Decision and Risk In-Course Assessment
Group 58

24 March 2023

Contents

Individual Contributions

19157580
- Write R code for Task 2 (a) and (b)
- Run the test models in R

20062064
- Write R code for Task 2 (a) and (b)
- Run the test models in R

20009310
- Comment on Task 2 (a) approach and working process
- Compare Task 2 (a) and (b) approaches and give analysis
- Consolidate Task 2 (a) and (c) in LaTeX

20022286
- Comment on Task 2 (a) approach and working process
- Compare Task 2 (a) and (b) approaches and give analysis
- Consolidate Task 2 (a) and (c) in LaTeX

20018539
- Developed questions, solutions, and marking scheme in Task 1
- Comment on Task 2 (b) approach and working process
- Consolidate Task 1 and Task 2 (b)in LaTeX

20015477
- Developed questions, solutions, and marking scheme in Task 1
- Comment on Task 2 (b) approach and working process
- Consolidate Task 1 and Task 2 (b) in LaTeX

1 Task 1

A company produces software in batches of 100. Due to variations in the coding process, each produced batch could be of three types: θ_1 is "nonfunctional", θ_2 is "functional", and θ_3 is "high-quality".

The company has to decide whether to release the software to customers or not. The losses corresponding to each action a_1 is to release and a_2 is to not release, and type θ_j, j = 1, 2, 3, are represented by the following loss matrix:

	a_1	a_2
θ_1	5	1
θ_2	-3	2
θ_3	-5	8

Find the minimax randomized action.

Solution:

Choose a_1 with probability p, and a_2 with probability $(1-p)$ respectively.
Consider the randomised action δ^* in the form of:

$$\delta^* = p\,\langle a_1\rangle + (1-p)\,\langle a_2\rangle$$

Since this is a no-data problem, the risk $R(\theta, \delta^*)$ is just the loss $L(\theta, \delta^*)$:

$$R(\theta, \delta^*) = L(\theta, \delta^*)= \begin{cases} 5p + 1(1-p) = 4p + 1 & \text{if } \theta = \theta_1 \\ -3p + 2(1-p) = -5p + 2 & \text{if } \theta = \theta_2 \\ -5p + 8(1-p) = -13p + 8 & \text{if } \theta = \theta_3 \end{cases}$$

(1 Mark)

Apply the Minimax Action among all randomised rules in $\mathscr{D}^*$:

$$\sup_{\theta\in\Theta} R(\theta, \delta^*) = \inf_{\delta^*\in\mathscr{D}^*} \sup_{\theta\in\Theta} R(\theta, \delta^*)$$

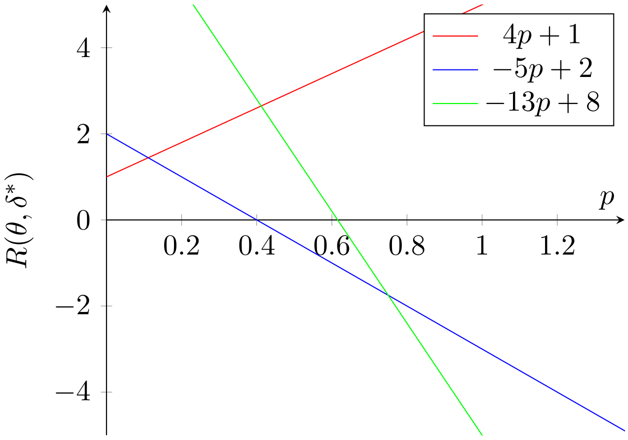
(1 Mark)

We have

$$\sup_{\theta} R(\theta, \delta^*_p) = \max\{4p + 1, -5p + 2, -13p + 8\}$$

(1 Mark)

Plotting these three functions:



(Plotting: 1 Mark)

The minimum value of $\max\{4p+1, -5p+2, -13p+8\}$ is $\frac{45}{17}$, which occurs at $p = \frac{7}{17}$.

(1 Mark)

Therefore, $\delta^*_{\frac{7}{17}}$ is the minimax action, and $\frac{45}{17}$ is the minimax value for the problem. Therefore, the company would choose a_1 with probability $\frac{7}{17}$ and choose a_2 with probability $\frac{10}{17}$.

(1 Mark)

2 Task 2

2.1 Part a

2.1.1 Data used

The adjusted closing weekly prices from SSE Composite Index and CAC 40 Index, from the week starting on 01/01/2000 to the week ending on 29/12/2018.

2.1.2 Working process

To construct an equally-weighted portfolio, we first use the formula $r_t = \log(\frac{p_t}{p_{t-1}})$ to work out the log returns for the two types of stocks.

```r
## Import and sort out SSE and CAC40 data
sse_data <- read.csv("SSE_weekly_2000-2018.csv")
cac_data <- read.csv("^FCHI.csv")

# Find and store rows where the Adj.Close value is "null"
headings <- which(sse_data$Adj.Close=='null')

# remove null data in SSE data
sse_data <- sse_data[-headings,]

# remove null data in CAC data
cac_data <- cac_data[-headings,]

## Create log returns for SSE and CAC40 data
op1 <- sse_data$Adj.Close
prices1 <- as.numeric(op1)
n1 <- length(prices1)
lret1 <- log(prices1[-1]/prices1[-n1])

op2 <- cac_data$Adj.Close
prices2 <- as.numeric(op2)
n2 <- length(prices2)
lret2 <- log(prices2[-1]/prices2[-n2])
```

