TASK 2

For this task, we aim to estimate VAR values using the Monte Carlo Simulation approach based on copula theory. Our portfolio will consist of FTSE AND CAC40.

Importing the relevant packages

```
library(VineCopula)
library(fGarch)
library(KScorrect)
library(stats)
library(ADGofTest)
library(tseries)
```

Obtaining the price history for FTSE and CAC40 between the years 2000 and 2021

```
p_ftse=matrix(get.hist.quote(instrument="^ftse", start="2000-01-01",
                             end="2021-01-01", quote="AdjClose", compression='w'))
```

```
## time series ends   2020-12-26
```

```
p_fchi=matrix(get.hist.quote(instrument="^fchi", start="2000-01-01",
                             end="2021-01-01", quote="AdjClose", compression='w'))
```

```
## time series ends   2020-12-26
```

Computes log differences.

```
p_ftse=diff(log(p_ftse))
p_fchi=diff(log(p_fchi))
```

Normality tests show that the log returns are not normally distributed. The qqplot is a visual representation of the distribution of the log returns. In both cases, it can be observed that the plotted points deviate from the normally distributed line in the extreme ends, indicating that the data has fat tails and so is unlikely to be normally distributed. These observations are supported by the p value of the Jarque-Bera test being smaller than 5%.

```
par(mfrow=c(1,2))
jarqueberaTest(p_ftse)@test
```

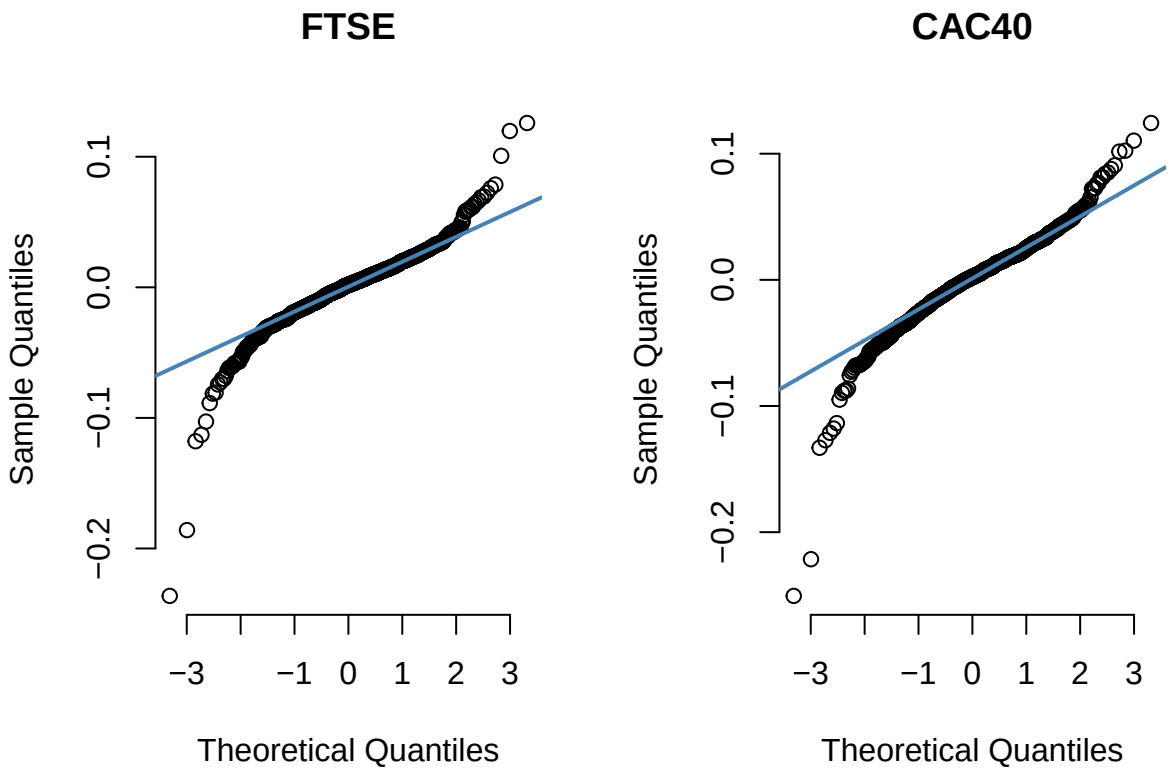
```
## $statistic
## X-squared
##  7337.474
##
## $p.value
## Asymptotic p Value
##                  0
##
## $method
## [1] "Jarque Bera Test"
##
## $data.name
## [1] "p_ftse"
```

```
qqnorm(p_ftse, pch = 1, frame = FALSE, main="FTSE")
qqline(p_ftse, col = "steelblue", lwd = 2)

jarqueberaTest(p_fchi)@test
```

```
## $statistic
## X-squared
##   2979.01
##
## $p.value
## Asymptotic p Value
##                  0
##
## $method
## [1] "Jarque Bera Test"
##
## $data.name
## [1] "p_fchi"
```

```
qqnorm(p_fchi, pch = 1, frame = FALSE, main="CAC40")
qqline(p_fchi, col = "steelblue", lwd = 2)
```

