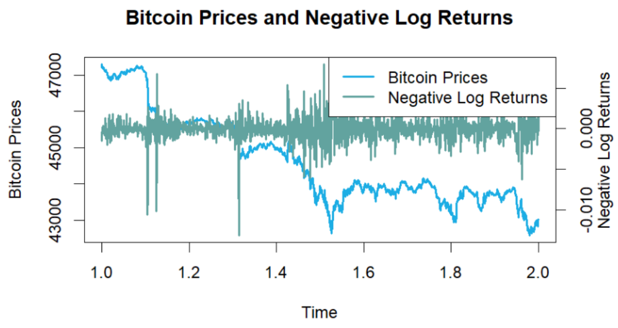
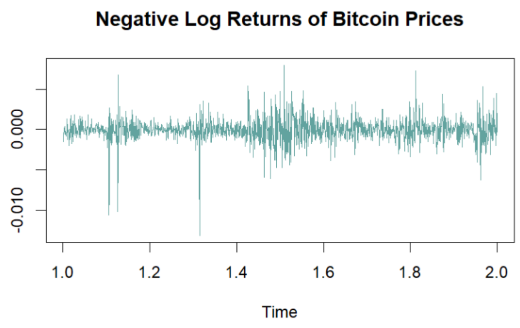
Practical 1

Part 1: Financial returns and normality

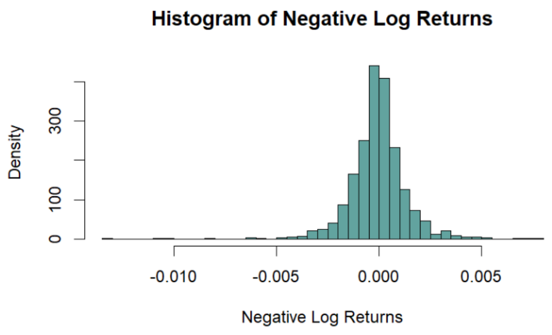
1. **Read Crypto data.csv. Then, assess the stationarity of the (raw) Bitcoin prices.**
The cumulative periodogram (Appendix 1.1.1) shows clearly that the data base **is not stationary** as the observations are outside the confidence interval of white noise stationarity. This statement is confirmed by the Dickey Fuller Test (Appendix 1.1.2) with a p value higher than 0.5.

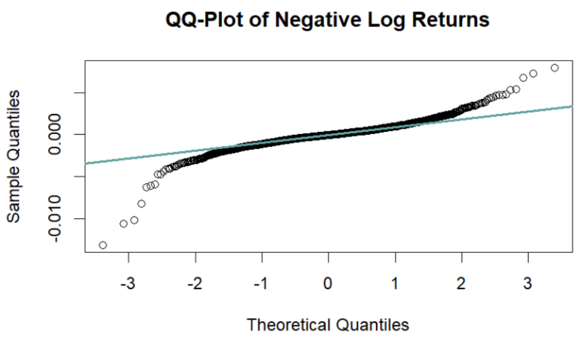
2. **Create a function to transform the Bitcoin prices into their negative log returns. Plot the latter series and assess their stationarity. To compare the series, also plot the negative log returns on a common scale.**
This graph suggests us that the negative log returns are more similar to a stationary data base than the original one.





3. **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.





```
            Anderson-Darling normality test

data:  bitcoin_log_returns
A = 26.277, p-value < 2.2e-16
```

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

4. **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 1.1.3 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.
5. **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 2: Financial time series, heteroscedasticity and the random walk hypothesis

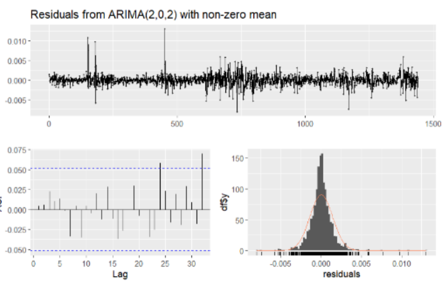
1. **Plot the ACF of the raw series as well as the negative log returns. Which one do you think are easier to model?**

The ACF of the raw Bitcoin price series (Appendix 1.2.1) shows high autocorrelation at multiple lags, indicating a strong positive, declining trend and non-stationarity. In contrast, the ACF of the negative log returns (Appendix 1.2.2) is mostly within the confidence intervals, suggesting little to no autocorrelation and a stationary series. Since the negative log returns exhibit less persistence and randomness, they are easier to model and more suitable for time series analysis.

2. **Use a Ljung-Box procedure to formally test for (temporal) serial dependence in the raw series andin the negative log return series. What is your conclusion?**

The Ljung-Box tests (Appendix 1.2.3) for both raw Bitcoin price series and the negative log returns, so we reject the null hypothesis. This confirms that the raw Bitcoin series is non-stationary and exhibits strong trends and patterns over time, as suggested by the ACF plot. While the negative log returns series is closer to stationarity (as indicated by the ACF plot), there is still a slight level of autocorrelation present.

3. **Propose ARIMA models for the negative log returns series, based on visualization tools (e.g. ACF, PACF).**



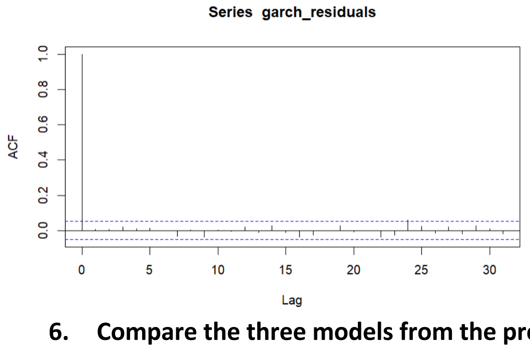
ARIMA(1,0,1) and ARIMA(2,0,2) are fitted (Appendix 1.2.4). The auto ARIMA(2, 0, 2) model is likely preferred because it has a lower AIC value. The first model provides a simpler representation but may not capture the underlying dynamics as effectively as the second model. We reject the null hypothesis of no autocorrelation in the residuals for manual ARIMA(1, 0, 1) model (Appendix 1.2.5). The model may not adequately capture the time series dynamics, suggesting that it might be beneficial to consider a more complex model. For auto ARIMA(2, 0, 2) we accept the null hypothesis of no autocorrelation in the residuals.

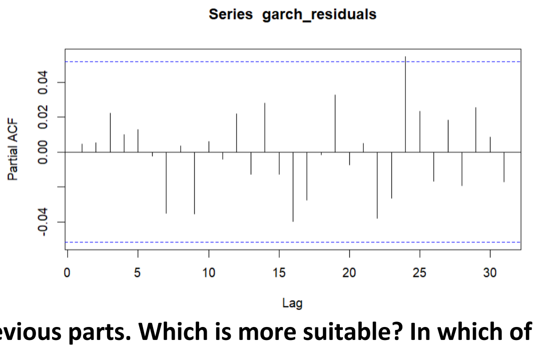
4. **Fit GARCH models to the negative log returns with both normal and standardized t-distributions, with order (1, 1), using the garchFit() function from the fGarch library. Assess the quality of the fit by evaluating the residuals.**

The GARCH(1,1) model with a standardized t-distribution provide a better fit for the negative log returns based on the log-likelihood, AIC, and the inclusion of a shape parameter (Appendix 1.2.7). Both models demonstrate significant coefficients for ω, α, and β1. While residuals from both models exhibit non-normality, they effectively capture the autocorrelation structure of the returns (Appendix 1.2.8).

5. **Residual serial correlation can be present when fitting a GARCH directly on the negative log returns. Proceed with the above recipe. Assess the quality of the above fit.**

The GARCH(1, 1) model on the residuals of the ARIMA(2, 0, 2) (Appendix 1.2.9) model indicates significant volatility persistence, as evidenced by the significant α1 and β1 coefficients. The Ljung-Box test results suggest that the residuals do not exhibit serial correlation, indicating a good fit for the model. ACF(Appendix 1.2.10) and PACF(Appendix 1.2.11) of the residuals indicate that GARCH model has adequately captured the temporal dependencies in the data suggesting that the model is a good fit for the data.





6. **Compare the three models from the previous parts. Which is more suitable? In which of these models is the homoscedasticity assumption violated?**

To compare the three models (ARIMA, GARCH, and ARIMA-GARCH) we consider various criteria such as model fit, statistical significance, and the homoscedasticity assumption. The ARIMA-GARCH model is the most suitable because it effectively captures both the mean and the variance of the negative log returns. It addresses the limitations of the ARIMA model alone, which may not handle volatility adequately, while also confirming the improvements in fit over the standalone GARCH model. The ARIMA model is where the homoscedasticity assumption is most likely violated, as it does not account for the changing variance inherent in financial time series data. In contrast, both the GARCH and ARIMA-GARCH models explicitly model volatility, thereby addressing this assumption.

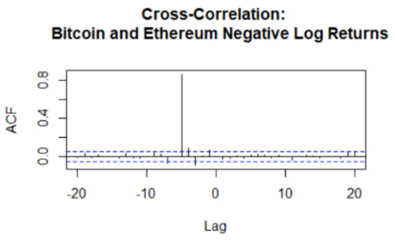
Part 3: Dependence between time series

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?**

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

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

2. **Calculate 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.

3. **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.

4. **Based on your answer in (c), answer the following questions:**
   4.1. 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.

