```r
suppressPackageStartupMessages({
  library(VineCopula) # Copula analysis
  library(ADGofTest) # Anderson-Darling Goodness-of-Fit test
  library(KScorrect) # (Lilliefors-Corrected) Kolmogorov-Smirnov Goodness-of-Fit test
  library(fGarch) # Time series analysis
  library(tseries) # Time series analysis
  library(zoo) # Data preparation
})
```

Data Preparation

Note that since it's common practice to use **adjusted closed prices**, which would account for dividends, stock splits and other factors that may significantly affect the performance of the indices, we have chosen to analyse the weekly adjusted close prices from **2000-01-01 to 2018-12-31**.

Before selecting our two indices, we import data for all six indices from Yahoo to check for missing data in each index. We concluded that it would be best to choose an index with minimal or no missing values, as this would provide the most accurate results.

```r
FTSE100 <- get.hist.quote(instrument = "^FTSE", start = "2000-01-01", end = "2018-12-31",
↪  quote = "Adjusted", provider = "yahoo", compression = "w")
```

```
## time series ends   2018-12-29
```

```r
SP500 <- get.hist.quote(instrument = "^GSPC", start = "2000-01-01", end = "2018-12-31",
↪  quote = "Adjusted", provider = "yahoo", compression = "w")
```

```
## time series ends   2018-12-29
```

```r
SSE <- get.hist.quote(instrument = "000001.SS", start = "2000-01-01", end = "2018-12-31",
↪  quote = "Adjusted", provider = "yahoo", compression = "w")
```

```
## time series ends   2018-12-29
```

```r
DAX <- get.hist.quote(instrument = "^GDAXI", start = "2000-01-01", end = "2018-12-31",
↪  quote = "Adjusted", provider = "yahoo", compression = "w")
```

```
## time series ends   2018-12-29
```

```r
Nikkei225 <- get.hist.quote(instrument = "^N225", start = "2000-01-01", end =
↪  "2018-12-31", quote = "Adjusted", provider = "yahoo", compression = "w")
```

```
## time series ends   2018-12-29
```

```r
CAC40 <- get.hist.quote(instrument = "^FCHI", start = "2000-01-01", end = "2018-12-31",
↪  quote = "Adjusted", provider = "yahoo", compression = "w")
```

```
## time series ends   2018-12-29
```

```r
# Return names of indices with no missing values
names(Filter(function(x)all(!is.na(x)), list(FTSE100 = FTSE100, SP500 = SP500, SSE = SSE,
↪  DAX = DAX, Nikkei225 = Nikkei225, CAC40 = CAC40)))
```

```
## [1] "FTSE100" "CAC40"
```

Therefore, we selected **FTSE100** and **CAC40** as our chosen indices, as they contained no missing values.

Next, we proceeded to calculate the **log returns** for each index. Note that log returns are defined as the natural logarithms of net returns.

```r
# Calculate FTSE100 weekly log-returns
FTSE100 <- as.data.frame(coredata(diff(log(FTSE100))))
colnames(FTSE100) <- "FTSE100"

# Calculate CAC40 weekly log-returns
CAC40 <- as.data.frame(coredata(diff(log(CAC40))))
colnames(CAC40) <- "CAC40"

# Combine the log-returns data into a single dataset
Indices_log_returns <- cbind(FTSE100, CAC40)
save(Indices_log_returns, file = "Indices_log_returns.RData")
```

Question (a)

With our data now prepared, we will proceed to determine the Value-at-Risk using the **parametric approach**.

The parametric approach using the Variance-Covariance method, assuming that two indices' log returns follow a jointly normal distribution with the portfolio's mean and variance calculated by the following formulas:

$$\mu_P = w_{FTSE100}\mu_{FTSE100} + w_{CAC40}\mu_{CAC40}$$

$$\sigma_P^2 = w_{FTSE100}^2\sigma_{FTSE100}^2 + w_{CAC40}^2\sigma_{CAC40}^2 + 2w_{FTSE100}w_{CAC40}\rho\sigma_{FTSE100}\sigma_{CAC40}$$

First, we will calculate the mean and variance of each index's log returns, as well as the correlation coefficient between the their log returns. Then, we will create an **equally weighted portfolio** ($w_{FTSE100} = w_{CAC40} = 0.5$) and determine its mean and variance using the above formulas. Finally, we will use the inverse normal distribution with the portfolio mean and standard deviation to derive the 99% and 95% VaR.

```r
FTSE100 <- Indices_log_returns$FTSE100
CAC40 <- Indices_log_returns$CAC40

# Calculate mean, standard deviation, and correlation
FTSE100_mu <- mean(FTSE100)
FTSE100_sigma <- sd(FTSE100)
CAC40_mu <- mean(CAC40)
CAC40_sigma <- sd(CAC40)
correlation <- cor(FTSE100, CAC40)
```

```r
# Calculate the portfolio mean and variance
port_mu <- 1/2 * (FTSE100_mu + CAC40_mu)
port_variance <- (1/2)^2 * (FTSE100_sigma^2 + CAC40_sigma^2) + 2 * 1/2 * 1/2 *
↪  correlation * FTSE100_sigma * CAC40_sigma

# Estimated 99% and 95% VaR
port_VaR <- qnorm(c(0.01, 0.05), mean = port_mu, sd = sqrt(port_variance))
message("Estimated 99% 1-week VaR: ", port_VaR[1], "\n", "Estimated 95% 1-week VaR: ",
↪  port_VaR[2])
```

```
## Estimated 99% 1-week VaR: -0.0604024160428703
## Estimated 95% 1-week VaR: -0.0427261138196969
```

Table 2: Estimated 99% and 95% VaR using the parametric approach:

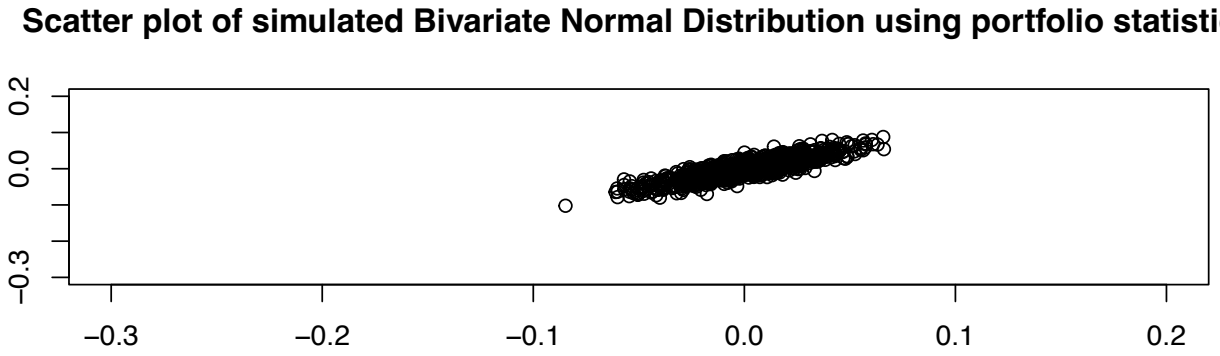
	99% Portfolio Value-at-Risk	**95% Portfolio Value-at-Risk**
Estimated Value-at-Risk	0.060402	0.042726

Strengths and weaknesses of using the parametric approach and log returns [Word Count 200]: Using the parametric approach to estimate the VaR for FTSE100 and CAC40 is computationally faster and easier to interpret than other approaches because it only requires the mean, variance, and correlation of the log returns, along with the well-known normal distribution.

