="Sample\ Quantiles",pch=$
$qqplotfunc(Xdat=StandardizedResiduals3,distrQuant=qnorm)$

Appendix E

Fit an GARCH model with t distributed noise to the log returns.

```
df4 = data.frame(p4 = integer(), q4 = integer(), AICC4 = double(), BIC4 = double())
i4 = 1
K2 = 14

    tryCatch(

    (
```

Fit the model you just specified to the training dataset

```
GF4 = ugarchfit(spec = GFSpec4, data = trainingSet, solver="nloptr", solver.control
= list(xtol_rel = 1e − 6, ftol_rel = 1e − 6, solver = 3))
AICC4 = infocriteria(GF4)[1]
BIC4 = infocriteria(GF4)[2]
df4[i4, ] < −data.frame(p4, q4, AICC4, BIC4)
i4 = i4 + 1
),

    error=function(error_message)(
print('ModelNotValid')
return(NULL)
))
)))for(p4in0 : K2)
(
for(q4in0 : K2)
(
if(p4 + q4 > 0p4 >= q4)
(
```

Specify the GARCH(p,q) model you want to fit on your data using the function ugarchsp

```
GFSpec4 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder =
c(p4, q4)), mean.model = list(armaOrder = c(0, 0), include.mean = TRUE), distribution.model =
"std")

    tryCatch(

    (
```

Fit the model you just specified to the training dataset

```
GF4 = ugarchfit(spec = GFSpec4, data = trainingSet, solver="nloptr", solver.control
= list(xtol_rel = 1e − 6, ftol_rel = 1e − 6, solver = 3))
AICC4 = infocriteria(GF4)[1]
BIC4 = infocriteria(GF4)[2]
df4[i4, ] < −data.frame(p4, q4, AICC4, BIC4)
i4 = i4 + 1
),

    error=function(error_message)(
print('ModelNotValid')
```

return(NULL)
))
)))
df4[which(df4[,3] in min(df4[,3])),] ***minimise AICC****df4[which(df4[,4] in min(df4[,4])),]* ***minimise B***
df41 = data.frame(p41 = integer(), q41 = integer(), AICC41 = double(), BIC41 = double())i41 = 1K21

tryCatch(

(
Fit the model you just specified to the training dataset
GF41 = ugarchfit(spec = GFSpec41, data = trainingSet, solver="nloptr", solver.control
= list(xtol$_r$*el* $= 1e-6, ftol_rel = 1e-6, solver = 3)$)
$AICC41 = infocriteria(GF41)[1]$
$BIC41 = infocriteria(GF41)[2]$
$df41[i41,] < -data.frame(p41, q41, AICC41, BIC41)$
$i41 = i41 + 1$
),

error=function(error$_m$*essage)(*
$print('ModelNotValid')$
$return(NULL)$
)))))
df41[which(df41[,3] in min(df41[,3])),] ***minimise AICC****df41[which(df41[,4] in min(df41[,4])),]* ***minim***

Residuals of GARCH(1,1) model with t distributed noise.
GFSpec4 = ugarchspec(variance.model = list(model="sGARCH",garchOrder=c(1,1)),
mean.model = list(armaOrder=c(0,0),include.mean=TRUE), distribution.model = "std")

StandardizedResiduals4 = stdize(as.numeric(residuals(GF4)[,1]))GF4 = ugarchfit(spec
= GFSpec4, data = trainingSet, solver="nloptr", solver.control = list(xtol$_r$*el* $= 1e-$
$6, ftol_rel = 1e-6, solver = 3)$)

StandardizedResiduals4 = stdize(as.numeric(residuals(GF4)[,1]))
Compute the sample autocorrelation function (ACF) of the log-returns.
ACF5 = acf(StandardizedResiduals4, lag = 50, type = "correlation", na.action = na.pass,
plot = FALSE)
n5 = length(StandardizedResiduals4)
Plot the sample autocorrelation function (ACF) of the log-returns.
plot(ACF5, xaxt = 'no', main = "ACF Plot", sub = expression("Sample ACF of the
residuals h = 0,...,50."), ylab = "ACF of the residuals", xlab = "Lag", col.sub = "red",
col.lab = "blue", xaxt = 'n', yaxt = 'n')
axis(1, at = seq(1,51,1), labels = seq(0,50,1), cex.axis = 0.6)
axis(2, at = seq(-1,1,0.1), labels = seq(-1,1,0.1), cex.axis = 0.6)
par(las = 2)
axis(2, at = 1.96/sqrt(n5), labels = expression(+1.96/sqrt(N)), cex.axis = 0.5, col.axis
= "green")
axis(2, at = -1.96/sqrt(n5), labels = expression(-1.96/sqrt(N)), cex.axis = 0.5, col.axis
= "green")

Compute whether autocorrelations are significant.

```
sum7 = 0
for (i in 1:51)
(
if (ACF5acf[i] > −1.96/sqrt(n5)ACF5acf[i] ¡ 1.96/sqrt(n5))
(
sum7 = sum7 + 1
)
end
)
end
print((sum7/51)*100)
```

Compute the sample autocorrelation function (ACF) of the log-returns.