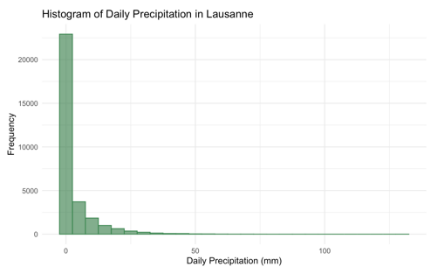
   4.2. We observe an extreme sudden drop in ETH stocks. What should we expect, that will happen with Bitcoin stocks?

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 2

Part 1: Block Maxima Approach

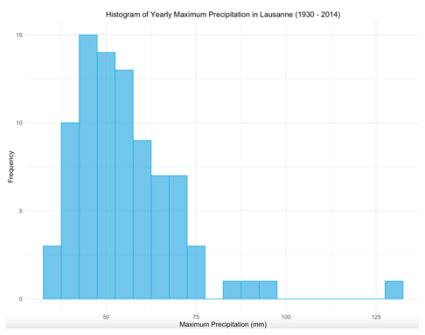
1. **Read in the data. Draw a histogram of the daily precipitation values. Which distribution would best fit the data?**



It appears that the data could fit into a Gumbel Distribution. There is a rapid decay in the frequency of observations and the Gumbel distribution can model this behavior well and considering that we are focusing on the extreme heavy rainfall events.

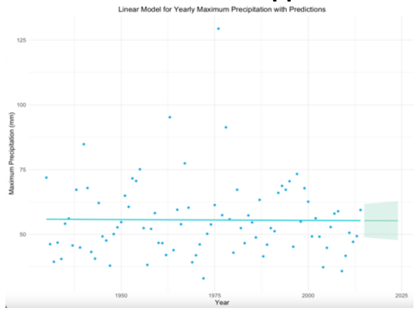
2. **Extract the yearly maximum values. Draw their histogram. Which distribution would best fit the data?**

Most of the yearly maximum precipitation values are clustered between 40 and 70 mm, indicating that these values are typical for Lausanne's climate during the period analyzed. Precipitation levels exceeding 100 mm are rare, as seen by the few bars on the far right of the histogram.



As seen in Appendix 2.1, by examining the scale parameters of the distributions, which indicate the variability of the data, we can observe that the Frechet distribution has the smallest scale parameter (9.97), closely followed by the Gumbel distribution (10.30), both indicating less variation in the extreme precipitation events. Both the Gumbel and Frechet distributions show a typical yearly maximum precipitation around 48 mm, which is consistent with what was observed in the histogram. Although both distributions have similar deviance values, the Frechet distribution shows the lowest deviance (666.94), compared to the Gumbel distribution (668.33), suggesting that Frechet provides the best fit for the data. However, the Gumbel distribution also performs reasonably well, and while the Frechet is the better fit, Gumbel remains a valid option for modeling the yearly maximum precipitation events.

3. **Fit a linear model to the yearly maximum precipitation values and predict the values for the next 10 years. Provide confidence intervals for your predictions and plot it. Do you think that this a reasonable approach?**

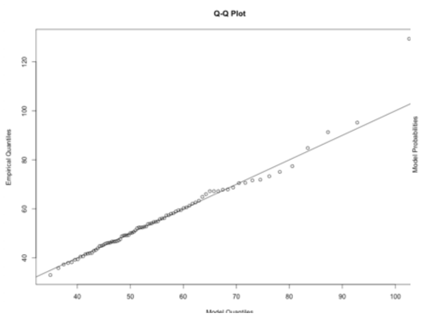


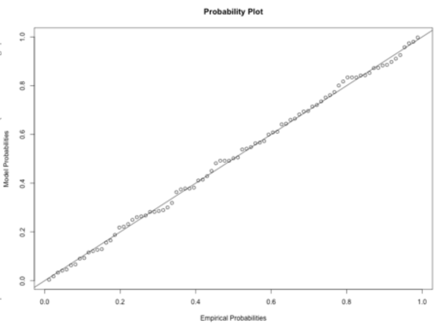
The coefficient for Year is -.006, indicating that the maximum precipitation is expected to decrease by approximately 0.006 mm per year. However, given that the p-value is higher than the standard significance level (0.05), we cannot conclude that Year is a significant predictor of maximum precipitation. Additionally, the model has a low explanatory power, as reflected by the R-squared value (Appendix 2.2). This suggests that the linear model with Year as the regressor may not be appropriate for capturing the trends in yearly maximum precipitation. It is not appropriate to assume that maximum precipitation levels will increase each year simply because time has passed.

4. **Fit a GEV with constant parameters to the historical yearly max values. Fit a second GEV model with time varying location parameter. Compare the two models using AIC or BIC. Which one do you recommend using?**

The constant-parameter model has a lower AIC and BIC compared to the time-varying location model (Appendix 2.3). This suggests that the constant model provides a better fit to the data based on these criteria, so the additional complexity introduced by this model does not significantly improve the fit. The constant model, by contrast, assumes that the average extreme precipitation values remain stable over time, which appears to align better with the data.

5. **Draw diagnostic plots of your GEV fit (for example, using gev.diag function). Is it a good fit?**

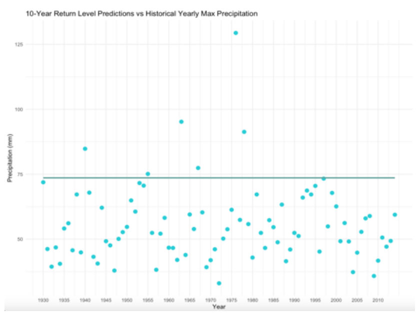




Using the probability plot, the points generally align along the diagonal, however there are some deviations at the extremes, indicating that the model struggles to fit the most extreme values. This issue is also observed in the Q-Q plot, which further suggests that the model is not accurate for predicting extreme events. While the GEV model provides a better fit, it still falls short in reliably predicting the most extreme precipitation events, particularly at the tails of the distribution

6. **Using the model chosen in the previous parts, predict the 10-year return level. Draw your predictions of the 10-year return levels together with your data.**

The blue dots represent the observed maximum precipitation for each year, while the green line represents the 10-year return level predicted by the constant-parameter GEV model. The horizontal nature of the red line reflects the assumption of a constant location parameter, meaning the threshold for extreme events does not change over time. While the observed blue points vary from year to year, there is no evidence of an increasing trend in extreme precipitation based on this model.



7. **Broadly speaking, each year, there is a chance of 1/10 that the observed value is above the 10-year return level. Comment on the results for both the linear model prediction (from c) and the GEV approach (from f). How many historical values were above this 10-year return level? Answer the same question with 20, 50 and 85-year return level.**

As expected, for the 10-year return level using the GEV model, there are 6 historical values where the maximum precipitation exceeds the 10-year return level. For the 20-year return level, there are 4 exceedances, while there are 2 exceedances for the 50-year return level and 1 exceedance for the 85-year return level. These results indicate that the GEV model performs well, capturing the frequency of extreme precipitation values accurately and in line with what is expected for rare events. In contrast, the linear model shows 0 exceedances across all return levels, meaning it fails to account for extreme values. This suggests that the linear model is unsuitable for predicting and modeling extreme events, as it does not adequately capture the distribution's tail.

8. **Using the fitted model, compute the return period of 100 mm of precipitation.**

Precipitation of 100 mm or more can occur once every 71.77 years.

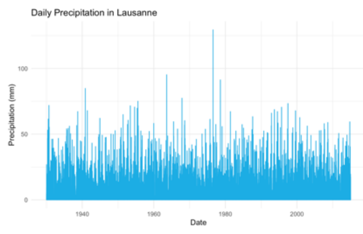
9. **Using the fitted model, compute the probability that there will be a day in the next year when the precipitation exceeds 150 mm.**

The probability of exceeding 150 mm on any day is .0642% and the probability of at least one day in the next year is 20.9%.

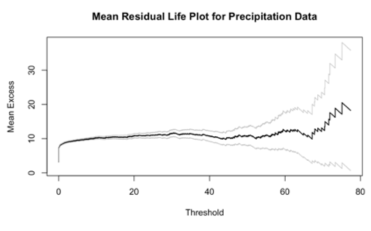
Part 2: Peaks-over-threshold approach

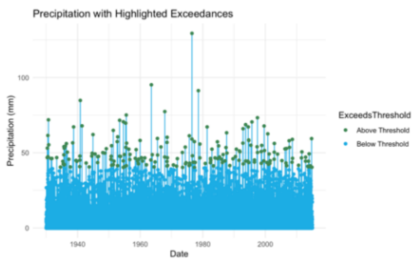
1. **Display a time series plot of the daily precipitation across the data range.**

This plot displays the precipitation in Lausanne over several decades. The data shows variability with occasional spikes.