Listing 1: Task 2 (a) R code

Then, the mean and variance of log-returns of both stock prices are calculated, together with the covariance between them.

```r
## Calculate mean and standard deviation
sse_mean <- mean(lret1) #0.000520
sse_sd <- sd(lret1) #0.0336
cac_mean <- mean(lret2) #-0.000176
cac_sd <- sd(lret2) #0.0299

## Calculate correlation
corr <- cor(lret1,lret2,use='complete.obs') #0.167
```

Listing 2: Task 2 (a) R code

Since it is assumed that the portfolio is equally weighted by SSE Composite and CAC40, portfolio weights of 50% should be used to compute the portfolio mean μ_p and variance σ_p^2.

```r
## Create an equally-weighted portfolio
w <- c(1/2,1/2)

##Calculate portfolio mean and variance
port_vari <- ((w[1])^2)*((sse_sd)^2)+
             ((w[2])^2)*((cac_sd)^2)+
             2*w[1]*w[2]*corr*sse_sd*cac_sd #0.000591
port_mean <- w[1]*sse_mean+w[2]*cac_mean #0.000172
```

Listing 3: Task 2 (a) R code

Thus, the Value-at-Risk can be calculated through the formula $\mathrm{VaR}_{\alpha,p} = \Phi^{-1}(\alpha)\sigma_p - \mu_p$.

```r
Port_VaR_99 <- qnorm(0.01)*sqrt(port_vari)-port_mean
Port_VaR_99 #0.0567

Port_VaR_95 <- qnorm(0.05)*sqrt(port_vari)-port_mean
Port_VaR_95 #0.0402
```

Listing 4: Task 2 (a) R code

Then the normality test for the data is carried out by using Jarque-Bera Test which is a goodness-of-fit test of whether the sample data (log-returns) have the skewness and kurtosis matching a normal distribution. Since the test shows a small p-value, we have sufficient evidence to reject the null hypothesis that the log-returns of both stock indices are normally distributed.

```r
## Check normality by Jarque-Bera Test
# Note that for SSE we use data with missing values
plot(lret1,lret2)
jarque.bera.test(lret0) #p-value < 2.2e^(-16)
jarque.bera.test(lret2) #p-value < 2.2e^(-16)
```

Listing 5: Task 2 (a) R code

2.1.3 Result

The estimated 99% and 95% 1-week VaR of the portfolio are -0.0567% and -0.0402%, which represent that if we invest one million in the portfolio, there are 1% and 5% probability of earning more than 567 and 402 during a single week. However, since the Jarque-Bera test against the normality assumption of parametric approach, it seems that other approaches should be considered.

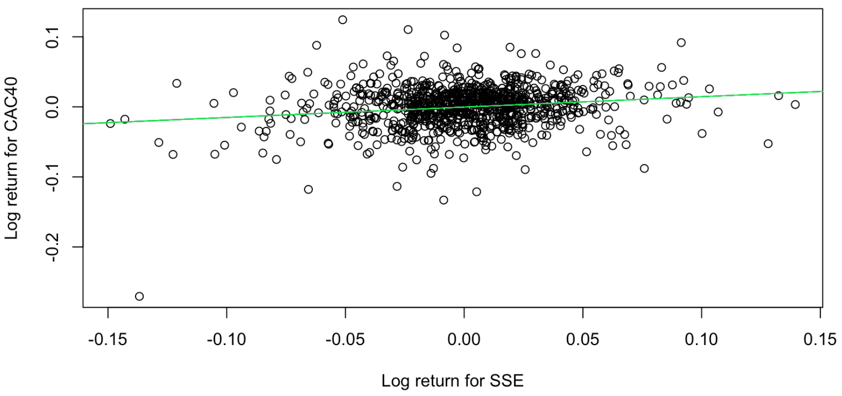


Figure 1: Correlation between log-returns of SSE and CAC40

2.1.4 Strengths and Weaknesses of Parametric Approach

Strengths The parametric approach is easy to calculate, understand and interpret in communicating the portfolio behaviour of SSE and CAC40. It is also convenient, especially in investigating a portfolio of two assets that have data with different variances. Furthermore, the assumption of normality in the parametric approach can be suitable for estimating the investment portfolio with large sample size. Although the approach can be applied to both log-returns and net returns, log-return is chosen because it is additive and has a symmetric range, which is useful in analysing the statistical behaviour of portfolio payoffs over time.

Weaknesses The extremely small p-value ($p < 2.2e^{-16}$) generated in Jarque-Bera test shows that the portfolio returns of SSE and CAC40 do not follow a normal distribution, so the parametric approach, no matter using log-return or net return, can be misleading. Moreover, the linear correlation coefficient may measure the dependence structure inaccurately, which can lower the accuracy of the result. As Figure 1 shows, the most of the SSE and CAC40 log-returns are concentrated in the middle and some of them are scattered around on the plot, which doesn't seem to have a linear correlation. The approach is also difficult to derive the parametric formula for VaR based on copulas.

2.2 Part b

2.2.1 Working Process

We now using the Monte Carlo Simulation approach to estimate the VaR of the portfolio based on the copula theory.

