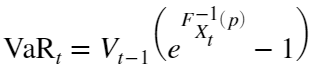
# Mattias Berg, Colton Cunov

# Project 2: Calculating Values at Risk

## 1.

Let the log returns  be given by . Let  be the cumulative distribution function of  given all information up to time . Show that

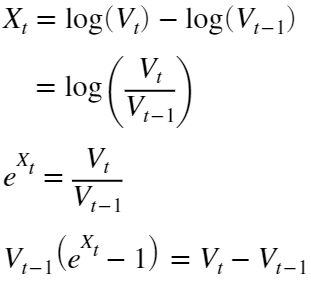


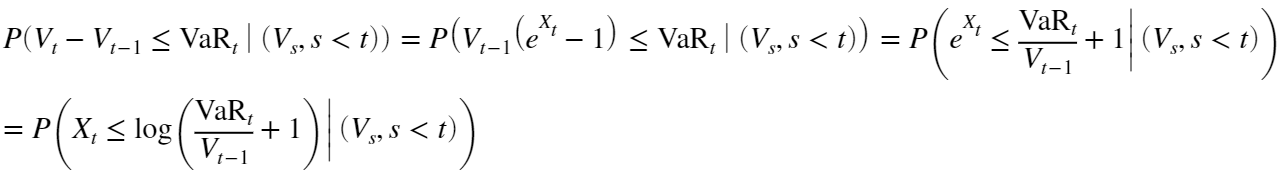


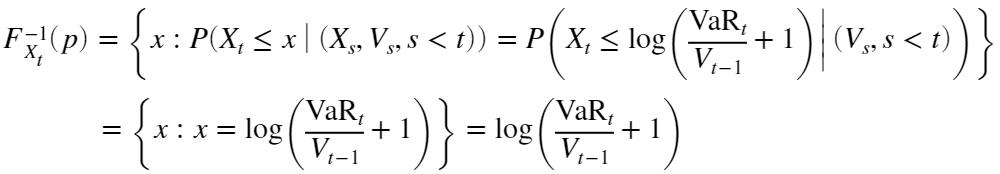


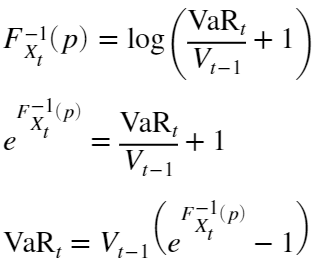


Above, is read as " such that  holds"









## 2.

Plot the (sample) autocorrelation and partial autocorrelation functions of X for lags h = 0,...,50

close all

V = readtable('dat\_intel.csv');

X = log(V.Close(2:end)) - log(V.Close(1:end-1));

subplot(1,2,1);

autocorr(X,'NumLags',50);

title('X');

ylim([-.2, 1]);

xlim([0, 50]);

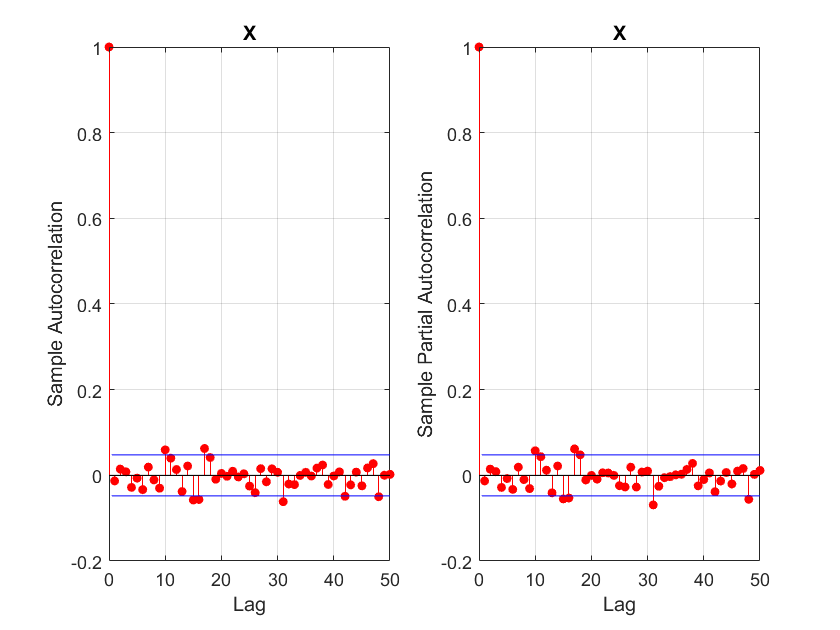
subplot(1,2,2);

parcorr(X,'NumLags',50);

title('X');

ylim([-.2, 1]);

xlim([0, 50]);



No large sized lags, but their still more than 5% of the lags that are outside the boundaries, indicating that the data is not only noise.

