**Practical 1**

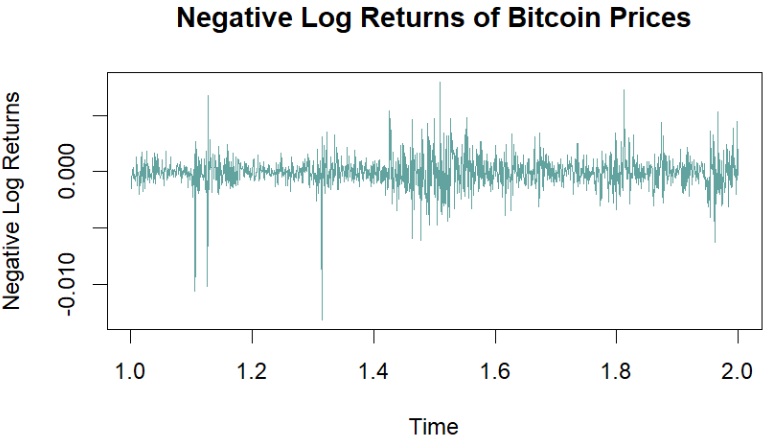
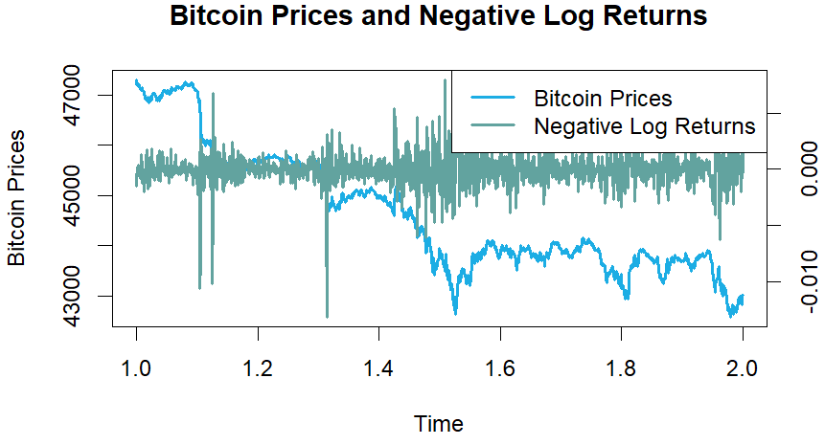
**Part 1: Financial returns and normality**

1. **Read Crypto data.csv. Then, assess the stationarity of the (raw) Bitcoin prices.**

The cumulative periodogram (Appendix 1.1) shows clearly that the data base **is not stationary** as the observations are outside the confidence interval of white noise stationarity. This statement is confirmed by the Dickey Fuller Test (Appendix 1.2) with a p value higher than 0.5.

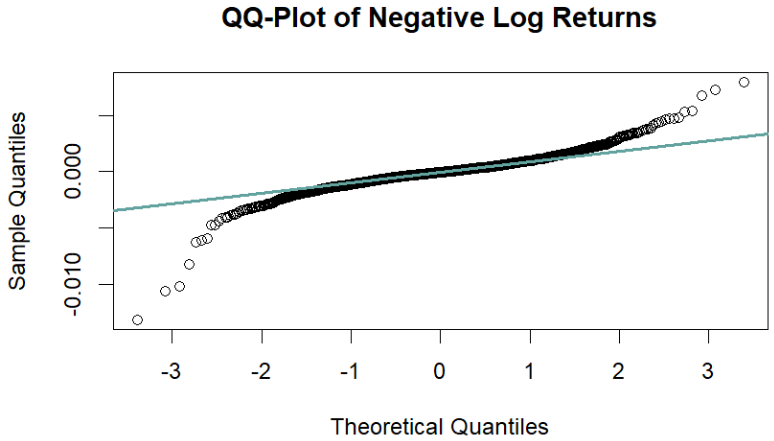
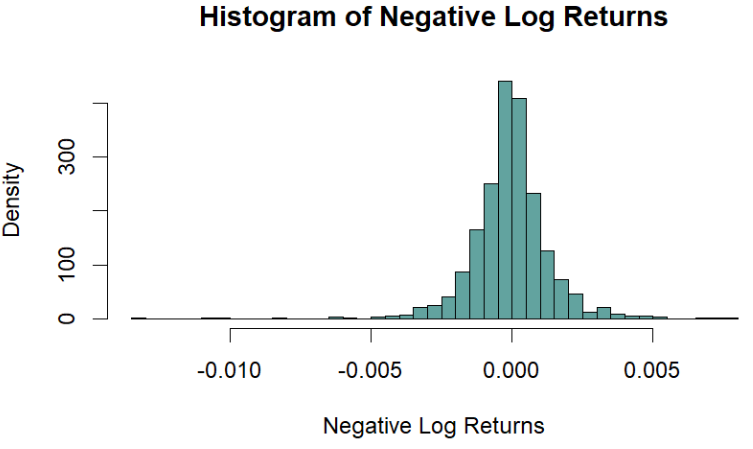
1. **Create a function to transform the Bitcoin prices into their negative log returns. Plot the latter series and assess their stationarity. To compare the series, also plot the negative log returns on a common scale.**