2. **We want to model the high precipitation levels using the POT approach. First step is choosing a threshold. Draw Mean Residual Life Plot (for example using mrlplot in POT library) for the full range of your data. Choose a reasonable threshold. In the plot from part a) highlight the data that exceeds this threshold.**

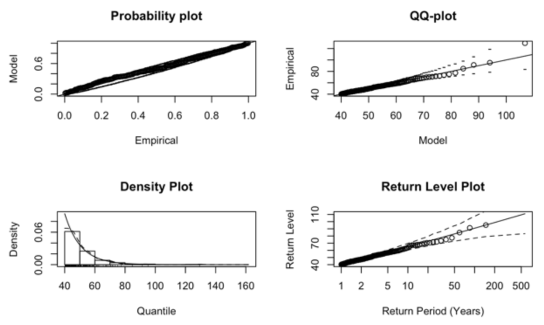




Between 20 and 40, the plot is relatively stable, with no strong upwards or downwards trend, and the mean excess remains quite constant. In the region around 45-50, the mean excess shows larger fluctuations, and the graph starts to act more erratic. Therefore, we choose 40 as the threshold value.

3. **Fit a GPD for the data exceeding the threshold and draw a diagnostic plot. Is it a reasonable fit? (Hint: if not, you may reconsider the choice of the threshold)**

With a threshold value of 40, we get the following diagnostic for our model:



- Probability plot: while our model tends to overestimate low values, it seems overall reliable.
- QQ-Plot: the fit is generally good. However, we notice an extreme value in the upper tail that is not properly captured by the model. Depending on the application, this could be an issue.
- Density Plot: despite some slight variation, our fitted values align generally well with the model.
- Return Level Plot: despite some slight variation in the 20-50 years period, the fit is generally good.

4. **Using the fitted model, compute the 10-year, 20-year, 50-year and 85-year return levels.**

Return levels for period units in years:

10-year level	20-year level	50-year level	85-year level
74.37276	82.17134	92.61834	98.74085

5. **Using the fitted model, compute the return period of 100 mm of precipitation.**

Every 221.82 years, we can expect >100mm of rain

6. **Using the fitted model, compute the probability that there will be a day in the next year when the precipitation exceeds 150 mm.**

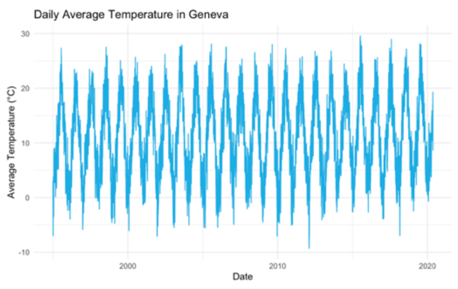
The probability that there is a day with >150mm rain next year is 7.03152640343374e-05, (= 0.007%).

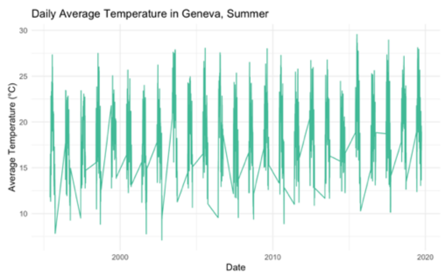
7. **Compare the results with the block maxima method. Explain the drawbacks and advantages of using the POT approach compared to the block maxima method. Which method do you prefer?**

Interestingly, our approach to the POT and Block Maxima methods yield similar (in the same range) results for the return period of >100mm rain. However, we find widely different results when computing the probability that there will be a day in the next year when the precipitation exceed 150mm. This could be due to several factor. The first hypothesis is that, because the POT method is better at modelling extreme events than the block maxima (one of the advantages), the probability found by the block maxima method widely over-estimate the risk. The second hypothesis is that the threshold selection for the POT method (40) introduced some bias in the distribution, therefore leading to high variance in estimates (Drawback of the POT, subjective choice, yet critical).

Part 3: Clustering and Seasonal Variations

1. **Upload the Geneva temperature data. Plot the data. Subset the data for the summer months (June to September).**



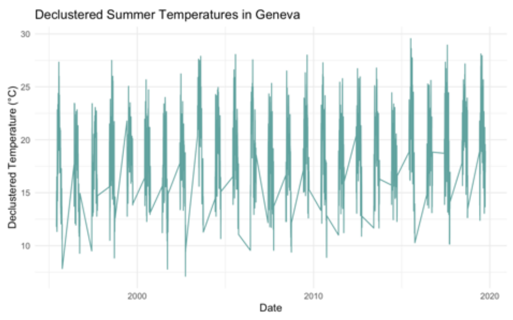


2. **Compute the extremal index of the subsetted series with appropriatelly chosen threshold (for example, you can use extremalindex function in extRemes package). Do the extremes occur in clusters? What is the probability that if the temperature today is extreme (above the chosen threshold) then tomorrow will be also extreme?**

Temperature (C)	Extremal index	Number of clusters
20	0.1518	162
25	0.2780	40

With a threshold of 20 degrees Celcius, our extremal index is close to 0 (0.15), indicating that extreme temperatures tend to happen in blocks (clusters). This can be illustrated for example by heatwaves in Summer. We can use the extremal index to approximate the probability that tomorrow is extreme, if today is extreme. For instance, if today is 25 °C, there is a 73% (1-0,27) probability that tomorrow is also extreme.

3. **Decluster the data using a suitable threshold. Plot the resulting declustered data.**



4. **Fit a Generalized Pareto Distribution (GPD) to the data, both raw and declustered. Compare the models and compute 10-year return level.**

10-year return level

Raw data	Declustered data
29.2317	29.1623

The raw summer data return level for 10-years is 29.23 degrees Celcius. It means that on average, the temperature of 29.23 C will be exceeded on average every 10 years. Without clustering of extreme events, the 10-years return level is 29.16 degrees Celcius. This is very close to the raw data and indicates that the clustering of extreme values does not have a significant impact on the 10-years return level.

Practical 3

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

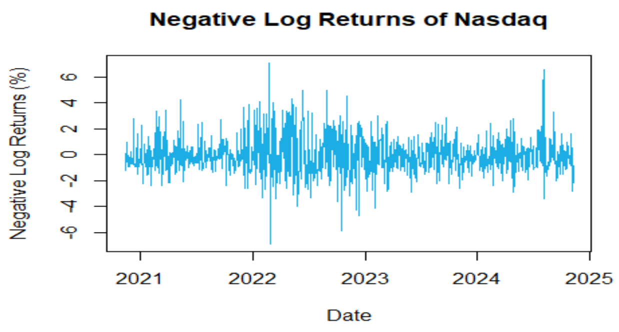
- Which risk management model, based on the POT or GEV method, provides a better understanding of extreme risks to enhance the profitability of a portfolio invested in the NASDAQ Composite Index during the period Nov 2020 – Nov 2024?

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

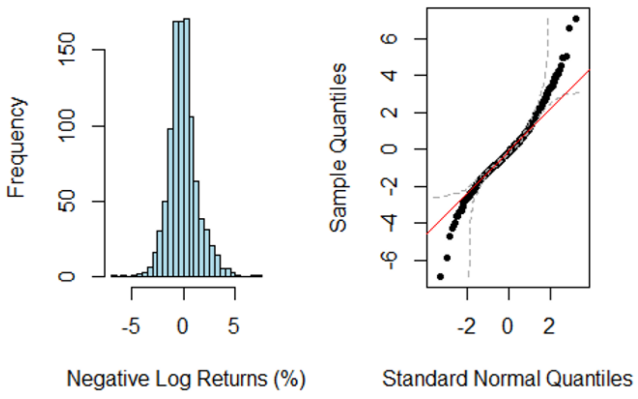
a. Checking Stationarity

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%.

b. 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.

2. Extreme Values Theory

a. Block Maxima

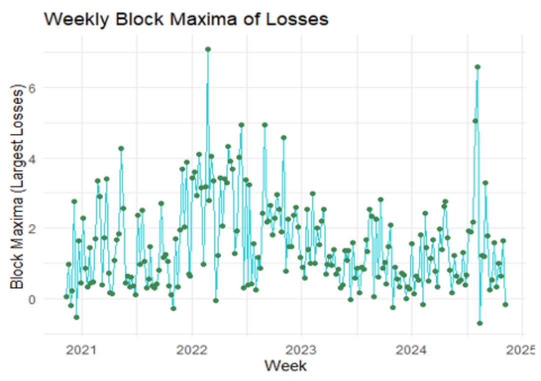
i.Calculation

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.

i.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.

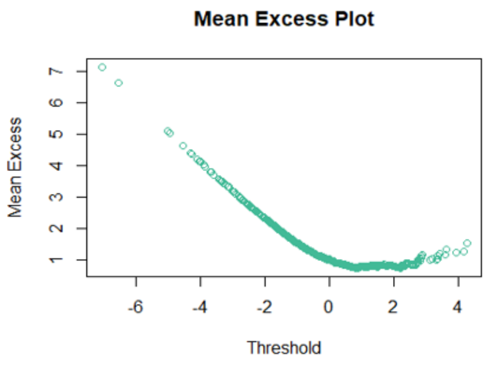
Confidence Level	Historical VaR	Historical ES	Parametric VaR	Parametric ES
95%	0.612%	-0.206%	0.049%	-0.871%
99%	0.272%	-0.504%	0.402%	-0.483%

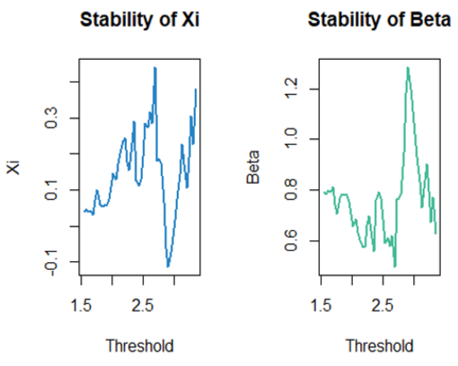
The parametric approach shows a wider range, especially in ES at 95%, where the estimated loss magnitude is significantly higher. This discrepancy reflects the sensitivity of the parametric method, which assumes a heavier-tailed GEV model. However, since the underlying distribution is known to be non-normal,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

b. Peaks Over Threshold

i. 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. Region 2.2 and 2.8 shows stability, making it reliable range for analyzing extreme events. At the end the chosen threshold for stability region was 2.5.

ii. 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.

iii. 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.