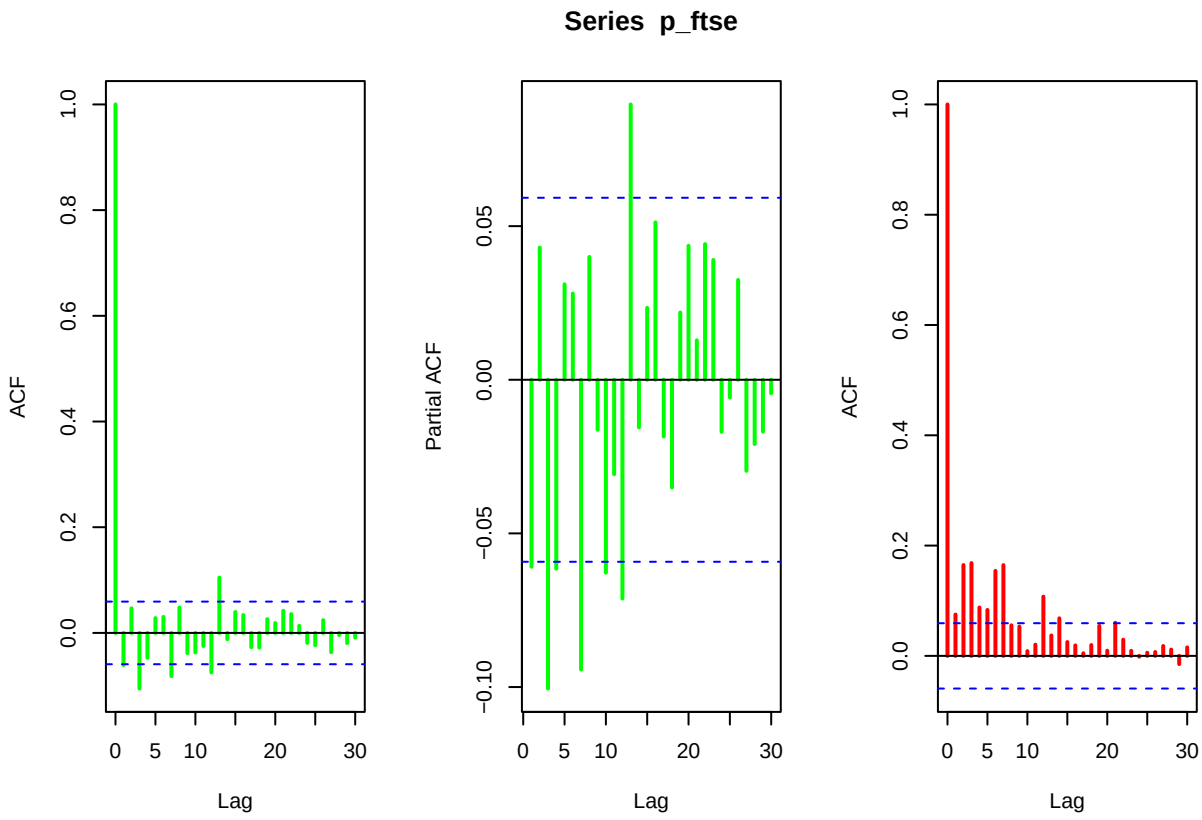


Building AR models: the Box - Jenkins approach

Step 1: Identification

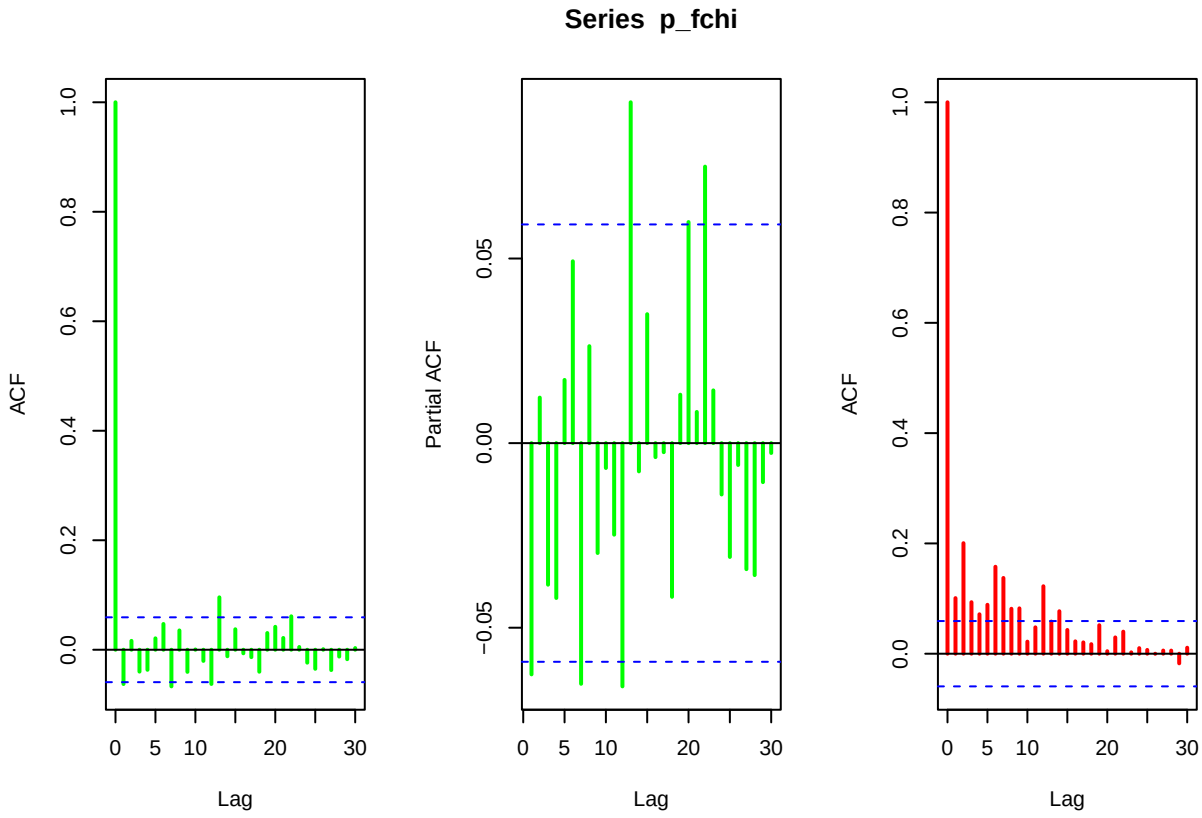
Returns 1: Significant values for the PACF indicate that the AR model can be of order 1, 3, 4 or 7. It is also clear that GARCH effects are present here as visible in the ACF of p_ftse^2.

```
par(mfrow=c(1,3))
acf(p_ftse, col="green", lwd=2, main="")
pacf(p_ftse, col="green", lwd=2)
acf(p_ftse^2, col="red", lwd=2, main="")
```



Returns 2: Significant values for the PACF indicate that the AR model can be of order 1 or 7. GARCH effects are also present here as visible in the ACF of p_fchi^2.

```
par(mfrow=c(1,3))
acf(p_fchi, col="green", lwd=2, main="")
pacf(p_fchi, col="green", lwd=2)
acf(p_fchi^2, col="red", lwd=2, main="")
```




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

Step 2: Estimation

To find the optimal models, we wrote a python script(see the r script file) that generates code for all the possible different types of models we could get, within a reasonable range of values, which we then ran in r. We exported this data to an excel file which made it easier to do visual checks using conditional formatting to identify the best model. We picked the models with the lowest BIC values that also passed the box and uniformity tests. BIC was used over AIC as the AIC tends to pick more complex models which we may not have encountered.