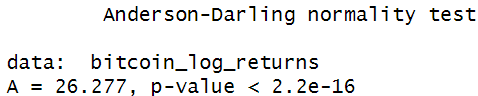
This graph suggests us that the negative log returns are more similar to a stationary data base than the original one.



1. **Are the negative log returns normally distributed? Draw histograms, check QQ-plots and use an Anderson-Darling testing procedure to answer this question.**

The histogram suggests a normal distribution while the QQplot shows some desviations at the tails that make us believe that there could be some problems with the normality.



Finally, as the p value is lower than 5% we can reject the normality and confirm that the negative log returns are not normally distributed

1. **Fit a t-distribution to the negative log returns using fitdistr(). Using a QQ-plot, decide whether the t is better than with a Normal distribution, based on your answer in (3).**

Comparing the appendix 1.3 QQ Plot graphs with the normal distribution plot, we can say that the normal distribution was better than the t distribution as it is closer to the blue line.

1. **Compare the tails of the densities of the t-distribution and the normal distribution. Can we expect more extreme, unexpected events in t-distribution or in normal distribution? What can you conclude about the extreme events of our bitcoin data?**

Both tails deviate significantly from the blue line, indicating that Bitcoin is prone to extreme events. However, the t-distribution exhibits greater tail density compared to the normal distribution, suggesting that the t-distribution allows for a higher likelihood of unexpected, extreme events.

**Part 3: Dependence between time series**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Statistic (t)** | **Degrees of Freedom (df)** | **p-value** | **95% Confidence Interval** | **Sample Correlation Estimate** |
| -0.11935 | 1437 | 0.905 | [-0.0548, 0.0485] | -0.0031 |

1. **Are the negative log returns of Bitcoin and ETH dependent? Compute the correlation using cor.test() function. Can we conclude that these series are independent?**

Given the extremely low correlation coefficient, the high p-value, and the confidence interval that includes zero, there is no no significant linear relationship. This implies that negative extreme events in Bitcoin prices are not related to negative extreme events in Ethereum prices.

1. **A graph with numbers and lines

   Description automatically generatedCalculate the cross-correlation function (CCF) between the negative log returns of Bitcoin and ETH. What do you observe?**

The CCF plot shows **no significant lead-lag relationship** between the negative log returns of Bitcoin and Ethereum. The only notable correlation occurs at **lag 0**, implying that the two assets' negative log returns are correlated when they happen **at the same time**, but there is no evidence of either asset consistently leading the other in terms of negative returns. This result is consistent with the Pearson correlation analysis.

1. **Is one of the time series good predictor of the second? Assess whether there is any predictive power between the negative log returns of Bitcoin and ETH. You can use grangertest() in the lmtest package with carefully chosen hyperparameter order. What is your conclusion?**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Lags** | **Direction** | **F-Statistic** | **p-value** | **Conclusion** |
| 2 | Bitcoin → Ethereum | 2.78 | 0.062 | Weak evidence of causality (significant at 10% level) |
| 2 | Ethereum → Bitcoin | 0.64 | 0.525 | No evidence of causality. |
| 3 | Bitcoin → Ethereum | 5.13 | 0.0016\*\* | Significant causality at 1% level. |
| 3 | Ethereum → Bitcoin | 0.46 | 0.712 | No evidence of causality. |
| 4 | Bitcoin → Ethereum | 7.09 | 1.19e-05\*\*\* | Strong evidence of causality at 0.1% level. |
| 4 | Ethereum → Bitcoin | 0.54 | 0.708 | No evidence of causality. |

Unidirectional Causality: Bitcoin's past negative log returns have significant predictive power over Ethereum’s returns, especially when using 3 or 4 lags. Stronger Predictive Power with More Lags: The evidence strengthens as the lag order increases, with the most robust relationship observed at 4 lags.