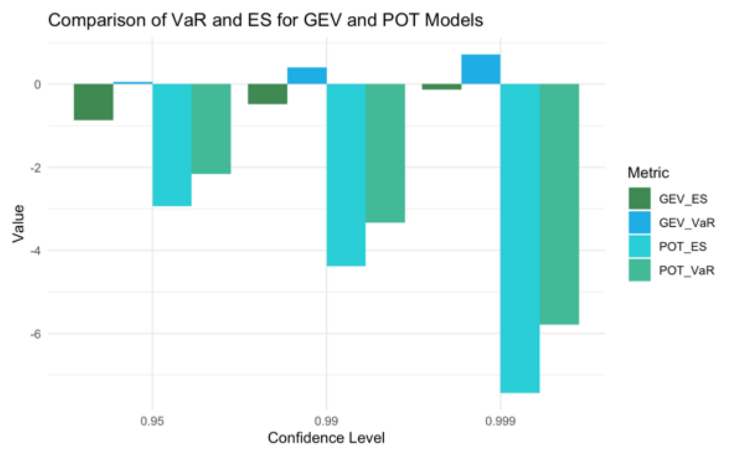
Confidence Level	Parametric VaR	Parametric ES	Historical VaR	Historical ES
95%	-2.163	-2.932	-6.175	-7.221
99.5%	-3.963	-5.163	-7.587	-7.736
99.9%	-5.791	- 7.429	-7.706	-7.736

The Parametric VaR and ES rely on assumptions of normality, which may underestimate tail risks, as shown in the earlier analysis (non-normal returns). While the Historical VaR and ES do not make these assumptions, instead relying on actual extreme events from the data, leading to more conservative estimates.

The Historical POT approach better captures extreme losses due to its reliance on observed data and focus on the most extreme values. This makes it more suitable for managing tail risks, especially in volatile markets like the NASDAQ Composite. Hence, it is more suitable for risk-averse investors who want to be prepared for worst-case scenarios.

3. Model Comparison

Looking at the results from the comparison between GEV and POT models' VaR and ES, POT model captures more extreme losses than GEV, which may be useful for risk averse scenarios. GEV model may be underestimating the risks given that it only considers block maxima, while POT is considering ALL exceedances above the threshold, making the model more sensitive to extreme events.



Given these results, for three different confidence intervals, POT model is better than GEV, especially for financial events which require a more rigourous analysis rather than the simplicity GEV offers.

4. Conclusion

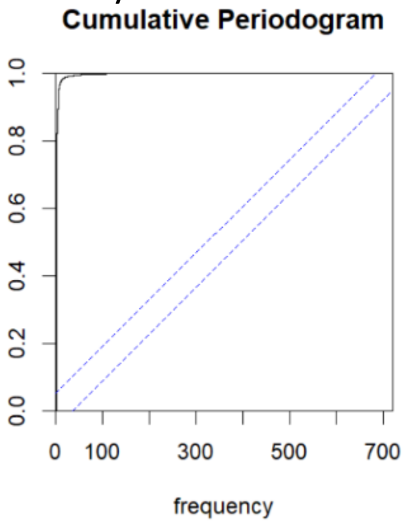
In order to evaluate extreme risks in the Nasdaq Composite Index using Extreme Value Theory, two models were implemented: Block Maxima (GEV), which focuses on capturing the largest loss in blocks and Peaks Over Threshold (POT), which analyzes losses exceeding a threshold. Through the Stop Loss Strategy, POT models demonstrated its ability to reduce losses during volatile periods, specifically in 2022.

Both models provided VaR and ES estimates for different confidence levels in order to determine which one is better. The results demonstrate POT model performs better for financial events. GEV provides a simple summary within predefined time blocks, so it underestimates extreme events because it only considers the largest loss within each block. This model is more appropriate for lower risk environments which require computational simplicity. In the other hand, POT is more sensitive to extreme events so it produced more conservative VaR and ES estimates, especially for 99% and 99.9% confidence levels. This model works better for volatile environments such as the financial one.

Appendix:

1. **Practical 1**

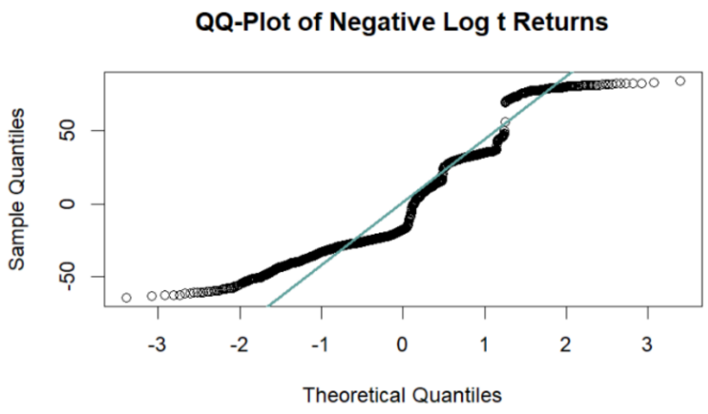
1.1.1 Dickey Fuller Test



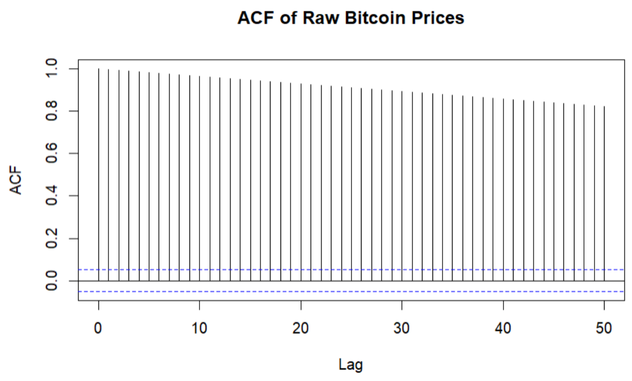
1.1.2 Dickey Fuller Test

```
	Augmented Dickey-Fuller Test

data:  crypto$Bitcoin
Dickey-Fuller = -2.4484, Lag order = 11, p-value = 0.3885
alternative hypothesis: stationary
```

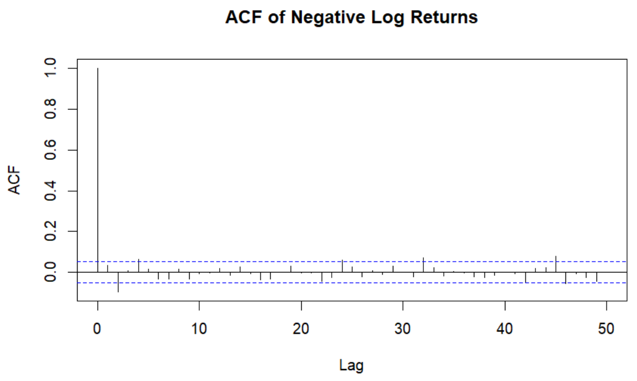
1.1.3 QQ plot t fitted



1.2.1 ACF of Raw Bitcoin Prices



1.2.2 ACF of Negative Log Returns



1.2.3 Box-Ljung test

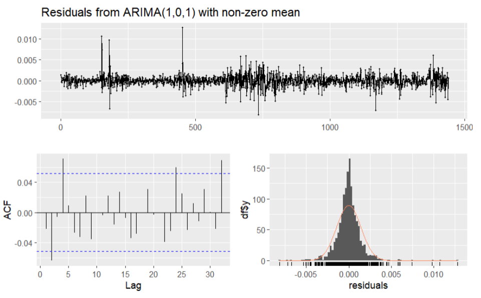
```
        Box-Ljung test

data:  crypto$Bitcoin
X-squared = 26873, df = 20, p-value < 2.2e-16


        Box-Ljung test

data:  neg_log_returns
X-squared = 33.356, df = 20, p-value = 0.03082
```

1.2.4 ARIMA models

```
Call:
arima(x = neg_log_returns, order = c(1, 0, 1))

Coefficients:
          ar1     ma1  intercept
      -0.4903  0.5515      1e-04
s.e.   0.1634  0.1554      0e+00

sigma^2 estimated as 2.041e-06:  log likelihood = 7385.16,  aic = -14762.32
Series: neg_log_returns
ARIMA(2,0,2) with non-zero mean

Coefficients:
          ar1      ar2     ma1     ma2   mean
      -0.0520  -0.5415  0.0853  0.4479  1e-04
s.e.   0.1717   0.1664  0.1824  0.1773  0e+00

sigma^2 = 2.029e-06:  log likelihood = 7391.82
AIC=-14771.65   AICc=-14771.59   BIC=-14740.02
```

```
        Ljung-Box test

data:  Residuals from ARIMA(1,0,1) with non-zero mean
Q* = 19.33, df = 8, p-value = 0.01319

Model df: 2.   Total lags used: 10


        Ljung-Box test

data:  Residuals from ARIMA(2,0,2) with non-zero mean
Q* = 4.7774, df = 6, p-value = 0.5727

Model df: 4.   Total lags used: 10
```