Finding models for marginal distributions:

1) Identification After checking normality of return series, we create a dataframe with portfoilo and log returns.

```r
port <- data.frame(sse_data$Date[-1],lret11,lret2)
colnames(port) = c("Date","Ret1","Ret2")
```

Listing 6: Task 2 (b) R code 1

Then, we use ACF and PACF plots for log-returns to observe potential autocorrelation relationship and determine whether we can use GARCH model.

```r
##log return 1
par(mfrow=c(2,2))
acf(lret0, col="green", lwd=2)
pacf(lret0, col="green", lwd=2)
acf(lret0^2, col="red", lwd=2)
par(mfrow=c(1,1))
##log returns 2
par(mfrow=c(2,2))
acf(lret2, col="green", lwd=2)
pacf(lret2, col="green", lwd=2)
acf(lret2^2, col="red", lwd=2)
par(mfrow=c(1,1))
```

Listing 7: Task 2 (b) R code 2

The ACF and PACF plots (Figure 2 and 3) are created for each return series (lret1 and lret2) and their squared values to investigate potential autocorrelation patterns. We see a spike at lag 0 from the ACF graph. The spikes in the ACF and PACF plots at different lags provide insights into the time-series properties of the log returns. We see 4 spikes exceeding the limits from the PACF graph.

And there are many spikes exceeding the limit for squared values' plots which indicate that there exsits an GARCH effect. Thus we select GARCH(1,1) model for further estimation.

By plotting the time series of log returns (Figure 4), we notice the existence of volatility clustering, and thus a GARCH(1,1) model is selected to best describe this phenomenon.

```r
year = as.Date(port$Date)
par(mfrow=c(1,2))
# Plot log return of idx1
```

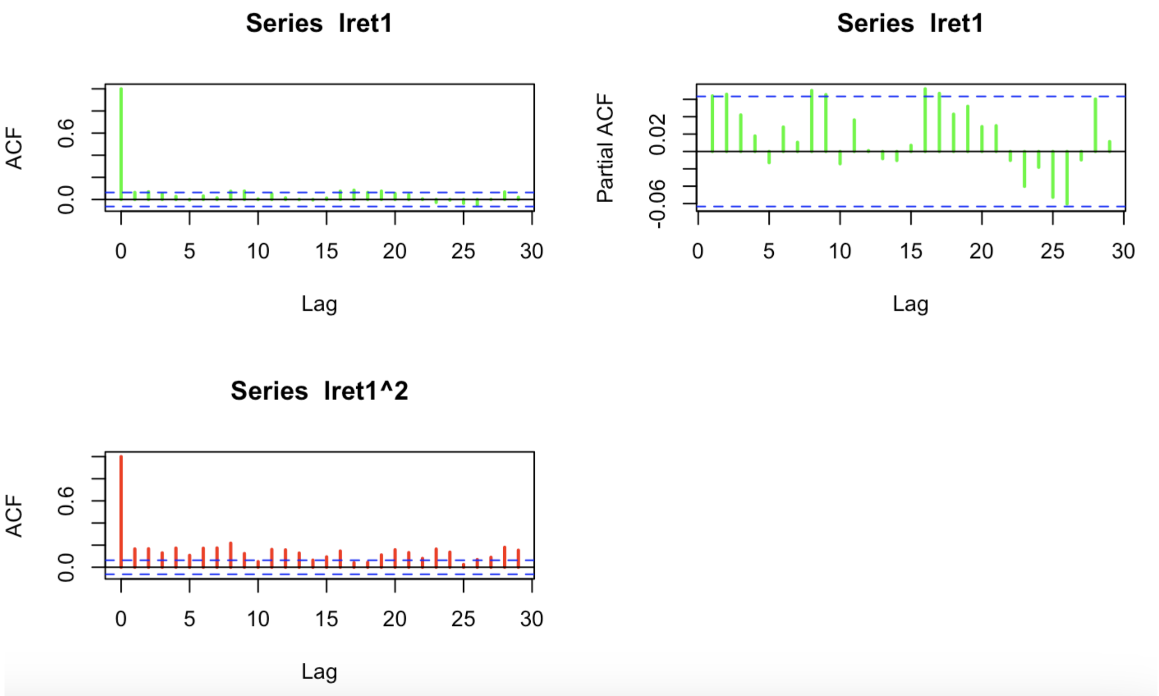


Figure 2: ACF and PACF plots for SSE data

```
plot(year,lret1,xlab="Year",ylab="Log return of idx1", main= "Time series of idx1"
     ,type="l",xaxt = "n")
axis(1,year,format(year,"%Y"),cex.axis=0.5,tick = FALSE)
# Plot log return of idx2
plot(year,lret2,xlab="Year",ylab="Log return of idx2", main= "Time series of idx2"
     ,type="l",xaxt="n")
axis(1,year,format(year,"%Y"),cex.axis=0.5,tick = FALSE)
```

Listing 8: Task 2 (b) R code 3

2) Estimation Setting up GARCH models with different conditional distributions to find the best fit by AIC/BIC values.

```
model1_1=garchFit(~arma(0,0)+garch(1,1),data=lret0,trace=F,cond.dist="norm")
model1_2=garchFit(~arma(0,0)+garch(1,1),data=lret0,trace=F,cond.dist="std")
model1_3=garchFit(~arma(0,0)+garch(1,1),data=lret0,trace=F,cond.dist="sstd")
model1_4=garchFit(~arma(0,0)+garch(1,1),data=lret0,trace=F,cond.dist="snorm")
model1_5=garchFit(~arma(0,0)+garch(1,1),data=lret0,trace=F,cond.dist="sged")
model1_6=garchFit(~arma(0,0)+garch(1,1),data=lret0,trace=F,cond.dist="ged")
ic1=rbind(model1_1@fit$ics,model1_2@fit$ics,model1_3@fit$ics,model1_4@fit$ics,model1_5
```