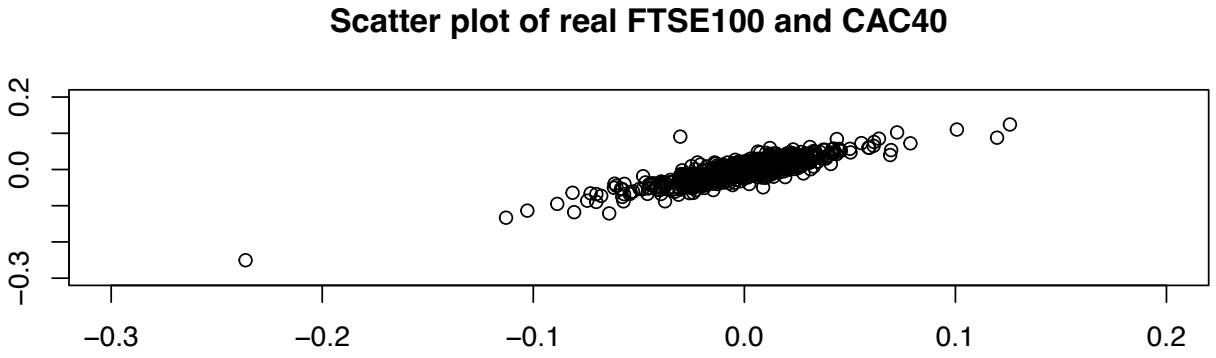
However, the simplicity of the approach comes with some costs. While it assumes that the log returns follow a bivariate normal distribution, our data exhibits non-normal behaviour, such as skewness and fatter tails (see the graph below and the formal tests in Question (b)). Also, the linear correlation coefficient is not enough to explain the nonlinear dependence structure in our data. As a result, the approach may not effectively capture these characteristics, underestimating portfolio risks.

Another drawback is the violation of the assumptions of constant volatility in our data, which may be due to the effects of some significant changes in various market factors over time. This limits the approach's ability to account for the uncertainty during periods of high market stress, compromising the estimation accuracy.

By using log returns, we obtain the flexibility of performing multi-period analysis, as they are additive. On the other hand, the estimated results may not be accurate, as log returns approximate worse for larger net returns, which is very likely for our weekly (instead of daily) data.



We simulate a bivariate normal distribution under parametric approach's assumption.



We observe more dispersing tails and more extreme values in our data.

Question (b)

Step 1: Identification

Before proceeding with Monte Carlo approach, we need to check if the approach is suitable for our data. Hence, we carry out a formal Jarque-Bera goodness-of-fit test to test whether the log returns of both indices have skewness and kurtosis similar to a normal distribution:

```
jarque.bera.test(FTSE100)
```

```
##
##  Jarque Bera Test
##
## data:  FTSE100
## X-squared = 6052.8, df = 2, p-value < 2.2e-16
```

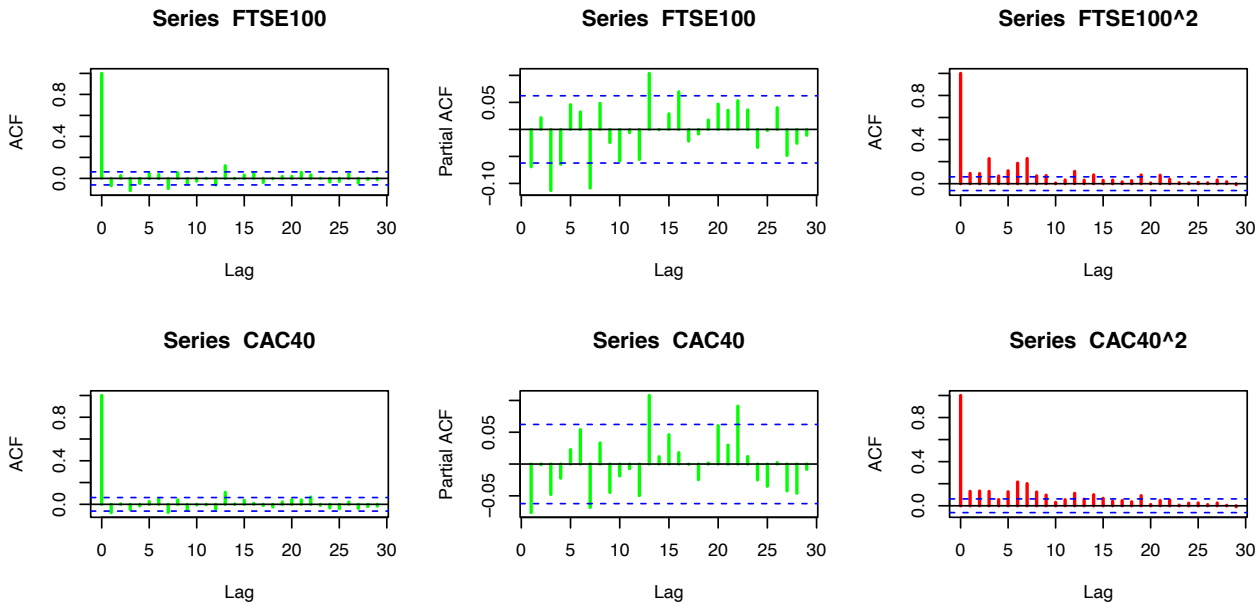
```
jarque.bera.test(CAC40)
```

```
##
##  Jarque Bera Test
##
## data:  CAC40
## X-squared = 1866.7, df = 2, p-value < 2.2e-16
```

The results of the test (tiny p-values of $< 2.2e\text{-}16$) indicate there's enough evidence to **reject the null hypothesis** that the marginal distributions are normally distributed. As result, we conclude that our data has non-normal-like behaviours and we need to specify a different distribution for our log returns and deem the parametric approach which assumes a jointly normal distribution to be unsuitable.

