

Practical 1,2 and 3 for Group 1 (In full)

Lodrik Adam, Sophie Daya, Jeff Macaraeg, Julien Perini, Alexandra Boado

December 10, 2024

Table of contents

Introduction	3
Practical 1: Financial returns and normality	4
Part 1: Financial returns and normality	4
a) Load Bitcoin data and assess price stationarity	4
b) Create and plot Bitcoin negative log returns, assess stationarity	5
c) Check negative log returns normality with histograms, QQ-plots, Anderson-Darling	9
d) Fit t-distribution, compare with Normal via QQ-plot analysis	10
e) Compare t-distribution and normal tails	12
Part 3: Financial returns and normality	14
a) Check Bitcoin-ETH dependence using correlation	14
b) Analyze Bitcoin-ETH cross-correlation function	14
c) Assess predictive power with Granger test	15
d) Predict mutual impacts of drops	16
Practical 2: Weather	17
Part 1: Block maxima approach	17
a) Best fit distribution	17
b) Best fit distribution on yearly maxima	17
c) Linear model approach	17
d) GEV distribution	17
e) Diagnostic plots	17
f) Return levels prediction	18
g) Interpretation	18
h) Return period 100 mm precipitation	18
i) Probability of exceeding 150 mm	18
Part 2: Peaks-over-threshold approach	18
a) Time series plot	18
b) Mean Residual Life Plot	18
c) Fit a GPD	19
d) Return levels prediction	19
e) Return period 100 mm precipitation	19
f) Probability of exceeding 150 mm	19
g) Comparison with block maxima	19
Part 2	20
Part 3	21

Practical 3: Heat wave in Switzerland	22
Part 3: Comparing GEV and POT approaches	23
Block Maxima Approach (GEV Distribution)	23
Comparison with Peaks-over-Threshold Approach (GPD Distribution)	26
Peaks-over-Threshold Approach	26
Our results	29
Comparison of Approaches:	30
Part 4: Return Period & Probabilities of New Record High Temperatures	31
Block Maxima Approach (GEV Distribution)	31
Peaks-over-Threshold Approach (GPD Distribution)	31
Return Period	31
Probability of New Record High Temperature	32
Results	33

Introduction

This document is the annex to our final report for group 1. It contains the full code for the practicals 1, 2 and 3 with the results and some discussions. This allows anyone to reproduce our results and to understand the code we used.

Note:

Certain sections of the code have been omitted from the generated PDF and HTML versions of this annex to enhance readability. Specifically, the code for data loading, package installation, and plot generation has been excluded. The full code is available in the QMD files located in the “code” folder of our [GitHub repository](#).

Practical 1: Financial returns and normality

Part 1: Financial returns and normality

The working directory is set to: /Users/lodrik/Documents/GitHub/RA_Practicals

a) Load Bitcoin data and assess price stationarity

Question

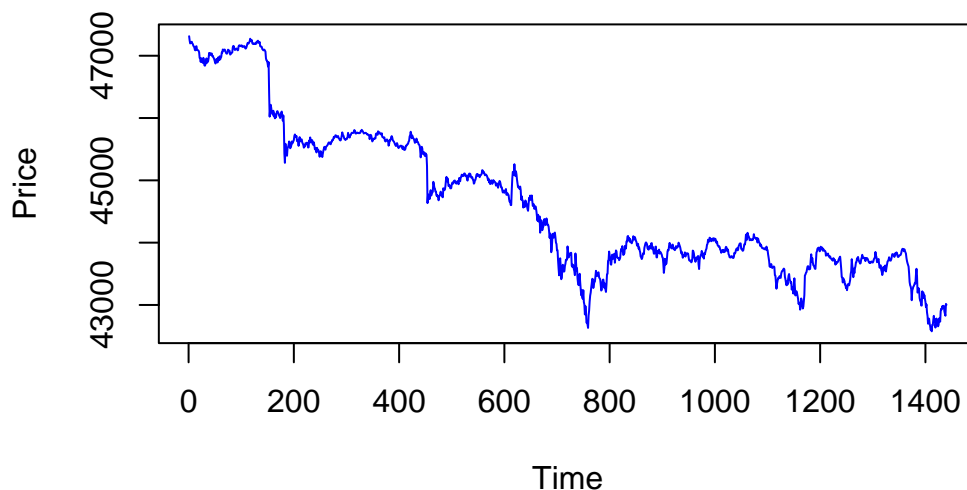
Read in the Bitcoin data from file Crypto data.csv. Then, assess the stationarity of the (raw) Bitcoin prices.

First, let's take a look at the Bitcoin Prices on a plot.

```
## Step 1: Extract the Bitcoin prices
bitcoin_prices <- crypto_data$Bitcoin

## Step 2: Plot the Bitcoin prices
plot(bitcoin_prices,
      type="l",
      col="blue",
      main="Bitcoin Prices",
      xlab="Time",
      ylab="Price")
```

Bitcoin Prices



The graph of the raw Bitcoin prices suggest that the series might not be stationary.

Let's perform the Augmented Dickey-Fuller test to check if the raw Bitcoin prices are stationary.

```
## Step 3: test for stationarity
adf.test(crypto_data$Bitcoin)
```

Augmented Dickey-Fuller Test

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

Since the p-value is significantly bigger than 0.05, we can not reject the null hypothesis and therefore, we can conclude that the raw Bitcoin prices are non-stationary.

b) Create and plot Bitcoin negative log returns, assess stationarity

Question

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

Let's create a function to compute the negative log returns of a given price series. We will then apply this function to the Bitcoin prices to compute the negative log returns.

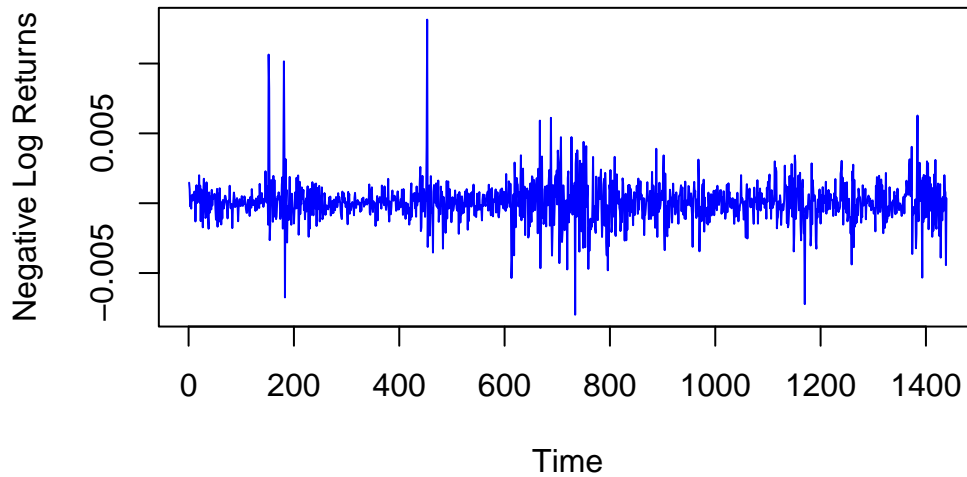
```
## Step 1: Create a function to compute negative log returns
negative_log_returns <- function(prices) {
  return(-diff(log(prices)))
}

## Step 2: Use the function on Bitcoin prices
neg_log_returns_bitcoin <- negative_log_returns(bitcoin_prices)
```

We can now plot the negative log returns series and the raw Bitcoin prices to compare.

```
## Step 3: Plot the negative log returns series
plot(neg_log_returns_bitcoin,
     type="l",
     col="blue",
     main="Negative Log Returns of Bitcoin Prices",
     xlab="Time",
     ylab="Negative Log Returns")
```

