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

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

December 16, 2024

Table of contents

Introduction

This documents 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 or .zip file.

Practical 1: Financial returns and normality

Part 1: Financial returns and normality

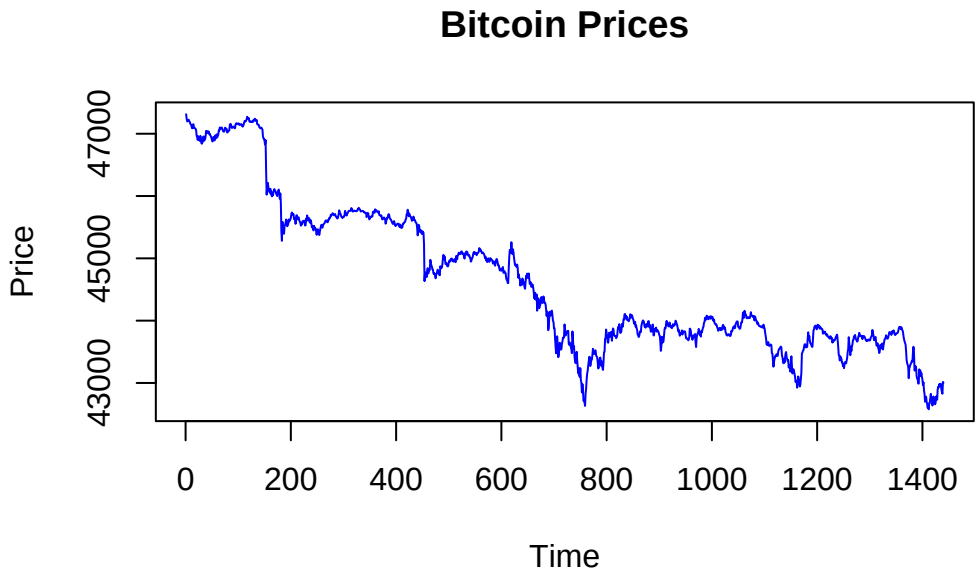
The working directory is set to: /Users/lodrik/Documents/GitHub/RA_Praticals

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")
```



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.

```r
## 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

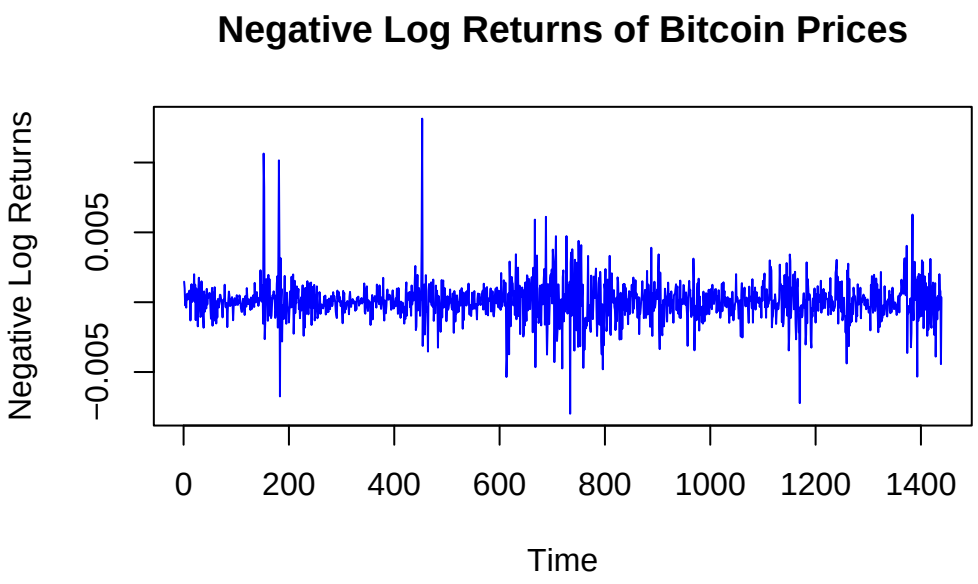
> Question
>
> Create a function to transform the Bitcoin prices into their negative log returns counterparts. Plot the latter series and assess their stationarity. To compare the series, also plot the negative log returns on a common scale.

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.

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

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

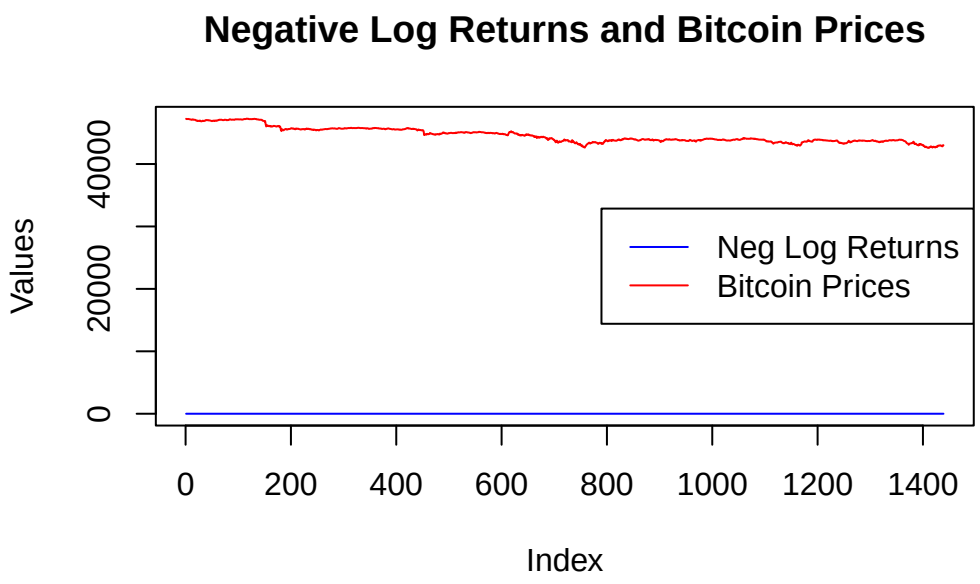
We can now plot the negative log returns series and the raw Bitcoin prices to compare.

```r
## 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")
```



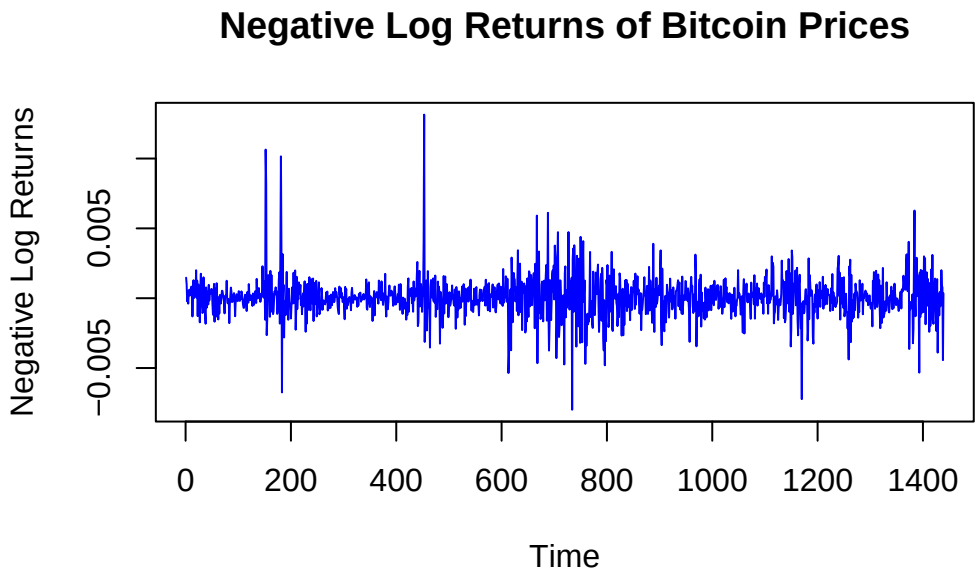
If we scale bitcoin prices and negative log returns, we can compare both time series on a plot with a common scale.

```
# 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)
```