Then, do a Ljung–Box test with h = 50 for X

alpha = .05;

lags = 1:50;

close all

[h, pValues, stat, cValue] = lbqtest(X, 'lags', lags);

c = (pValues < .05)' .\* [1 0 0];

scatter(lags', pValues', 15, c, 'filled');

hold on

plot(lags, pValues, 'k--')

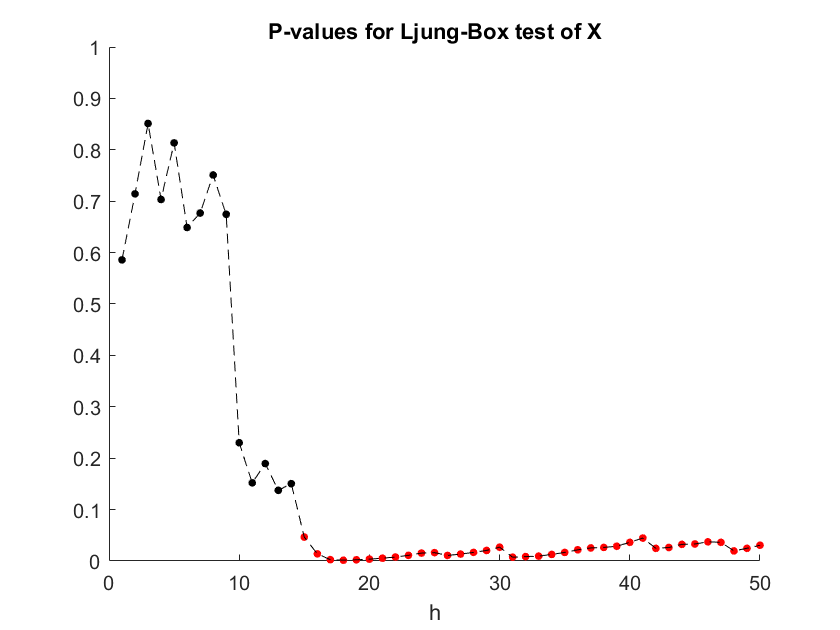
hold off

ylim([0, 1]);

xlim([0,50]);

title('P-values for Ljung-Box test of X');

xlabel('h');



if pValues(find(lags==50)) < alpha

disp('Reject H0 at h=50')

else

disp('Fail to reject H0 at h=50')

end

Reject H0 at h=50

Repeat these two steps with the squared returns . We start with the (sample) autocorrelation and partial autocorrelation functions of  for lags h = 0,...,50

subplot(1,2,1)

autocorr(X.^2,'NumLags',50)

title('X^2')

ylim([-.2, 1])

xlim([0, 50])

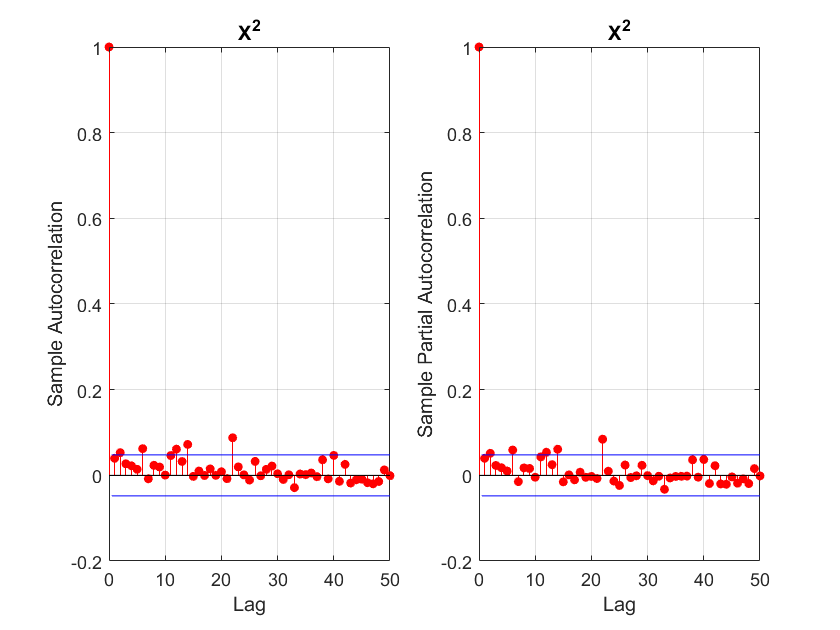
subplot(1,2,2)

parcorr(X.^2,'NumLags',50)

title('X^2')

ylim([-.2, 1])

xlim([0, 50])



We can see that their are some larger sized lags, and that there is more than 5% of the lag that are outside the boundaries, indicating that the data is not only noise.

do a Ljung–Box test with h = 50 for X^2

close all

[h, pValues, stat, cValue] = lbqtest(X.^2, 'lags', lags, 'Alpha', alpha);

c = (pValues < .05)' .\* [1 0 0];

scatter(lags', pValues', 15, c, 'filled');

hold on

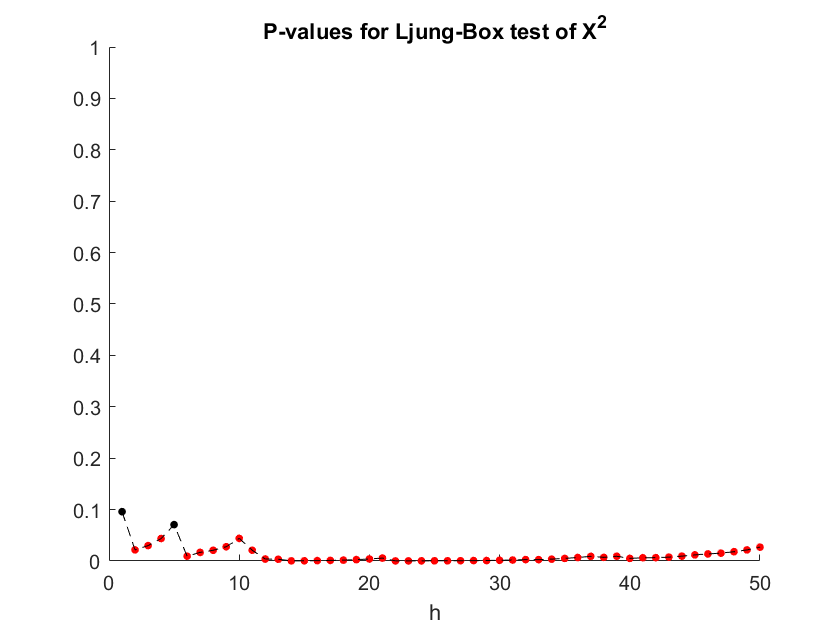
plot(lags,pValues,'k--')