Next, we determine if AR and GARCH models are suitable for modelling our data. We plot auto-correlation functions (ACF), partial auto-correlation functions (PACF) and ACF squared for both indices, to determine the lags of the model and inclusion of the GARCH effect:

```
par(mfrow = c(2, 3))
acf(FTSE100, col = "green", lwd = 2)
pacf(FTSE100, col = "green", lwd = 2)
acf(FTSE100^2, col = "red", lwd = 2)
acf(CAC40, col = "green", lwd = 2)
pacf(CAC40, col = "green", lwd = 2)
acf(CAC40^2, col = "red", lwd = 2)
```




```
par(mfrow = c(1, 1))
```

We should note each plot measures significance of a lag using a 95% confidence interval, where all lags (below 10) that exceed this threshold (indicated by the dashed blue line) are statistically significant in terms of their correlations with the original time series.

We begin by analysing the **FTSE100** index plots, and observe the following:

In the PACF plot, we observe multiple significant autocorrelation spikes (namely lags **1, 3, 4, and 7**) and a geometric decay of the lags in the ACF plot. This indicates the time series is non-random and the **presence of autocorrelation** (suggested by significant spikes in both ACF and PACF plots for the same lags). Additionally, analysis of the ACF squared plot indicates significant spikes, suggesting the **presence of the GARCH effect** in our data.

Next, we analyse the **CAC40** index plots, and observe the following:

In the PACF plot, we observe multiple significant autocorrelation spikes (namely lags **1, and 7**) and a geometric decay of the lags in the ACF plot. This indicates the time series is non-random and the **presence of autocorrelation** (suggested by significant spikes in both ACF and PACF plots for the same lags).

Additionally, analysis of the ACF squared plot, indicates significant spikes, suggesting the **presence of the GARCH effect** in our data.

So, we conclude from our observations of the plots, our models for both log returns must **include both the AR and GARCH models** to account for the volatility clustering (GARCH effect) and autocorrelation.

Given many possible combinations of AR and GARCH (from `garch(1, 1)` up to `garch(3, 3)`) models, we use a for loop to further determine the best-fit combination by **minimizing the AIC and BIC** for the following conditional distributions: Normal (`norm`), Skew Normal (`snorm`), Student-t (`std`), Skew Student-t (`sstd`), Generalized Error (`ged`), and Skew Generalized Error (`sged`). The following block of codes will return a table showing, for each conditional distribution, the combinations of AR and GARCH models that have the lowest AIC and BIC values.