No Evidence of Reverse Causality: Ethereum’s past returns do not Granger cause Bitcoin’s returns, as no significant relationship was found for any lag order.

Practical Implication: The results suggest that Bitcoin's market behavior influences Ethereum, but Ethereum's behavior does not influence Bitcoin. This insight could be useful for forecasting Ethereum’s movements using Bitcoin’s historical data.

1. **Based on your answer in (c), answer the following questions:**
   1. **We observe an extreme sudden drop in Bitcoin stocks. What should we expect that will happen with ETH stocks?**

The Granger causality suggests that Bitcoin’s past behavior influences Ethereum’s future performance, so a significant negative movement in Bitcoin (such as a sudden drop) could lead to a similar negative impact on Ethereum over the next few days (based on the lag structure). The effect might not be immediate, but it could unfold over the subsequent few days, particularly within 3 to 4 days after the drop in Bitcoin.

* 1. **We observe an extreme sudden drop in ETH stocks. What should we expect, that will happen with Bitcoin stocks?**

According to the Granger causality test, if there is an extremely sudden drop in Ethereum stocks, we should not expect Bitcoin stocks to be directly influenced by this drop, at least not in a predictable or systematic way based on the historical relationship between the two assets.

**Practical 3**

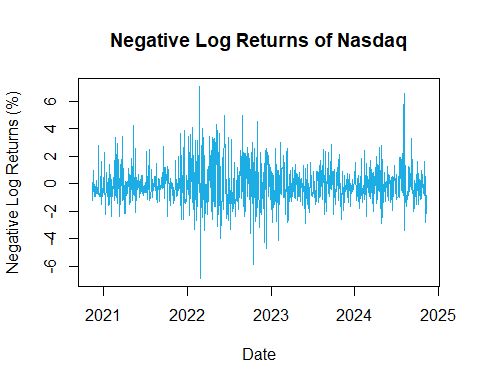
In the following report we will be analyzing the NASDAQ Composite Index during the periods of Nov 2020 – Nov 2024 to answer the following research questions:

* If an investor had implemented a stop-loss strategy based on historical VaR levels, how many times would it have triggered between 2021 and 2024, and would it have reduced overall losses?
* How well do common risk models (like historical VaR) predict actual losses during the most volatile periods in the Nasdaq Composite Index, and could an alternative model improve accuracy?

The objective is to understand the movements of the index to manage the risk for a potential financial portfolio. To do it, we will review the Value at Risk, Expected Shortfall, and Extreme Values Theory methods to **select the best Stop Loss Strategy**.

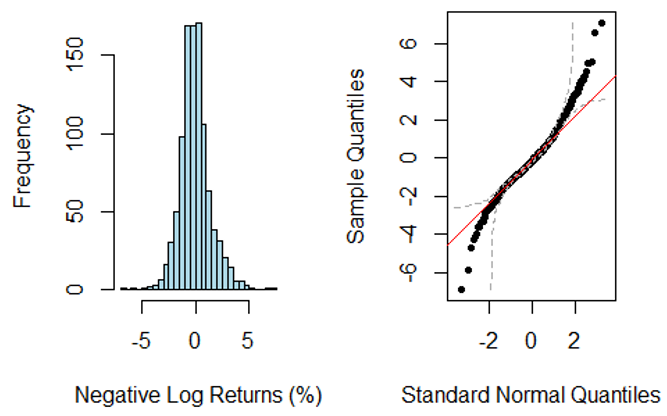
1. **Data Pre-Processing**
   1. **Checking Stationarity**

It is simple to see that the index historical open price is not stationary (Appendix 3.1). Then, we proceed to calculate the negative log returns.



Together the plot and the Dickey-Fuller test (Appendix 3.2), we can conclude that the negative log return is stationary as the p value of the test is less than 5%.

* 1. **Checking Normality**

****

Confirmed by the Shapiro test (Appendix 3.3), the daily returns are not normally distributed. Therefore, using parametric Value at Risk and Expected Shortfall could result in under/overestimating the risk.