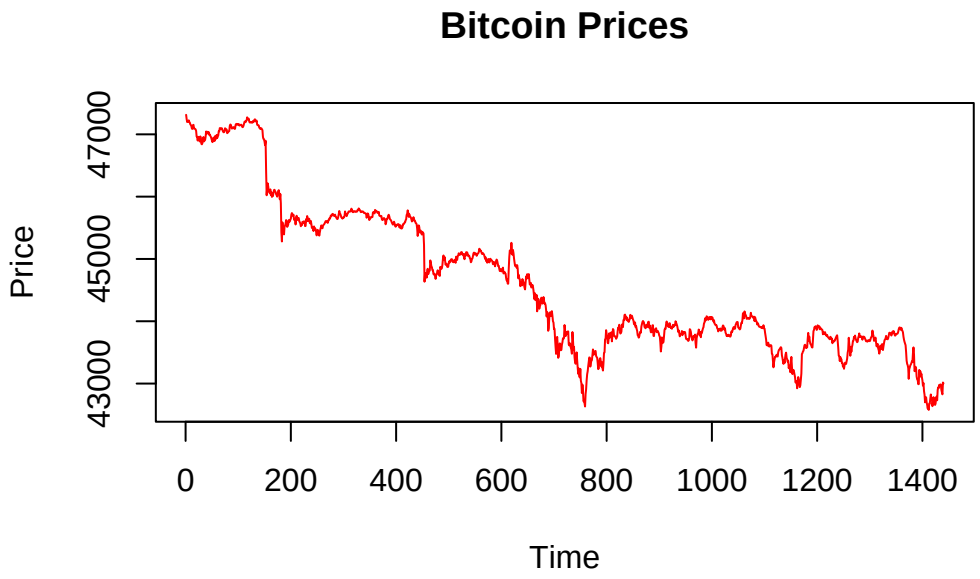



```r
# Reset the plotting area to default settings (optional, for future plots)
par(mfrow=c(1,1))
```

```r
# 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")
```




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



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 tes
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
```

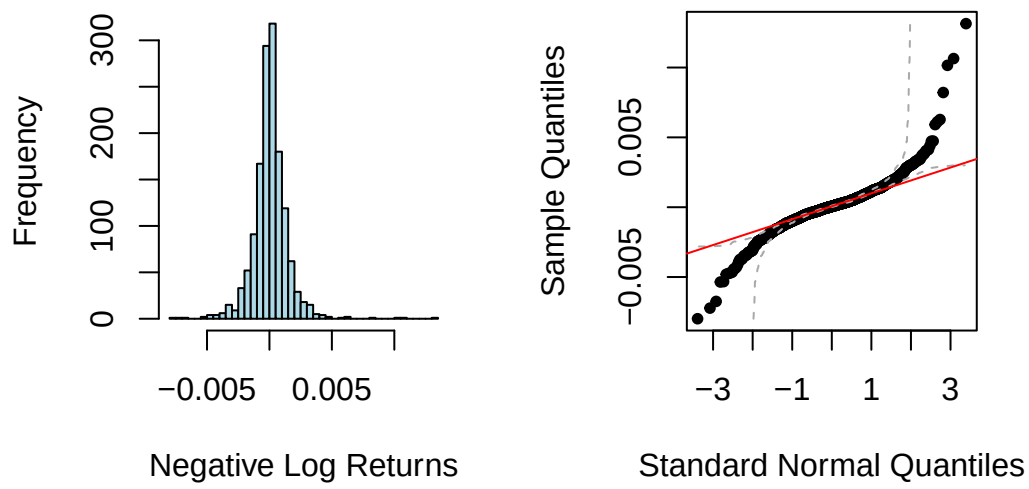
Since the p-value is significantly smaller than 0.05, we can reject the null hypothesis and conclude that the negative log returns series is stationary.

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")
```




```
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.

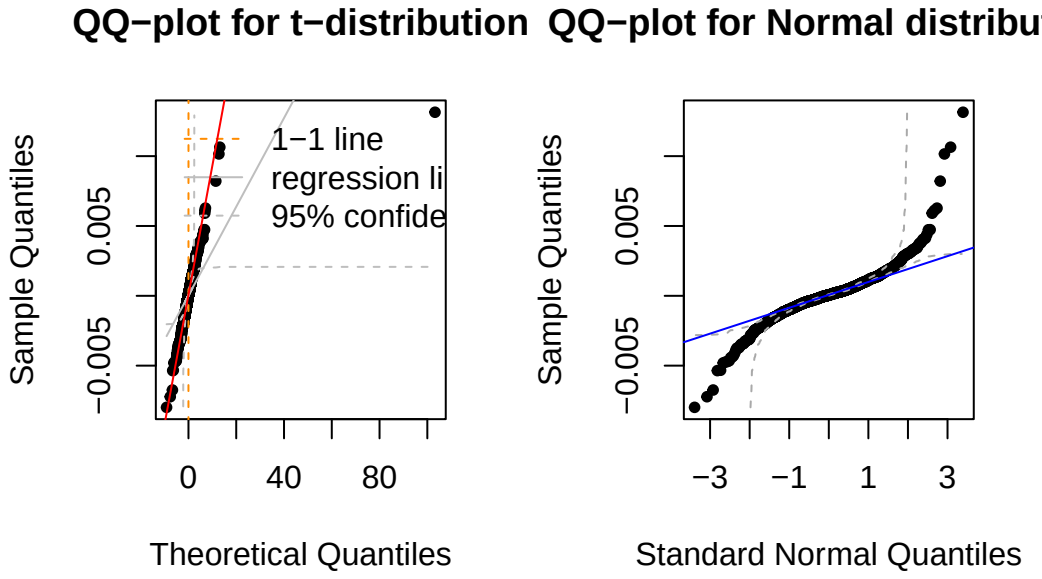
```
## Step 1: Fit a t-distribution to the negative log returns
fit_t <- fitdistr(scaling_factor * neg_log_returns_bitcoin, "t") # Multiply by 100000 to avoid nu
```

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")
```




```
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

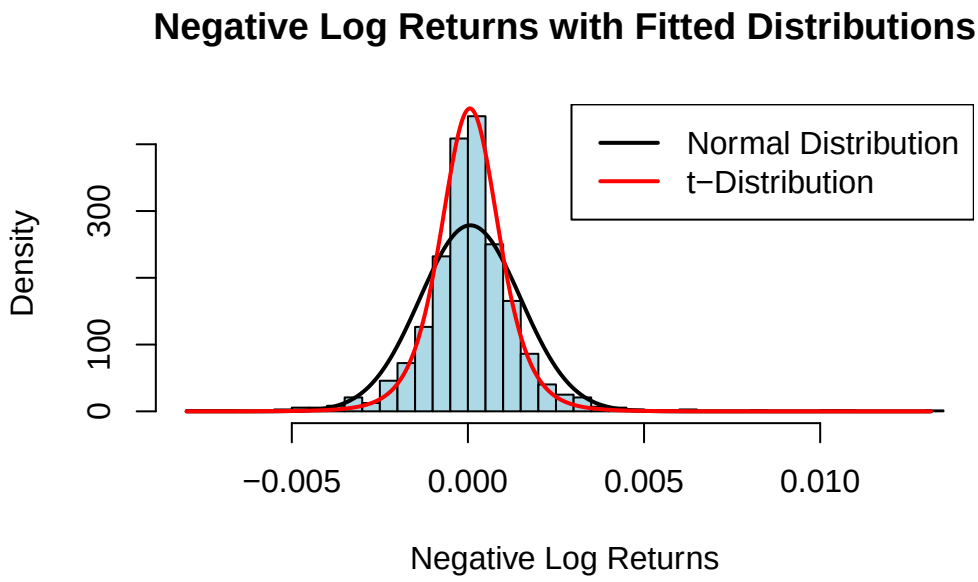
Question

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)
```