$ACF6 = acf(StandardizedResiduals4^2, lag = 50, type = "correlation", na.action = na.pass, plot = FALSE)$

$n6 = length(StandardizedResiduals4^2)$

Plot the sample autocorrelation function (ACF) of the log-returns.

```
plot(ACF6, xaxt = 'no', main = "ACF Plot", sub = expression("Sample ACF of the
squared residuals h = 0,...,50."), ylab = "ACF of the squared residuals", xlab = "Lag",
col.sub = "red", col.lab = "blue", xaxt = 'n', yaxt = 'n')
axis(1, at = seq(1,51,1), labels = seq(0,50,1), cex.axis = 0.6)
axis(2, at = seq(-1,1,0.1), labels = seq(-1,1,0.1), cex.axis = 0.6)
par(las = 2)
axis(2, at = 1.96/sqrt(n6), labels = expression(+1.96/sqrt(N)), cex.axis = 0.5, col.axis
= "green")
axis(2, at = -1.96/sqrt(n6), labels = expression(-1.96/sqrt(N)), cex.axis = 0.5, col.axis
= "green")
```

Compute whether autocorrelations are significant.

```
sum8 = 0
for (i in 1:51)
(
if (ACF6acf[i] > −1.96/sqrt(n6)andACF6acf[i] ¡ 1.96/sqrt(n6))
(
sum8 = sum8 + 1
)
end
)
end
print((sum8/51)*100)
qqplotfunc(Xdat=StandardizedResiduals4,distrQuant=qt,df=30)
```

Appendix F

VaR estimation for GARCH(1,1) model with Gaussian noise.

```
(
garch_spec = ugarchspec(variance.model = list(model = "sGARCH", garchOrder =
c(1,1)), mean.model = list(armaOrder = c(0,0), include.mean = FALSE), distribution.model =
"norm")
ugfit = ugarchfit(spec = garch_spec, data = X[1:t-1], solver = "nloptr", solver.control =
list(xtol_rel = 1e-6, ftol_rel = 1e-6, solver = 3))forecast = ugarchforecast(ugfit, 1)
valueatrisk = VaR(R = X[1:t-1], pvalue, sigma = sigma(forecast))
if(V[t] - V[t-1] <= valueatrisk)(ind = ind + 1))
,
    error=function(error_message)(
return(NULL)
)))print(pvalue)
```

count the number of breaches

```
print(ind)
)for(pvalueinc(0.01, 0.05, 0.1))
(
ind = 0
for(tin1526 : 1750)
(
tryCatch(
```

We specifically told you NOT to do this.

```
(
garch_spec = ugarchspec(variance.model = list(model = "sGARCH", garchOrder =
c(1,1)), mean.model = list(armaOrder = c(0,0), include.mean = FALSE), distribution.model =
"norm")
ugfit = ugarchfit(spec = garch_spec, data = X[1:t-1], solver = "nloptr", solver.control =
list(xtol_rel = 1e-6, ftol_rel = 1e-6, solver = 3))forecast = ugarchforecast(ugfit, 1)
valueatrisk = VaR(R = X[1:t-1], pvalue, sigma = sigma(forecast))
if(V[t] - V[t-1] <= valueatrisk)(ind = ind + 1))
,
    error=function(error_message)(
return(NULL)
)))print(pvalue)
```

count the number of breaches

```
print(ind)
)
```

VaR estimation for GARCH(1,1) model with t-distributed noise.

```
(
garch_spec = ugarchspec(variance.model = list(model = "sGARCH", garchOrder =
c(1,1)), mean.model = list(armaOrder = c(0,0), include.mean = FALSE), distribution.model =
"std")
ugfit = ugarchfit(spec = garch_spec, data = X[1:t-1], solver = "nloptr", solver.control =
list(xtol_rel = 1e-6, ftol_rel = 1e-6, solver = 3))forecast = ugarchforecast(ugfit, 1)
```

$valueatrisk = VaR(R = X[1 : t-1], pvalue, sigma = sigma(forecast))$
$if(V[t] - V[t-1] <= valueatrisk)$
(
$ind = ind + 1$
)
)
,
$error$=$function(error_message)$(
$return(NULL)$
))
$)print(pvalue)$
count the number of breaches
$print(ind)$
)

Simple model?