Negative Log Returns of Bitcoin Prices



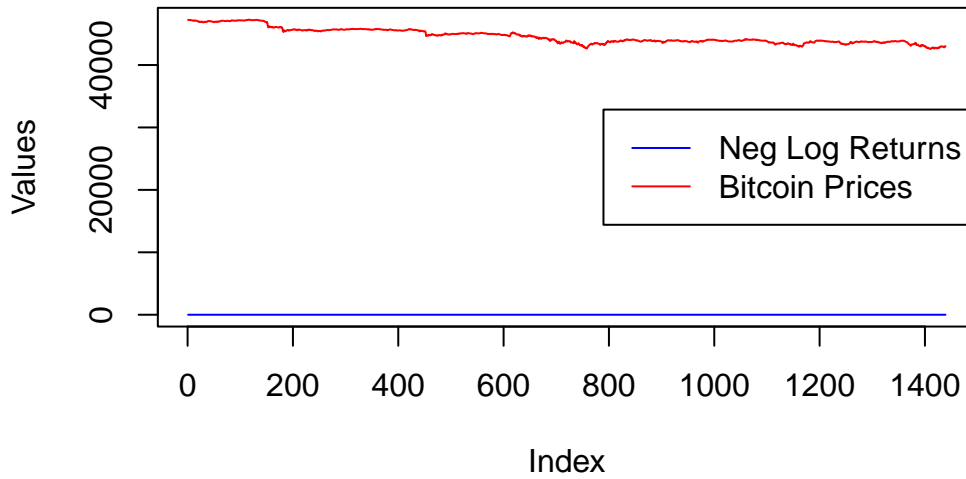
If we scale bitcoin prices and negative log returns, we can compare both time series on a plot with a common scale.

```
# Top Plot: Plot both time series on the same graph
trimmed_bitcoin_prices <- bitcoin_prices[-1] # Make sure lengths match
plot(neg_log_returns_bitcoin,
     type = "l",
     col = "blue",
     ylab = "Values",
     xlab = "Index",
     main = "Negative Log Returns and Bitcoin Prices",
     ylim = range(c(neg_log_returns_bitcoin, trimmed_bitcoin_prices)))

# Add Bitcoin prices on the same plot
lines(trimmed_bitcoin_prices, col = "red")

# Add a legend
legend("right",
     legend = c("Neg Log Returns", "Bitcoin Prices"),
     col = c("blue", "red"),
     lty = 1)
```

Negative Log Returns and Bitcoin Prices

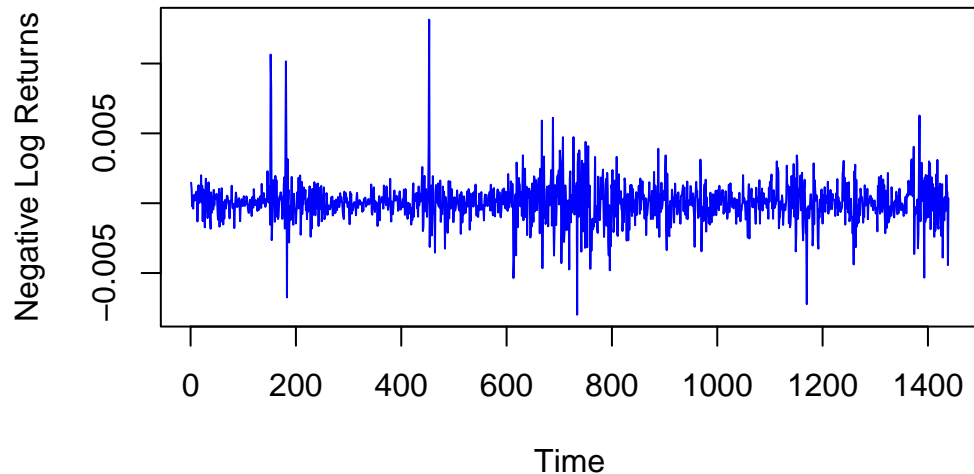


```
# Reset the plotting area to default settings (optional, for future plots)
par(mfrow=c(1,1))

# Set up the plotting area to have 3 rows and 1 column
par(mfrow=c(1,1))

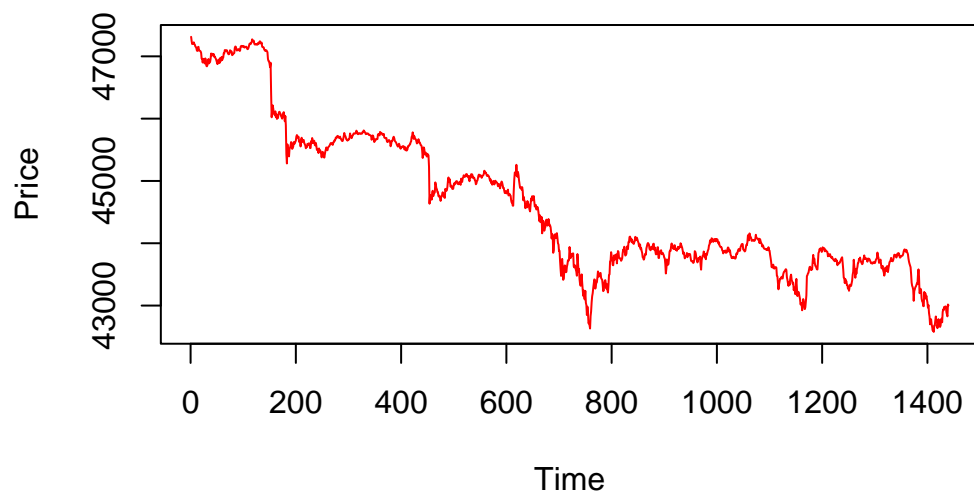
# Bottom Left Plot: Plot the negative log returns series
plot(neg_log_returns_bitcoin,
     type = "l",
     col = "blue",
     main = "Negative Log Returns of Bitcoin Prices",
     xlab = "Time",
     ylab = "Negative Log Returns")
```

Negative Log Returns of Bitcoin Prices



```
# Bottom Right Plot: Plot the Bitcoin prices
plot(bitcoin_prices,
     type = "l",
     col = "red",
     main = "Bitcoin Prices",
     xlab = "Time",
     ylab = "Price")
```

Bitcoin Prices



Visually, the negative log returns series does not appear to indicate a clear trend or seasonality. The variance, although it fluctuates in the middle, seems relatively constant. This observation suggests that the series may be stationary. To confirm this, we will perform the Augmented Dickey-Fuller test to assess the stationarity of the negative log returns.


```
## Step 5: Test the stationarity of the negative log returns with the Augmented Dickey-Fuller test
adf.test(neg_log_returns_bitcoin)
```

Augmented Dickey-Fuller Test

```
data: neg_log_returns_bitcoin
Dickey-Fuller = -11.035, Lag order = 11, p-value = 0.01
alternative hypothesis: stationary
```

Since the p-value is significantly smaller than 0.05, we can reject the null hypothesis and conclude that the negative log returns series is stationary.

c) Check negative log returns normality with histograms, QQ-plots, Anderson-Darling