FTSE

```
model1=garchFit(formula=~arma(1,0)+garch(1,1),data=p_ftse,trace=F,cond.dist="sstd")
```

CAC40

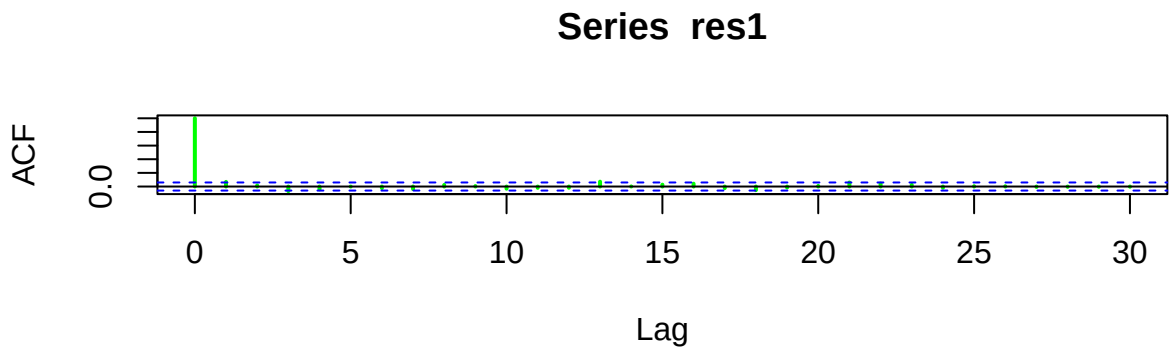
```
model2=garchFit(formula=~arma(1,0)+garch(1,1),data=p_fchi,trace=F,cond.dist="sstd")
```

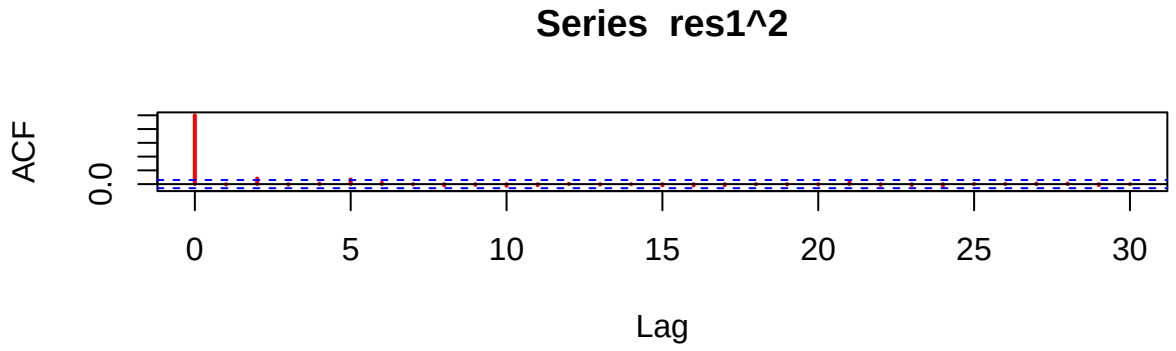
Step 3: Model checking

Model 1 Residuals:

ACF graphs and box tests provide visual and numerical tests to show that the residual data is not autocorrelated using a 5% confidence level. Res1 narrowly passes the box test as shown below. This could be due to the model not being the most accurate fit for the data, however, we can still use this as it has passed the confidence level for both box tests but we should note that this is a weakness of our model as it could be a false positive. Using this model, we have successfully modelled the AR and GARCH effects, since they are no longer present in the ACF plots of the residuals.

```
res1=residuals(model1, standardize=TRUE)
par(mfrow=c(2,1))
acf(res1, col="green", lwd=2)
acf(res1^2, col="red", lwd=2)
```






```
par(mfrow=c(1,1))
Box.test(res1, lag=10, type=c("Ljung-Box"), fitdf=1)#$p.value
```

```
##
##  Box-Ljung test
##
## data:  res1
## X-squared = 16.421, df = 9, p-value = 0.05859
```

```
Box.test(res1^2, lag=10, type=c("Ljung-Box"), fitdf=1)#$p.value
```

```
##
##  Box-Ljung test
##
## data:  res1^2
## X-squared = 14.204, df = 9, p-value = 0.1153
```

```
model1@fit$ics
```

```
##       AIC       BIC       SIC      HQIC
## -4.918682 -4.886728 -4.918763 -4.906590
```

The following computes the PIT values for the residuals and checks for uniformity using KS tests and AD

tests with 5% significance. The histogram allows for a visual check for the uniformity. Here we may observe that there is approximate uniformity in the data which is supported by the numerical tests. As we are using the arma(1,0) model for both cases, we notice that our first entry for u1 and u2 is 0 so we skip the first value in the data returned from the PIT function.