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 2: Financial time series heteroskedasticity and the random walk hypothesis

Another crucial hypothesis in asset pricing is the so-called homoscedasticity, i.e. constant variance of theresiduals. We would also like to check this assumption. We use the same Bitcoin data as in Part 1.

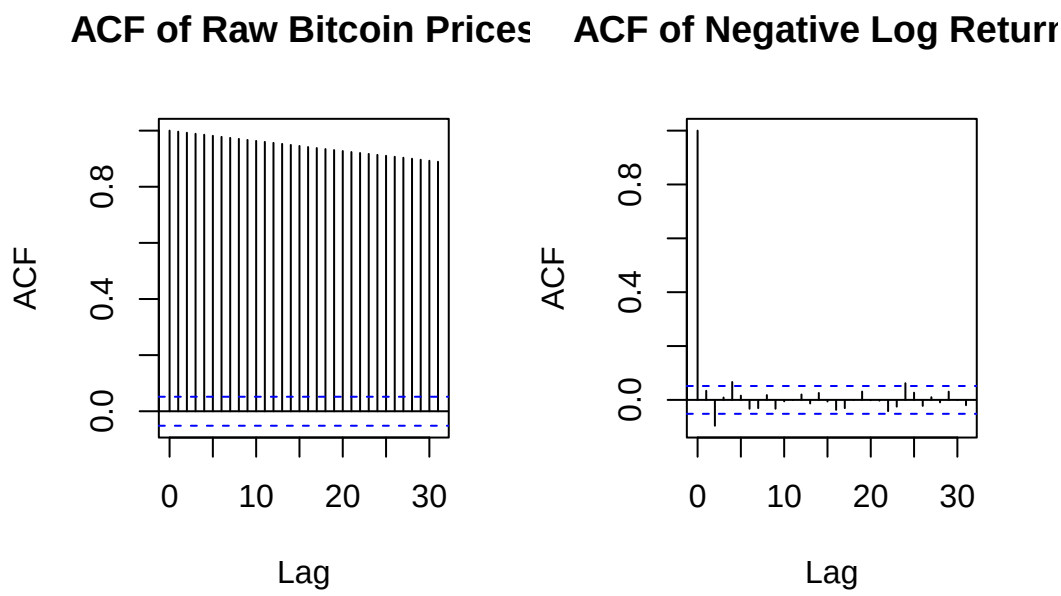
```
# load the required packages and install them if they are not.
source(here::here("code","setup.R"))

# getiing the working directory
wd <- here::here()

# Loading the data
crypto_data <- read.csv(here("data", "crypto_data.csv"))

# Extract the Bitcoin prices
bitcoin_prices <- crypto_data$Bitcoin

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

# Use the fucntion on Bitcoin prices
neg_log_returns_bitcoin <- negative_log_returns(bitcoin_prices)
```

a) ACF & negative log returns

Question

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

```
# Set the plotting area to have 1 row and 2 columns
par(mfrow = c(1, 2))

# Tracer l'ACF de la série brute des prix du Bitcoin
acf(bitcoin_prices, main = "ACF of Raw Bitcoin Prices")

# Tracer l'ACF des rendements logarithmiques négatifs (bitcoin_log_returns)
acf(neg_log_returns_bitcoin, main = "ACF of Negative Log Returns")
```



1. ACF or Raw Bitcoin Prices: The ACF plot for the Raw Bictoin Prices shows strong autocorrelation. The values are strongly correlated with their past values. This indicates that the raw series is non-stationary and has a long-term dependency.
2. ACF of Negative Log Returns: The ACF of Negative Log Returns shows that most of the correlations at higher lags fall withing the confidence interval. It implies that the negative log returns are more likely to be stationary and have less long-term dependence.

Conclusion: The Negative Log Returns are likely easier to model due to their more stationary nature and lack of significant autocorrelation.