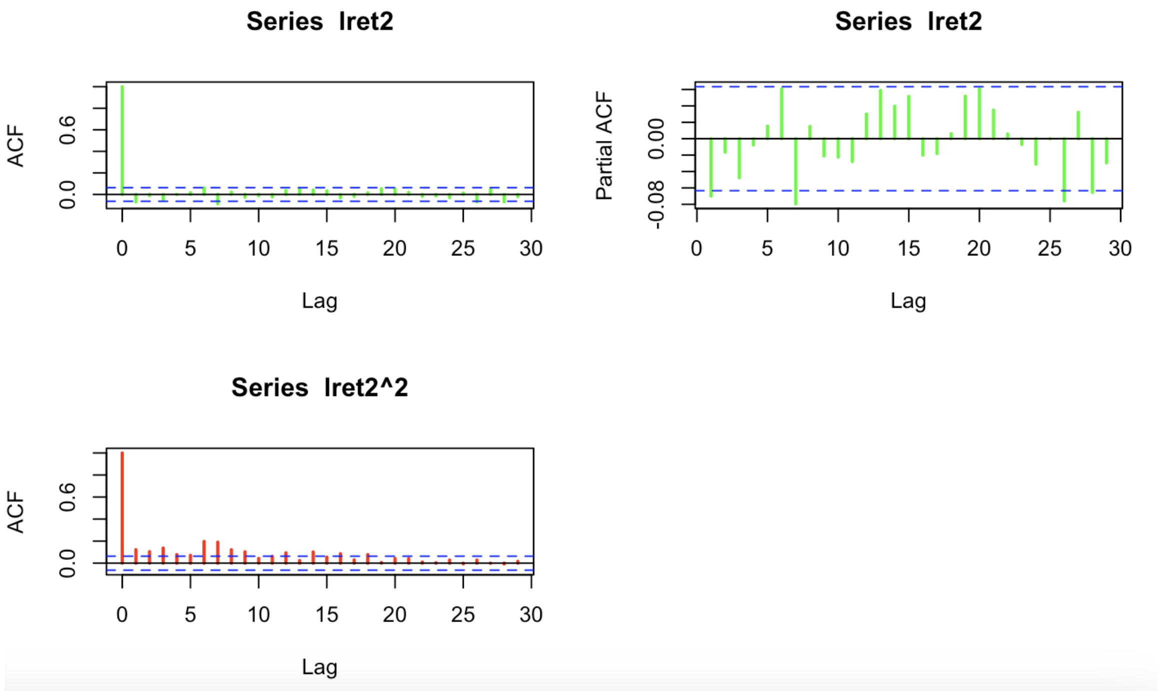


Figure 3: ACF and PACF plots for CAC 40 data

```
          @fit$ics,model1_6@fit$ics)
rownames(ic)=c("norm", "std","sstd", "snorm", "sged", "ged" )
ic1
# Find the model with the lowest BIC for SSE
min_bic1_model <- rownames(ic1)[which.min(ic1[,"BIC"])]
cat("Model with lowest BIC:", min_bic1_model, "\n")
```

Listing 9: Task 2 (b) R code 4 for SSE

(Similar coding process for the CAC 40 distribution selection.)

From the table 1 and code above, we select Standard Student's t distribution for SSE and Skewed-Student's t distribution for CAC 40 since they give the lowest BIC values respectively.

Then, we use `garchFit` function again to determine the order of AR models for SSE and CAC 40 by applying conditional distributions to the models.

```
model_a1=garchFit(~arma(0,0)+garch(1,1),data=lret1,trace=F,cond.dist="sstd")
model_b1=garchFit(~arma(1,0)+garch(1,1),data=lret1,trace=F,cond.dist="sstd")
model_c1=garchFit(~arma(2,0)+garch(1,1),data=lret1,trace=F,cond.dist="sstd")
model_d1=garchFit(~arma(3,0)+garch(1,1),data=lret1,trace=F,cond.dist="sstd")
model_e1=garchFit(~arma(4,0)+garch(1,1),data=lret1,trace=F,cond.dist="sstd")
model_f1=garchFit(~arma(5,0)+garch(1,1),data=lret1,trace=F,cond.dist="sstd")
```