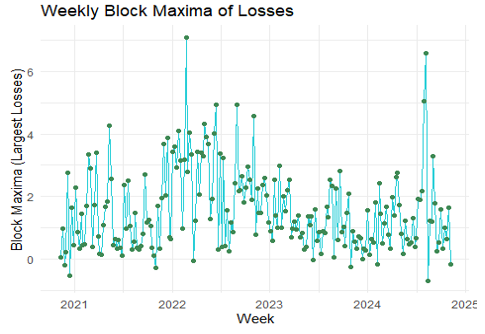
1. **Extreme Values Theory**
   1. **Block Maxima**
      1. **Calculation**

Weekly blocks were created by grouping the data into weeks based on the Date column. For each week, the **maximum return** (largest loss) was calculated, representing the **worst weekly loss** due to the negative log transformation used in the return calculation.

The table shows the **weekly block maxima** (largest losses) for the first six weeks in the dataset:

|  |  |
| --- | --- |
| **Year - Week** | **Block Max** |
| 2020-45 | 0.0641 |
| 2020-46 | 0.989 |
| 2020-47 | -0.189 |
| 2020-48 | 0.224 |
| 2020-49 | 2.77 |
| 2020-50 | -0.539 |

Each row highlights the **largest weekly loss** for the index. For example, in the 49th week of 2020, the worst weekly loss was **2.7711**, which reflects a significant drop in returns during that week.



The graph visualizes the weekly **largest losses** over time, highlighting periods of heightened market stress, such as in 2022, where weekly losses reached their most extreme values. Outside of 2022, the general trend indicates that most weekly losses are less severe, reflecting a relatively stable risk profile during those periods. This visualization provides a clear timeline of extreme losses, offering insights into when and where the market experienced the most significant volatility.

1. **VaR and ES**

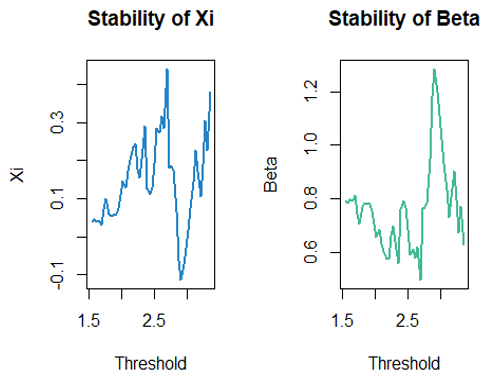
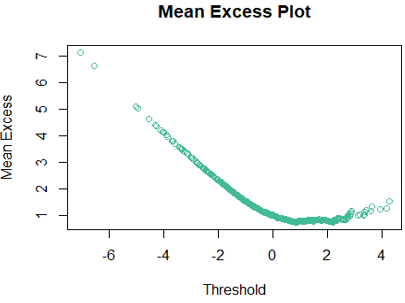
The table compares Value at Risk (VaR) and Expected Shortfall (ES) for 95% and 99% confidence levels using historical and parametric approaches.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Confidence Level** | **Historical VaR** | **Historical ES** | **Parametric VaR** | **Parametric ES** |
| 95% | 0.612% | -0.206% | 0.049% | -0.871% |
| 99% | 0.272% | -0.504% | 0.402% | -0.483% |

The parametric approach shows a wider range, especially in ES at 95%, where the estimated loss magnitude is significantly higher. This discrepancy reflects the sensitivity of the parametric method, which assumes a heavier-tailed GEV model. However, since the underlying distribution is known to be non-normal, the parametric estimates might overstate risks, particularly at lower confidence levels, emphasizing the need for careful model validation.

* 1. **Peaks Over Threshold**
     1. **Parametrical POT**

The choice of threshold is crucial in the POT method. An appropriate threshold balances bias and variance in parameter estimates. The mean excess plot helps identify a suitable threshold where the mean excess over the threshold is linear. Based on the plot, a suitable threshold would be around -2 to 0, where the mean excess stabilizes and becomes linear.



The parameter stability plots provided a guidance to select a threshold. The region 2.2 and 2.8 shows stability, making it reliable range for analyzing extreme events.