hold off

ylim([0, 1])

title('P-values for Ljung-Box test of X^2')

xlabel('h')



if pValues(find(lags==50)) < alpha

disp('Reject H0 at h=50')

else

disp('Fail to reject H0 at h=50')

end

Reject H0 at h=50

### Discussion

Both  and  (the squared returns) have outliers on the autocorrelation function plots. For both  and  we reject the Ljung-Box null hypotheses that  and  are realizations of  at the 5% level for . This means we might be able to tease out relationships among the data, as they are likely not entirely white noise.

## 

## 3)

Now split the time series  into a training set ( , t = 2,...,1526) and a test set ( , t = 1527, . . . , 1751).

trainX = X(2:1526);

testX = X(1527:1750);

### 3. Gaussian distribution

For all values of p and q in the range 0, 1, . . . , 15 and such that p+q > 0 and p ≥ 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 (this will take a couple of minutes). Which models minimize the two information criteria?

i=1;

clear pq mdl logL numParam WN\_variance\_MLE EstMdl

pq = [];

for p = 0:15

for q = 0:15

if p+q>0 && p>=q

mdl(i) = arima(p, 0, q);

mdl(i).Constant = mean(trainX.^2);

pq = [pq, [p; q]];

i = i + 1;

end

end

end

p = pq(1,:);

q = pq(2,:);

Options = optimoptions(@fmincon,'ConstraintTolerance',1e-7,'Algorithm','interior-point');

for i = 1:length(mdl)

[EstMdl(i), ~, logL(i)] = estimate(mdl(i),trainX.^2,'Display','off','Options',Options);

results = summarize(EstMdl(i));

numParam(i) = sum(results.Table.Value ~= 0) - 1; % =M from hints (-1 to remove variance)

WN\_variance\_MLE(i) = results.VarianceTable.Value; % maximum likelihood estimate of white noise variance

end

AICC = calculateAICC(length(trainX), numParam, logL); % both defined at end of file

BIC = calculateBIC(length(trainX), p, q, WN\_variance\_MLE, trainX.^2, numParam, 'M');

i\_AICC = find(AICC == min(AICC));

i\_BIC = find(BIC == min(BIC));

disp(["Model which minimizes [BD]'s AICC:" mdl(i\_AICC).Description; "Model which minimizes [BD]'s BIC:" mdl(i\_BIC).Description])

"Model which minimizes [BD]'s AICC:" "ARIMA(2,0,2) Model (Gaussian Distribution)"

"Model which minimizes [BD]'s BIC:" "ARIMA(2,0,2) Model (Gaussian Distribution)"

best\_ARMA\_mdl\_Gauss\_AICC = EstMdl(i\_AICC);

best\_ARMA\_mdl\_Gauss\_BIC = EstMdl(i\_BIC);

max\_order\_Gauss = max([p([i\_AICC, i\_BIC]) q([i\_AICC, i\_BIC])]);

### 3. Student’s *t* distribution

i = 1;

clear pq mdl logL numParam WN\_variance\_MLE EstMdl

pq = [];

for p = 0:15

for q = 0:15

if p+q>0 && p>=q

mdl(i) = arima(p,0,q);

mdl(i).Constant = mean(trainX.^2);

mdl(i).Distribution="t";

pq = [pq, [p; q]];

i = i + 1;

end

end

end

p = pq(1,:);

q = pq(2,:);

Options = optimoptions(@fmincon,'ConstraintTolerance',1e-7,'Algorithm','interior-point');

for i = 1:length(mdl)

try

[EstMdl(i),~,logL(i)] = estimate(mdl(i),trainX.^2,'Display','off','Options', Options);

results = summarize(EstMdl(i));

numParam(i) = sum(results.Table.Value ~= 0) - 2; % = M from hints (-2 to remove DoF and variance)

WN\_variance\_MLE(i) = results.VarianceTable.Value; % maximum likelihood estimate of white noise variance

catch

disp("error")

disp(mdl(i).Description)

disp("")

end

end

AICC = calculateAICC(length(trainX), numParam, logL); % both defined at end of file

BIC = calculateBIC(length(trainX), p, q, WN\_variance\_MLE, trainX.^2, numParam, 'M');

i\_AICC = find(AICC == min(AICC));

i\_BIC = find(BIC == min(BIC));

disp(["Model which minimizes [BD]'s AICC:" mdl(i\_AICC).Description; "Model which minimizes [BD]'s BIC:" mdl(i\_BIC).Description])

"Model which minimizes [BD]'s AICC:" "ARIMA(3,0,2) Model (t Distribution)"

"Model which minimizes [BD]'s BIC:" "ARIMA(3,0,2) Model (t Distribution)"

best\_ARMA\_mdl\_t\_AICC = EstMdl(i\_AICC);

best\_ARMA\_mdl\_t\_BIC = EstMdl(i\_BIC);

max\_order\_t = max([p([i\_AICC, i\_BIC]) q([i\_AICC, i\_BIC])]);

[res\_Gauss\_AICC, v\_Gauss\_AICC] = infer(best\_ARMA\_mdl\_Gauss\_AICC, trainX.^2);