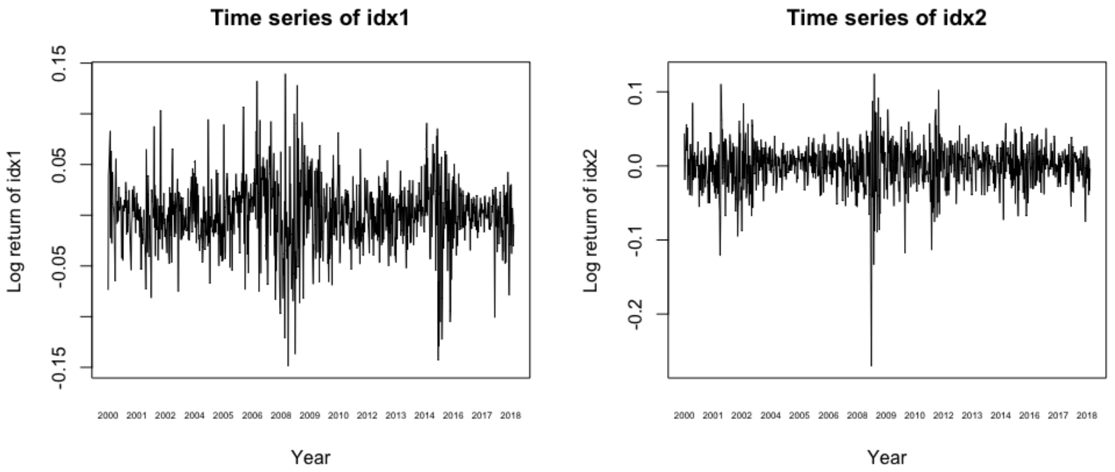


Figure 4: Time Series plot

	SSE		CAC	
Conditional Distribution	AIC	BIC	AIC	BIC
Normal	-4.139420	-4.119090	-4.399550	-4.358891
Student-t	-4.167255	-4.141843	-4.453261	-4.407520
Skew Student-t	-4.167832	-4.137338	-4.492718	-4.441895
Skew Normal	-4.456733	-4.410992	-4.484679	-4.440190
Skew Generalized Error	-4.160076	-4.129582	-4.480647	-4.429823
Generalized Error	-4.159328	-4.133917	-4.432251	-4.386510

Table 1: Selecting conditional distirbution for AR(0)-GARCH(1,1) model

```r
model_g1=garchFit(~arma(10,0)+garch(1,1),data=lret1,trace=F,cond.dist="sstd")
ic.1=rbind(model_a1@fit$ics,model_b1@fit$ics,model_c1@fit$ics,model_d1@fit$ics,model_e1@fit$ics,
           model_f1@fit$ics,model_g1@fit$ics)
rownames(ic.1)=c("AR(0)","AR(1)","AR(2)","AR(3)","AR(4)","AR(5)","AR(10)")
ic.1
# Find the model with the lowest AIC and BIC
min_aic_model.1 <- rownames(ic.1)[which.min(ic.1[,"AIC"])]
min_bic_model.1 <- rownames(ic.1)[which.min(ic.1[,"BIC"])]
cat("Model with lowest AIC:", min_aic_model.1, "\n")#output AR(10)
cat("Model with lowest BIC:", min_bic_model.1, "\n")#output AR(2)
```

Listing 10: Task 2 (b) R code 5 for SSE

(Similar coding process for the CAC 40 AR model selection, outputting Model with the lowest AIC is AR(10) and Model with the lowest BIC is AR(2).)

Model	AIC	BIC	PValue	MSE	Combined Score
AR(0)	-4.167255	-4.141843	0.006422453	0.001137857	0.2478990
AR(1)	-4.172248	-4.141754	0.047634054	0.001137142	0.3225620
AR(2)	-4.177749	-4.142172	0.256197465	0.001133933	0.5268697
AR(3)	-4.179420	-4.138761	0.680157775	0.001129828	0.7367063
AR(4)	-4.184102	-4.138361	0.639338491	0.001130410	0.7443326
AR(5)	-4.184311	-4.133488	0.643568710	0.001130905	0.6769141
AR(10)	-4.196199	-4.119964	0.652355010	0.001130792	0.6330776

Table 2: Selecting the order of AR model for SSE

Model	AIC	BIC	PValue	MSE	Combined Score
AR(0)	-4.470095	-4.439601	0.130931073	0.0005230598	0.2290749
AR(1)	-4.479557	-4.443981	0.358091901	0.0005228494	0.4956748
AR(2)	-4.487056	-4.446397	0.078141651	0.0005224342	0.6829193
AR(3)	-4.489294	-4.443553	0.075427360	0.0005229823	0.4623089
AR(4)	-4.492718	-4.441895	0.041447521	0.0005229326	0.4935499
AR(5)	-4.492140	-4.436234	0.039666363	0.0005229791	0.4242592
AR(10)	-4.496048	-4.414730	0.004827651	0.0005229816	0.2824382

Table 3: Selecting the order of AR model for CAC

As figures in table 2 and 3 shows, AIC and BIC do not agree on the best model for SSE and CAC 40, so we should consider other factors, i.e. Lujung Box test and Mean Squared Error (MSE).

```r
# Load necessary packages
library(forecast)
library(lmtest)
library(rugarch)

# Split the data into training and testing sets
n <- length(lret1)
train_size <- floor(0.8 * n)
train_data <- lret1[1:train_size]
test_data <- lret1[(train_size + 1):n]

# Fit the models on the training set
models <- list(model_a1, model_b1, model_c1, model_d1, model_e1, model_f1,
    model_g1)

# Initialize variables to store the results
ljung_box_pvalues <- numeric(length(models))
mse <- numeric(length(models))
```

```

# Evaluate the models
for (i in 1:length(models)) {
  # Refit the model on the training data
  model_spec <- ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
                           mean.model = list(armaOrder = c(i - 1, 0), include.mean = TRUE),
                           distribution.model = "sstd")
  refitted_model <- ugarchfit(spec = model_spec, data = train_data)

  # Ljung-Box test on residuals
  ljung_box_pvalues[i] <- Box.test(residuals(refitted_model), type = "Ljung-Box", lag = 10)$p.value

  # Forecast and compute the Mean Squared Error (MSE)
  forecasted_values <- ugarchforecast(refitted_model, n.ahead = length(test_data))@forecast$seriesFor
  mse[i] <- mean((test_data - forecasted_values)^2)
}

# Combine the results
model_names <- c("AR(0)", "AR(1)", "AR(2)", "AR(3)", "AR(4)", "AR(5)", "AR(10)")
results <- data.frame(Model = model_names, AIC = ic.1[,"AIC"], BIC = ic.1[,"BIC"],
                      Ljung_Box_PValue = ljung_box_pvalues, MSE = mse)
results
```