Question

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

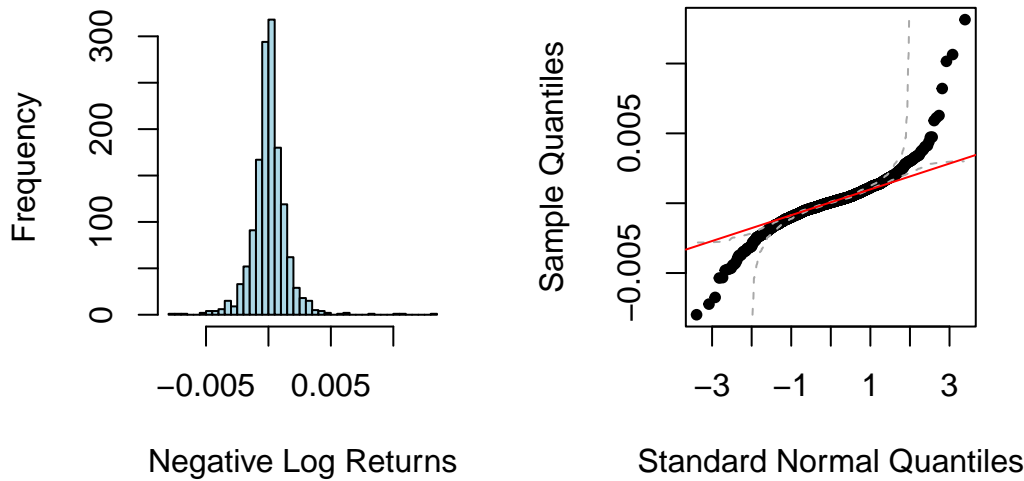
Let's first plot the histogram and QQ-plot of the negative log returns to visually assess the normality.

```
## Step 1: Plot the histogram and QQ-plot of the negative log returns
par(mfrow=c(1, 2))

# Plot the histogram of the negative log returns
hist(neg_log_returns_bitcoin,
     breaks=50,
     col="lightblue",
     main="Histogram of Negative Log Returns",
     xlab="Negative Log Returns")

# Plot the QQ-plot of the negative log returns
qqnorm(neg_log_returns_bitcoin)
qqline(neg_log_returns_bitcoin,
      col="red")
```

Histogram of Negative Log Returns



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

The histogram of the negative log returns suggests that the data may follow a normal distribution. However, we need to perform a formal test to confirm this.

```
## Step 2: Perform Anderson-Darling test for normality
ad.test(neg_log_returns_bitcoin)
```

Anderson-Darling normality test

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

Even though the Histogram suggest that the negative log returns follows a normal distribution, the p-value when performing the Andersen-Darling test is smaller than 5%. It indicates that the data does not follow a normal distribution. The Normal Q-Q plot suggest also that the data does not follow a normal distribution.

d) Fit t-distribution, compare with Normal via QQ-plot analysis

Question

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

Let's fit a t-distribution to the negative log returns and compare it with the normal distribution using a QQ-plot.

```
## Step 1: Fit a t-distribution to the negative log returns
fit_t <- fitdistr(scaling_factor * neg_log_returns_bitcoin, "t") # Multiply by 100000 to avoid nu
```

The t-distribution fit parameters are:

- mean: 5.65
- standard deviation: 84.15
- degrees of freedom: 2.77

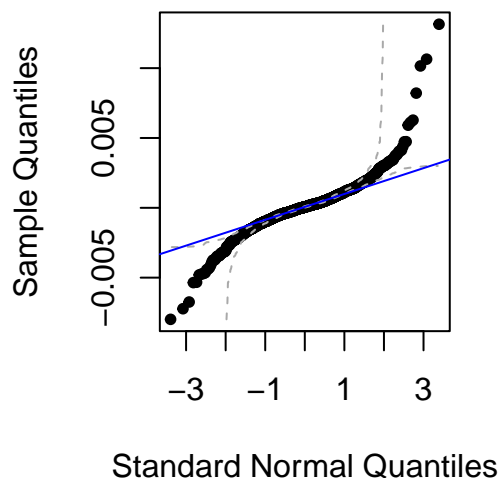
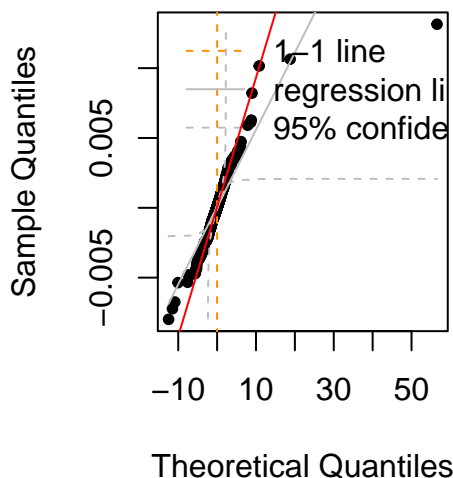
We can now compare the QQ-plot of the t-distribution with the QQ-plot of the normal distribution of [question c](#)).

```
## Step 2: Create a QQ-plot for the t-distribution and the normal distribution
par(mfrow = c(1, 2))

# Generate QQ-plot for t-distribution
df_t <- fit_t$estimate[3] # Degrees of freedom from the fit
qqplot(rt(length(neg_log_returns_bitcoin),
            df=df_t),
        neg_log_returns_bitcoin,
        main="QQ-plot for t-distribution",
        xlab="Theoretical Quantiles",
        ylab="Sample Quantiles")
qqline(neg_log_returns_bitcoin,
        col="red")

# Generate QQ-plot for normal distribution
qqnorm(neg_log_returns_bitcoin,
        main="QQ-plot for Normal distribution")
qqline(neg_log_returns_bitcoin,
        col="blue")
```

QQ-plot for t-distribution QQ-plot for Normal distribut



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

As we can see, the QQ-plot for the t-distribution is closer to the 45-degree line than the QQ-plot for the normal distribution. This suggests that the t-distribution is a better fit for the negative log returns than the normal distribution.

e) Compare t-distribution and normal tails

Question

Compare the tails of the density 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?

To compare the tails of the t-distribution and the normal distribution, we will plot the density functions of both distributions and visually assess the differences.

```
## Step 1: Fit the normal distribution to the negative log returns
fit_norm <- fitdistr(scaling_factor * neg_log_returns_bitcoin, "normal")

# Generate a sequence of values for the x-axis (log returns)
x <- seq(min(neg_log_returns_bitcoin), max(neg_log_returns_bitcoin), length = 1000)

## Step 2: Scale back the mean and sd for plotting (for both normal and t-distributions)
# For normal distribution
scaled_mean_norm <- fit_norm$estimate[1] / scaling_factor
scaled_sd_norm <- fit_norm$estimate[2] / scaling_factor

# For t-distribution
scaled_mean_t <- fit_t$estimate[1] / scaling_factor
scaled_sd_t <- fit_t$estimate[2] / scaling_factor

# Density for the normal distribution using the scaled mean and sd
dens_norm <- dnorm(x, mean = scaled_mean_norm, sd = scaled_sd_norm)

# Density for the t-distribution using the scaled parameters
dens_t <- dt((x - scaled_mean_t) / scaled_sd_t, df = fit_t$estimate[1]) / scaled_sd_t

## Step 3 : Plot the histogram of negative log returns
hist(neg_log_returns_bitcoin,
     breaks = 50,
     col = "lightblue",
     freq = FALSE, # For density plot
     main = "Negative Log Returns with Fitted Distributions",
     xlab = "Negative Log Returns")

# Add the normal distribution curve
lines(x,
      dens_norm,
      col = "black",
      lwd = 2,
      lty = 1)

# Add the t-distribution curve
lines(x,
      dens_t,
      col = "red",
```