[res\_Gauss\_BIC, v\_Gauss\_BIC] = infer(best\_ARMA\_mdl\_Gauss\_BIC, trainX.^2);

[res\_t\_AICC, v\_t\_AICC] = infer(best\_ARMA\_mdl\_t\_AICC, trainX.^2);

[res\_t\_BIC, v\_t\_BIC] = infer(best\_ARMA\_mdl\_t\_BIC, trainX.^2);

res\_Gauss\_AICC = res\_Gauss\_AICC ./ sqrt(v\_Gauss\_AICC);

res\_Gauss\_BIC = res\_Gauss\_BIC ./ sqrt(v\_Gauss\_BIC);

res\_t\_AICC = res\_t\_AICC ./ sqrt(v\_t\_AICC);

res\_t\_BIC = res\_t\_BIC ./ sqrt(v\_t\_BIC);

lags = 1:50;

close all

set(gcf,'Position',[100, 100, 1000, 800])

subplot(2,2,1)

[h, pValues, stat, cValue] = lbqtest(res\_Gauss\_AICC, 'lags', lags);

c = (pValues < .05)' .\* [1 0 0];

scatter(lags', pValues', 15, c, 'filled')

hold on

plot(lags, pValues, 'k--')

hold off

ylim([0 1.1])

title('LB-test p-values for Gaussian-distributed AICC minimizer')

subplot(2,2,2)

[h, pValues, stat, cValue] = lbqtest(res\_Gauss\_BIC, 'lags', lags);

c = (pValues < .05)' .\* [1 0 0];

scatter(lags', pValues', 15, c, 'filled')

hold on

plot(lags, pValues, 'k--')

hold off

ylim([-0.1 1.1])

title('LB-test p-values for Gaussian-distributed BIC minimizer')

subplot(2,2,3)

[h, pValues, stat, cValue] = lbqtest(res\_t\_AICC, 'lags', lags);

c = (pValues < .05)' .\* [1 0 0];

scatter(lags', pValues', 15, c, 'filled')

hold on

plot(lags, pValues, 'k--')

hold off

ylim([0 1.1])

title('LB-test p-values for t-distributed AICC minimizer')

subplot(2,2,4)

[h, pValues, stat, cValue] = lbqtest(res\_t\_BIC, 'lags', lags);

c = (pValues < .05)' .\* [1 0 0];

scatter(lags', pValues', 15, c, 'filled')

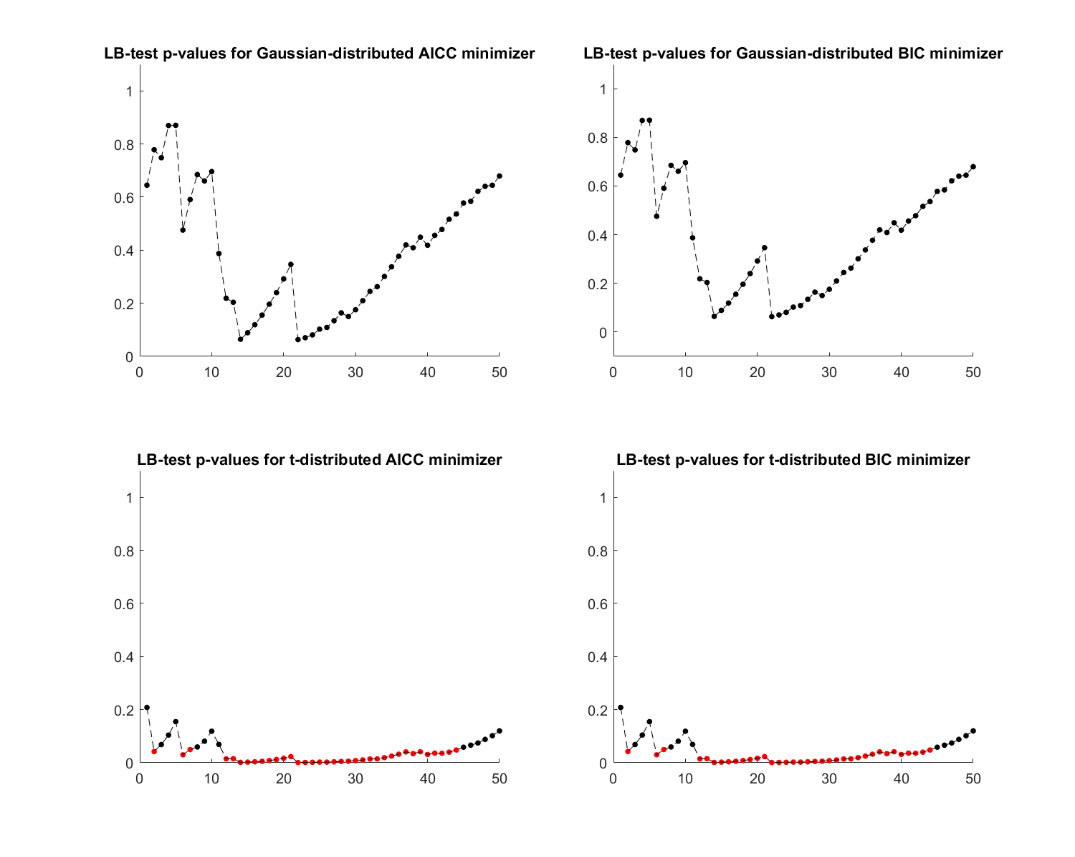
hold on

plot(lags, pValues, 'k--')

hold off

ylim([0 1.1])

title('LB-test p-values for t-distributed BIC minimizer')



There are no statistically significant p-values up to h=50 for the Gaussian distributed ARMA residuals, so we do not reject the null hypothesis that the residuals are IID noise.