FTSE100 For index FTSE100:

```
# Create a data frame to store the results
FTSE100_ICs_results <- data.frame(Lowest_AIC_Model = c("","","","","",""), AIC = c(0, 0,
↪  0, 0, 0, 0), Lowest_BIC_Model = c("","","","","",""), BIC = c(0, 0, 0, 0, 0, 0),
↪  row.names = c("Normal (norm)", "Skew Normal (snorm)", "Student-t (std)", "Skew
↪  Student-t (sstd)", "Generalized Error (ged)", "Skew Generalized Error (sged)"))

# Normal
ICs <- data.frame(Model = "", AIC = 0, BIC = 0)
q <- 0
for (k in c(1, 3, 4, 7)){
  for (i in 1:3){
   for (j in 1:3){
    q <- q + 1
    fit <- garchFit(substitute(~ arma(r, 0) + garch(p, q), list(p = i, q = j, r = k)),
↪  data = FTSE100, trace = F, cond.dist = 'norm')
    ICs <- rbind(ICs, data.frame(Model = paste("AR(", k, ")", "+ garch(", i, ",", j,
↪  ")"), AIC = fit@fit$ics[[1]], BIC = fit@fit$ics[[2]]))
    }
  }
}
FTSE100_ICs_results[1, 1] <- ICs[which.min(ICs$AIC),][[1]]
FTSE100_ICs_results[1, 2] <- ICs[which.min(ICs$AIC),][[2]]
FTSE100_ICs_results[1, 3] <- ICs[which.min(ICs$BIC),][[1]]
FTSE100_ICs_results[1, 4] <- ICs[which.min(ICs$BIC),][[3]]

# Skew Normal
ICs <- data.frame(Model = "", AIC = 0, BIC = 0)
q <- 0
for (k in c(1, 3, 4, 7)){
  for (i in 1:3){
   for (j in 1:3){
    q <- q + 1
    fit <- garchFit(substitute(~ arma(r, 0) + garch(p, q), list(p = i, q = j, r = k)),
↪  data = FTSE100, trace = F, cond.dist = 'snorm')
    ICs <- rbind(ICs, data.frame(Model = paste("AR(", k, ")", "+ garch(", i, ",", j,
↪  ")"), AIC = fit@fit$ics[[1]], BIC = fit@fit$ics[[2]]))
    }
  }
}
```

```
FTSE100_ICs_results[2, 1] <- ICs[which.min(ICs$AIC),][[1]]
FTSE100_ICs_results[2, 2] <- ICs[which.min(ICs$AIC),][[2]]
FTSE100_ICs_results[2, 3] <- ICs[which.min(ICs$BIC),][[1]]
FTSE100_ICs_results[2, 4] <- ICs[which.min(ICs$BIC),][[3]]

# Student-t
ICs <- data.frame(Model = "", AIC = 0, BIC = 0)
q <- 0
for (k in c(1, 3, 4, 7)){
  for (i in 1:3){
   for (j in 1:3){
    q <- q + 1
    fit <- garchFit(substitute(~ arma(r, 0) + garch(p, q), list(p = i, q = j, r = k)),
↪  data = FTSE100, trace = F, cond.dist = 'std')
    ICs <- rbind(ICs, data.frame(Model = paste("AR(", k, ")", "+ garch(", i, ",", j,
↪  ")"), AIC = fit@fit$ics[[1]], BIC = fit@fit$ics[[2]]))
    }
  }
}
FTSE100_ICs_results[3, 1] <- ICs[which.min(ICs$AIC),][[1]]
FTSE100_ICs_results[3, 2] <- ICs[which.min(ICs$AIC),][[2]]
FTSE100_ICs_results[3, 3] <- ICs[which.min(ICs$BIC),][[1]]
FTSE100_ICs_results[3, 4] <- ICs[which.min(ICs$BIC),][[3]]

# Skew Student-t
ICs <- data.frame(Model = "", AIC = 0, BIC = 0)
q <- 0
for (k in c(1, 3, 4, 7)){
  for (i in 1:3){
   for (j in 1:3){
    q <- q + 1
    fit <- garchFit(substitute(~ arma(r, 0) + garch(p, q), list(p = i, q = j, r = k)),
↪  data = FTSE100, trace = F, cond.dist = 'sstd')
    ICs <- rbind(ICs, data.frame(Model = paste("AR(", k, ")", "+ garch(", i, ",", j,
↪  ")"), AIC = fit@fit$ics[[1]], BIC = fit@fit$ics[[2]]))
    }
  }
}
FTSE100_ICs_results[4, 1] <- ICs[which.min(ICs$AIC),][[1]]
FTSE100_ICs_results[4, 2] <- ICs[which.min(ICs$AIC),][[2]]
FTSE100_ICs_results[4, 3] <- ICs[which.min(ICs$BIC),][[1]]
FTSE100_ICs_results[4, 4] <- ICs[which.min(ICs$BIC),][[3]]

# Generalized Error
ICs <- data.frame(Model = "", AIC = 0, BIC = 0)
q <- 0
for (k in c(3, 4, 7)){
  for (i in 1:3){
   for (j in 1:3){
    q <- q + 1
    fit <- garchFit(substitute(~ arma(r, 0) + garch(p, q), list(p = i, q = j, r = k)),
↪  data = FTSE100, trace = F, cond.dist = 'ged')
    ICs <- rbind(ICs, data.frame(Model = paste("AR(", k, ")", "+ garch(", i, ",", j,
↪  ")"), AIC = fit@fit$ics[[1]], BIC = fit@fit$ics[[2]]))
```

```
    }
  }
}
FTSE100_ICs_results[5, 1] <- ICs[which.min(ICs$AIC),][[1]]
FTSE100_ICs_results[5, 2] <- ICs[which.min(ICs$AIC),][[2]]
FTSE100_ICs_results[5, 3] <- ICs[which.min(ICs$BIC),][[1]]
FTSE100_ICs_results[5, 4] <- ICs[which.min(ICs$BIC),][[3]]

# Skew Generalized Error
ICs <- data.frame(Model = "", AIC = 0, BIC = 0)
q <- 0
for (k in c(3, 4, 7)){
  for (i in 1:3){
   for (j in 1:3){
    q <- q + 1
    fit <- garchFit(substitute(~ arma(r, 0) + garch(p, q), list(p = i, q = j, r = k)),
↪  data = FTSE100, trace = F, cond.dist = 'sged')
    ICs <- rbind(ICs, data.frame(Model = paste("AR(", k, ")", "+ garch(", i, ",", j,
↪  ")"), AIC = fit@fit$ics[[1]], BIC = fit@fit$ics[[2]]))
    }
  }
}
FTSE100_ICs_results[6, 1] <- ICs[which.min(ICs$AIC),][[1]]
FTSE100_ICs_results[6, 2] <- ICs[which.min(ICs$AIC),][[2]]
FTSE100_ICs_results[6, 3] <- ICs[which.min(ICs$BIC),][[1]]
FTSE100_ICs_results[6, 4] <- ICs[which.min(ICs$BIC),][[3]]

FTSE100_ICs_results
```

```
##                                       Lowest_AIC_Model       AIC
## Normal (norm)                AR( 4 ) + garch( 1 , 1 ) -4.870823
## Skew Normal (snorm)          AR( 4 ) + garch( 1 , 1 ) -4.912673
## Student-t (std)              AR( 4 ) + garch( 1 , 1 ) -4.939202
## Skew Student-t (sstd)        AR( 7 ) + garch( 1 , 1 ) -4.972314
## Generalized Error (ged)      AR( 4 ) + garch( 1 , 1 ) -4.916930
## Skew Generalized Error (sged) AR( 7 ) + garch( 1 , 1 ) -4.962513
##                                       Lowest_BIC_Model       BIC
## Normal (norm)                AR( 3 ) + garch( 1 , 1 ) -4.835321
## Skew Normal (snorm)          AR( 3 ) + garch( 1 , 1 ) -4.869965
## Student-t (std)              AR( 1 ) + garch( 1 , 1 ) -4.899696
## Skew Student-t (sstd)        AR( 3 ) + garch( 1 , 1 ) -4.919508
## Generalized Error (ged)      AR( 3 ) + garch( 1 , 1 ) -4.875888
## Skew Generalized Error (sged) AR( 4 ) + garch( 1 , 1 ) -4.908987
```

The result shows that an `AR(7) + garch(1, 1)` model with conditional distribution `Skew Student-t (sstd)` has the lowest AIC; and an `AR(3) + garch(1, 1)` model with conditional distribution `Skew Student-t (sstd)` has the lowest BIC.

CAC40 For index CAC40, we perform exactly the same step as above (codes not shown again):

```
##                                       Lowest_AIC_Model       AIC
## Normal (norm)                AR( 7 ) + garch( 2 , 3 ) -4.426896
```

```
## Skew Normal (snorm)           AR( 7 ) + garch( 2 , 3 ) -4.482006
## Student-t (std)               AR( 7 ) + garch( 1 , 1 ) -4.476179
## Skew Student-t (sstd)         AR( 7 ) + garch( 1 , 1 ) -4.516670
## Generalized Error (ged)       AR( 7 ) + garch( 1 , 1 ) -4.457174
## Skew Generalized Error (sged) AR( 7 ) + garch( 1 , 1 ) -4.506512
##                                       Lowest_BIC_Model       BIC
## Normal (norm)                 AR( 1 ) + garch( 1 , 1 ) -4.389980
## Skew Normal (snorm)           AR( 1 ) + garch( 1 , 1 ) -4.428693
## Student-t (std)               AR( 1 ) + garch( 1 , 1 ) -4.439145
## Skew Student-t (sstd)         AR( 1 ) + garch( 1 , 1 ) -4.462914
## Generalized Error (ged)       AR( 1 ) + garch( 1 , 1 ) -4.418344
## Skew Generalized Error (sged) AR( 1 ) + garch( 1 , 1 ) -4.452093
```

The result shows that an `AR(7) + garch(1, 1)` model with conditional distribution `Skew Student-t (sstd)` has the lowest AIC; and an `AR(1) + garch(1, 1)` model with conditional distribution `Skew Student-t (sstd)` has the lowest BIC.