* + 1. **Historical POT**

Instead of analyzing all returns, the Historical POT method focuses exclusively on the most extreme losses, ensuring that the potential severity of these events is not underestimated. This approach provides a conservative and realistic estimate of potential losses, making it highly effective for assessing extreme risk scenarios without relying on parametric assumptions.

For this the **95th percentile threshold** was selected as 2.46201, representing the cut-off point for extreme negative returns. The excesses over this threshold were extracted (e.g., 1.059, 0.271, 0.831), which will be used to calculate the Value at Risk (VaR) and Expected Shortfall (ES).

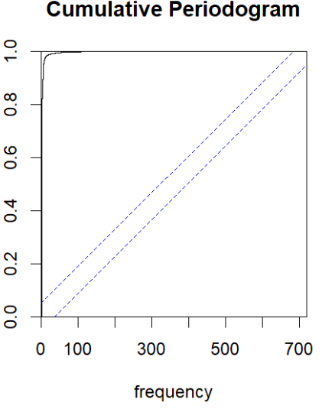
* + 1. **VaR and ES**

The table compares Value at Risk (VaR) and Expected Shortfall (ES) for 95% and 99% confidence levels using historical and parametric approaches.

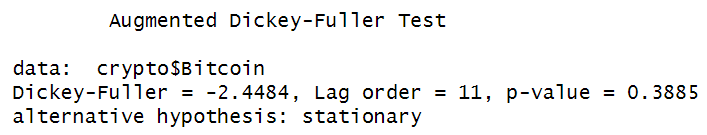
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Confidence Level** | **Parametric VaR** | **Parametric ES** | **Historical VaR** | **Historical ES** |
| 95% | -3.979958 | -4.880986 | -6.600801 | -6.83753 |
| 99.5% | -4.612842 | -5.501092 | -6.719166 | -6.83753 |
| 99.9% | -6.047954 | -6.907230 | -6.813857 | -6.83753 |

The Parametric VaR and ES highlights the increasing severity of potential losses as confidence levels rise, with VaR and ES growing more extreme at 99.9% confidence. Regarding the Historical POT, The VaR grows as the confidence level increases, which is expected because higher confidence levels capture rarer, more extreme events. The 99% VaR indicates that losses will not exceed 6.37% on 99 out of 100 trading days, while the ES shows that when losses exceed the VaR, the average loss is 6.88%. As confidence levels increase, the VaR captures progressively worse losses (up to 6.83% at 99.9% confidence), showing the potential for larger losses in rarer scenarios.

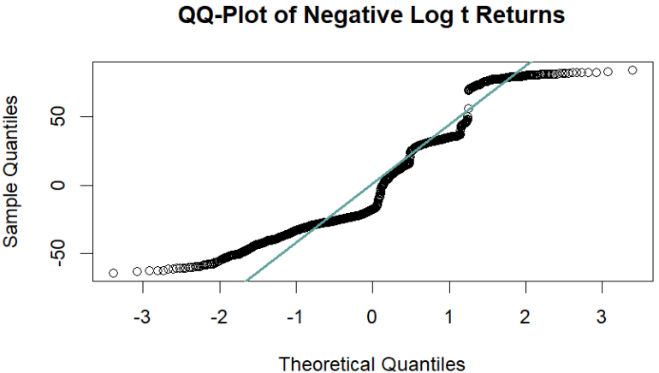
1. **Stop-Loss Strategy**
2. **Conclusions**
3. **Recommendations**
4. **Appendix:**
   1. **Dickey Fuller Test**

****

* 1. **Dickey Fuller Test**



**1.3 QQ plot t fitted**



1.3. **Dependence between time series**

#Reading the csv

crypto <- readr::read\_csv("C:/Users/Marcela/Documents/Documentos/3rd Semester/Risk analytics/Week 1/Crypto\_data.csv")

# Calculate log returns for Bitcoin and Ethereum (convert to numeric in case of any issues)

crypto <- crypto %>%

mutate(Bitcoin = as.numeric(Bitcoin),

Ethereum = as.numeric(Ethereum),

Bitcoin\_log\_return = log(Bitcoin / lag(Bitcoin)),

Ethereum\_log\_return = log(Ethereum / lag(Ethereum)))

# Calculate negative log returns for both Bitcoin and Ethereum

crypto <- crypto %>%

mutate(Bitcoin\_negative\_log\_return = -Bitcoin\_log\_return,

Ethereum\_negative\_log\_return = -Ethereum\_log\_return)