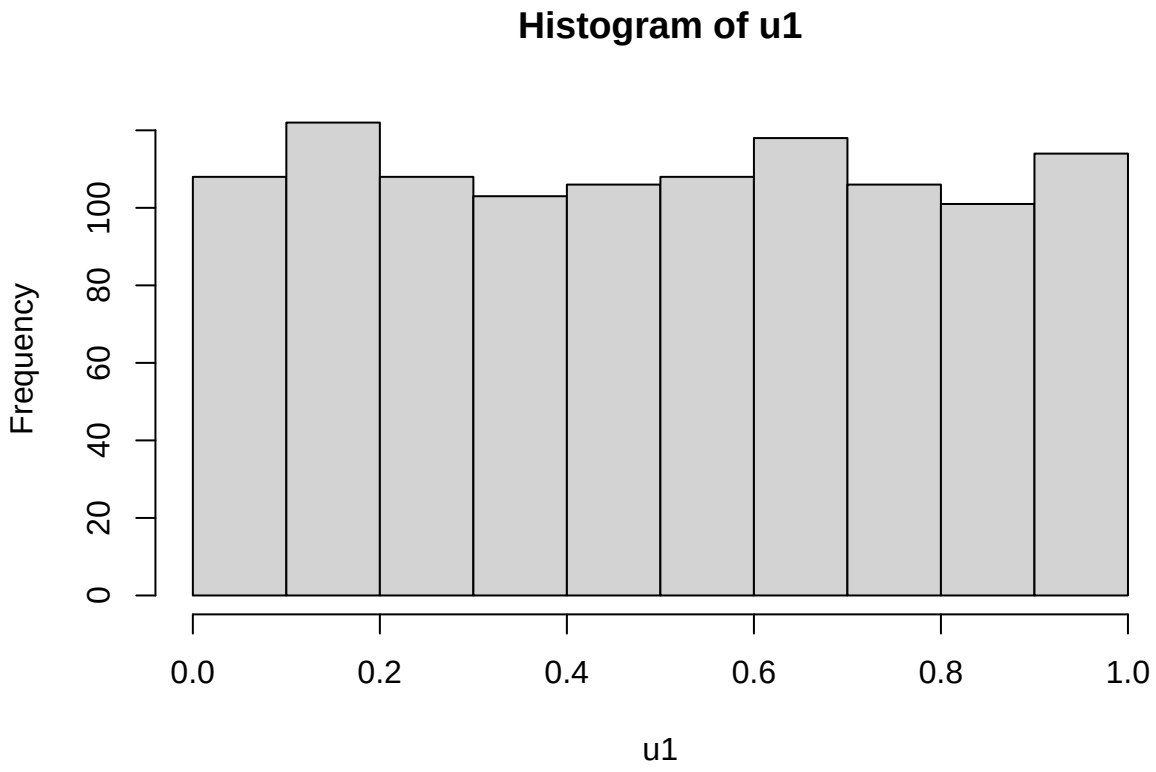
```
u1=psstd(res1, mean=0, sd=1, nu=tail(coef(model1), n=1),
         xi=coef(model1)[length(coef(model1))-1])[2:length(p_ftse)]
cat("KS test p value: ", LcKS(u1, cdf = "punif")$p.value, "\n")
```

```
## KS test p value:  0.8054
```

```
cat("AD test p value: ", ad.test(u1, null="punif")$p.value)
```

```
## AD test p value:  0.8731842
```