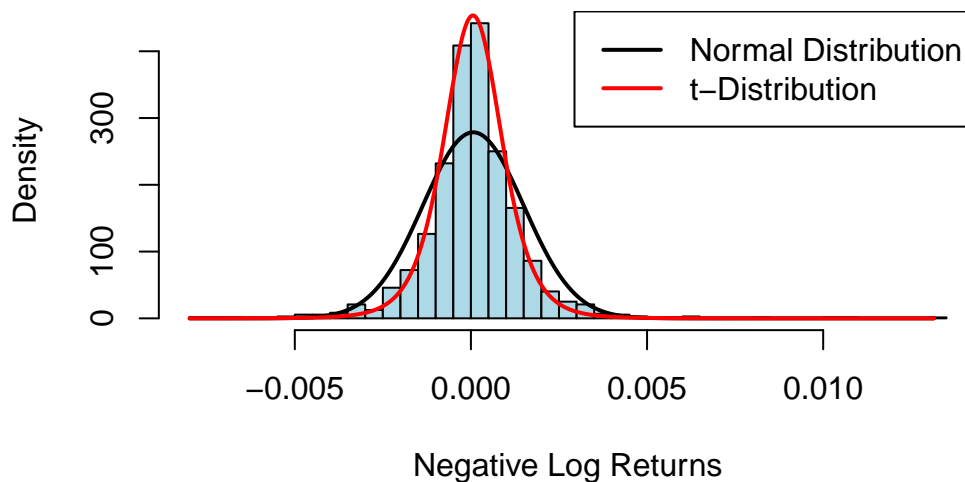
```

    lwd = 2,
    lty = 1)

# Add a legend
legend("topright",
      legend = c("Normal Distribution", "t-Distribution"),
      col = c("black", "red"),
      lty = c(1, 1),
      lwd = 2)

```

Negative Log Returns with Fitted Distributions



Visually, the tails of the t-distribution are heavier than those of the normal distribution. This means that the t-distribution assigns more probability to extreme events than the normal distribution. Therefore, we can expect more extreme, unexpected events in the t-distribution than in the normal distribution. This observation is consistent with the QQ-plot analysis in [question d](#)), where the t-distribution was a better fit for the negative log returns than the normal distribution.

Part 3: Financial returns and normality

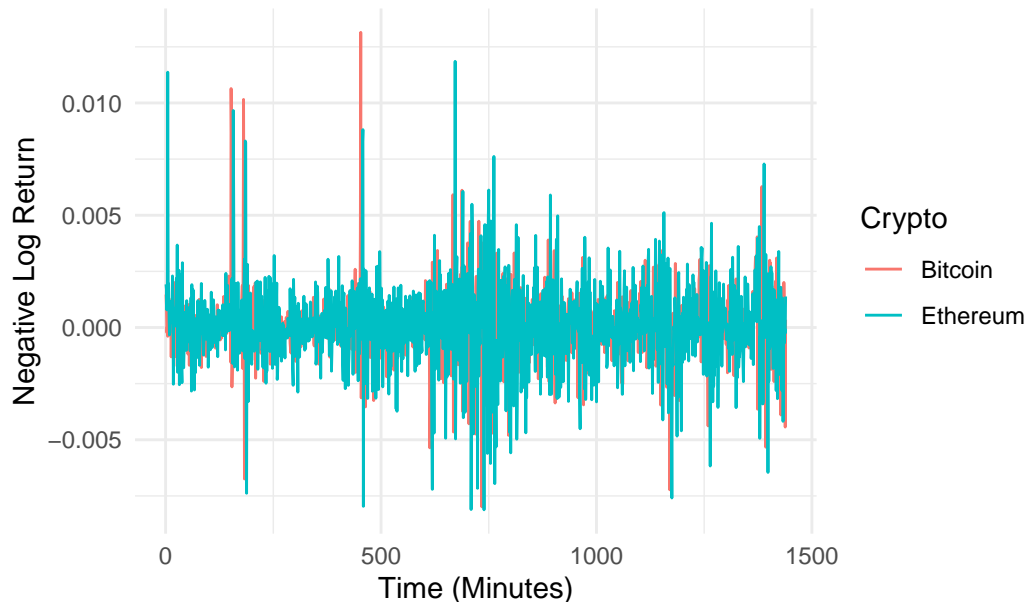
a) Check Bitcoin-ETH dependence using correlation

Question

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?

To answer this question, we will first plot both negative log returns and then perform a correlation test.

Negative Log Returns of Bitcoin and Ethereum



The graph could suggest that the negative log returns of Bitcoin and Ethereum are dependent. However, we will perform a correlation test to confirm this.

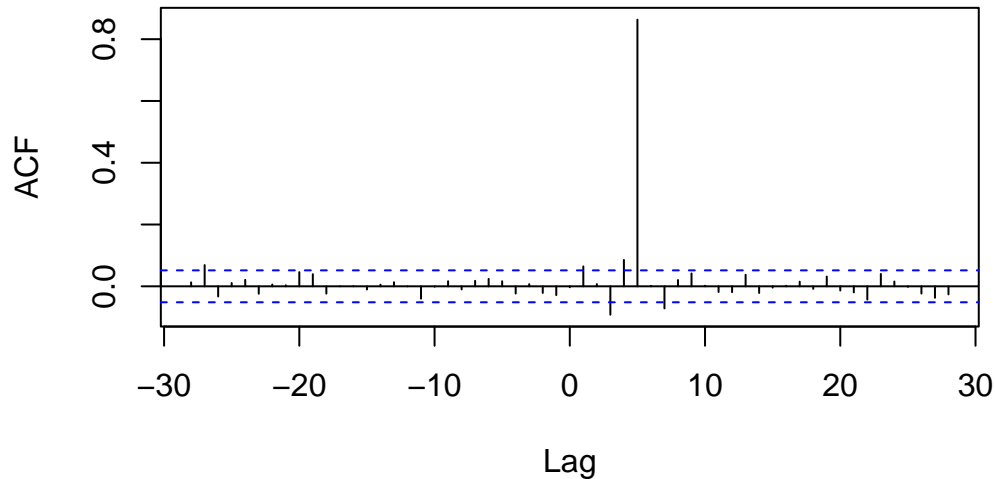
The computed correlation between the negative log returns of Bitcoin and Ethereum using the `cor.test()` function is -0.0031 and is not statistically significant (p-value = 0.905). However, the lack of significance in the correlation test does not necessarily imply independence, as there may be a non-linear relationship or other factors influencing the association between the two variables.

b) Analyze Bitcoin-ETH cross-correlation function

Question

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

is-Correlation Function: Ethereum vs. Bitcoin Negative Log



We can clearly observe a significant spike at lag 5 indicates that there is a notable correlation between Bitcoin's returns and Ethereum's returns when Ethereum's returns are shifted by 5 minutes. Ethereum's returns are correlated with past values of Bitcoin's returns and thus may predict Ethereum's returns with a 5 minutes delay. The spike indicates the strength of the correlation. In our case, the spike at lag 5 crosses the significance threshold (horizontal dashed lines). We can say with confidence that it is statistically significant and not likely due to random noise. However, it is important to keep in mind that cross-correlation does not imply causation. Other factors or indirect relationships might also explain this pattern.

c) Assess predictive power with Granger test

Question

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?

We performed here two grander test using the `grandertest()`function. The first tests if Ethereum's past values do not Granger-cause Bitcoin's returns. With a p-value of 0.7132, we cannot reject the null hypothesis. Therefore, there is no evidence to suggest that Ethereum's past values have predictive power for Bitcoin's returns.

The second test examines if Bitcoin's past values Granger-cause Ethereum's returns. With a p-value of 0, Bitcoin's past values significantly Granger-cause Ethereum's returns, meaning there is strong evidence of predictive power.

d) Predict mutual impacts of drops

Question

Based on your answer in (c), answer the following questions:

d.1) Predict ETH reaction to Bitcoin drop

Question

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

Given our results in [question c](#)), the sudden drop in Bitcoin stocks should have a significant impact on the Ethereum stocks price. Given our analysis, the Ethereum stocks should drop 5 minutes (lag of 5 / [question b](#))) after the Bitcoin stocks drop with a value close to 80% of the Bitcoin drop.

d.2) Predict Bitcoin reaction to ETH drop

Question

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

Based on the Granger causality test results from [question c](#)), Ethereum's past values do not Granger-cause Bitcoin's returns, suggesting no significant predictive relationship. If Ethereum experiences an extreme sudden drop, it is less likely that Bitcoin will exhibit a direct or immediate reaction based on historical relationships. However, Bitcoin and Ethereum share common market factors and investor sentiment, so an indirect reaction (e.g., due to market-wide panic) is still possible, though it is not strongly supported by the causality analysis.