The AIC and BIC suggest different combinations for our models, we decided to use the **BIC** as our final reference for model selection. The main reason for this choice is that BIC imposes a stronger penalty for model complexity than AIC. While over-fitting can be a significant concern in financial time series modelling, choosing BIC as our reference would reduce the risk of over-fitting our log returns. Thus, we will use an **`AR(3) + garch(1, 1)`** model with conditional distribution **`Skew Student-t (sstd)`** for **FTSE100**, and an **`AR(1) + garch(1, 1)` model** with conditional distribution **`Skew Student-t (sstd)`** for **CAC40**.

Step 2: Estimation

We have now determined the combinations of AR, GARCH, and conditional distribution with the lowest BIC to model our log returns, so we now proceed with the estimation.

According to the results from **Step 1**, we fit the following two models for FTSE100 and CAC40:

```
# FTSE100
model1 <- garchFit(formula = ~ arma(3, 0) + garch(1, 1), data = FTSE100, trace = F,
↪  cond.dist = "sstd")

# CAC40
model2 <- garchFit(formula = ~ arma(1, 0) + garch(1, 1), data = CAC40, trace = F,
↪  cond.dist = "sstd")
```

Step 3: Model Checking

We now carry out diagnostics for the fitted models to determine if the autocorrelation and GARCH effect have been eliminated.

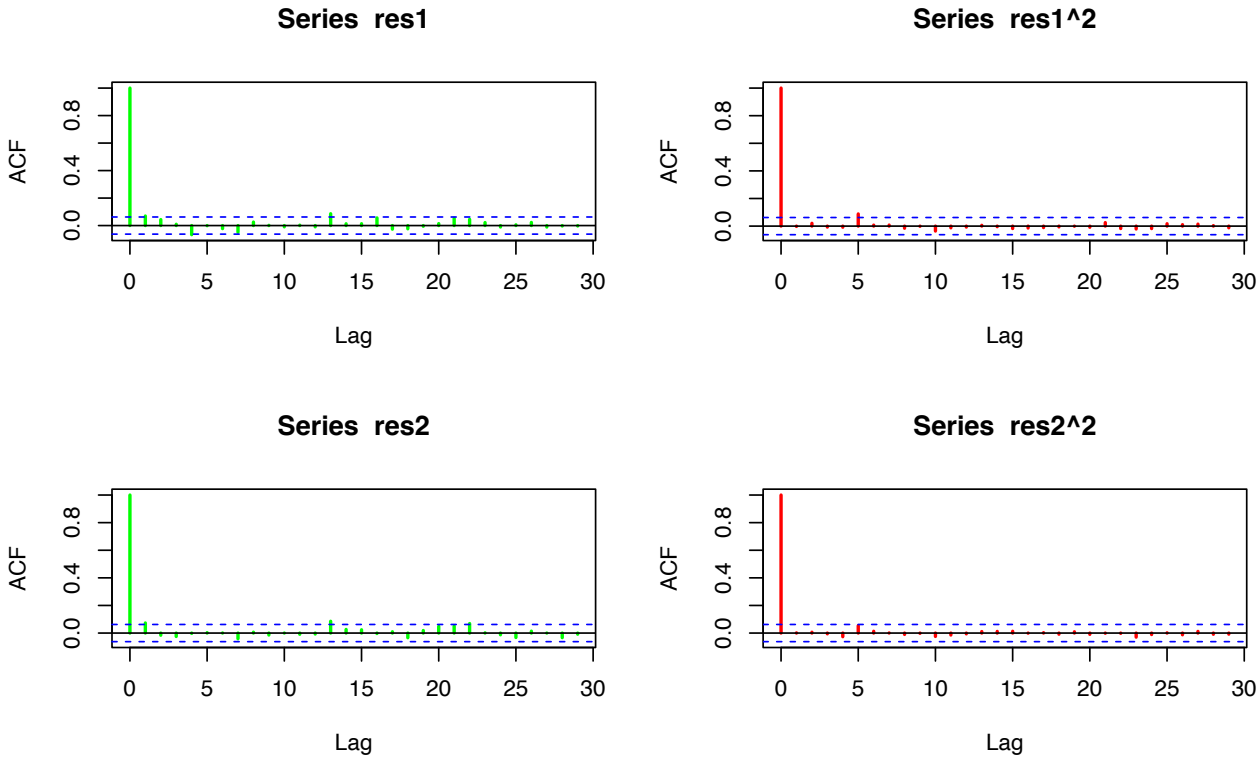
First, we compute the standardised residuals for two models:

```
res1 <- residuals(model1, standardize = TRUE)
res2 <- residuals(model2, standardize = TRUE)
```

Then, we analyse the ACF plots of (squared) standardised residuals to assess our models' effectiveness in eliminating autocorrelation and the GARCH effect. Compared with the original ACF plots of the (squared) indices log returns, both the number of lags spiking beyond the 95% confidence threshold and the magnitudes of the correlation decrease significantly. This suggests that our models have adequately captured the

underlying structure and volatility dynamics of the indices log returns. However, we can still observe a very limited number of significant spikes on both plots. Thus, we would need to formally test for the absence of autocorrelation and GARCH effect within our data.

```
# Plots
par(mfrow = c(2, 2))
acf(res1, col = "green", lwd = 2)
acf(res1^2, col = "red", lwd = 2)
acf(res2, col = "green", lwd = 2)
acf(res2^2, col = "red", lwd = 2)
```




```
par(mfrow = c(1, 1))
```

Following the visual inspection of the ACF plots, we proceed to carry out the formal **Box-Ljung** test to test for the presence of autocorrelation and the GARCH effect in the first 10 lags, as a period longer than 10 weeks is likely too extended to significantly affect current events, making it irrelevant for short-term risk analysis. Note that we set `fitdf` equal to the the lags of the AR models in `model1` and `mode2`. The null hypothesis of the Box-Ljung test is the autocorrelation in the data is zero.

```
# Box-Ljung Test
Box.test(res1, lag = 10, type = c("Ljung-Box"), fitdf = 3)
```

```
##
##  Box-Ljung test
##
## data:  res1
## X-squared = 14.753, df = 7, p-value = 0.03931
```

```
Box.test(res1^2, lag = 10, type = c("Ljung-Box"), fitdf = 3)
```

```
##
##  Box-Ljung test
##
## data:  res1^2
## X-squared = 9.9181, df = 7, p-value = 0.1933
```

```
Box.test(res2, lag = 10, type = c("Ljung-Box"), fitdf = 1)
```

```
##
##  Box-Ljung test
##
## data:  res2
## X-squared = 8.1533, df = 9, p-value = 0.5188
```

```
Box.test(res2^2, lag = 10, type = c("Ljung-Box"), fitdf = 1)
```

```
##
##  Box-Ljung test
##
## data:  res2^2
## X-squared = 4.5574, df = 9, p-value = 0.8711
```

After conducting the tests, we obtain **very high p-values** for the tests on the squared standardised residuals of model1, the standardised residuals of model2, and the squared standardised residuals of model2, resulting in the **rejection of the null hypothesis**.