b) Ljung-Box procedure

Question

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

```
# Apply Ljung-Box test on raw Bitcoin prices
ljung_box_raw <- Box.test(bitcoin_prices, lag = 20, type = "Ljung-Box")

# Apply Ljung-Box test on negative log returns
ljung_box_returns <- Box.test(neg_log_returns_bitcoin, lag = 20, type = "Ljung-Box")

# Print results
print(ljung_box_raw)
```

```
    Box-Ljung test

data:  bitcoin_prices
X-squared = 26873, df = 20, p-value < 2.2e-16
```

```r
print(ljung_box_returns)
```

```
    Box-Ljung test

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

The Ljung-Box test checks for serial dependence (autocorrelation) in the series. If the p-value is small (typically < 0.05), it suggests that there is serial dependence, meaning the series is not independent over time.

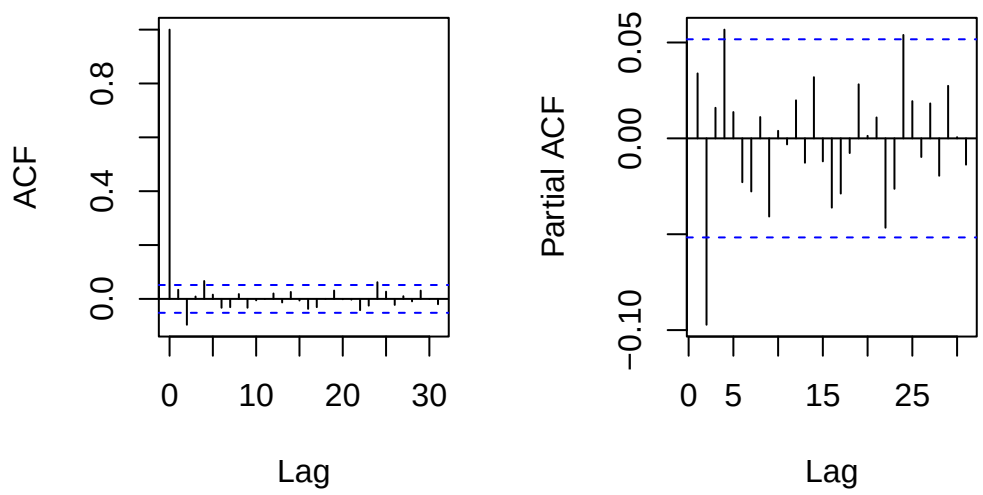
For the raw series: Since price data tends to show trends, we often expect serial dependence. For the negative log returns: These are typically expected to be more random (i.e., closer to white noise), so the test might indicate less serial dependence.

Based on the results of the Ljung-Box tests: For the raw Bitcoin prices: - Raw Bictoin Prices: p-value $<$ 2.2e-16, the p-value is extremely small, which mean that we reject the null hypothesis of no autocorrelation in the raw Bictoin prices. The values are highly dependent on previous values, it confirms that the series is non-stationary. - Negative Log Returns: p-value $= 0.03082$, the p-value is also small, but higher than the raw prices. It indicates that there is still some autocorrelation in the series, although it is less pronounced compared to the raw Bitcoin prices. Ideally, negative log returns should behave more likke white noise, meaning no serial dependence

c) ARIMA models for the negative log returns series

> Question
>
> Propose ARIMA models for the negative log returns series, based on visualization tools (e.g. ACF, PACF). Select an ARIMA model using auto.arima() (forecast package) for the negative log returns series. Comment on the difference. Assess the residuals of the resulting models.

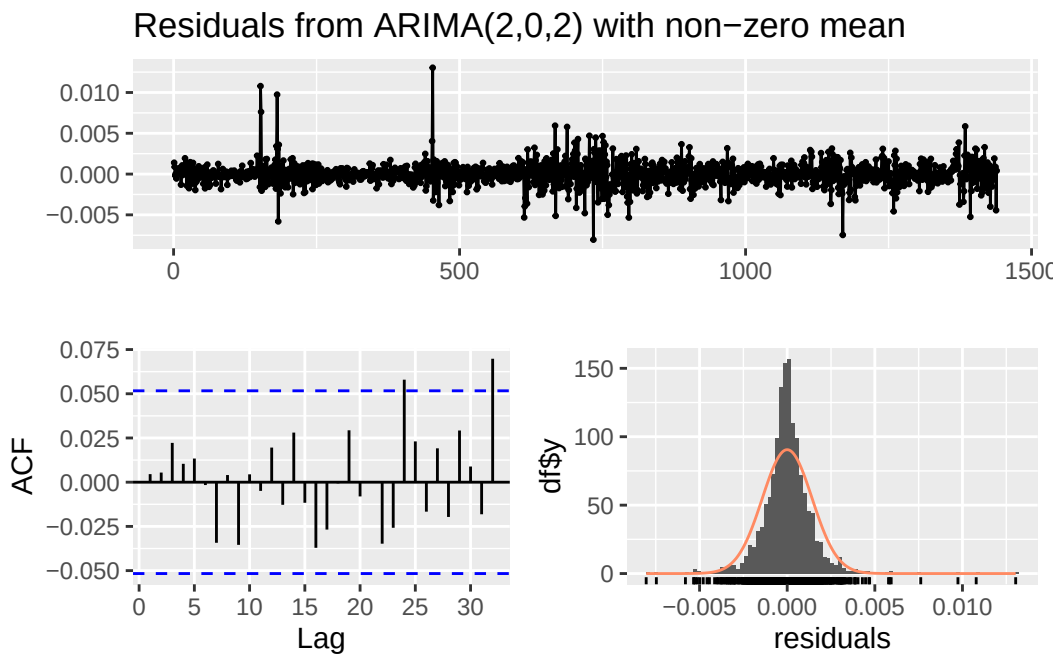
```r
# Set the plotting area to have 1 row and 2 columns
par(mfrow = c(1, 2))

# Step 1: Visualize ACF and PACF for negative log returns
acf(neg_log_returns_bitcoin, main = "ACF of Negative Log Returns")
pacf(neg_log_returns_bitcoin, main = "PACF of Negative Log Returns")
```




```r
# Step 2: Use auto.arima() to find the best ARIMA model for negative log returns
auto_arima_model <- auto.arima(neg_log_returns_bitcoin)
summary(auto_arima_model)
```

```
Series: neg_log_returns_bitcoin
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 -1.965777e-07 0.001421946 0.0009423239 100.013 131.3896 0.7133069
                    ACF1
Training set 0.00455059
```

```r
# Step 3: Plot residuals of the ARIMA model to assess the goodness of fit
checkresiduals(auto_arima_model)
```




```
    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
```

```r
# Additional: Ljung-Box test on residuals to check if they are white noise
Box.test(residuals(auto_arima_model), type="Ljung-Box")
```

```
    Box-Ljung test

data:  residuals(auto_arima_model)
X-squared = 0.029861, df = 1, p-value = 0.8628
```

The results from the ARIMA model fitting for the negative log returns of Bitcoin and the residual analysis suggest the following:

ARIMA Model: The selected ARIMA model is ARIMA(2,0,2), meaning:

AR(2): Two autoregressive terms are included. MA(2): Two moving average terms are included. d = 0: No differencing was applied, indicating that the series is already stationary (which aligns with the fact that negative log returns tend to be stationary). Coefficients:

The AR1 and AR2 coefficients are -0.0520 and -0.5415, respectively. The MA1 and MA2 coefficients are 0.0853 and 0.4479, respectively. The mean of the series is very close to zero (1e-04). Error metrics:

RMSE (Root Mean Square Error): 0.00142, which is relatively low, indicating that the model fits the data well. MAE (Mean Absolute Error): 0.000942, which is also quite low. ACF1 of residuals: 0.00455, suggesting that the residuals do not exhibit significant autocorrelation. Ljung-Box Test: The Ljung-Box test on residuals gives a p-value of 0.8628, which is much larger than 0.05. This indicates that there is no significant autocorrelation left in the residuals, implying that the model fits the data well. Conclusion: The ARIMA(2,0,2) model selected

by auto.arima() seems to be a good fit for the negative log returns of Bitcoin, as evidenced by the low RMSE and MAE, as well as the results of the Ljung-Box test. The residuals behave like white noise, meaning that the model has successfully captured the patterns in the data. There is no significant temporal dependence left in the residuals, which supports the adequacy of this ARIMA model for the series. Overall, the ARIMA model chosen by auto.arima() fits the data well and leaves no significant autocorrelation in the residuals.

d) GARCH models

Question

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.