1.2.5 Residuals from ARIMA(1,0,1)



1.2.6 GARCH model with normal distribution

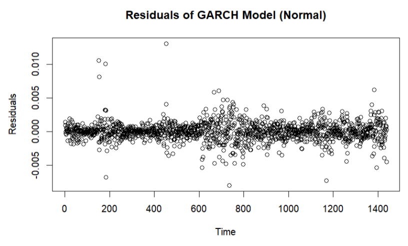
```
Series Initialization:
 ARMA Model:                arma
 Formula Mean:              ~ arma(0, 0)
 GARCH Model:               garch
 Formula Variance:          ~ garch(1, 1)
 ARMA Order:                0 0
 Max ARMA Order:            0
 GARCH Order:               1 1
 Max GARCH Order:           1
 Maximum Order:             1
 Conditional Dist:          norm
 h.start:                   2
 llh.start:                 1
 Length of Series:          1439
 Recursion Init:            mci
 Series Scale:              0.001432575

Parameter Initialization:
 Initial Parameters:          $params
 Limits of Transformations:   $U, $V
 Which Parameters are Fixed?  $includes
 Parameter Matrix:
 Index List of Parameters to be Optimized:
    mu  omega alpha1  beta1
     1      2      3      5
 Persistence:                  0.9


--- START OF TRACE ---
Selected Algorithm: nlminb

R coded nlminb Solver:

  0:     1868.9969: 0.0463426 0.100000 0.100000 0.800000
  1:     1845.3472: 0.0463386 0.0722160 0.108922 0.785987
  2:     1826.7217: 0.0463319 0.0658435 0.139406 0.794821
  3:     1820.4722: 0.0463247 0.0360279 0.147892 0.785497
  4:     1803.5420: 0.0463069 0.0398160 0.175916 0.801247
  5:     1800.7708: 0.0462703 0.0144484 0.195086 0.807274
  6:     1795.8088: 0.0461648 0.0310054 0.216934 0.790207
  7:     1794.3055: 0.0460237 0.0337098 0.231903 0.761794
  8:     1793.2439: 0.0454747 0.0231887 0.257991 0.772579
  9:     1793.0719: 0.0454603 0.0264773 0.256090 0.769843
 10:     1793.0153: 0.0452897 0.0245525 0.255405 0.767770
 11:     1792.8826: 0.0450869 0.0261267 0.256011 0.767644
 12:     1792.8412: 0.0448848 0.0261102 0.255634 0.765953
 13:     1792.7972: 0.0446729 0.0270512 0.256076 0.765836
 14:     1792.0632: 0.0361519 0.0272497 0.270440 0.757038
 15:     1791.3631: 0.0276382 0.0265526 0.254898 0.767018
 16:     1791.2157: 0.0233757 0.0276512 0.253946 0.761841
 17:     1790.9926: 0.0191108 0.0266855 0.253412 0.766055
 18:     1790.9060: 0.0148424 0.0277822 0.254301 0.764302
 19:     1790.8888: 0.0105756 0.0259119 0.253338 0.766956
 20:     1790.8688: 0.0105983 0.0268364 0.253507 0.766342
 21:     1790.8687: 0.0106081 0.0268227 0.253559 0.766278
 22:     1790.8687: 0.0106070 0.0268181 0.253546 0.766295
 23:     1790.8687: 0.0106071 0.0268185 0.253547 0.766294
```

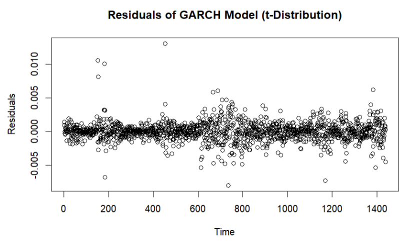
```
 24:     1686.7522: 0.0304619 0.0196569 0.184215 0.824216  4.16471
 25:     1686.6779: 0.0291031 0.0191586 0.191152 0.818296  4.24715
 26:     1686.6453: 0.0255874 0.0193998 0.193192 0.817815  4.20266
 27:     1686.6273: 0.0245673 0.0203923 0.190409 0.816604  4.30929
 28:     1686.6226: 0.0248337 0.0199501 0.190837 0.817469  4.28128
 29:     1686.6226: 0.0248426 0.0199336 0.190819 0.817521  4.27986
 30:     1686.6226: 0.0248430 0.0199353 0.190825 0.817516  4.27984

Final Estimate of the Negative LLH:
 LLH:  -7736.355    norm LLH:  -5.376202
          mu        omega       alpha1        beta1        shape
3.558952e-05 4.091266e-08 1.908245e-01 8.175158e-01 4.279836e+00

R-optimhess Difference Approximated Hessian Matrix:
                  mu         omega        alpha1         beta1         shape
mu     -2.035760e+09  2.882255e+10 -7.068135e+04 -1.420219e+05 -6.295524e+03
omega   2.882255e+10 -1.776641e+16 -7.870029e+09 -1.471487e+10 -2.520044e+08
alpha1 -7.068135e+04 -7.870029e+09 -7.415721e+03 -1.059650e+04 -2.069406e+02
beta1  -1.420219e+05 -1.471487e+10 -1.059650e+04 -1.737162e+04 -3.062470e+02
shape  -6.295524e+03 -2.520044e+08 -2.069406e+02 -3.062470e+02 -1.009168e+01
attr(,"time")
Time difference of 0.08910513 secs

--- END OF TRACE ---
```

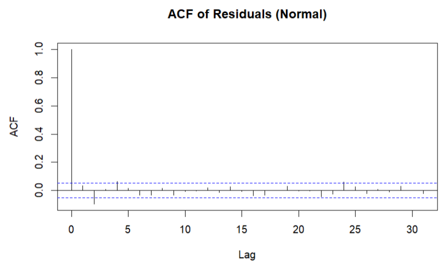
1.2.7 GARCH model with t-distribution

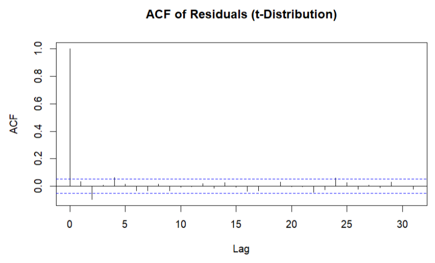
```
Title:
 GARCH Modelling

Call:
 garchFit(formula = ~garch(1, 1), data = neg_log_returns, cond.dist = "std")

Mean and Variance Equation:
 data ~ garch(1, 1)
<environment: 0x000001ca20dc8cc8>
 [data = neg_log_returns]

Conditional Distribution:
 std

Coefficient(s):
        mu       omega      alpha1       beta1       shape
3.5590e-05  4.0913e-08  1.9082e-01  8.1752e-01  4.2798e+00

Std. Errors:
 based on Hessian


Error Analysis:
        Estimate  Std. Error  t value Pr(>|t|)
mu     3.559e-05   2.224e-05    1.600   0.1095
omega  4.091e-08   1.593e-08    2.568   0.0102 *
alpha1 1.908e-01   3.935e-02    4.849 1.24e-06 ***
beta1  8.175e-01   3.248e-02   25.170  < 2e-16 ***
shape  4.280e+00   4.884e-01    8.763  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Log Likelihood:
 7736.355    normalized:  5.376202


Standardised Residuals Tests:
                                Statistic   p-Value
 Jarque-Bera Test   R    Chi^2  3278.9537544 0.0000000
 Shapiro-Wilk Test  R    W         0.9421334 0.0000000
 Ljung-Box Test     R    Q(10)    11.0874954 0.3507421
 Ljung-Box Test     R    Q(15)    12.2447831 0.6604143
 Ljung-Box Test     R    Q(20)    13.4702881 0.8563071
 Ljung-Box Test     R^2  Q(10)    12.3216177 0.2641083
 Ljung-Box Test     R^2  Q(15)    13.1928753 0.5874032
 Ljung-Box Test     R^2  Q(20)    14.0749451 0.8266758
 LM Arch Test       R    TR^2     12.3122128 0.4209424

Information Criterion Statistics:
      AIC       BIC       SIC      HQIC
-10.74545 -10.72714 -10.74548 -10.73862
```