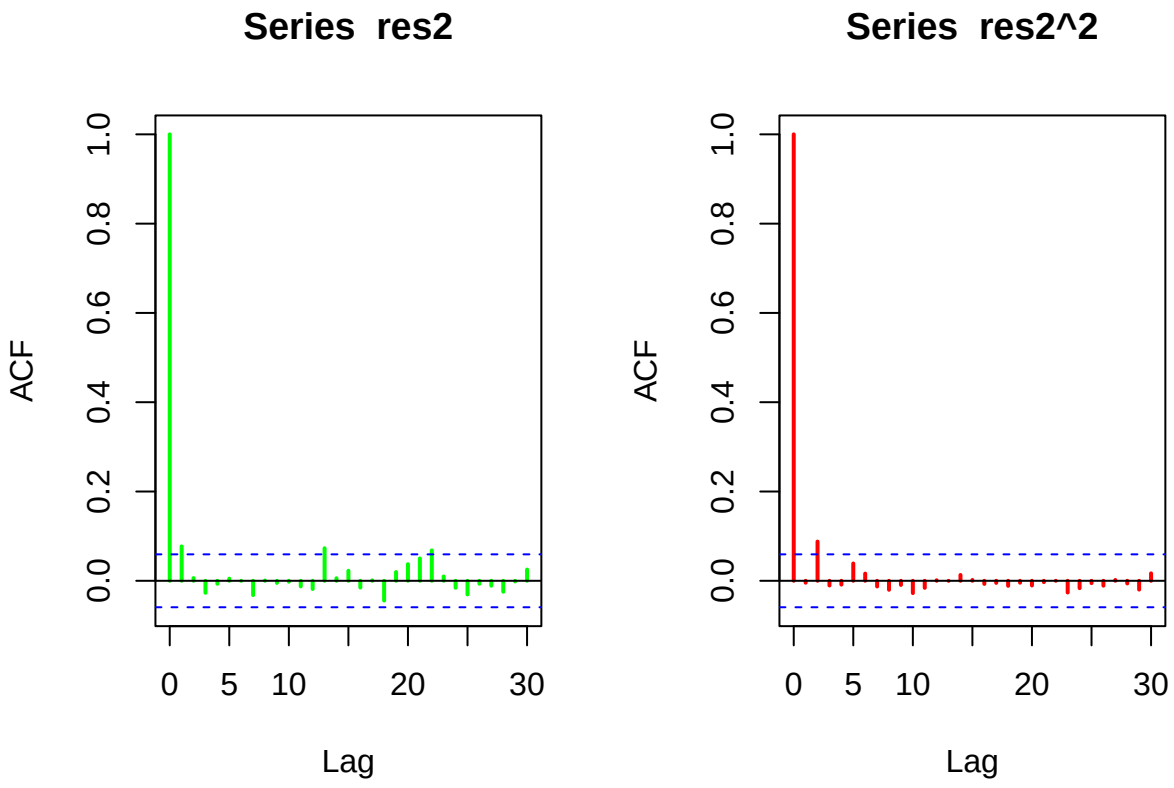
```
hist(u1)
```



Model 2 Residuals

We also compute the box tests and show the ACF plots for res2. In this case, the box tests provide a high p value, indicating that there is a higher chance of this being a suitable model for the CAC40 log returns. The ACF plots for res2 also show that the AR and GARCH effects have been modelled accurately.

```
res2=residuals(model2, standardize=TRUE)
par(mfrow=c(1,2))
acf(res2, col="green", lwd=2)
acf(res2^2, col="red", lwd=2)
```




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

```
##
##  Box-Ljung test
##
## data:  res2
## X-squared = 8.6093, df = 9, p-value = 0.4741
```

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

```
##
##  Box-Ljung test
##
## data:  res2^2
## X-squared = 12.243, df = 9, p-value = 0.1999
```

```
model2@fit$ics
```

```
##       AIC       BIC       SIC      HQIC
## -4.471672 -4.439718 -4.471753 -4.459581
```

Similarly for model2, the uniformity tests show that the PIT gives us a unform distribution, thus indicating that the conditional distribution was chosen appropriately.