```
# Fit GARCH(1,1) model with normal distribution
garch_normal <- garchFit(~ garch(1, 1), data = neg_log_returns_bitcoin, cond.dist = "norm")

# Summary of the model
summary(garch_normal)

# Fit GARCH(1,1) model with standardized t-distribution
garch_t <- garchFit(~ garch(1, 1), data = neg_log_returns_bitcoin, cond.dist = "std")

# Summary of the model
summary(garch_t)

# Residuals from the normal GARCH model
residuals_normal <- residuals(garch_normal)

# Residuals from the t-distribution GARCH model
residuals_t <- residuals(garch_t)

# Plot residuals for the normal GARCH model
plot(residuals_normal, main = "Residuals of GARCH(1,1) with Normal Distribution", type = "l")
```

```
# Plot residuals for the t-distribution GARCH model
plot(residuals_t, main = "Residuals of GARCH(1,1) with t-Distribution", type = "l")
```

```
# ACF of residuals for the normal GARCH model
acf(residuals_normal, main = "ACF of Residuals (Normal GARCH Model)")
```

```
# ACF of residuals for the t-distribution GARCH model
acf(residuals_t, main = "ACF of Residuals (t-Distribution GARCH Model)")
```

```
# Ljung-Box test for normal GARCH model residuals
Box.test(residuals_normal, lag = 20, type = "Ljung-Box")

# Ljung-Box test for t-distribution GARCH model residuals
```

```r
Box.test(residuals_t, lag = 20, type = "Ljung-Box")
```

The results for fitting GARCH(1,1) models with both normal and standardized t-distributions to the negative log returns are as follows:

GARCH(1,1) with Normal Distribution:

The log-likelihood value is 7632.108. The coefficients for the GARCH model (omega, alpha1, and beta1) are significant (p-values < 0.05), indicating that the model is well-fitted. The Ljung-Box test for the residuals shows a p-value of 0.3419 for 10 lags, which indicates no significant autocorrelation in the residuals, meaning the model fits well in terms of residual serial dependence. GARCH(1,1) with Standardized t-Distribution:

The log-likelihood value is 7736.355, which is slightly better than the normal distribution model, indicating a potentially better fit. The coefficients are also significant (p-values < 0.05), with the shape parameter of the t-distribution (shape $= 4.28$) indicating a heavier tail than the normal distribution. The Ljung-Box test for the residuals shows a p-value of 0.3507 for 10 lags, similar to the normal model, suggesting that there is no significant autocorrelation in the residuals. Conclusion: Both the GARCH(1,1) models (with normal and t-distributions) provide a good fit, with no significant residual autocorrelation based on the Ljung-Box test. However, the GARCH model with the standardized t-distribution has a higher log-likelihood and captures heavier tails (as indicated by the shape parameter), suggesting that it may be a better fit for the data due to the presence of tail risk or more extreme variations in the negative log returns of Bitcoin.

e) Residual serial correlation

Question

Residual serial correlation can be present when fitting a GARCH directly on the negative log returns. Hence, in order to circumvent this problem, it is possible to use the following two-step approach:
- Fit an ARIMA(p, d, q) on the negative log returns with the choices p, d and q from part (c);
- Fit a GARCH(1, 1) on the residuals of the ARIMA(p, d, q) fit.

Proceed with the above recipe. Assess the quality of the above fit.

```r
# Step 1: Fit an ARIMA(p, d, q) model on the negative log returns
# From part (c), we decided ARIMA(2, 0, 2)
arima_model <- arima(neg_log_returns_bitcoin, order = c(2, 0, 2))

# Extract the residuals from the ARIMA model
arima_residuals <- residuals(arima_model)

# Step 2: Fit a GARCH(1,1) model on the residuals from the ARIMA model
garch_model_arima_residuals <- garchFit(~ garch(1, 1), data = arima_residuals, cond.dist = "norm"
```

```
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:              0.001422441

Parameter Initialization:
 Initial Parameters:          $params
 Limits of Transformations:   $U, $V
 Which Parameters are Fixed?  $includes
 Parameter Matrix:
                       U             V        params includes
    mu     -0.001381975 1.381975e-03 -0.0001381975     TRUE
    omega   0.000001000 1.000000e+02  0.1000000000     TRUE
    alpha1  0.000000010 1.000000e+00  0.1000000000     TRUE
    gamma1 -0.999999990 1.000000e+00  0.1000000000    FALSE
    beta1   0.000000010 1.000000e+00  0.8000000000     TRUE
    delta   0.000000000 2.000000e+00  2.0000000000    FALSE
    skew    0.100000000 1.000000e+01  1.0000000000    FALSE
    shape   1.000000000 1.000000e+01  4.0000000000    FALSE
 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:     1877.5170: -0.000138197 0.100000 0.100000 0.800000
  1:     1855.7714: -0.000138198 0.0726602 0.109431 0.786204
  2:     1839.1695: -0.000138198 0.0680171 0.139755 0.795457
  3:     1829.7062: -0.000138198 0.0532235 0.143295 0.790116
  4:     1810.0200: -0.000138198 0.0267149 0.199381 0.807723
  5:     1809.1646: -0.000138198 0.0219331 0.198987 0.805030
  6:     1808.7267: -0.000138198 0.0249406 0.202100 0.801633
  7:     1808.3177: -0.000138199 0.0257164 0.207720 0.792203
  8:     1807.4361: -0.000138210 0.0315621 0.224320 0.778984
  9:     1806.9719: -0.000138218 0.0251042 0.230515 0.781780
 10:     1806.9152: -0.000138230 0.0268312 0.234701 0.782044
 11:     1806.6955: -0.000138240 0.0271249 0.235641 0.777616
 12:     1806.4274: -0.000138242 0.0278457 0.249636 0.766085
 13:     1806.3620: -0.000138242 0.0290636 0.249980 0.766517
 14:     1806.3497: -0.000138251 0.0292906 0.250326 0.765248
 15:     1806.3450: -0.000138323 0.0305057 0.251714 0.763384
 16:     1806.3146: -0.000138526 0.0299426 0.253707 0.762550
 17:     1806.3008: -0.000139110 0.0300663 0.256681 0.761179
 18:     1806.2912: -0.000139772 0.0303903 0.258077 0.759281
```

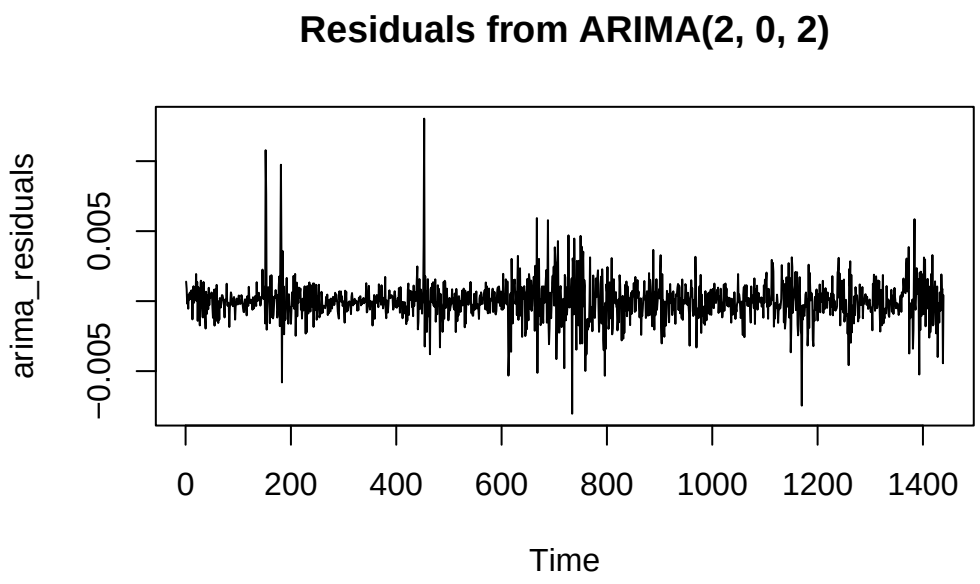
```
 19:     1806.2891: -0.000140480 0.0304056 0.259412 0.758500
 20:     1806.2883: -0.000141953 0.0305967 0.259976 0.757903
 21:     1806.2879: -0.000145229 0.0306869 0.260363 0.757537
 22:     1806.2873: -0.000152139 0.0307685 0.260698 0.757217
 23:     1806.2854: -0.000177344 0.0309333 0.261376 0.756569
 24:     1806.2808: -0.000251181 0.0312215 0.262556 0.755442
 25:     1806.2700: -0.000436287 0.0316540 0.264316 0.753760
 26:     1806.2435: -0.000905201 0.0323112 0.266960 0.751230
 27:     1806.1898: -0.00138197 0.0322376 0.266573 0.751590
 28:     1806.1562: -0.00138197 0.0309672 0.261581 0.756386
 29:     1806.1541: -0.00138197 0.0305619 0.259903 0.757984
 30:     1806.1541: -0.00138197 0.0305834 0.259961 0.757926
 31:     1806.1541: -0.00138197 0.0305829 0.259962 0.757926