# Check the results

head(crypto)

A screenshot of a computer

Description automatically generated

# Plot the log returns of Bitcoin and Ethereum

plot(crypto$Bitcoin\_log\_return, type = "l", col = "#27CED7",

main = "Log Returns of Bitcoin and Ethereum", ylab = "Log Returns")

lines(crypto$Ethereum\_log\_return, col = "#62A39F")

legend("topright", legend = c("Bitcoin", "Ethereum"), col = c("#27CED7", "#62A39F"), lty = 1)

A graph of a graph

Description automatically generated with medium confidence

* 1. Are the negative log returns of Bitcoin and ETH dependent? Compute the correlation using cor.test() function. Can we conclude that these series are independent?

# Check if the columns are numeric

str(crypto)

# Drop the first row with NA values from the log return calculations (due to lag)

crypto <- na.omit(crypto)

# Check if there are any NA values remaining

sum(is.na(crypto$Bitcoin\_negative\_log\_return)) # Should return 0

sum(is.na(crypto$Ethereum\_negative\_log\_return)) # Should return 0

# Perform the correlation test between Bitcoin and Ethereum negative log returns

correlation\_test <- cor.test(crypto$Bitcoin\_negative\_log\_return, crypto$Ethereum\_negative\_log\_return)

# Print the results of the correlation test

print(correlation\_test)

A computer screen shot of a computer code

Description automatically generated

* 1. Calculate the cross-correlation function (CCF) between the negative log returns of Bitcoin and ETH. What do you observe?

# Calculate the cross-correlation function (CCF) with a smaller title font size

ccf\_result <- ccf(

crypto$Bitcoin\_negative\_log\_return,

crypto$Ethereum\_negative\_log\_return,

lag.max = 20,

plot = TRUE,

main = "Cross-Correlation:

Bitcoin and Ethereum Negative Log Returns",

cex.main = 0.6 # Adjust title size (smaller font)

)

A graph with a line

Description automatically generated

* 1. Is one of the time series good predictor of the second? Assess whether there is any predictive power between the negative log returns of Bitcoin and ETH. You can use grangertest() in the lmtest package with carefully chosen hyperparameter order. What is your conclusion?

# Fit ARIMA model to Bitcoin negative log returns and auto-select order based on AIC/BIC

fit\_btc <- auto.arima(crypto$Bitcoin\_negative\_log\_return, ic = "aic")

# Check the selected ARIMA order (p, d, q)

summary(fit\_btc)

# Similarly, for Ethereum

fit\_eth <- auto.arima(crypto$Ethereum\_negative\_log\_return, ic = "aic")

# Check the selected ARIMA order for Ethereum

summary(fit\_eth)

A screenshot of a computer program

Description automatically generated

# Granger causality test with 2 lags based on ARIMA models

granger\_test\_btc\_to\_eth <- grangertest(Ethereum\_negative\_log\_return ~ Bitcoin\_negative\_log\_return, order = 2, data = crypto)

granger\_test\_eth\_to\_btc <- grangertest(Bitcoin\_negative\_log\_return ~ Ethereum\_negative\_log\_return, order = 2, data = crypto)

# Print results

print("Granger causality test: Bitcoin causing Ethereum")

print(granger\_test\_btc\_to\_eth)

print("Granger causality test: Ethereum causing Bitcoin")

print(granger\_test\_eth\_to\_btc)

A screenshot of a computer code

Description automatically generated

# Granger Causality Test with 3 Lags

granger\_test\_btc\_to\_eth\_lag3 <- grangertest(Ethereum\_negative\_log\_return ~ Bitcoin\_negative\_log\_return, order = 3, data = crypto)

granger\_test\_eth\_to\_btc\_lag3 <- grangertest(Bitcoin\_negative\_log\_return ~ Ethereum\_negative\_log\_return, order = 3, data = crypto)

# Print results for 3 lags

print("Granger causality test (3 lags): Bitcoin causing Ethereum")

print(granger\_test\_btc\_to\_eth\_lag3)

print("Granger causality test (3 lags): Ethereum causing Bitcoin")

print(granger\_test\_eth\_to\_btc\_lag3)