Practical 2: Weather

Part 1: Block maxima approach

a) Best fit distribution

Question

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

b) Best fit distribution on yearly maxima

Question

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

c) Linear model approach

Question

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?

d) GEV distribution

Question

Fit a GEV with constant parameters to the historical yearly max values. We recommend using `fevd` function in `extRemes` library or `gev.fit` function in `ismev` library. Fit a second GEV model with time varying location parameter. Compare the two models using AIC or BIC. Which one do you recommend using?

e) Diagnostic plots

Question

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

f) Return levels prediction

Question

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.

g) Interpretation

Question

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.

h) Return period 100 mm precipitation

Question

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

i) Probability of exceeding 150 mm

Question

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

Part 2: Peaks-over-threshold approach

a) Time series plot

Question

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

b) Mean Residual Life Plot

Question

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.

c) Fit a GPD

Question

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)

d) Return levels prediction

Question

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

e) Return period 100 mm precipitation

Question

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

f) Probability of exceeding 150 mm

Question

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

g) Comparison with block maxima

Question

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?

Part 2

Part 3

Practical 3: Heat wave in Switzerland

Part 3: Comparing GEV and POT approaches

Data Loading and Preparation

```
# load the required packages and install them if they are not.
source(here::here("code", "setup.R"))

# Load the data
data_temp <- read.csv(here("data", "Cleaned_Stations_Data.csv"))

# Filter data for all cities and remove any rows with NA in TMAX
# Geneva
data_geneva <- data_temp %>%
  filter(!is.na(TMAX)) %>%
  filter(NAME == "Genève") %>%
  dplyr::select(DATE, TMAX) %>%
  mutate(DATE = as.Date(DATE))

# Santis
data_santis <- data_temp %>%
  filter(!is.na(TMAX)) %>%
  filter(NAME == "Saentis") %>%
  dplyr::select(DATE, TMAX) %>%
  mutate(DATE = as.Date(DATE))

# Lugano
data_lugano <- data_temp %>%
  filter(!is.na(TMAX)) %>%
  filter(NAME == "Lugano") %>%
  dplyr::select(DATE, TMAX) %>%
  mutate(DATE = as.Date(DATE))
```

We ensure that we have the necessary data for the analysis by filtering the temperature data for all three cities and removing any rows with missing values. We removed 395 rows.

Block Maxima Approach (GEV Distribution)

Question

Can a Generalized Extreme Value (GEV) distribution accurately model annual maximum temperatures in Switzerland?

In this analysis, we investigate whether a Generalized Extreme Value (GEV) distribution can accurately model annual maximum temperatures in Switzerland. Additionally, we compare the results of the block Maxima approach with the Peaks-over-Threshold (POT) approach using declustering for extreme temperatures.

We calculate the annual maximum temperatures for Geneva, Santis and Lugano and fit a GEV distribution to these maxima.

```
# Calculate annual maximum temperatures
# Geneva
```

```
annual_maxima_geneva <- data_geneva %>%
  mutate(Year = year(DATE)) %>%
  group_by(Year) %>%
  summarize(MaxTemp = max(TMAX), .groups = 'drop')

# Santis
annual_maxima_santis <- data_santis %>%
  mutate(Year = year(DATE)) %>%
  group_by(Year) %>%
  summarize(MaxTemp = max(TMAX), .groups = 'drop')

# Lugano
annual_maxima_Lugano <- data_lugano %>%
  mutate(Year = year(DATE)) %>%
  group_by(Year) %>%
  summarize(MaxTemp = max(TMAX), .groups = 'drop')

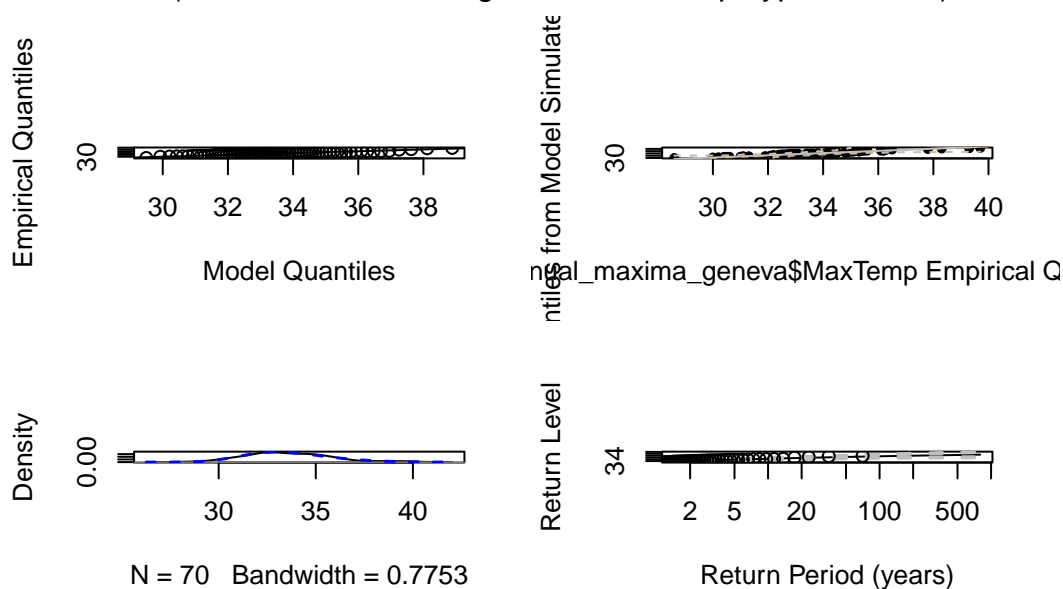
# Fit a GEV distribution to the annual maxima
# Genève
gev_fit_geneva <- fevd(annual_maxima_geneva$MaxTemp, type = "GEV")

# Santis
gev_fit_santis <- fevd(annual_maxima_santis$MaxTemp, type = "GEV")

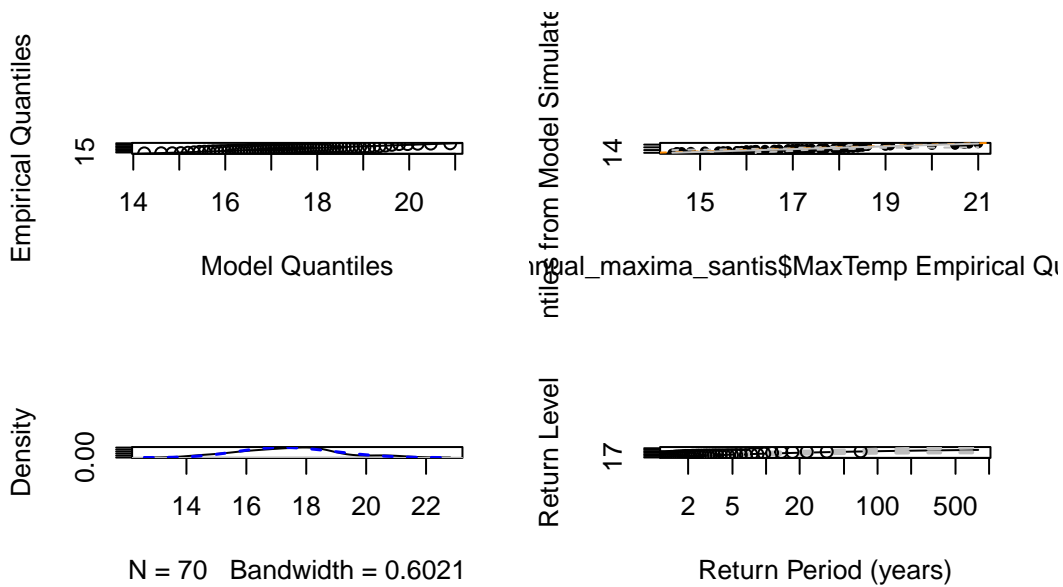
# Lugano
gev_fit_Lugano <- fevd(annual_maxima_Lugano$MaxTemp, type = "GEV")

# Diagnostic plots for the GEV fit
# Geneva
par(mfrow = c(2, 2))
plot(gev_fit_geneva)
```