Final Estimate of the Negative LLH:
 LLH:  -7627.039    norm LLH:  -5.300236
           mu         omega        alpha1         beta1
-1.965777e-06  6.187958e-08  2.599616e-01  7.579261e-01

R-optimhess Difference Approximated Hessian Matrix:
                  mu         omega        alpha1         beta1
mu     -1.554923e+09  2.507936e+11  9.975053e+04 -2.026553e+04
omega   2.507936e+11 -1.709771e+16 -6.864037e+09 -1.486634e+10
alpha1  9.975053e+04 -6.864037e+09 -6.383094e+03 -9.576435e+03
beta1  -2.026553e+04 -1.486634e+10 -9.576435e+03 -1.823892e+04
attr(,"time")
Time difference of 0.004616976 secs

--- END OF TRACE ---


Time to Estimate Parameters:
 Time difference of 0.02679801 secs
```

```r
# Summary of the GARCH(1,1) model
summary(garch_model_arima_residuals)
```

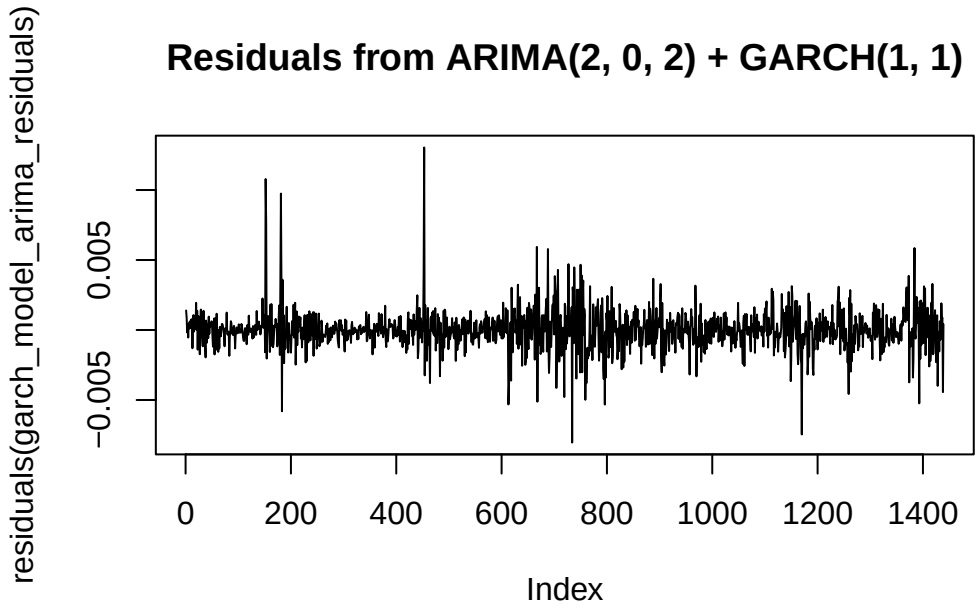
```
Title:
 GARCH Modelling

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

Mean and Variance Equation:
 data ~ garch(1, 1)
<environment: 0x12af139e0>
 [data = arima_residuals]

Conditional Distribution:
 norm
```

```
Coefficient(s):
         mu        omega       alpha1        beta1
-1.9658e-06   6.1880e-08   2.5996e-01   7.5793e-01

Std. Errors:
 based on Hessian

Error Analysis:
         Estimate  Std. Error  t value Pr(>|t|)
mu     -1.966e-06   2.566e-05   -0.077    0.939
omega   6.188e-08   1.534e-08    4.034 5.49e-05 ***
alpha1  2.600e-01   2.936e-02    8.854  < 2e-16 ***
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 16 15:37:32 2024 by user:


Standardised Residuals Tests:
                                Statistic   p-Value
 Jarque-Bera Test   R    Chi^2  2537.823714 0.0000000
 Shapiro-Wilk Test  R    W         0.945703 0.0000000
 Ljung-Box Test     R    Q(10)    10.890175 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)    12.237348 0.2694857
 Ljung-Box Test     R^2  Q(15)    13.290578 0.5798648
 Ljung-Box Test     R^2  Q(20)    14.030759 0.8289330
 LM Arch Test       R    TR^2     12.413093 0.4130994

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

```r
# Plot the residuals from the ARIMA + GARCH model
plot(arima_residuals, main = "Residuals from ARIMA(2, 0, 2)", type = "l")
```



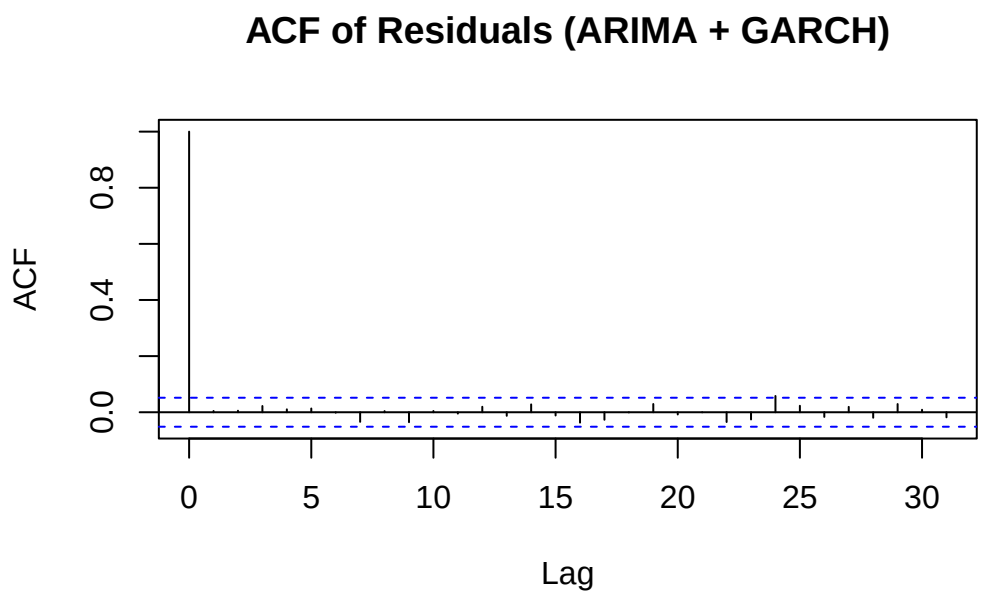

```
plot(residuals(garch_model_arima_residuals), main = "Residuals from ARIMA(2, 0, 2) + GARCH(1, 1)"
```




```
# Assess the quality of the fit using the Ljung-Box test on the GARCH model residuals
ljung_box_test <- Box.test(residuals(garch_model_arima_residuals), lag = 20, type = "Ljung-Box")
print(ljung_box_test)
```

```
    Box-Ljung test

data:  residuals(garch_model_arima_residuals)
X-squared = 11.355, df = 20, p-value = 0.9365
```

```r
# Plot the ACF of the residuals to visually assess if there's still serial correlation
acf(residuals(garch_model_arima_residuals), main = "ACF of Residuals (ARIMA + GARCH)")
```