For the t-distribution we have many statistically significant p-values, for each of which we reject the null hypothesis. This means our residuals for these lags are not entirely white noise, and we may be able to further tease out relationships.

Under the assumption that the GARCH process  a , so by looking in the reverse direction of this, on the four best ARMA models found, we can deduce four GARCH models that might be suitble model for the data;

Both the AICC and BIC for Gaussian models was minimized by **ARMA(2,2)**  **GARCH(1,2) or GARCH(2,2)**

Both the AICC and BIC for Student's t models was minimized by **ARMA(3,2)**  **GARCH(3,2)**

ARMA\_to\_GARCH\_Gauss\_params = struct('p',[2 2],'q',[2 1]);

ARMA\_to\_GARCH\_t\_params = struct('p',2,'q',3);

## 4)

Let be the maximal order of the ARMA models with Gaussian noise fitted in the previous task. For all values of  and  such that  and  fit a  process driven by Gaussian noise to the training data and compute the BIC and AICC (this will take a couple of minutes). Which models minimize the two information criteria? Write these candidate models down including the assumption on the noise.

clearvars EstMdl numParam\_garch logL\_garch

Options = optimoptions(@fmincon,'ConstraintTolerance',1e-7,'Algorithm','interior-point');

i = 1;

K = max\_order\_Gauss; %max(q,p) of the gaussian models in task 3

qp\_garch = [];

for p = 1:K

for q = 1:K

mdl\_temp = garch('GARCHLags',1:q,'ARCHLags',1:p,'Offset',mean(trainX.^2));

mdl\_temp.Offset = mean(trainX);

[EstMdl(i), ~, logL\_garch(i), ~] = estimate(mdl\_temp, trainX, 'Display', 'off', 'Options', Options);

results = summarize(EstMdl(i));

numParam\_garch(i) = sum(results.Table.Value ~= 0) - 1; % subtract the user-defined offset

qp\_garch = [qp\_garch, [q; p]];

i = i + 1;

end

end

q\_garch = qp\_garch(1,:);

p\_garch = qp\_garch(2,:);

AICC\_garch = calculateAICC(length(trainX), numParam\_garch, logL\_garch); % defined at end of file

BIC\_garch = -2\*logL\_garch + numParam\_garch\*log(length(trainX));

i\_AICC = find(AICC\_garch==min(AICC\_garch));

i\_BIC = find(BIC\_garch==min(BIC\_garch));

best\_GARCH\_mdl\_Gauss\_AICC = EstMdl(i\_AICC);

best\_GARCH\_mdl\_Gauss\_BIC = EstMdl(i\_BIC);

disp(["Model which minimizes [BD]'s AICC:" EstMdl(i\_AICC).Description; "Model which minimizes hint's BIC:" EstMdl(i\_BIC).Description])

"Model which minimizes [BD]'s AICC:" "GARCH(2,2) Conditional Variance Model with Offset (Gaussian Distrib…"

"Model which minimizes hint's BIC:" "GARCH(1,1) Conditional Variance Model with Offset (Gaussian Distrib…"

## 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?

close all

set(gcf,'Position',[100, 100, 1300, 800])

cond\_var\_AICC = infer(best\_GARCH\_mdl\_Gauss\_AICC, trainX);

cond\_var\_BIC = infer(best\_GARCH\_mdl\_Gauss\_BIC, trainX);

res\_AICC = (trainX - best\_GARCH\_mdl\_Gauss\_AICC.Offset) ./ sqrt(cond\_var\_AICC);

res\_BIC = (trainX - best\_GARCH\_mdl\_Gauss\_BIC.Offset) ./ sqrt(cond\_var\_BIC);

subplot(3,2,1)

[acf, lags, bounds] = autocorr(res\_AICC,50);

num\_significant = sum(abs(acf)-bounds(1) > 0);

autocorr(res\_AICC,50)

title(['Sample Autocorr for AICC-minimizing Gaussian model residuals [' num2str(num\_significant) ']'])

subplot(3,2,3)

[acf, lags, bounds] = autocorr(res\_AICC.^2,50);

num\_significant = sum(abs(acf)-bounds(1) > 0);

autocorr(res\_AICC.^2,50)

title(['Sample Autocorr for AICC-minimizing Gaussian model squared residuals [' num2str(num\_significant) ']'])

subplot(3,2,5)

qqplot(res\_AICC)

title('QQ plot for AICC-minimizing Gaussian model residuals')

subplot(3,2,2)

[acf, lags, bounds] = autocorr(res\_BIC,50);

num\_significant = sum(abs(acf)-bounds(1) > 0);

autocorr(res\_BIC,50)

title(['Sample Autocorr for BIC-minimizing Gaussian model residuals [' num2str(num\_significant) ']'])

subplot(3,2,4)

[acf, lags, bounds] = autocorr(res\_BIC.^2,50);

num\_significant = sum(abs(acf)-bounds(1) > 0);