# Granger Causality Test with 4 Lags

granger\_test\_btc\_to\_eth\_lag4 <- grangertest(Ethereum\_negative\_log\_return ~ Bitcoin\_negative\_log\_return, order = 4, data = crypto)

granger\_test\_eth\_to\_btc\_lag4 <- grangertest(Bitcoin\_negative\_log\_return ~ Ethereum\_negative\_log\_return, order = 4, data = crypto)

# Print results for 4 lags

print("Granger causality test (4 lags): Bitcoin causing Ethereum")

print(granger\_test\_btc\_to\_eth\_lag4)

print("Granger causality test (4 lags): Ethereum causing Bitcoin")

print(granger\_test\_eth\_to\_btc\_lag4)

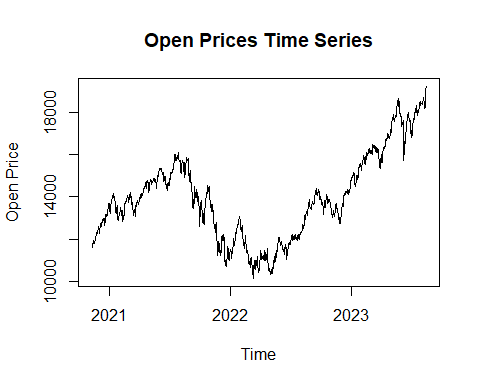
**A screenshot of a computer program

Description automatically generated**

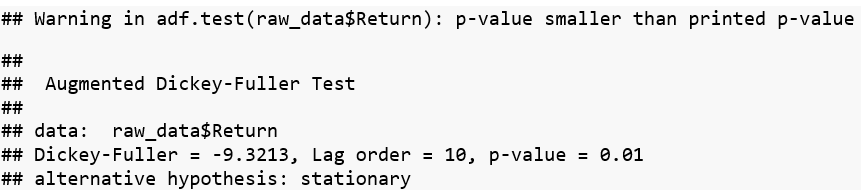
**A computer screen shot of text

Description automatically generated**

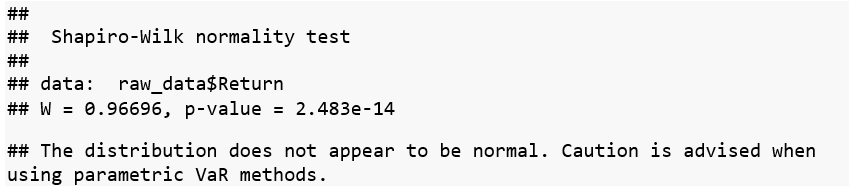
1. **Practical 2**
2. **Practical 3**
   1. **Open Prices Time Series**



* 1. **Dickey Fuller Test**

****

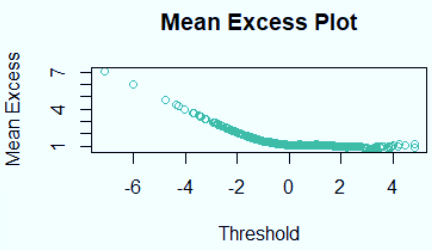
* 1. **Shapiro Wilk Normality Test**

****

* 1. **Extreme Values Theory**
     1. **Block Maxima**
     2. **Peaks**
        1. **POT Parametric**

# Mean Excess Plot

meplot(-raw\_data$Return, main = "Mean Excess Plot", col = "#42BA97")



# Define a sequence of thresholds

thresholds <- seq(quantile(-raw\_data$Return, 0.90, na.rm = TRUE),

quantile(-raw\_data$Return, 0.99, na.rm = TRUE),

length.out = 50)

# Initialize vectors to store parameters

xi\_values <- numeric(length(thresholds))

beta\_values <- numeric(length(thresholds))

# Fit the GPD for each threshold and store parameters

for (i in seq\_along(thresholds)) {

threshold <- thresholds[i]

# Fit GPD only if threshold is not NA

if (!is.na(threshold)) {

fit <- gpd(-raw\_data$Return, threshold = threshold)

xi\_values[i] <- fit$par.ests["xi"]

beta\_values[i] <- fit$par.ests["beta"]

} else {

xi\_values[i] <- NA

beta\_values[i] <- NA

}

}

# Create a data frame for plotting

gpd\_params\_df <- data.frame(

Threshold = thresholds,

Xi = xi\_values,

Beta = beta\_values

)