Quality Assessment of the ARIMA + GARCH(1,1)

Fit:

Parameter Estimates: The fitted ARIMA + GARCH(1,1) model has significant coefficients for the GARCH components: Alpha1 (0.2599) and Beta1 (0.7579) are both highly significant (p-value < 2e-16), indicating that the GARCH(1,1) model has captured the volatility clustering effectively. The intercept (Mu) is not significant, suggesting that the mean of the residuals is close to zero, which is expected in a well-fitted model.

Log-Likelihood and AIC: The log-likelihood of the model is high (7627.039), and the AIC (-10.59491) and BIC (-10.58026) indicate a good fit for the model. Lower AIC/BIC values suggest a better model fit.

Residuals Analysis: Ljung-Box Test for Serial Correlation: The p-value of the Ljung-Box test (p = 0.9365) for the residuals of the ARIMA + GARCH(1,1) model is very high, indicating no significant serial correlation. This suggests that the model has captured the dependencies in the data well, and the residuals are essentially white noise.

Jarque-Bera and Shapiro-Wilk Tests for Normality: Both tests indicate non-normality of the residuals (p-value = 0.000). This is common in financial time series data, as they often exhibit heavy tails and skewness.

Ljung-Box Test on Squared Residuals: The p-values for the Ljung-Box test on squared residuals are also high, indicating no remaining conditional heteroscedasticity in the model, suggesting that the GARCH model has effectively modeled the conditional variance.

Conclusion: The ARIMA + GARCH(1,1) model fits the data well, capturing both the autocorrelation in the returns and the conditional heteroscedasticity (volatility clustering). The residuals show no significant autocorrelation, and the GARCH model appears to have adequately handled the volatility. The model might still exhibit non-normality, but this is expected in financial data due to extreme returns or fat tails.

f) Model Comparison

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

Comparison of the Three Models:

Model 1: GARCH(1,1) with Normal Distribution

Fit: This model captured the volatility clustering well. The Ljung-Box test on residuals and squared residuals showed no significant serial correlation, indicating that the model accounted for the time-varying volatility. Homoscedasticity: By definition, a GARCH model assumes heteroscedasticity, meaning that the volatility changes over time. Therefore, the assumption of constant variance (homoscedasticity) is explicitly violated in this model. Conclusion: This model is good for capturing conditional heteroscedasticity (changing volatility), and thus it is appropriate for financial time series data where volatility clustering is observed.

Model 2: GARCH(1,1) with Standardized t-Distribution

Fit: Similar to the GARCH with normal distribution, this model accounts for volatility clustering and captures extreme returns better due to the heavy tails of the t-distribution. Homoscedasticity: The GARCH structure still assumes heteroscedasticity, and therefore the homoscedasticity assumption is violated in this model as well. Conclusion: This model is better suited for capturing heavy-tailed data, like extreme price movements in financial markets, while still accounting for changing volatility. It can outperform the GARCH(1,1) with a normal distribution for financial data with fat tails.

Model 3: ARIMA + GARCH(1,1) (Two-Step Approach)

Fit: This two-step model first applies an ARIMA to remove autocorrelation from the log returns and then applies a GARCH(1,1) to model the remaining volatility clustering. The Ljung-Box tests on residuals showed no significant serial correlation, suggesting that the ARIMA model handled the autoregressive components well, and the GARCH captured the volatility. Homoscedasticity: As in the other GARCH models, this model also assumes heteroscedasticity, meaning the homoscedasticity assumption is violated here as well. Conclusion: This model is the most comprehensive because it first removes any autocorrelation in the series (via ARIMA) before modeling the volatility. It is typically more suitable for financial time series where both autocorrelation and volatility clustering are present.

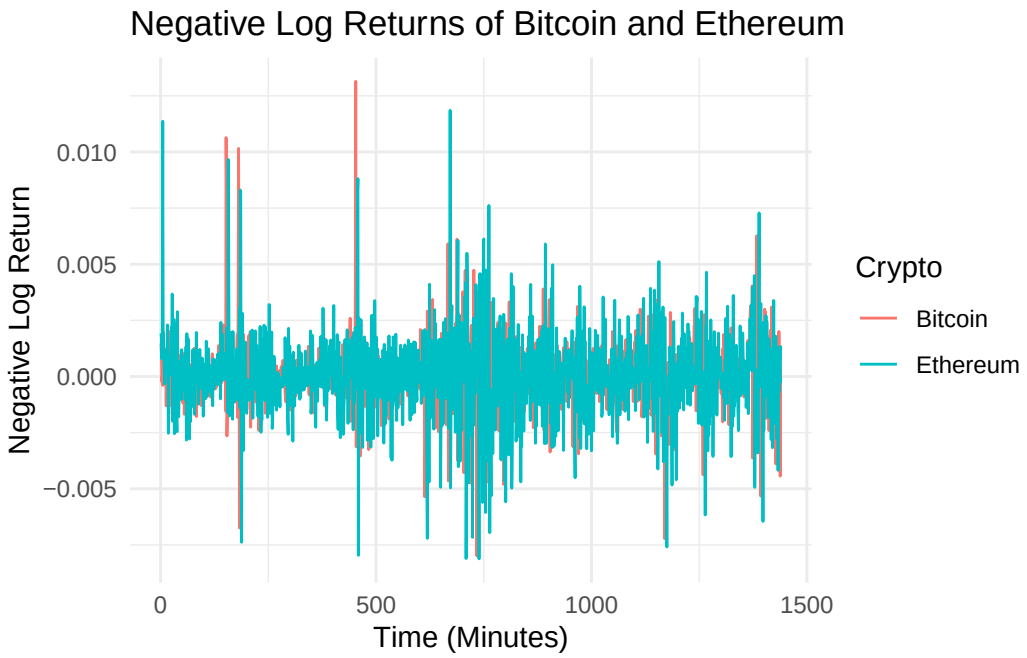
Conclusion: Most Suitable Model: The ARIMA + GARCH(1,1) model is the most suitable because it addresses both the autocorrelation in the log returns and the heteroscedasticity (changing volatility). It captures both the time-varying nature of volatility and any serial dependence in the data. Homoscedasticity Violation: In all three models, the homoscedasticity assumption is violated since all models incorporate the GARCH(1,1) structure, which models conditional heteroscedasticity (changing volatility over time).

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.