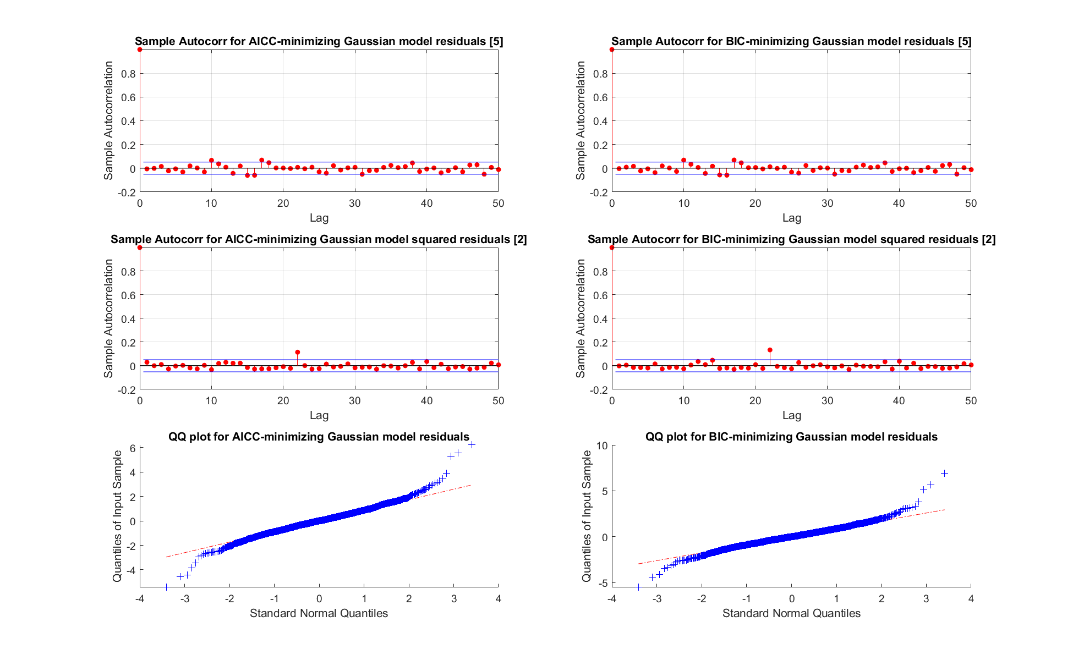
autocorr(res\_BIC.^2,50)

title(['Sample Autocorr for BIC-minimizing Gaussian model squared residuals [' num2str(num\_significant) ']'])

subplot(3,2,6)

qqplot(res\_BIC)

title('QQ plot for BIC-minimizing Gaussian model residuals')



If the model is correct the residuals should follow the underlying assumed distribution of the model (in this case, Gaussian).

The QQ-plots should follow a straight line if the residuals are normally distributed as we are fitting a GARCH process driven by Gaussian noise. There are some outliers on either end, indicating our residuals follow the central part of the assumed distribution but have heavier tails.

Looking at the Sample autocorrelation we see that there are few significant values, indicating that there is little autocorrelation among the residuals.

A normal distribution might not be appropriate here.

## 6)

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

clearvars EstMdl numParam\_garch logL\_garch

Options = optimoptions(@fmincon,'ConstraintTolerance',1e-7,'Algorithm','interior-point');

i = 1;

K = max\_order\_t;

qp\_garch = [];

for p = 1:K

for q = 1:K

mdl\_temp = garch('GARCHLags',1:q,'ARCHLags',1:p,'Distribution',struct('Name','t','DoF',nan));

[EstMdl(i), ~, logL\_garch(i), ~] = estimate(mdl\_temp, trainX, 'Display', 'off', 'Options', Options);

results = summarize(EstMdl(i));

numParam\_garch(i) = sum(results.Table.Value ~= 0) - 1; % removing DoF

DoF(i) = results.Table.Value(end);

qp\_garch = [qp\_garch, [q; p]];

i = i + 1;

end

end

q\_garch = qp\_garch(1,:);

p\_garch = qp\_garch(2,:);

AICC\_garch = calculateAICC(length(trainX), numParam\_garch, logL\_garch); % both defined at end of file

BIC\_garch = -2\*logL\_garch + numParam\_garch\*log(length(trainX));

i\_AICC = find(AICC\_garch==min(AICC\_garch));

i\_BIC = find(BIC\_garch==min(BIC\_garch));

best\_GARCH\_mdl\_t\_AICC = EstMdl(i\_AICC);

best\_GARCH\_mdl\_t\_BIC = EstMdl(i\_BIC);

disp(["Model which minimizes [BD]'s AICC:" EstMdl(i\_AICC).Description; "Model which minimizes hint's BIC:" EstMdl(i\_BIC).Description])

"Model which minimizes [BD]'s AICC:" "GARCH(2,2) Conditional Variance Model (t Distribution)"

"Model which minimizes hint's BIC:" "GARCH(1,1) Conditional Variance Model (t Distribution)"

close all

set(gcf,'Position',[100, 100, 1300, 800])

t\_dist\_AICC = makedist('tLocationScale','nu',DoF(i\_AICC));

t\_dist\_BIC = makedist('tLocationScale','nu',DoF(i\_BIC));

cond\_var\_AICC = infer(best\_GARCH\_mdl\_t\_AICC, trainX);

cond\_var\_BIC = infer(best\_GARCH\_mdl\_t\_BIC, trainX);

res\_AICC = (trainX - best\_GARCH\_mdl\_t\_AICC.Offset) ./ sqrt(cond\_var\_AICC);

res\_BIC = (trainX - best\_GARCH\_mdl\_t\_BIC.Offset) ./ sqrt(cond\_var\_BIC);

subplot(3,2,1)

[acf, lags, bounds] = autocorr(res\_AICC,50);

num\_significant = sum(abs(acf)-bounds(1) > 0);

autocorr(res\_AICC,50)

title(['Sample Autocorr for AICC-minimizing model residuals [' num2str(num\_significant) ']'])

subplot(3,2,3)