Listing 11: Task 2 (b) R code 6 for SSE

We want a model with a high p-value from the Ljung-Box test (indicating no significant autocorrelation in the residuals) and a low MSE value (indicating better forecasting accuracy). Therefore, we consider the best model based on a combination of AIC, BIC, Ljung-Box p-value, and MSE.

```
# Define a weight for each metric
aic_weight <- 0.25
bic_weight <- 0.25
ljung_box_weight <- 0.25
mse_weight <- 0.25

# Normalize the AIC, BIC, and MSE values (the lower, the better)
norm_aic <- 1 - (results$AIC - min(results$AIC)) / (max(results$AIC) - min(results$AIC))
norm_bic <- 1 - (results$BIC - min(results$BIC)) / (max(results$BIC) - min(results$BIC))
norm_mse <- 1 - (results$MSE - min(results$MSE)) / (max(results$MSE) - min(results$MSE))

# Calculate the combined score for each model
combined_score <- aic_weight * norm_aic +
  bic_weight * norm_bic +
  ljung_box_weight * results$Ljung_Box_PValue +
  mse_weight * norm_mse

# Add the combined score to the results data frame
results$Combined_Score <- combined_score

# Find the best model based on the combined score
best_model_index <- which.max(results$Combined_Score)
best_model1 <- results[best_model_index, ]
best_model1
```

Listing 12: Task 2 (b) R code 7 for SSE

(Similar coding process for the CAC 40.)

Under the consideration of all four factors above, we can find out the best AR model for SSE data is AR(4) since it gives out the highest combined score as table 2 shows. And similarly, we need select AR(2) as the best AR model for CAC 40 as it is shown in table 3.

Therefore, the final model selection is

```
model_sse = garchFit(formula=~arma(4,0)+garch(1,1),data=lret1,trace=F,cond.
    dist="std")
model_cac = garchFit(formula=~arma(2,0)+garch(1,1),data=lret2,trace=F,cond.
    dist="sstd")
```

Listing 13: Task 2 (b) R code 8 for SSE

3) Model Checking Once we select the specific model, we use the residual diagnostics method to determine whether this model is adequate. And then check its autocorrelation and GARCH effect by ACF plots and the Ljung-Box test.

We firstly use ACF plots to visualise the standardised residuals and their squared values for SSE and CAC 40 respectively.

```
# Standardize the residuals of the GARCH models
resid_1 <- residuals(model_sse, standardize=TRUE)
resid_2 = residuals(model_cac, standardize=TRUE)
# Plot ACF for the standardized residuals and their squared values
par(mfrow=c(1,2))
acf(resid_1, col="green", lwd=2)
acf(resid_1^2, col="red", lwd=2)
par(mfrow=c(1,2))
acf(resid_2, col="green", lwd=2)
acf(resid_2^2, col="red", lwd=2)
par(mfrow=c(1,1))
```

Listing 14: Task 2 (b) R code 9

Then, we apply Ljung-Box test to test the autocorrelations.

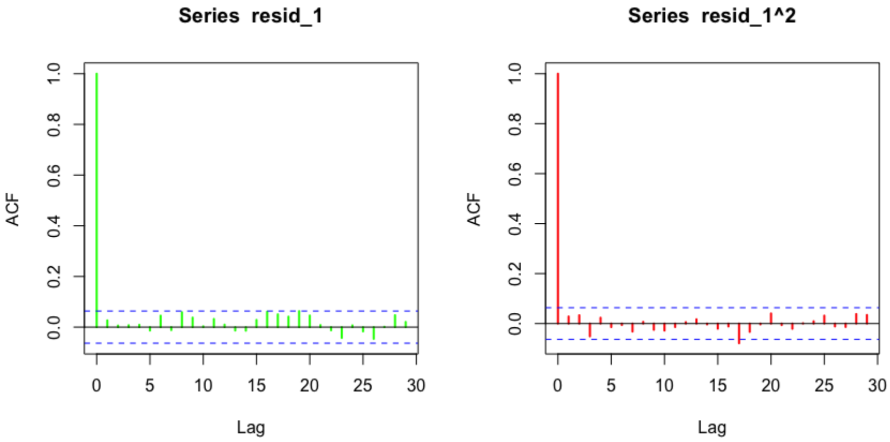


Figure 5: ACF plots of residuals and squared residuals for SSE

```
Box.test(resid_1, lag = 10, type = c("Ljung-Box"), fitdf = 1)#p-value =
    0.5525
Box.test(resid_1^2, lag = 10, type = c("Ljung-Box"), fitdf = 1)#p-value =
    0.5859
Box.test(resid_2, lag = 10, type = c("Ljung-Box"), fitdf = 1)#p-value =
    0.103
Box.test(resid_2^2, lag = 10, type = c("Ljung-Box"), fitdf = 1)#p-value =
    0.9999
```

Listing 15: Task 2 (b) R code 10

These tests and plots above help us to assess the goodness of fit of the GARCH models by checking if there is any remaining autocorrelation in the standardized residuals and their squared values. If there is no significant autocorrelation, i.e. p-value > 0.05, it suggests that the GARCH models have adequately captured the dynamics of the data.