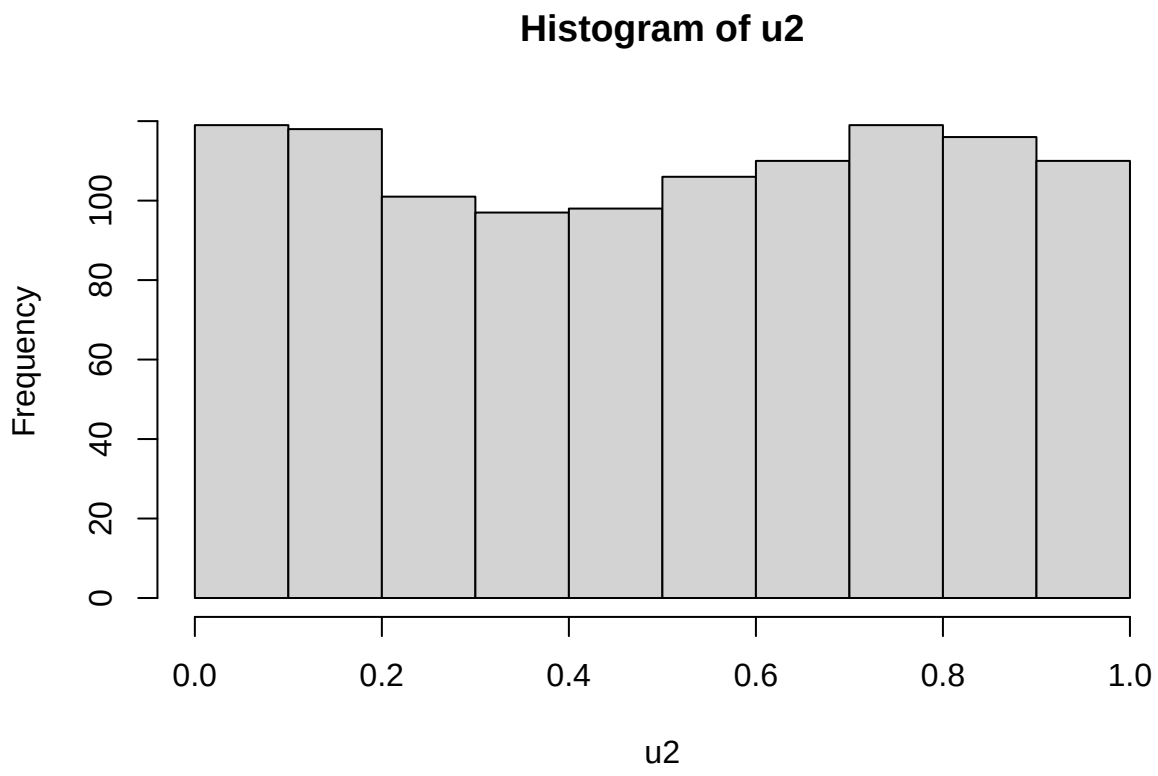
```
u2=psstd(res2, mean=0, sd=1,nu=tail(coef(model1), n=1),
         xi=coef(model1)[length(coef(model2))-1])[2:length(p_ftse)]
cat("KS test p value: ", LcKS(u2, cdf = "punif")$p.value, "\n")
```

```
## KS test p value:  0.6626
```

```
cat("AD test p value: ", ad.test(u2, null="punif")$p.value)
```

```
## AD test p value:  0.536942
```

```
hist(u2)
```



Copula modelling

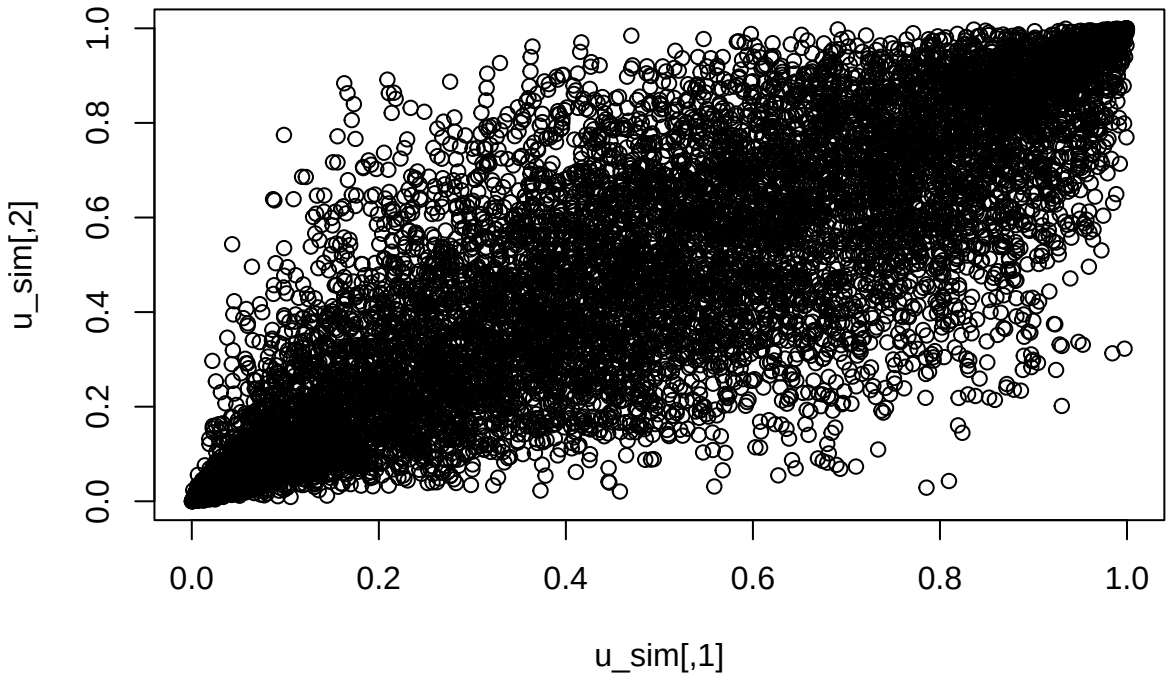
BiCopSelect chooses the most appropriate copula to model the data. The BB1 copula has been selected. This copula combines the dependency of both tails from Clayton and Gumbel so it is both upper and lower tail dependent.