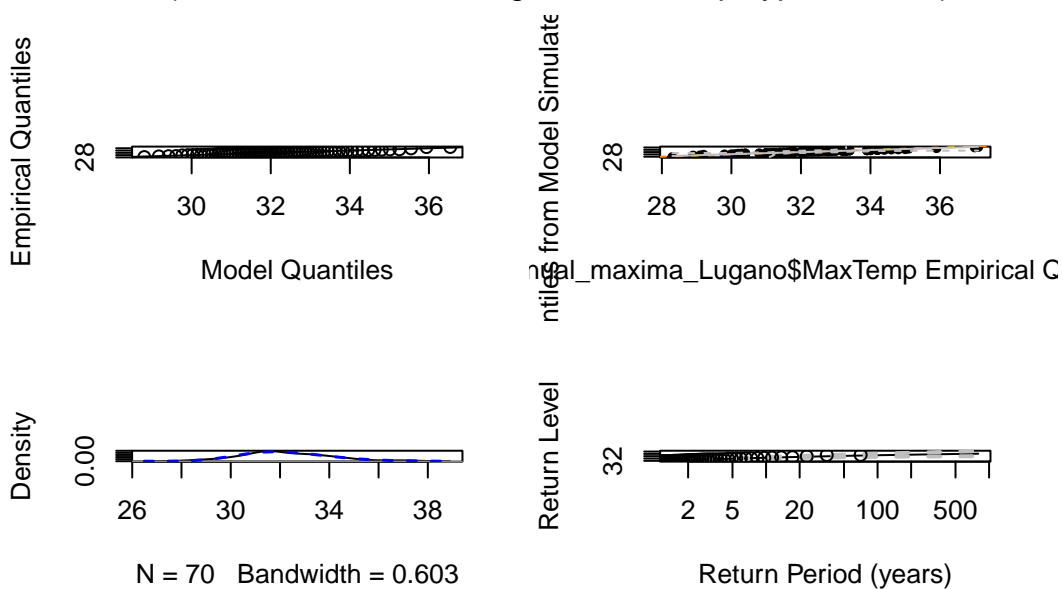
```
fevd(x = annual_maxima_geneva$MaxTemp, type = "GEV")
```




```
fevd(x = annual_maxima_santis$MaxTemp, type = "GEV")
```



```
fevd(x = annual_maxima_Lugano$MaxTemp, type = "GEV")
```



We have here the diagnostic plots for the GEV fit, which include the quantile-quantile plot, the return level plot, the probability plot, and the density plot. We see that overall the fit is good.

```
# Calculate return levels for 10, 50, and 100-year return periods
# Genève
gev_return_levels_geneva <- return.level(gev_fit_geneva, return.period = c(10, 50, 100))

# Santis
gev_return_levels_santis <- return.level(gev_fit_santis, return.period = c(10, 50, 100))

# Lugano
gev_return_levels_Lugano <- return.level(gev_fit_Lugano, return.period = c(10, 50, 100))
```

Comparison with Peaks-over-Threshold Approach (GPD Distribution)

Question

How do the results of the block maxima approach compare with the Peaks-over-Threshold (POT) approach for temperature extremes?

Peaks-over-Threshold Approach

```
# Filter the data for the summer months (June to September)
# Geneva
geneva_summer <- subset(data_geneva, format(DATE, "%m") %in% c("06", "07", "08", "09"))

# Santis
santis_summer <- subset(data_santis, format(DATE, "%m") %in% c("06", "07", "08", "09"))

# Lugano
lugano_summer <- subset(data_lugano, format(DATE, "%m") %in% c("06", "07", "08", "09"))

# Define a threshold at the 95th percentile
# Geneva
threshold_geneva <- quantile(geneva_summer$TMAX, 0.95)

# Santis
threshold_santis <- quantile(santis_summer$TMAX, 0.95)

# Lugano
threshold_lugano <- quantile(lugano_summer$TMAX, 0.95)

# Number of exceedances over the threshold
# Geneva
num_exceedances_geneva <- sum(geneva_summer$TMAX > threshold_geneva)

# Santis
num_exceedances_santis <- sum(santis_summer$TMAX > threshold_santis)

# Lugano
```

```
num_exceedances_Lugano <- sum(lugano_summer$TMAX > threshold_lugano)
```