However, the the p-value of the test on the standardised residuals of model1 is only 0.039, which is slightly below the 0.05 threshold. This may be due to the limitation of our models (we only use AR and GARCH models) and the relatively more conservative suggestion from BIC. As a result, we may try to increase the number of lags of the AR model for `model1`, although this would increase the BIC. We run the for loop in **Step 1** for **FTSE100** again, excluding AR(3) model for the conditional distribution **`Skew Student-t (sstd)`**, to get the combination of AR and GARCH models having the **second smallest** BIC result (codes not shown):

```
##                                     Lowest_AIC_Model       AIC
## Normal (norm)                AR( 4 ) + garch( 1 , 1 ) -4.870823
## Skew Normal (snorm)          AR( 4 ) + garch( 1 , 1 ) -4.912673
## Student-t (std)              AR( 4 ) + garch( 1 , 1 ) -4.939202
## Skew Student-t (sstd)        AR( 7 ) + garch( 1 , 1 ) -4.972314
## Generalized Error (ged)      AR( 4 ) + garch( 1 , 1 ) -4.916930
## Skew Generalized Error (sged) AR( 7 ) + garch( 1 , 1 ) -4.962513
##                                     Lowest_BIC_Model       BIC
## Normal (norm)                AR( 3 ) + garch( 1 , 1 ) -4.835321
## Skew Normal (snorm)          AR( 3 ) + garch( 1 , 1 ) -4.869965
## Student-t (std)              AR( 1 ) + garch( 1 , 1 ) -4.899696
## Skew Student-t (sstd)        AR( 4 ) + garch( 1 , 1 ) -4.919391
## Generalized Error (ged)      AR( 3 ) + garch( 1 , 1 ) -4.875888
## Skew Generalized Error (sged) AR( 4 ) + garch( 1 , 1 ) -4.908987
```

The result shows that after excluding AR(3) model for the conditional distribution `Skew Student-t (sstd)`, an **`AR(4) + garch(1, 1)`** model with conditional distribution **`Skew Student-t (sstd)`** has the lowest BIC.

Following the result above, we update our model for FTSE100 (`model1`) as follows:

```
# FTSE100
model1 <- garchFit(formula = ~ arma(4, 0) + garch(1, 1), data = FTSE100, trace = F,
↪  cond.dist = "sstd")
```

We carry out the formal **Box-Ljung** test to test for the presence of autocorrelation and the GARCH effect in the first 10 lags for the updated `model1` again:

```
res1 <- residuals(model1, standardize = TRUE)
Box.test(res1, lag = 10, type = c("Ljung-Box"), fitdf = 4)
```

```
##
##  Box-Ljung test
##
## data:  res1
## X-squared = 12.503, df = 6, p-value = 0.05165
```

```
Box.test(res1^2, lag = 10, type = c("Ljung-Box"), fitdf = 4)
```

```
##
##  Box-Ljung test
##
## data:  res1^2
## X-squared = 9.3475, df = 6, p-value = 0.155
```

The p-value of the test on the standardised residuals of the updated `model1` is now 0.05165, which is higher the 0.05 threshold. Thus, we conclude that our models have adequately address the issue of autocorrelation and GARCH effects in our data.

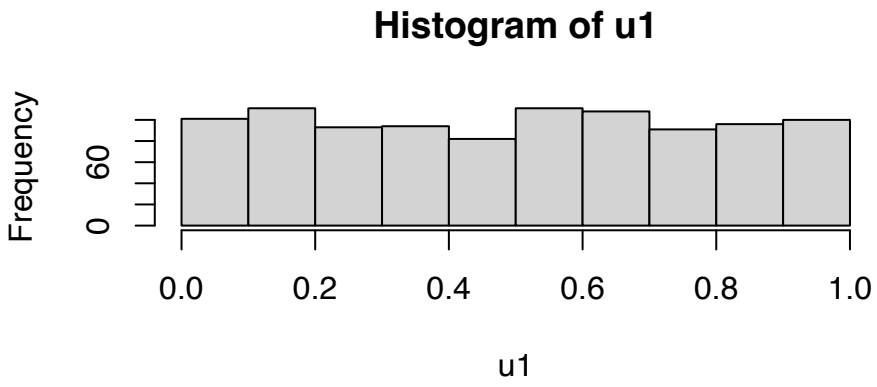
Given the conclusion above, we can proceed with carrying Probability Integral Transform (PIT) on the standardized residuals. If our chosen models are effective in capturing the behaviour of our data, we except to see approximate uniform distributions for the transformed results.

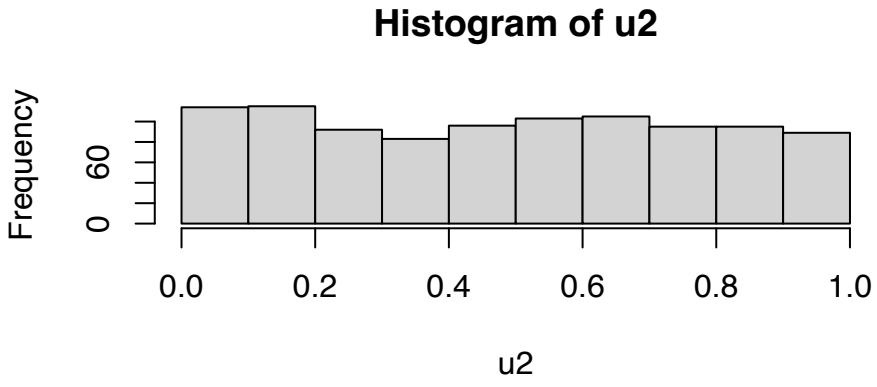
We apply Probability Integral Transform (PIT) on the standardised residuals to get two corresponding uniformly distributed variables, using the conditional distribution skewed student-t and the parameters estimated by ours models. Note that the length of our data begins from the 5^{th} transformed result as we fit an AR(4) model for index FTSE100 and the `BiCopSelect` function requires both residuals to be of the same length.

```
u1 <- psstd(res1, mean = 0, sd = 1, nu = model1@fit$par["shape"], xi =
↪  model1@fit$par["skew"])[5:length(FTSE100)]
u2 <- psstd(res2, mean = 0, sd = 1, nu = model2@fit$par["shape"], xi =
↪  model2@fit$par["skew"])[5:length(CAC40)]
```