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

```
## Bivariate copula: BB1 (par = 1.03, par2 = 1.79, tau = 0.63)
```

Value-at-Risk using Monte Carlo simulation

```
N=10000
set.seed(0123)
u_sim=BiCopSim(N, family=model$family, model$par,  model$par2)
plot(u_sim)
```




```
res1_sim=qnorm(u_sim[,1], mean = 0, sd = 1)
res2_sim=qnorm(u_sim[,2], mean = 0, sd = 1)
```

"res1_sim" and "res2_sim" are i.i.d. So, the next step is to re-introduce autocorrelation and GARCH effects observed in data.

In order to reintroduce AR/GARCH effects to our simulated data, we must first compute the conditional standard deviation which, when multiplied by the simulated residuals, produce the simulated GARCH effects. This is then summed with the mean and the product of ar1 and the previous log return. We only need to consider ar1 as we are using the arma(1,0) model.

```
nonStandardised_res1=residuals(model1, standardize=FALSE)
hat_variance1 = model1@h.t
variance_t_add1=coef(model1)[3]+coef(model1)[4]*nonStandardised_res1[1095]^2+
  coef(model1)[5]*hat_variance1[1095]
sd_t_add1=sqrt(variance_t_add1)
y1simulated=coef(model1)[1]+coef(model1)[2]*p_ftse[1095]+sd_t_add1*res1_sim

nonStandardised_res2=residuals(model2, standardize=FALSE)
hat_variance2 = model2@h.t
variance_t_add1=coef(model2)[3]+coef(model2)[4]*nonStandardised_res2[1095]^2+
  coef(model2)[5]*hat_variance2[1095]
sd_t_add1=sqrt(variance_t_add1)
y2simulated=coef(model2)[1]+coef(model2)[2]*p_fchi[1095]+sd_t_add1*res2_sim
```

Computing the 99% and 95% var estimates

```
portsim <- matrix(0, nrow = N, ncol = 1)
varsim <- matrix(0, nrow = 1, ncol = 2)

portsim=log(1+((exp(y1simulated)-1)+(exp(y2simulated)-1))*(1/2))
varsim=quantile(portsim,c(0.01,0.05))
-varsim
```

```
##        1%        5%
## 0.0646580 0.0453304
```

The model produces the 99% and 95% estimated VAR for the portfolio: 6.47% and 4.53%, respectively. These values could be more accurately estimated if we were to use more trials for the Monte Carlo simulation.

Advantages and Disadvantages of the Monte Carlo Simulation Approach Based on Copula Theory

A key advantage of using the Monte Carlo Simulation approach for estimating VAR is the flexibility of the model and its ability to account for a variety of modelling assumptions. For instance, the historical and parametric approaches would not be able to account for the dynamically evolving volatility which is apparent in GARCH models. Using GARCH, we can capture the autocorrelation and volatility clustering that occurs in the time series which is not possible using the other approaches. Thus, the model is changing over time in accordance with the dependence structure of the time series and so is better at predicting values further into the future. Additionally, there is a plethora of copula models that can be chosen from, allowing for a higher chance of finding a more accurate fit for the data.

However, a significant disadvantage of this approach is its complexity. Finding an appropriate model that accurately fits the data set can be very time consuming as well as computationally intense. When finding the model for our dataset, we used Python to generate code for an exhaustive list of models of our combinations of conditional distributions as well as ranges of parameters such as the possible lags visible from the PACF. This resulted in us having to check up to 1000 different models for each stock index. For some of these indices, all the models failed to pass the box or uniformity tests resulting in an excessive expense of time and being unable to model that stock index. Furthermore, this approach is highly sensitive to changes as deviations from the correct copula parameters can cause the copula to not fit the true dependence structure and Monte Carlo simulation might provide data that deviates significantly from reality. Also, the addition of new data to the dataset has impacted the compatibility of our current model. This required us to recompute the test for each model to find a better fit since the AIC/BIC values and the results from the uniformity and box tests had changed significantly. Another limitation of our approach was using a 5% confidence level for the box tests. Selecting a higher confidence level would result in us being unable to find a suitable model for the FTSE data. This leads us to a higher likelihood of having found a false positive model over the true model of the data.