We now apply the Peaks-over-Threshold (POT) approach using a suitable threshold. With a 95% threshold, we have 413 exceedances over the threshold for Genève, 413 for Santis, and 406 for Lugano.

```
# Decluster the data using the chosen threshold for each station
# Geneva
geneva_declustered <- extRemes::decluster(geneva_summer$TMAX, threshold = threshold_geneva, run.1

# Santis
santis_declustered <- extRemes::decluster(santis_summer$TMAX, threshold = threshold_santis, run.1

# Lugano
lugano_declustered <- extRemes::decluster(lugano_summer$TMAX, threshold = threshold_lugano, run.1

# Add the declustered data to the corresponding datasets
# Geneva
geneva_summer$declustered <- ifelse(geneva_summer$TMAX >= threshold_geneva, geneva_declustered, N

# Santis
santis_summer$declustered <- ifelse(santis_summer$TMAX >= threshold_santis, santis_declustered, N

# Lugano
lugano_summer$declustered <- ifelse(lugano_summer$TMAX >= threshold_lugano, lugano_declustered, N

# Extract declustered extreme values
# Geneva
extreme_values_geneva <- geneva_summer[["declustered"]][!is.na(geneva_summer[["declustered"]])]

# Santis
extreme_values_santis <- santis_summer[["declustered"]][!is.na(santis_summer[["declustered"]])]

# Lugano
extreme_values_lugano <- lugano_summer[["declustered"]][!is.na(lugano_summer[["declustered"]])]

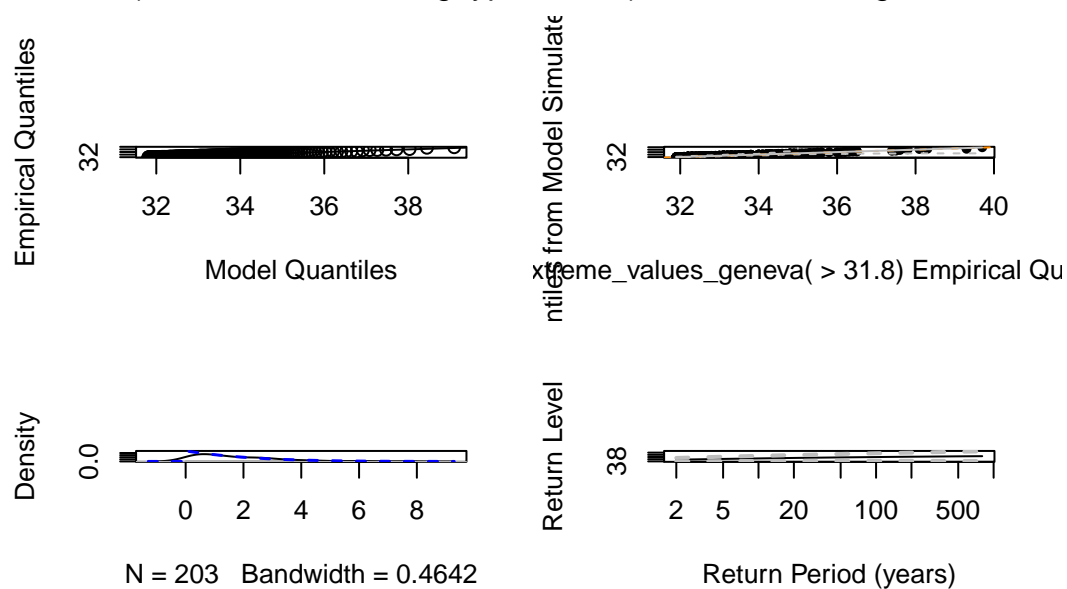
# Fit a GPD to the exceedances
# Genève
gpd_fit_geneva <- fevd(extreme_values_geneva, threshold = threshold_geneva, type = "GP")

# Santis
gpd_fit_santis <- fevd(extreme_values_santis, threshold = threshold_santis, type = "GP")

# Lugano
gpd_fit_lugano <- fevd(extreme_values_lugano, threshold = threshold_lugano, type = "GP")

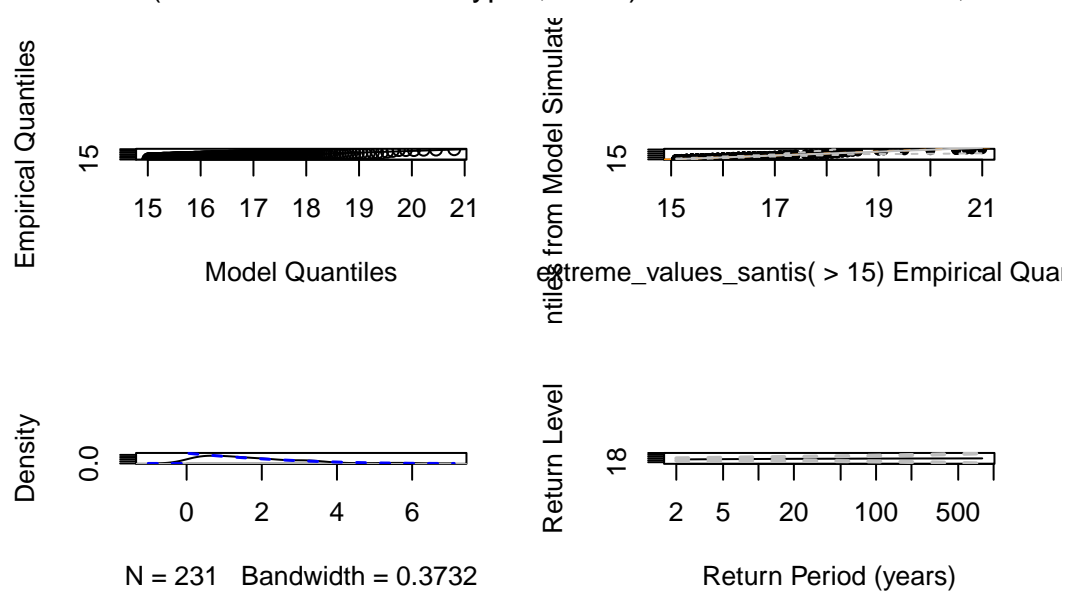
# Diagnostic plots for the GPD fit
# Genève
par(mfrow = c(2, 2))
plot(gpd_fit_geneva)
```

```
fevd(x = extreme_values_geneva, type = "Q", hold = threshold_geneva,
```

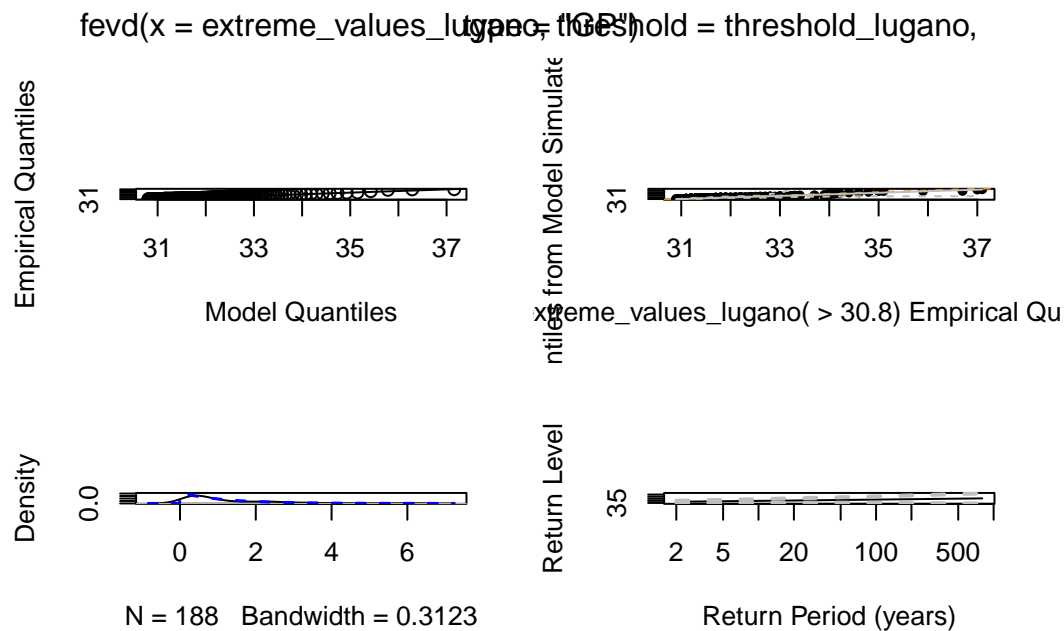


```
# Santis
par(mfrow = c(2, 2))
plot(gpd_fit_santis)
```

```
fevd(x = extreme_values_santis, type = "Q", hold = threshold_santis,
```



```
# Lugano
par(mfrow = c(2, 2))
plot(gpd_fit_lugano)
```



Again, we have the diagnostic plots for the GPD fit, which include the quantile-quantile plot, the return level plot, the probability plot, and the density plot. We see that overall the fit is good.

```
# Calculate return levels for 10, 50, and 100-year return periods
# Geneva
gpd_return_levels_geneva <- return.level(gpd_fit_geneva, return.period = c(10, 50, 100))

# Santis
gpd_return_levels_santis <- return.level(gpd_fit_santis, return.period = c(10, 50, 100))

# Lugano
gpd_return_levels_lugano <- return.level(gpd_fit_lugano, return.period = c(10, 50, 100))
```

Our results

Block Maxima Approach (GEV Distribution):

Parameter Estimates:

Station	Elevation	location parameter	scale parameter	shape parameter
Geneva	375m	32.69	1.97	-0.15
Santis	2500m	16.84	1.5	-0.23
Lugano	275m	31.51	1.65	-0.17

GEV Return Levels:

Station	Elevation	10-year RL (°C)	50-year RL (°C)	100-year RL (°C)
Geneva	375m	36.45	38.51	39.23
Santis	2500m	19.47	20.69	21.08

Station	Elevation	10-year RL (°C)	50-year RL (°C)	100-year RL (°C)
Lugano	275m	34.61	36.24	36.81

Peaks-over-Threshold Approach (GPD Distribution):

Threshold Selection:

The threshold was set at the 95th percentile, which is 31.8 degrees Celsius for Geneva

The threshold was set at the 95th percentile, which is 15 degrees Celsius for Santis.