[acf, lags, bounds] = autocorr(res\_AICC.^2,50);

num\_significant = sum(abs(acf)-bounds(1) > 0);

autocorr(res\_AICC.^2,50)

title(['Sample Autocorr for AICC-minimizing model squared residuals [' num2str(num\_significant) ']'])

subplot(3,2,5)

qqplot(res\_AICC, t\_dist\_AICC)

title('QQ plot for AICC-minimizing model residuals')

subplot(3,2,2)

[acf, lags, bounds] = autocorr(res\_BIC,50);

num\_significant = sum(abs(acf)-bounds(1) > 0);

autocorr(res\_BIC,50)

title(['Sample Autocorr for BIC-minimizing model residuals [' num2str(num\_significant) ']'])

subplot(3,2,4)

[acf, lags, bounds] = autocorr(res\_BIC.^2,50);

num\_significant = sum(abs(acf)-bounds(1) > 0);

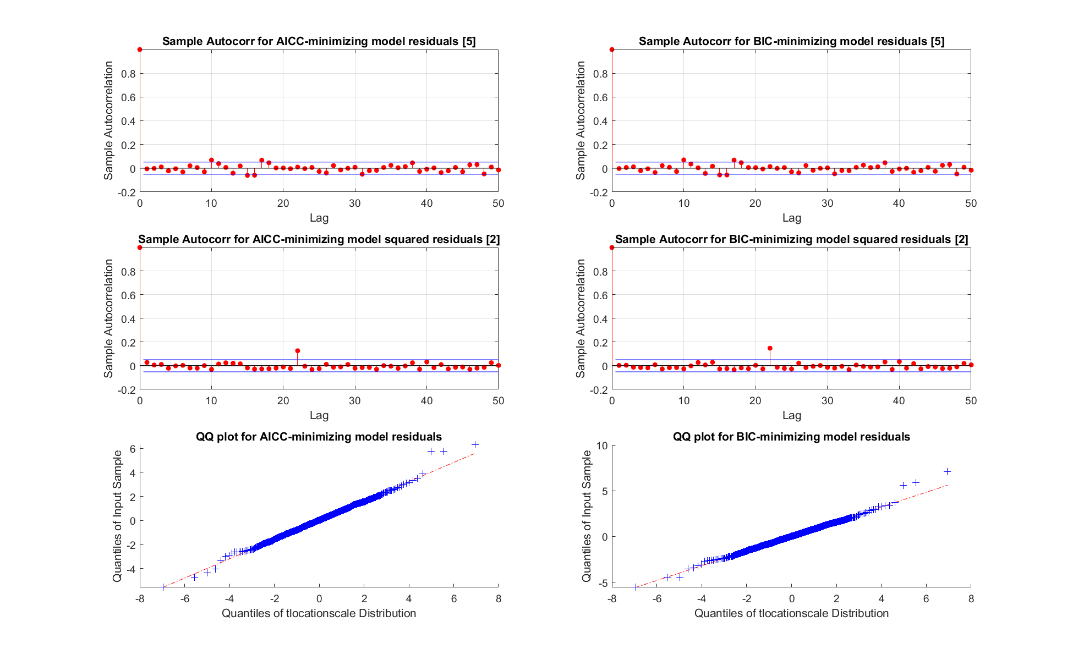
autocorr(res\_BIC.^2,50)

title(['Sample Autocorr for BIC-minimizing model squared residuals [' num2str(num\_significant) ']'])

subplot(3,2,6)

qqplot(res\_BIC, t\_dist\_BIC)

title('QQ plot for BIC-minimizing model residuals')



If the residuals follow a t-distribution, we would expect a straight line. The QQ-plots here look better than for the normal distribution. We have a only several outliers.

Looking at the Sample autocorrelation we see that we have some minor outliers for training data of X, and as there are more than 5% of the number of lags outside, that could indicate that we have some correlation left. The squared residuals looks better, with only one lag outside of the bounds, indicating that we have a good model for the data.

Compared to the gained model in task 3 for t-distribution, one of the models correspond with the gained result in this task; GARCH(2,2)).

## 7)

Estimate  and evaluate the estimates for 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  given  including its parameters (this will either be a Gaussian or a (generalized) *t*-distribution
* For each  of the test data set, compute  given  using (1) for  Do not reestimate parameters of the models at each time  but use the observations  to update the conditional variances as needed
* Count the number of  breaches, i.e., for  count the number of times that 

After doing this, also compute  under the simple model that , where is the sample variance of  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  would you recommend for this stock and why? (You are 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).

models = {};

%Create gaussian model from task 3

models{1} = garch('GARCHLags',1:ARMA\_to\_GARCH\_Gauss\_params.p(1),'ARCHLags',1:ARMA\_to\_GARCH\_Gauss\_params.q(1));

models{1}.Offset = mean(trainX);

models{2} = garch('GARCHLags',1:ARMA\_to\_GARCH\_Gauss\_params.p(2),'ARCHLags',1:ARMA\_to\_GARCH\_Gauss\_params.q(2));

models{2}.Offset = mean(trainX);

models{3} = garch('GARCHLags',1:ARMA\_to\_GARCH\_t\_params.p,'ARCHLags',1:ARMA\_to\_GARCH\_t\_params.q,...