1.2.8 Plots for GARCH models









1.2.9 GARCH(1, 1) model on the ARIMA residuals

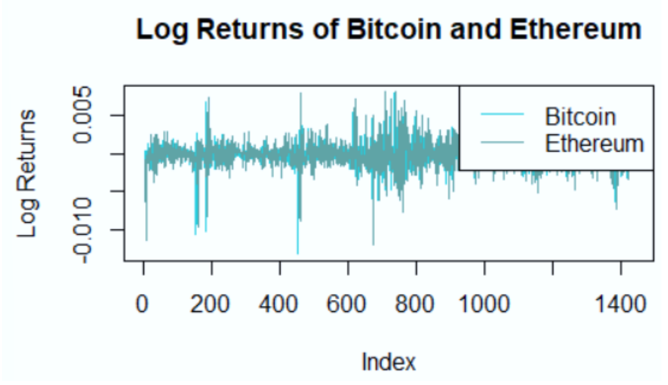
```
beta1   7.579e-01   2.434e-02   31.134  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Log Likelihood:
 7627.039    normalized:  5.300236

Description:
 Mon Dec  9 21:04:16 2024 by user: Владелец


Standardised Residuals Tests:
                                    Statistic   p-Value
 Jarque-Bera Test   R    Chi^2  2537.823850 0.0000000
 Shapiro-Wilk Test  R    W         0.945703 0.0000000
 Ljung-Box Test     R    Q(10)    10.890174 0.3661383
 Ljung-Box Test     R    Q(15)    11.971102 0.6812151
 Ljung-Box Test     R    Q(20)    13.626059 0.8489384
 Ljung-Box Test     R^2  Q(10)    12.237344 0.2694860
 Ljung-Box Test     R^2  Q(15)    13.290573 0.5798652
 Ljung-Box Test     R^2  Q(20)    14.030754 0.8289332
 LM Arch Test       R    TR^2     12.413089 0.4130997

Information Criterion Statistics:
      AIC       BIC       SIC      HQIC 
-10.59491 -10.58026 -10.59493 -10.58944 
```

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)
```

Bitcoin <dbl>	Ethereum <dbl>	Bitcoin_log_return <dbl>
47313.35	3401.124	NA
47243.93	3394.598	-1.468342e-03
47197.27	3388.820	-9.881028e-04
47207.95	3386.160	2.261376e-04
47206.35	3381.049	-3.386291e-05
47224.81	3342.820	3.909424e-04

```
# 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)
```



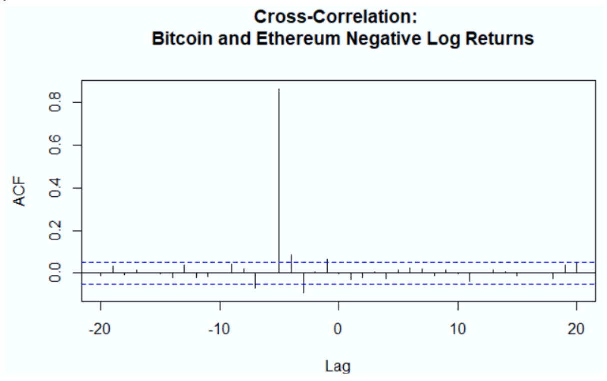
a. Are the negative log returns of Bitcoin and ETH dependent? Compute the correlation using cor.test() function. Can we conclude that these series are independent?

```
# 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)
```

```
        Pearson's product-moment correlation

data:  crypto$Bitcoin_negative_log_return and crypto$Ethereum_negative_log_return
t = -0.11935, df = 1437, p-value = 0.905
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.05481486  0.04853492
sample estimates:
        cor
-0.00314838
```

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

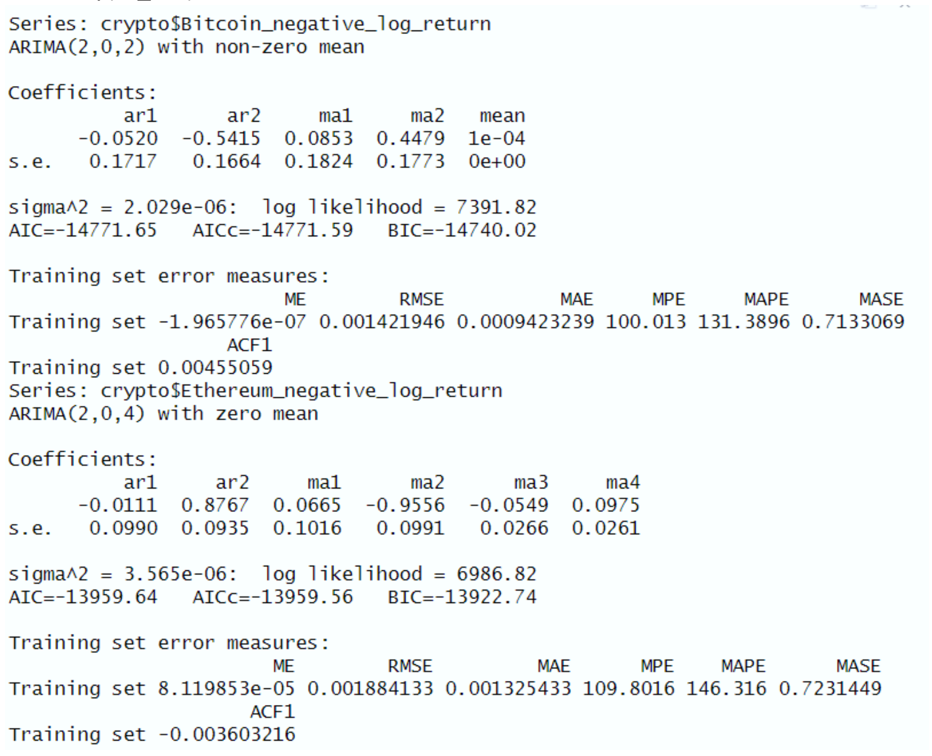
```
# Calculate the cross-correlation function (CCF) with a smaller title font size
ccf_result <- ccf(
  crypto$Bitcoin_negative_log_return,
  crypto$Ethereum_negative_log_return,
  lag.max = 20,
  plot = TRUE,
  main = "Cross-Correlation:
  Bitcoin and Ethereum Negative Log Returns",
  cex.main = 0.6  # Adjust title size (smaller font)
)
```



c. 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?

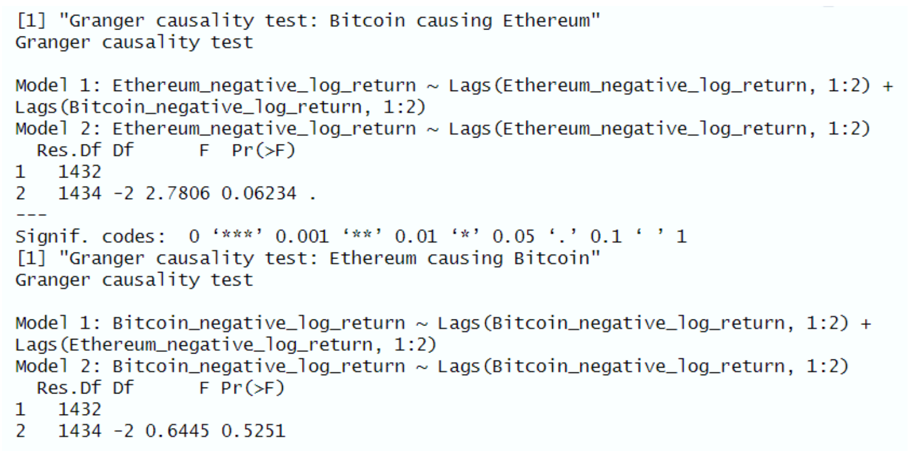
```
# Fit ARIMA model to Bitcoin negative log returns and auto-select order based on AIC/BIC
fit_btc <- auto.arima(crypto$Bitcoin_negative_log_return, ic = "aic")

# 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)
```



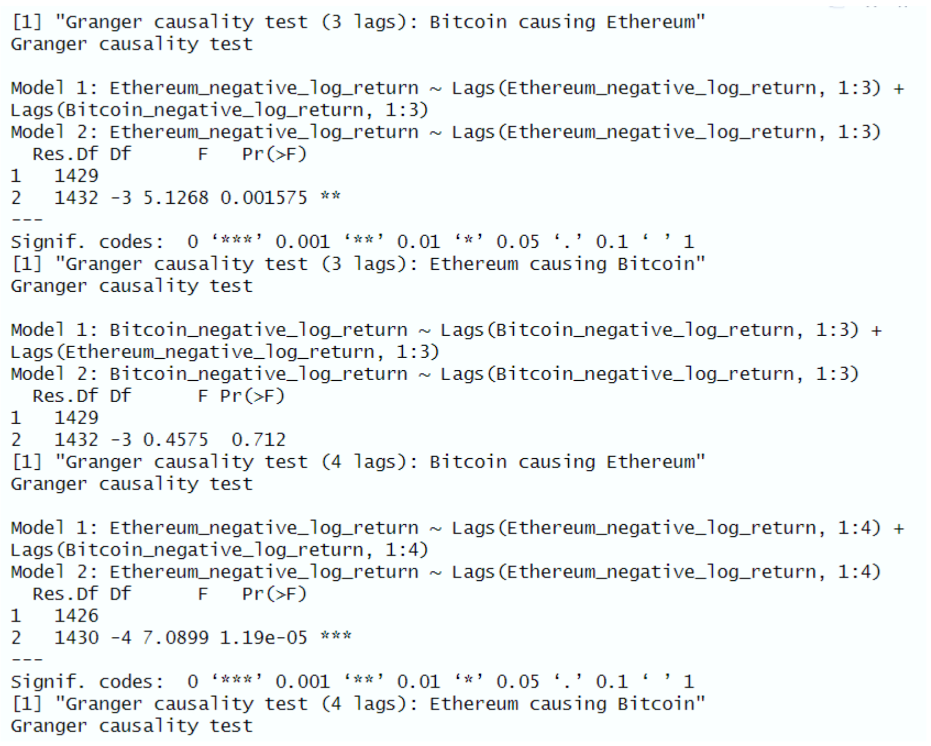
```
Series: crypto$Bitcoin_negative_log_return
ARIMA(2,0,2) with non-zero mean