Now, we will plot a histogram of our resulting transformed uniformly-distributed variables:

```r
# Histograms
par(mfrow = c(2, 1))
hist(u1)
hist(u2)
```






```r
par(mfrow = c(1, 1))
```

Although we observe an approximately uniform plot, indicating that our model is adequate, we cannot base our analysis solely on the results of one plot and whether it appears uniform or not. Therefore, we will proceed to carry out two formal tests for uniformity: the **Kolmogorov-Smirnov (KS) test** and the **Anderson-Darling (AD) test**. It is important to note that the null hypothesis for both tests is that the data follows a uniform distribution. Additionally, we should note that the key distinction between the two tests, while they both test for goodness of fit, is that the Anderson-Darling test is more sensitive to the tails of the distribution, making it more powerful in detecting non-uniformity in the tails of the distribution.

```r
# Kolmogorov-Smirnov test
KStest1 <- LcKS(u1, cdf = "punif")
KStest1$p.value
```

```
## [1] 0.7576
```

```r
KStest2 <- LcKS(u2, cdf = "punif")
KStest2$p.value
```

```
## [1] 0.1348
```

```
# Anderson-Darling test
ADtest1 <- ad.test(u1, null = "punif")
ADtest1$p.value
```

```
##        AD
## 0.8504737
```

```
ADtest2 <- ad.test(u2, null = "punif")
ADtest2$p.value
```

```
##        AD
## 0.1625578
```

After conducting the tests, we find that the p-values for both tests are much higher than the 0.05 threshold, indicating that we fail to reject the null hypothesis and **pass the test for uniformity** and that our models are good.

Step 4: Copula Modelling

After confirming that we've successfully obtained two uniformly distributed variables, we can now proceed to fit a copula model. We use the `BiCopSelect` function, which fits various copulas to our data sets and determines the best copula to model the dependence structure of the two input variables (`u1` and `u2`) based on minimising the BIC:

```
model <- BiCopSelect(u1, u2, familyset = NA, selectioncrit = "BIC", indeptest = TRUE,
↪  level = 0.05, se = TRUE)
model
```

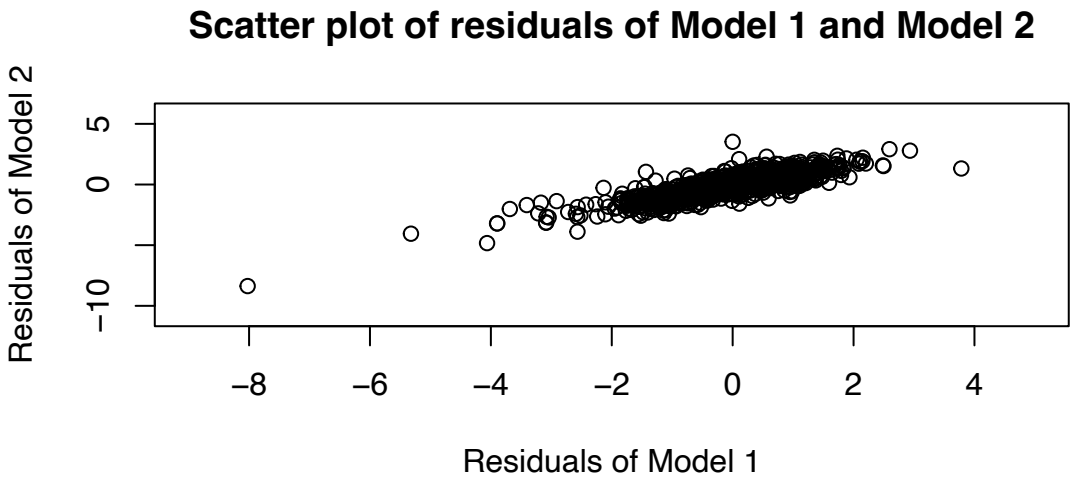
```
## Bivariate copula: Survival BB1 (par = 0.2, par2 = 2.53, tau = 0.64)
```

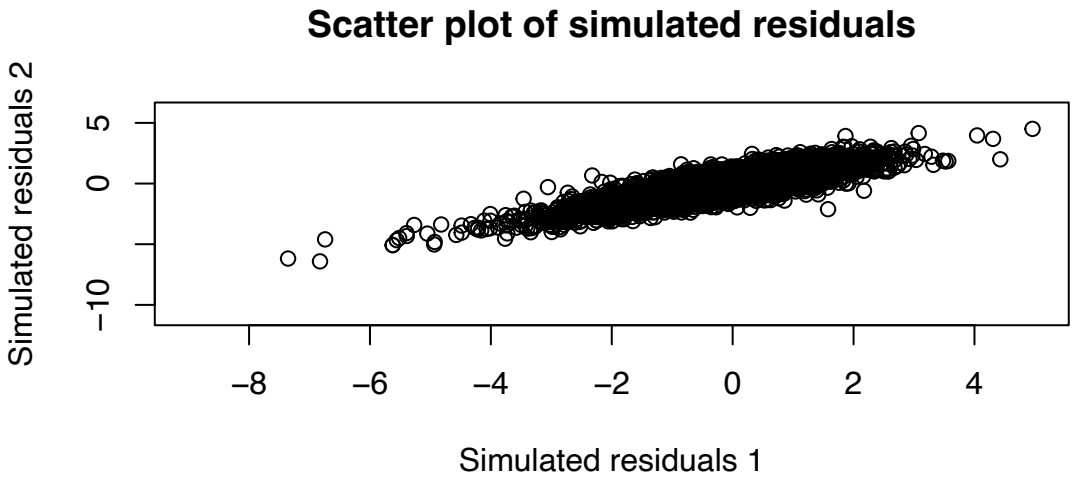
According to the result, the best-fit model based on minimising BIC is `Survival BB1 (par = 0.2, par2 = 2.53, tau = 0.64)`.

Next, we carry out Monte Carlo simulations using the best-fit copula model to simulate 20,000 uniformly distributed variables (10,000 `u1_sim` and 10,000 `u2_sim`) that capture the dependence structure of `u1` and `u2`. Then, to obtain the simulated residuals, we apply the **Inverse Probability Integral Transform (IPIT)** using the skewed student-t distribution and parameters estimated by our `model1` and `model2`:

```
N <- 10000
set.seed(888)
u_sim <- BiCopSim(N, obj = model)
# Apply Inverse Probability Integral Transform (IPIT) to the simulated results for both
↪  indices.
res1_sim <- qsstd(u_sim[,1], mean = 0, sd = 1, nu = model1@fit$par["shape"], xi =
↪  model1@fit$par["skew"])
res2_sim <- qsstd(u_sim[,2], mean = 0, sd = 1, nu = model2@fit$par["shape"], xi =
↪  model2@fit$par["skew"])
```

Following this, we plot the simulated residuals against each other and compare it to the plot of our original models' residuals:





We observe that the simulated residuals exhibit a similar pattern to the original residual, indicating we've successfully simulated the behaviour of the residuals.