'Distribution',struct('Name','t','DoF',nan));

for i = 1:3

[models{i}, ~, ~] = estimate(models{i}, trainX, 'Display', 'off', 'Options', Options);

end

models{4} = best\_GARCH\_mdl\_Gauss\_AICC;

models{5} = best\_GARCH\_mdl\_Gauss\_BIC;

models{6} = best\_GARCH\_mdl\_t\_AICC;

models{7} = best\_GARCH\_mdl\_t\_BIC;

s = {};

disp('Candidate models')

Candidate models

for i = 1:length(models)

disp(models{i}.Description)

if models{i}.Distribution.Name=='t'

dist = makedist('tLocationScale','nu',models{i}.Distribution.DoF);

s{i} = struct('model',models{i},'distribution',dist,'pre\_defined\_dist',true);

else

s{i} = struct('model',models{i},'distribution',nan,'pre\_defined\_dist',false);

end

end

GARCH(2,2) Conditional Variance Model with Offset (Gaussian Distribution)

GARCH(2,1) Conditional Variance Model with Offset (Gaussian Distribution)

GARCH(2,3) Conditional Variance Model (t Distribution)

GARCH(2,2) Conditional Variance Model with Offset (Gaussian Distribution)

GARCH(1,1) Conditional Variance Model with Offset (Gaussian Distribution)

GARCH(2,2) Conditional Variance Model (t Distribution)

GARCH(1,1) Conditional Variance Model (t Distribution)

t\_list = 1527:1751;

p\_list = [.1 .05 .01];

VaR = zeros(length(t\_list), length(p\_list));

for i = 1:length(s)

VaR = zeros(length(t\_list), length(p\_list));

breaches = zeros(length(t\_list), length(p\_list));

for iT = 1:length(t\_list)

t = t\_list(iT);

X\_tmp = X(1:t-1);

if s{i}.pre\_defined\_dist

dist = s{i}.distribution;

else

condVar = infer(s{i}.model,X\_tmp);

sigma = sqrt(condVar(end));

dist = makedist('normal','mu',mean(X\_tmp),'sigma',sigma);

end

for iP = 1:length(p\_list)

p = p\_list(iP);

VaR(iT,iP) = V.Close(t-1) \* (exp(dist.icdf(p))-1);

breaches(iT,iP) = V.Close(t) - V.Close(t-1) <= VaR(iT,iP);

end

end

s{i}.VaR = VaR;

s{i}.breaches = sum(breaches);

end

% simple model

VaR = zeros(length(t\_list), length(p\_list));

for iT = 1:length(t\_list)

t = t\_list(iT);

dist = makedist('norma','mu',mean(X(1:t-1)),'sigma',std(X(1:t-1)));

for iP = 1:length(p\_list)

p = p\_list(iP);

VaR(iT,iP) = V.Close(t-1) \* (exp(dist.icdf(p))-1);

breaches(iT,iP) = V.Close(t) - V.Close(t-1) <= VaR(iT,iP);

end

end

simple\_breaches = sum(breaches);

% tabulating

model\_names = [];

p\_1 = [];

p\_05 = [];

p\_01 = [];

for i = 1:length(s)

model\_names = [model\_names; s{i}.model.Description];

p\_1 = [p\_1; s{i}.breaches(1)];

p\_05 = [p\_05; s{i}.breaches(2)];

p\_01 = [p\_01; s{i}.breaches(3)];

end

model\_names = [model\_names; 'Simple'];

p\_1 = [p\_1; simple\_breaches(1)];

p\_05 = [p\_05; simple\_breaches(2)];

p\_01 = [p\_01; simple\_breaches(3)];

table(model\_names,p\_1,p\_05,p\_01,'VariableNames',{'Model Name','p=0.1','p=0.05','p=0.01'})

ans = 8×4 table

|  | **Model Name** | **p=0.1** | **⋯** |
| --- | --- | --- | --- |
| **1** | "GARCH(2,2) Conditional Variance Model with Offset (Gaussian Distribution)" | 15 |  |
| **2** | "GARCH(2,1) Conditional Variance Model with Offset (Gaussian Distribution)" | 14 |  |
| **3** | "GARCH(2,3) Conditional Variance Model (t Distribution)" | 0 |  |
| **4** | "GARCH(2,2) Conditional Variance Model with Offset (Gaussian Distribution)" | 15 |  |
| **5** | "GARCH(1,1) Conditional Variance Model with Offset (Gaussian Distribution)" | 14 |  |
| **6** | "GARCH(2,2) Conditional Variance Model (t Distribution)" | 0 |  |
| **7** | "GARCH(1,1) Conditional Variance Model (t Distribution)" | 0 |  |
| **8** | "Simple" | 11 |  |

From this we see that a t-distributed GARCH(1,1) model would probably be the most appropriate model for this stock. As we want to keep the order as low as possible (avoid over fitting to the data) and both the Gaussian GARCH models and the “simple” model performed worse than the t-distributed GARCH models.

## Appendix: Functions

function AICC = calculateAICC(n, M, logL)

factor = (2\*n\*M) ./ (n-M-1);

AICC = -2\*logL + factor;

end

function BIC = calculateBIC(n, p, q, var, X, M, which)

if which == 'pq' % include zero-valued parameters

left = (n-p-q) .\* log( (n.\*var) ./ (n-p-q) );

middle = n \* (1 + log(sqrt(2\*pi)));

right = (p+q) .\* log((sum(X.^2) - n.\*var) ./ (p+q));

BIC = left + middle + right;

else % do not include zero-valued parameters

left = (n-M) .\* log( (n.\*var) ./ (n-M));

middle = n \* (1 + log(sqrt(2\*pi)));

right = M .\* log((sum(X.^2) - n.\*var) ./ M);

BIC = left + middle + right;

end

end