# Plot parameter stability with custom colors

par(mfrow = c(1, 2)) # Arrange plots side by side

# Plot for Xi (Stability of Xi) with a custom color

plot(gpd\_params\_df$Threshold, gpd\_params\_df$Xi, type = "l",

col = "#2683C6", # Custom color for Xi

xlab = "Threshold", ylab = "Xi",

main = "Stability of Xi", lwd = 2)

# Plot for Beta (Stability of Beta) with a custom color

plot(gpd\_params\_df$Threshold, gpd\_params\_df$Beta, type = "l",

col = "#42BA97", # Custom color for Beta

xlab = "Threshold", ylab = "Beta",

main = "Stability of Beta", lwd = 2)

# Reset plotting parameters

par(mfrow = c(1, 1))

A graph of stability and stability of the stability of the data

Description automatically generated with medium confidence

# Step 1: Set the threshold based on stability analysis

u <- 2.5 # Chosen threshold from stability region

# Step 2: Extract exceedances over the threshold

excesses <- -raw\_data$Return[raw\_data$Return > u] - u

# Print the threshold and some of the exceedances for verification

print(paste("Selected Threshold (u):", u))

print(head(excesses))



# Fit the GPD to the exceedances

gpd\_fit <- gpd(-raw\_data$Return, threshold = u)

# Summarize the fitted model

kable(summary(gpd\_fit))

# Estimated parameters

xi <- as.numeric(gpd\_fit$par.ests["xi"])

beta <- as.numeric(gpd\_fit$par.ests["beta"])

# Sample sizes

N <- length(raw\_data$Return)

N\_exc <- sum(-raw\_data$Return > u)

# Confidence levels

p\_levels <- c(0.99, 0.995, 0.999)

# Calculate VaR

VaR\_POT <- sapply(p\_levels, function(p) {

VaR <- u + (beta / xi) \* (((N\_exc / (N \* (1 - p)))^xi - 1))

return(-VaR) # Convert back to negative return

})

# Create a data frame for results

VaR\_POT\_df <- data.frame(

Confidence\_Level = p\_levels,

VaR = VaR\_POT

)

# Display results

kable(print(VaR\_POT\_df))

A screenshot of a computer

Description automatically generated

# Calculate ES

ES\_POT <- sapply(1:length(p\_levels), function(i) {

p <- p\_levels[i]

VaR\_p <- -VaR\_POT[i] # Use positive value for calculation

ES <- (VaR\_p / (1 - xi)) + ((beta - xi \* u) / (1 - xi))

return(-ES) # Convert back to negative return

})

# Add ES to the data frame

VaR\_POT\_df$ES <- ES\_POT

# Display results

kable(print(VaR\_POT\_df))

A screenshot of a computer

Description automatically generated

* + - 1. **POT Historic**

# Define the threshold (e.g., 95th percentile of negative returns)

threshold <- quantile(-raw\_data$Return, 0.95, na.rm = TRUE)

# Print the selected threshold

cat("Selected Threshold:", threshold, "\n")



# Extract excesses over the threshold

excesses <- -raw\_data$Return[-raw\_data$Return > threshold] - threshold

# Print the first few exceedances

cat("Excesses over the threshold:\n")

print(head(excesses))

#Calculating the VaR historical with POT

# Define confidence levels

confidence\_levels <- c(0.99, 0.995, 0.999)

# Calculate Historical VaR using POT

historical\_var\_pot <- sapply(confidence\_levels, function(cl) {

quantile(excesses, probs = cl, na.rm = TRUE) + threshold # Add back the threshold

})

# Print the Historical VaR results

historical\_var\_pot\_df <- data.frame(

Confidence\_Level = paste0(confidence\_levels \* 100, "%"),

Historical\_VaR\_POT = -historical\_var\_pot # Convert back to negative returns

)

# Calculate Historical ES using POT

historical\_es\_pot <- sapply(1:length(confidence\_levels), function(i) {

var\_threshold <- historical\_var\_pot[i] - threshold # Find the excess threshold

mean(excesses[excesses >= var\_threshold], na.rm = TRUE) + threshold # Add back the threshold

})

# Add ES to the DataFrame

historical\_var\_pot\_df$Historical\_ES\_POT <- -historical\_es\_pot # Convert back to negative returns

**A white rectangular object with a black stripe

Description automatically generated**

* 1. **S**