The next step is to **reintroduce the GARCH effect and autocorrelation** to the data. We begin by extracting the parameters for the fitted AR and GARCH models (`model1` and `model2`), followed by using the equations below to reintroduce the GARCH effect and the autorrelation:

$$\hat{\sigma^2_{992}} = \hat{\omega} + \hat{\alpha} u^2_{991} + \hat{\beta}\hat{\sigma_{991}}$$

$$\hat{y_{992}} = \hat{\mu} + \hat{\alpha_1} y_{991} + ... + \hat{\alpha_k} y_{991-k+1} + \hat{\sigma_{992}}\varepsilon_{sim}$$

And after that we can simulate 10,000 1-week ahead log returns for each index.

```
## FTSE100
# Extract parameters from model1 for later calculations
mu_1 <- model1@fit$par["mu"]
ar1_1 <- model1@fit$par["ar1"]
```

```r
ar2_1 <- model1@fit$par["ar2"]
ar3_1 <- model1@fit$par["ar3"]
ar4_1 <- model1@fit$par["ar4"]
omega_1 <- model1@fit$par["omega"]
alpha_1 <- model1@fit$par["alpha1"]
beta_1 <- model1@fit$par["beta1"]

# Create a vector to store the simulated 1-week ahead log-returns
y1_sim <- numeric(N)

# Reintroduce the GARCH effects
sigma2s_1 <- omega_1 + alpha_1 * model1@residuals[991]^2 + beta_1 * model1@sigma.t[991]^2

# Reintroduce the AR effects and store the simulated results
for (i in 1:N) {
  y1_sim[i] <- mu_1 + ar1_1 * FTSE100[991] + ar2_1 * FTSE100[990] + ar3_1 * FTSE100[989]
↪  + ar4_1 * FTSE100[988] + sqrt(sigma2s_1) * res1_sim[i]
}

## CAC40
# Extract parameters from model2 for later calculations
mu_2 <- model2@fit$par["mu"]
ar1_2 <- model2@fit$par["ar1"]
omega_2 <- model2@fit$par["omega"]
alpha_2 <- model2@fit$par["alpha1"]
beta_2 <- model2@fit$par["beta1"]

# Create a vector to store the simulated 1-week ahead log-returns
y2_sim <- numeric(N)

# Reintroduce the GARCH effects
sigma2s_2 <- omega_2 + alpha_2 * model2@residuals[991]^2 + beta_2 * model2@sigma.t[991]^2

# Reintroduce the AR effects and store the simulated results
for (i in 1:N) {
  y2_sim[i] <- mu_2 + ar1_2 * CAC40[991] + sqrt(sigma2s_2) * res2_sim[i]
}
```

Finally, we construct an **equally weighted portfolio** and calculate its 99% and 95% VaR. Note that to construct a portfolio, we must convert the individual log returns back to net returns as log returns aren't additive over assets classes, and then convert back to get the portfolio log returns after the portfolio construction.

```r
port_sim <- matrix(0, nrow = N, ncol = 1)
VaR_sim <- matrix(0, nrow = 1, ncol = 2)

port_sim <- log(1 + ((exp(y1_sim) - 1) + (exp(y2_sim) - 1)) * (1/2))
VaR_sim <- quantile(port_sim, c(0.01, 0.05))
message("Estimated 99% 1-week VaR: ", VaR_sim[1], "\n", "Estimated 95% 1-week VaR: ",
↪  VaR_sim[2])
```

```
## Estimated 99% 1-week VaR: -0.05591217675269
## Estimated 95% 1-week VaR: -0.0327359029450898
```

Table 3: Estimated 99% and 95% using Monte Carlo simulation approach based on Copula theory:

	99% Portfolio Value-at-Risk	**95% Portfolio Value-at-Risk**
Estimated Value-at-Risk	0.055912	0.032736

Strengths and weaknesses of using the Monte Carlo simulation approach based on copula theory [Word Count 199]: Through this approach we have more freedom – modelling the dependence structure of the data. Since our data shows non-normal features such as heavy tails and more extreme values, the use of survival BB1 copula that deals with more flexible tail dependence structure would provide better fit. Combined with Monte Carlo simulations, this approach would yield more precise results, as it would better represent the joint behaviour our data during extreme events to which normality modelling are less robust.

This approach is also not without its drawbacks: it's very computationally inefficient and complicated to simulate a large number of the log returns and residuals. In particular this can lead to problems when the VaR estimate is requested frequently. Also, as we model our FTSE100 log returns, choosing the model based on BIC doesn't necessary guarantee the most suitable model. We need to assess other information, such as plots and test results to specify a choice in the marginal distribution and other parameters of the model. The uncertainty in this choice can have a bad effect on the estimated VaR. If the fitted model and the copula model are misspecified, then the estimated results may greatly deviate from the true value.

Question (c)

Table 4: Estimated Value-at-Risks using different approaches:

	Estimated 99% Value-at-Risk	**Estimated 95% Value-at-Risk**
Parametric Approach	0.060402	0.042726
Monte Carlo Simulation	0.055912	0.032736

[**Word Count 200**] The table above reveals that the Monte Carlo simulation approach based on copula theory returns smaller VaR estimates. Given that our data exhibits non-normal characteristics, this finding has important implications for the selection of the estimation technique.

First, the parametric approach relies heavily on the assumption of normal distribution, limiting its ability to account for the asymmetric distribution and higher probability of extreme values observed in our data. As a result, it's reasonable that we get inflated VaR estimates as this approach has fundamentally, expected by the normal distribution, assumed a lower occurrence of extreme values. On the other hand, the Monte Carlo simulation based on copula theory provides greater flexibility in modelling the dependence structures and marginal distribution of our data. Given that we specify an adequate model, it would explain more variation observed in our non-normal data. Also, rather than based purely on historical data, the implement of simulations allows getting results from a wider range of potential scenarios. Thus, we can expect that the VaR estimates by this approach would return more accurate representation of portfolio risks.

In sum, given the non-normal structure our data, we favour the Monte Carlo simulation based on copula theory for estimating VaR.