Applying Probability Integral Transform (PIT) to log-returns: For an adequate model, we can apply Probability Integral Transform (PIT) to data respectively to construct a histogram with the standard uniform distribution.

```
## PIT
u1 = pstd(resid_1, mean=0, sd=1,nu=coef(model_sse)['shape'])
u2 = psstd(resid_2, mean=0, sd=1,nu=coef(model_cac)['shape'],xi=coef(model_
    cac)[7]
par(mfrow=c(1,2))
hist(u1)
hist(u2)
```

Listing 16: Task 2 (b) R code 11

Then we use Kolmogorov-Smirnov test and Anderson-Darling test to test the uniformity.

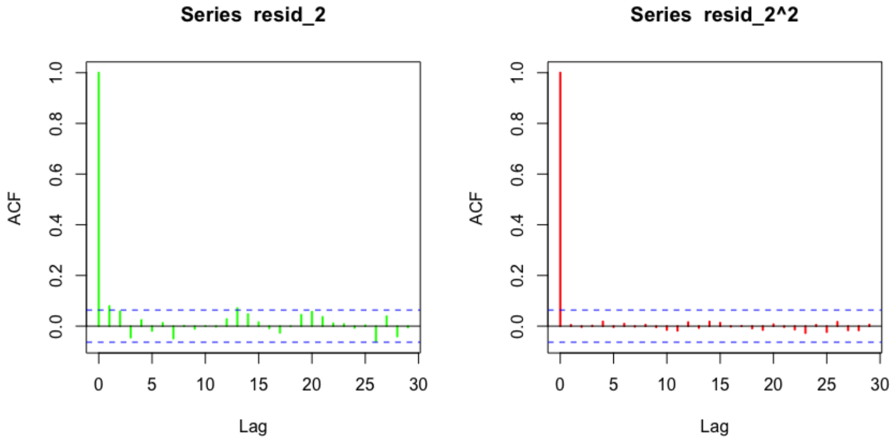


Figure 6: CF plots of residuals and squared residuals for CAC 40

```
KStest1 <- LcKS(u1, cdf = "punif")
KStest1$p.value #p-value = 0.9462
ADtest1 <- ad.test(u1, null="punif")
ADtest1$p.value #p-value = 0.8939862

KStest2 <- LcKS(u2, cdf = "punif")
KStest2$p.value #p-value = 0.3348
ADtest2 <- ad.test(u2, null="punif")
ADtest2$p.value #p-value = 0.1967728
```

Listing 17: Task 2 (b) R code 12

As the p-values indicate are larger than 0.05, both datasets are perfect fits for the uniform distribution.

Fitting various copula models: Next, we apply `BiCopSelect` to select suitable copula models and this ouputs Bivariate copula: Survival Gumbel (par = 1.1, tau = 0.09) for the best fit. Then, we define the AR-GARCH(1,1) model specification for SSE and CAC 40 data and fit the AR-GARCH model to the corresponding set of log returns using the ugarchfit function.

```
# Copula modelling
model <- BiCopSelect(u1, u2, familyset=NA, selectioncrit="AIC", indeptest=
    TRUE, level=0.05,se = TRUE)
model
# Fit AR-GARCH model for SSE and CAC
spec1 <- ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c
    (1, 1)),
                      mean.model = list(armaOrder = c(1, 0), include.mean =
    TRUE))
fit1 <- ugarchfit(spec1, lret1)
```