Coefficients:
          ar1      ar2     ma1     ma2    mean
      -0.0520  -0.5415  0.0853  0.4479   1e-04
s.e.   0.1717   0.1664  0.1824  0.1773   0e+00

sigma^2 = 2.029e-06:  log likelihood = 7391.82
AIC=-14771.65   AICc=-14771.59   BIC=-14740.02

Training set error measures:
                        ME        RMSE          MAE     MPE     MAPE      MASE
Training set -1.965776e-07 0.001421946 0.0009423239 100.013 131.3896 0.7133069
                   ACF1
Training set 0.00455059
Series: crypto$Ethereum_negative_log_return
ARIMA(2,0,4) with zero mean

Coefficients:
          ar1     ar2     ma1      ma2      ma3     ma4
      -0.0111  0.8767  0.0665  -0.9556  -0.0549  0.0975
s.e.   0.0990  0.0935  0.1016   0.0991   0.0266  0.0261

sigma^2 = 3.565e-06:  log likelihood = 6986.82
AIC=-13959.64   AICc=-13959.56   BIC=-13922.74

Training set error measures:
                       ME        RMSE         MAE      MPE    MAPE      MASE
Training set 8.119853e-05 0.001884133 0.001325433 109.8016 146.316 0.7231449
                     ACF1
Training set -0.003603216
```

```
# 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)
```

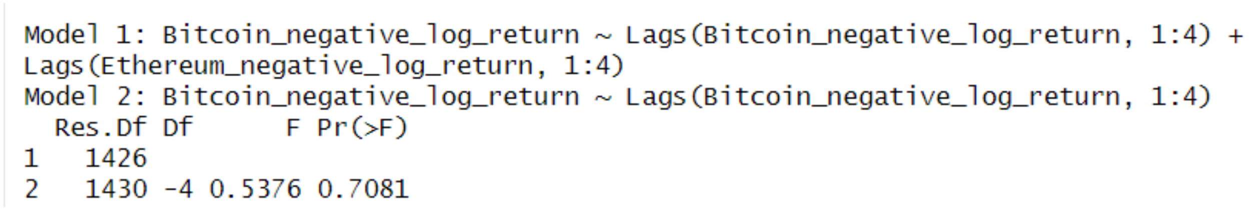


```
[1] "Granger causality test: Bitcoin causing Ethereum"
Granger causality test

Model 1: Ethereum_negative_log_return ~ Lags(Ethereum_negative_log_return, 1:2) +
Lags(Bitcoin_negative_log_return, 1:2)
Model 2: Ethereum_negative_log_return ~ Lags(Ethereum_negative_log_return, 1:2)
  Res.Df Df      F  Pr(>F)
1   1432
2   1434 -2 2.7806 0.06234 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
[1] "Granger causality test: Ethereum causing Bitcoin"
Granger causality test

Model 1: Bitcoin_negative_log_return ~ Lags(Bitcoin_negative_log_return, 1:2) +
Lags(Ethereum_negative_log_return, 1:2)
Model 2: Bitcoin_negative_log_return ~ Lags(Bitcoin_negative_log_return, 1:2)
  Res.Df Df      F Pr(>F)
1   1432
2   1434 -2 0.6445 0.5251
```

```
# 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)
```



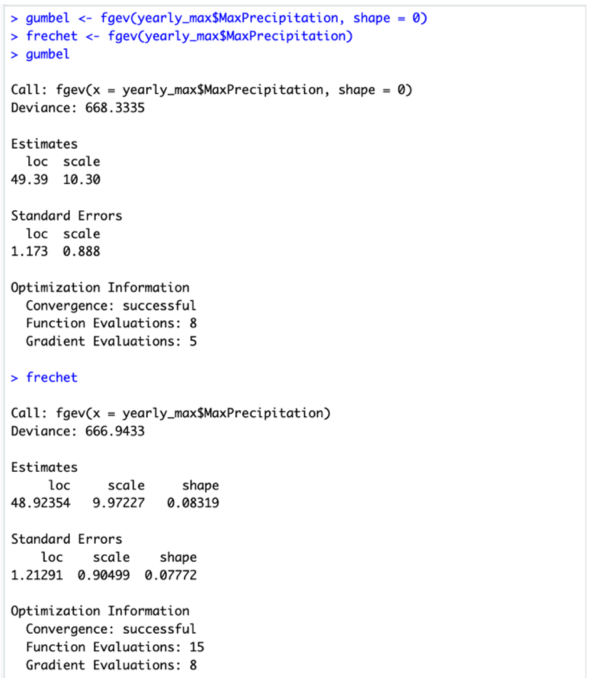
```
[1] "Granger causality test (3 lags): Bitcoin causing Ethereum"
Granger causality test

Model 1: Ethereum_negative_log_return ~ Lags(Ethereum_negative_log_return, 1:3) +
Lags(Bitcoin_negative_log_return, 1:3)
Model 2: Ethereum_negative_log_return ~ Lags(Ethereum_negative_log_return, 1:3)
  Res.Df Df      F   Pr(>F)
1   1429
2   1432 -3 5.1268 0.001575 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
[1] "Granger causality test (3 lags): Ethereum causing Bitcoin"
Granger causality test

Model 1: Bitcoin_negative_log_return ~ Lags(Bitcoin_negative_log_return, 1:3) +
Lags(Ethereum_negative_log_return, 1:3)
Model 2: Bitcoin_negative_log_return ~ Lags(Bitcoin_negative_log_return, 1:3)
  Res.Df Df      F Pr(>F)
1   1429
2   1432 -3 0.4575  0.712
[1] "Granger causality test (4 lags): Bitcoin causing Ethereum"
Granger causality test

Model 1: Ethereum_negative_log_return ~ Lags(Ethereum_negative_log_return, 1:4) +
Lags(Bitcoin_negative_log_return, 1:4)
Model 2: Ethereum_negative_log_return ~ Lags(Ethereum_negative_log_return, 1:4)
  Res.Df Df      F   Pr(>F)
1   1426
2   1430 -4 7.0899 1.19e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
[1] "Granger causality test (4 lags): Ethereum causing Bitcoin"
Granger causality test
```



```
Model 1: Bitcoin_negative_log_return ~ Lags(Bitcoin_negative_log_return, 1:4) +
Lags(Ethereum_negative_log_return, 1:4)
Model 2: Bitcoin_negative_log_return ~ Lags(Bitcoin_negative_log_return, 1:4)
  Res.Df Df      F Pr(>F)
1   1426
2   1430 -4 0.5376 0.7081
```

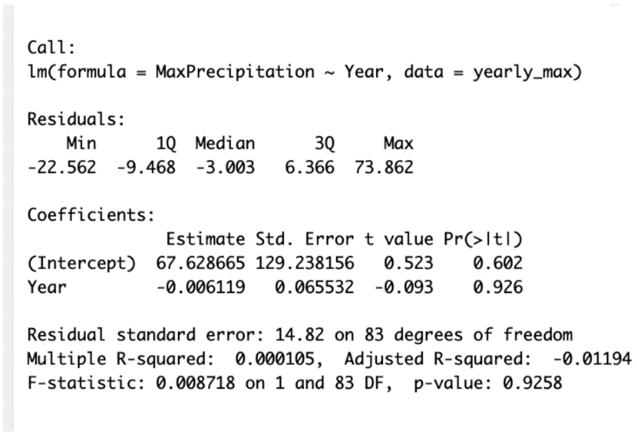
2. Practical 2

2.1 Gumbel and Frechet Distributions



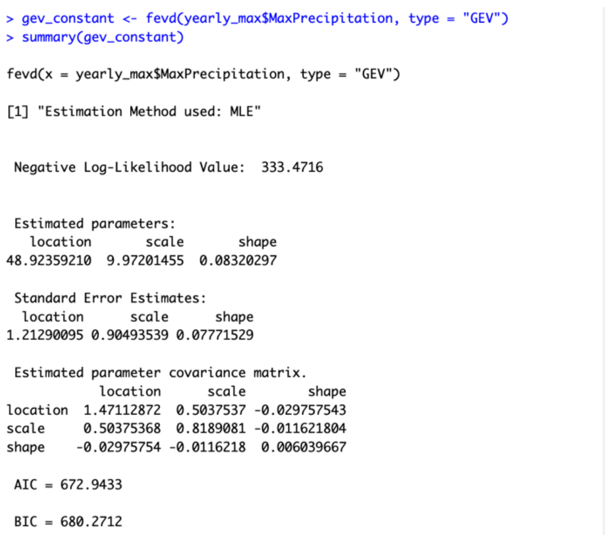
```
> gumbel <- fgev(yearly_max$MaxPrecipitation, shape = 0)
> frechet <- fgev(yearly_max$MaxPrecipitation)
> gumbel

Call: fgev(x = yearly_max$MaxPrecipitation, shape = 0)
Deviance: 668.3335

Estimates
  loc  scale
49.39  10.30

Standard Errors
  loc  scale
1.173  0.888

Optimization Information
  Convergence: successful
  Function Evaluations: 8
  Gradient Evaluations: 5