The threshold was set at the 95th percentile, which is 30.8 degrees Celsius for Lugano.

Number of Exceedances:

There are 413 exceedances over the threshold for Geneva.

There are 413 exceedances over the threshold for Santis.

There are 406 exceedances over the threshold for Lugano.

GPD Return Levels:

The return levels are:

Station	Elevation	10-year RL (°C)	50-year RL (°C)	100-year RL (°C)
Geneva	375m	40.72	41.63	41.96
Santis	2500m	21.52	21.84	21.94
Lugano	275m	39.88	42.01	42.95

Comparison of Approaches:

In comparing the block maxima approach (GEV) to the Peaks-over-Threshold approach (GPD) for modeling temperature extremes, we find that the POT method often provides more nuanced information about the tail behavior of the distribution. While the GEV model uses only the annual maximum values, which can lead to relatively large uncertainty due to a limited amount of extreme data, the POT approach utilizes all temperature values above a chosen high threshold. This generally results in more data points to characterize the tail, potentially yielding more stable and reliable estimates of parameters and return levels. However, the POT approach depends on careful threshold selection and declustering, which introduces additional steps not required by the simpler block maxima method. In practice, the POT approach can lead to different (often more precise) return level estimates, giving a potentially more accurate picture of the frequency and magnitude of extreme temperature events.

Part 4: Return Period & Probabilities of New Record High Temperatures

In this part, we will discuss the return periods & probability analysis of having a new record of heatwaves in the cities of Geneva, Säntis and Lugano. As seen when comparing the Block Maxima and the Peaks-over-Threshold approaches, the Block Maxima one seems to be more accurate to use. Thus, we will only use this approach when calculating the return period and the probabilities of obtaining a new record of temperature for 1, 3 and 10 years. To begin with, we will take the code from part 3 to obtain the GEV distribution for all cities.

Block Maxima Approach (GEV Distribution)

See part 3 for the code.

Peaks-over-Threshold Approach (GPD Distribution)

See part 3 for the code.

Question

Can a Generalized Extreme Value (GEV) distribution accurately model annual maximum temperatures in Switzerland?

Return Period

Now, we will analyse the return period for each city. The return period is the expected time (here, in years) between occurrences of an extreme event to be at least as extreme as a given value.

```
# Load required library
library(extRemes)

# Extract GEV parameters for Geneva
location_geneva <- gev_fit_geneva$results$par["location"]
scale_geneva <- gev_fit_geneva$results$par["scale"]
shape_geneva <- gev_fit_geneva$results$par["shape"]

# Extract GEV parameters for Säntis
location_santis <- gev_fit_santis$results$par["location"]
scale_santis <- gev_fit_santis$results$par["scale"]
shape_santis <- gev_fit_santis$results$par["shape"]

# Extract GEV parameters for Lugano
location_lugano <- gev_fit_Lugano$results$par["location"]
scale_lugano <- gev_fit_Lugano$results$par["scale"]
shape_lugano <- gev_fit_Lugano$results$par["shape"]

# Set the temperature threshold
temp_threshold <- 35
temp_threshold_santis <- 19

# Function to calculate return period
```

```

calculate_return_period <- function(location, scale, shape, temp_threshold) {
  cdf_value <- pevd(temp_threshold, loc = location, scale = scale, shape = shape, type = "GEV")
  return_period <- 1 / (1 - cdf_value)
  return(return_period)
}

# Calculate return periods
return_period_geneva <- calculate_return_period(location_geneva, scale_geneva, shape_geneva, temp_threshold)
return_period_santis <- calculate_return_period(location_santis, scale_santis, shape_santis, temp_threshold)
return_period_lugano <- calculate_return_period(location_lugano, scale_lugano, shape_lugano, temp_threshold)

# Print the results
cat("The return period for 35°C in Geneva is approximately", round(return_period_geneva, 2), "years\n")

```

The return period for 35°C in Geneva is approximately 4.16 years.

```

cat("The return period for 35°C in Sântis is approximately", round(return_period_santis, 2), "years\n")

```

The return period for 35°C in Sântis is approximately 6.29 years.

```

cat("The return period for 35°C in Lugano is approximately", round(return_period_lugano, 2), "years\n")

```

The return period for 35°C in Lugano is approximately 14.04 years.

As a result, we can observe that the return period is around 4.16 years in Geneva, Inf for Sântis and 14.04 years for Lugano.

Probability of New Record High Temperature

Using the previously fitted model

```

# Historical record highs for each city (replace with actual values)
record_high_geneva <- max(annual_maxima_geneva$MaxTemp)
record_high_santis <- max(annual_maxima_santis$MaxTemp)
record_high_lugano <- max(annual_maxima_lugano$MaxTemp)

# Function to calculate the probability of a new record high
calculate_new_record_probability <- function(location, scale, shape, record_high, years) {
  # Probability of not exceeding the record high in 1 year
  prob_not_exceed <- pevd(record_high, loc = location, scale = scale, shape = shape, type = "GEV")
  # Probability of exceeding the record high in the given time horizon
  prob_exceed <- 1 - (prob_not_exceed^years)
  return(prob_exceed)
}

# Calculate probabilities for Geneva

```



```

prob_new_record_geneva_1yr <- calculate_new_record_probability(location_geneva, scale_geneva, sha
prob_new_record_geneva_3yr <- calculate_new_record_probability(location_geneva, scale_geneva, sha
prob_new_record_geneva_10yr <- calculate_new_record_probability(location_geneva, scale_geneva, sh

# Calculate probabilities for Sântis
prob_new_record_santis_1yr <- calculate_new_record_probability(location_santis, scale_santis, sha
prob_new_record_santis_3yr <- calculate_new_record_probability(location_santis, scale_santis, sha
prob_new_record_santis_10yr <- calculate_new_record_probability(location_santis, scale_santis, sh

# Calculate probabilities for Lugano
prob_new_record_lugano_1yr <- calculate_new_record_probability(location_lugano, scale_lugano, sha
prob_new_record_lugano_3yr <- calculate_new_record_probability(location_lugano, scale_lugano, sha
prob_new_record_lugano_10yr <- calculate_new_record_probability(location_lugano, scale_lugano, sh

# Print the results
cat("Probability of a new record high in Geneva:\n",
    "1 year:", round(prob_new_record_geneva_1yr, 4), "\n",
    "3 years:", round(prob_new_record_geneva_3yr, 4), "\n",
    "10 years:", round(prob_new_record_geneva_10yr, 4), "\n\n")

```

Probability of a new record high in Geneva:

```

1 year: 0.0061
3 years: 0.0183
10 years: 0.0596

```

```

cat("Probability of a new record high in Sântis:\n",
    "1 year:", round(prob_new_record_santis_1yr, 4), "\n",
    "3 years:", round(prob_new_record_santis_3yr, 4), "\n",
    "10 years:", round(prob_new_record_santis_10yr, 4), "\n\n")

```

Probability of a new record high in Sântis:

```

1 year: 0.0116
3 years: 0.0343
10 years: 0.1098

```

```

cat("Probability of a new record high in Lugano:\n",
    "1 year:", round(prob_new_record_lugano_1yr, 4), "\n",
    "3 years:", round(prob_new_record_lugano_3yr, 4), "\n",
    "10 years:", round(prob_new_record_lugano_10yr, 4), "\n\n")

```

Probability of a new record high in Lugano:

```

1 year: 0.0068
3 years: 0.0201
10 years: 0.0655

```

Results

As a result, we can observe that ...