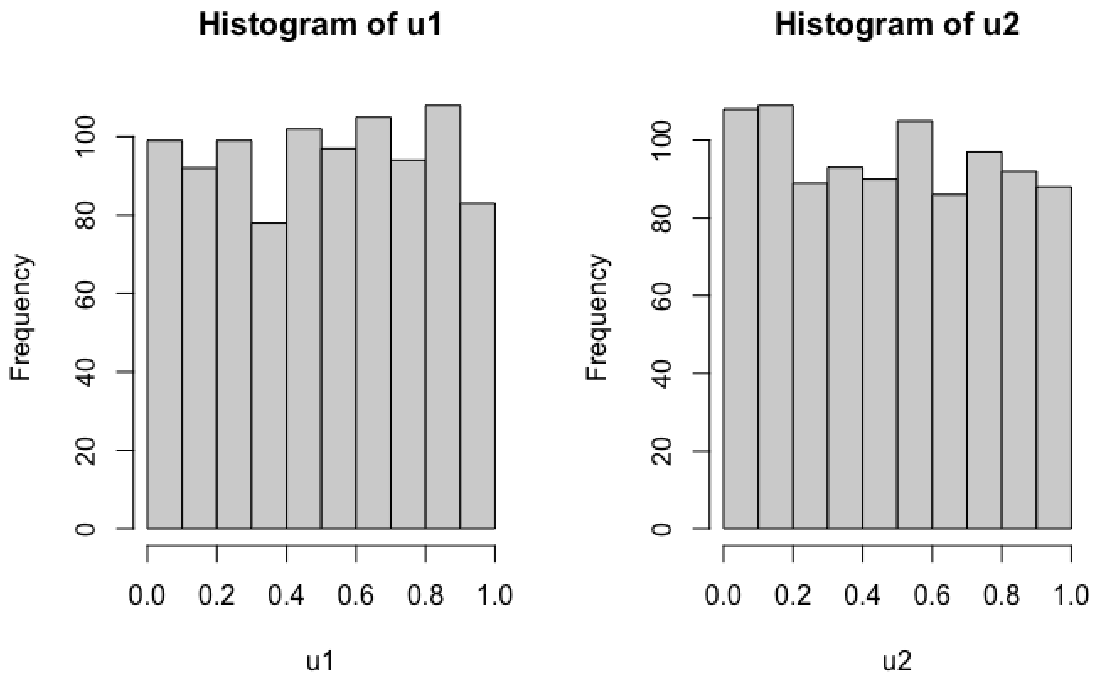


Figure 7: Histograms

```
spec2 <- ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c
    (1, 1)),
                    mean.model = list(armaOrder = c(1, 0), include.mean =
    TRUE))
fit2 <- ugarchfit(spec2, lret2)
```

Listing 18: Task 2 (b) R code 13

Using Monte Carlo simulation to estimate VaR: Finally, we can calculate VaR by Monte Carlo simulation. The Monte Carlo simulation based on copula model generates correlated uniform samples. And then, they are transformed into their respective distributions (i.e. Student's t distribution and Skewed Student's t distribution).

```
# Perform Monte Carlo simulation using the copula model to generate
    correlated uniform samples
N <- 10000
set.seed(111)
u_sim <- BiCopSim(N, family = model$family, model$par, model$par2)

# Transform the uniform samples into the respective distributions
res1_sim <- qdist("std", u_sim[, 1], shape = coef(model_sse)['shape']) #
    Student's t distribution
```

```
res2_sim <- qdist("sstd", u_sim[, 2], shape = coef(model_cac)['shape'], skew
    = coef(model_cac)[7]) # Skewed Student's t distribution
```

Listing 19: Task 2 (b) R code 14

Next, we generate AR-GARCH residuals and combine them with the standard normal samples to get correlated residuals for the respective distributions. These correlated residuals are then used to compute the portfolio returns. Finally, we can calculate the 1% and 5% quantiles of the resulting portfolio returns.

```
# Simulate AR-GARCH residuals
y1simulated <- fitted(ugarchsim(fit1, n.sim = N, m.sim = 1, startMethod = "
    sample"))
y2simulated <- fitted(ugarchsim(fit2, n.sim = N, m.sim = 1, startMethod = "
    sample"))

# Combine the simulated residuals with the standard normal samples
res1_correlated <- res1_sim * sd(y1simulated) + mean(y1simulated)
res2_correlated <- res2_sim * sd(y2simulated) + mean(y2simulated)

# Compute the portfolio returns
portsim <- matrix(0, nrow = N, ncol = 1)
portsim=log(1+((exp(y1simulated)-1)+(exp(y2simulated)-1))*(1/2))
varsim <- matrix(0, nrow = 1, ncol = 2)
varsim <- quantile(portsim, c(0.01, 0.05))
varsim
```

Listing 20: Task 2 (b) R code 15

2.2.2 Result

The estimated 99% and 95% 1-week VaR of the portfolio are -0.0550% and -0.0358%, which represent that if we invest one million in the portfolio, there are 1% and 5% probability of earning more than 550 and 358 during a single week.

2.2.3 Strengths and Weaknesses of Monte Carlo Simulation Approach

Strengths The β_1 coefficient in GARCH models governs the persistence of conditional variance, resulting in an improved fit to actual financial data issues. Additionally, simulation approach is flexible to different models, and all kinds of distribution can be modeled. In particular, it can define multivariate distributions and model multivariate data.

Weaknesses Copula-based Monte Carlo simulations rely on historical data to estimate dependence structures. When an event such as the 2008 recession happens, the model is unable to forecast future risk. It requires a large number of iterations for accurate VaR estimation, which can be computationally expensive and time-consuming. Also, the model is highly dependent on assumptions, but the assumptions may not accurately reflect reality. In this approach, we had to estimate the marginal distributions and the dependence structure, which could lead to an estimation error. Additionally, Monte Carlo simulations produce different results each time due to their reliance on random sampling. By setting a random seed, we can ensure consistent random number sequences and achieve the same outcomes in each simulation run.

2.3 Part c

The differences between 99% and 95% VaR under the parametric and Monte Carlo simulation approaches are 0.0017% and 0.0044% respectively, which are insignificant given the portfolio sizes. Based on these results, it is difficult to justify which approach is better; however, the Monte Carlo model would be more suitable because of the following reasons.

The selection of approach should depend on the distribution of the portfolio returns. Although the parametric method is simple, it is only useful for estimating the data with a bivariate normal distribution. The stock indices we used, SSE and CAC40, do not fit in a bivariate normal distribution by the Jarque-Bera Test, so the parametric approach may wrongly estimate the risk of our portfolio return. Specifically, it underestimates the risk of portfolio return since both VaR results are larger than those in the Monte Carlo approach.

On the other hand, Monte Carlo simulation does not have a normality assumption and can be applied in times of frequent financial disturbance. Although due to random nature of the simulation the outcome may vary between iterations, it was possible to fix this using R.

Thus, in order to deliver a comparatively accurate result, Monte Carlo model would be preferred.