> frechet

Call: fgev(x = yearly_max$MaxPrecipitation)
Deviance: 666.9433

Estimates
     loc     scale     shape
48.92354   9.97227   0.08319

Standard Errors
    loc     scale     shape
1.21291   0.90499   0.07772

Optimization Information
  Convergence: successful
  Function Evaluations: 15
  Gradient Evaluations: 8
```

2.2 Linear Model to the Yearly Maximum Precipitation

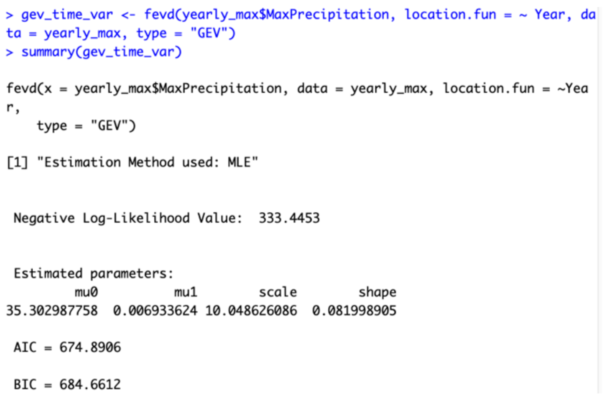


```
Call:
lm(formula = MaxPrecipitation ~ Year, data = yearly_max)

Residuals:
    Min      1Q  Median      3Q     Max 
-22.562  -9.468  -3.003   6.366  73.862 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  67.628665 129.238156   0.523    0.602
Year         -0.006119   0.065532  -0.093    0.926

Residual standard error: 14.82 on 83 degrees of freedom
Multiple R-squared:  0.000105,  Adjusted R-squared:  -0.01194 
F-statistic: 0.008718 on 1 and 83 DF,  p-value: 0.9258
```

2.3 GEV with Constant and Time Varying Parameters



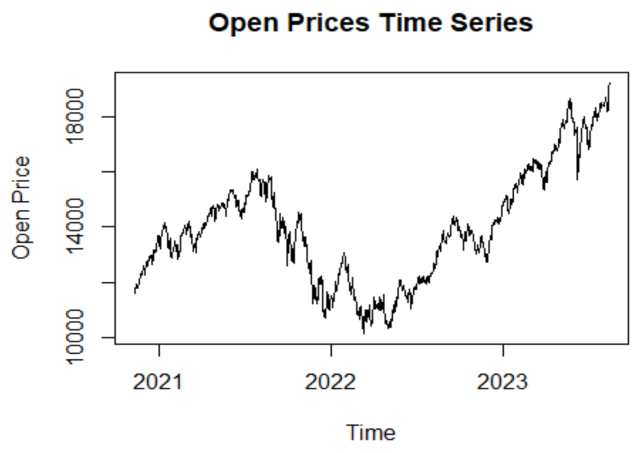
```
> gev_constant <- fevd(yearly_max$MaxPrecipitation, type = "GEV")
> summary(gev_constant)

fevd(x = yearly_max$MaxPrecipitation, type = "GEV")

[1] "Estimation Method used: MLE"


 Negative Log-Likelihood Value:  333.4716


 Estimated parameters:
   location       scale       shape 
48.92359210  9.97201455  0.08320297 

 Standard Error Estimates:
  location      scale      shape 
1.21290095 0.90493539 0.07771529 

 Estimated parameter covariance matrix.
            location      scale        shape
location  1.47112872  0.5037537 -0.029757543
scale     0.50375368  0.8189081 -0.011621804
shape    -0.02975754 -0.0116218  0.006039667

 AIC = 672.9433 

 BIC = 680.2712 
```



```
> gev_time_var <- fevd(yearly_max$MaxPrecipitation, location.fun = ~ Year, da
ta = yearly_max, type = "GEV")
> summary(gev_time_var)

fevd(x = yearly_max$MaxPrecipitation, data = yearly_max, location.fun = ~Yea
r, 
    type = "GEV")

[1] "Estimation Method used: MLE"


 Negative Log-Likelihood Value:  333.4453


 Estimated parameters:
         mu0          mu1        scale        shape 
35.302987758  0.006933624 10.048626086  0.081998905 

 AIC = 674.8906 

 BIC = 684.6612 
```

3. Practical 3

3.1 Open Prices Time Series



3.2 Dickey Fuller Test

```
## Warning in adf.test(raw_data$Return): p-value smaller than printed p-value

##
##  Augmented Dickey-Fuller Test
##
## data:  raw_data$Return
## Dickey-Fuller = -9.3213, Lag order = 10, p-value = 0.01
## alternative hypothesis: stationary
```

3.3 Shapiro Wilk Normality Test

```
##
##  Shapiro-Wilk normality test
##
## data:  raw_data$Return
## W = 0.96696, p-value = 2.483e-14

## The distribution does not appear to be normal. Caution is advised when
using parametric VaR methods.
```

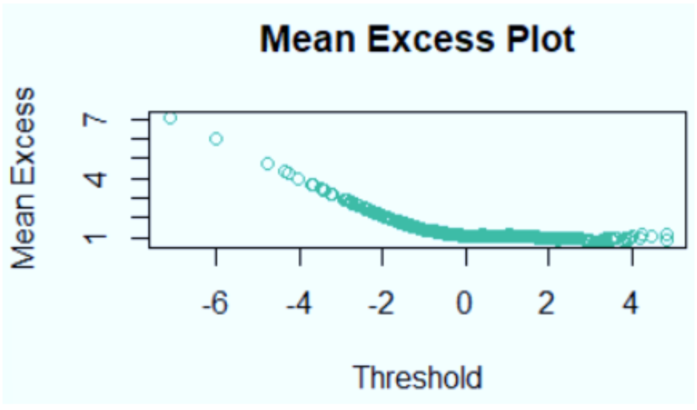
3.4 Extreme Values Theory

3.4.1 Block Maxima

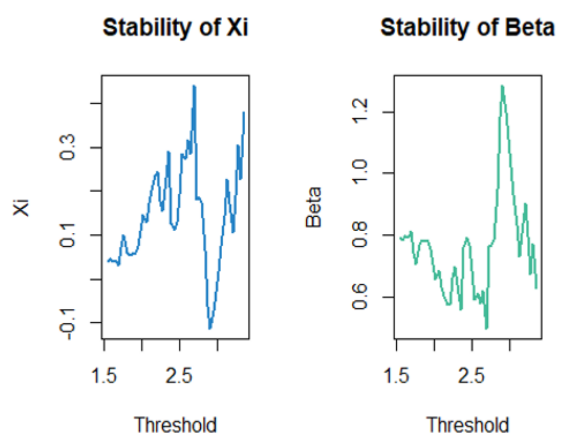
3.4.2 Peaks

3.4.2.1 POT Parametric

```
# Mean Excess Plot
meplot(-raw_data$Return, main = "Mean Excess Plot", col = "#42BA97")
```



```r
# 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))
```



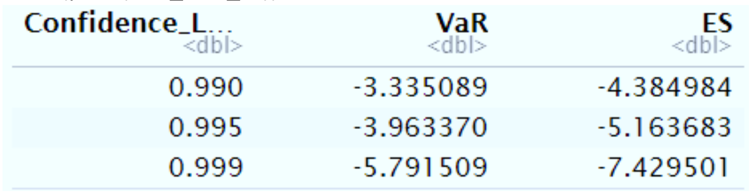
```
# Step 1: Set the threshold based on stability analysis
u <- 2.5  # Chosen threshold from stability region
# Step 2: Extract exceedances over the threshold
excesses <- -raw_data$Return[raw_data$Return > u] - u
# Print the threshold and some of the exceedances for verification
print(paste("Selected Threshold (u):", u))
print(head(excesses))
```

```
 [1] "Selected Threshold (u): 2.5"
```

```
# 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))
```

Confidence_Level <dbl>	VaR <dbl>
0.990	-3.335089
0.995	-3.963370
0.999	-5.791509

```
# 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))
```



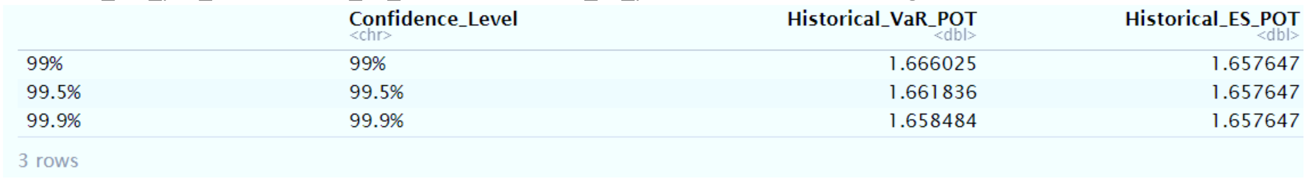
Confidence_L... <dbl>	VaR <dbl>	ES <dbl>
0.990	-3.335089	-4.384984
0.995	-3.963370	-5.163683
0.999	-5.791509	-7.429501

3.4.2.2 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")
```

```
Selected Threshold: 2.187711
```

```
# 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
```



	Confidence_Level <chr>	Historical_VaR_POT <dbl>	Historical_ES_POT <dbl>
99%	99%	1.666025	1.657647
99.5%	99.5%	1.661836	1.657647
99.9%	99.9%	1.658484	1.657647

3 rows