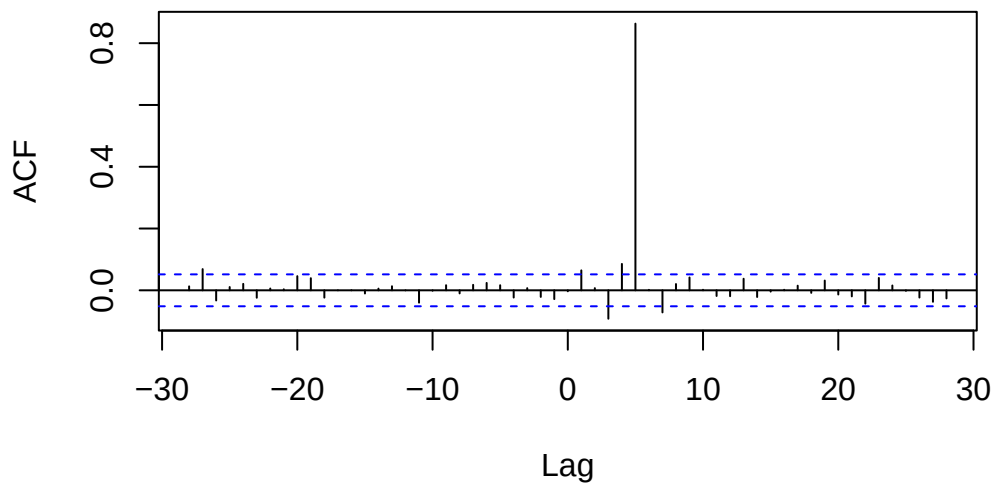


The graph could suggest that the negative log returns of Bitcoin and Ethereum are dependent. However, we will perform a correlation test to confirm this.

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?



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 Etherum'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 Etherum 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.

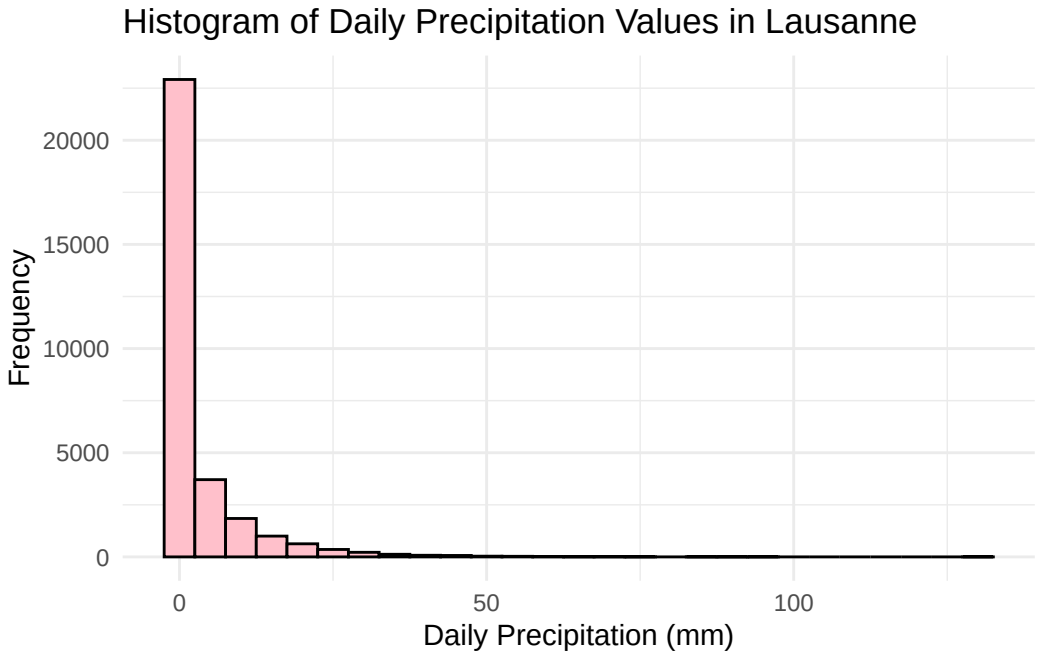
Practical 2: Weather

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 ?

```r
data_clean_df <- read.csv(here::here("data", "Precipitation_lausanne_full.csv"))

data_clean_df$Date <- as.Date(data_clean_df$Date, format = "%m/%d/%Y")
data_clean_df$Precipitation <- as.numeric(data_clean_df$Precipitation)
data_clean_df <- na.omit(data_clean_df)
```



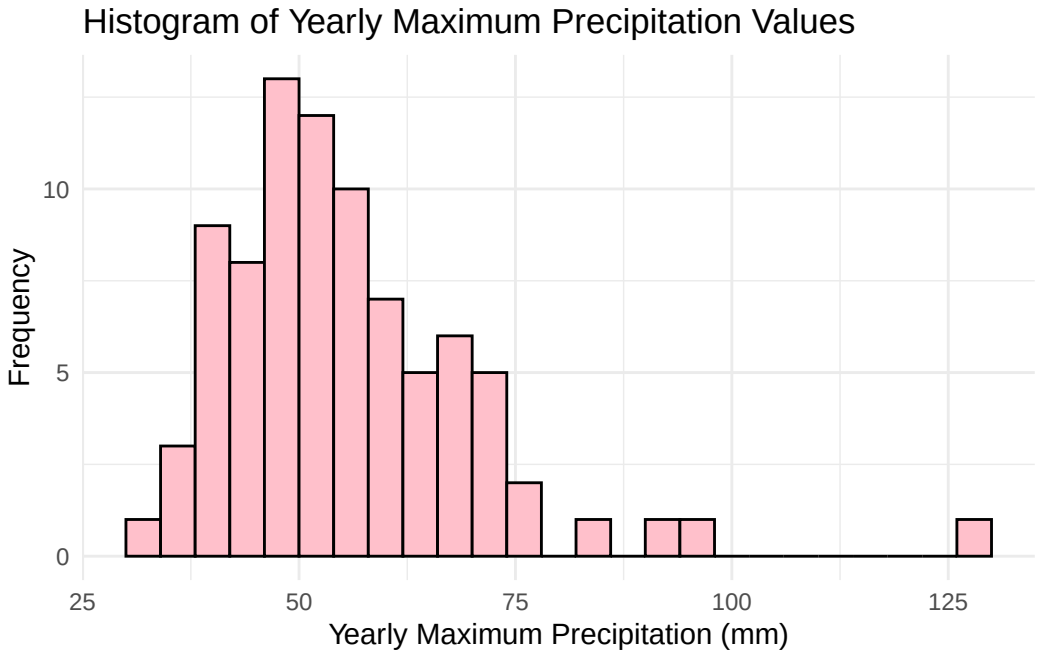
Based on the histogram, the data is heavily skewed to the right. The majority of the precipitation values are concentrated around zero and a long tail for higher values. There are many small precipitation events and a few extreme ones.

The distribution that will best fit the data is the **gamma distribution** and **Weibull**. Gamma and Weibull distributions are usually used in meteorological processes such as precipitation quantities.

b) Best fit distribution on yearly maxima

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

```r
# Extract yearly maximum values
data_clean_df$Year <- format(data_clean_df$Date, "%Y")
yearly_max <- aggregate(Precipitation ~ Year, data = data_clean_df, max)
```



Based on the characteristics of the data (yearly maxima and right-skewed nature), the Generalised Extreme Value (GEV) distribution seems to be the best fit. The GEV distribution is designed to model extreme values like yearly maxima and can accommodate the heavy-tailed nature of the data, where a few large values are observed but are rare.

c) Linear model approach

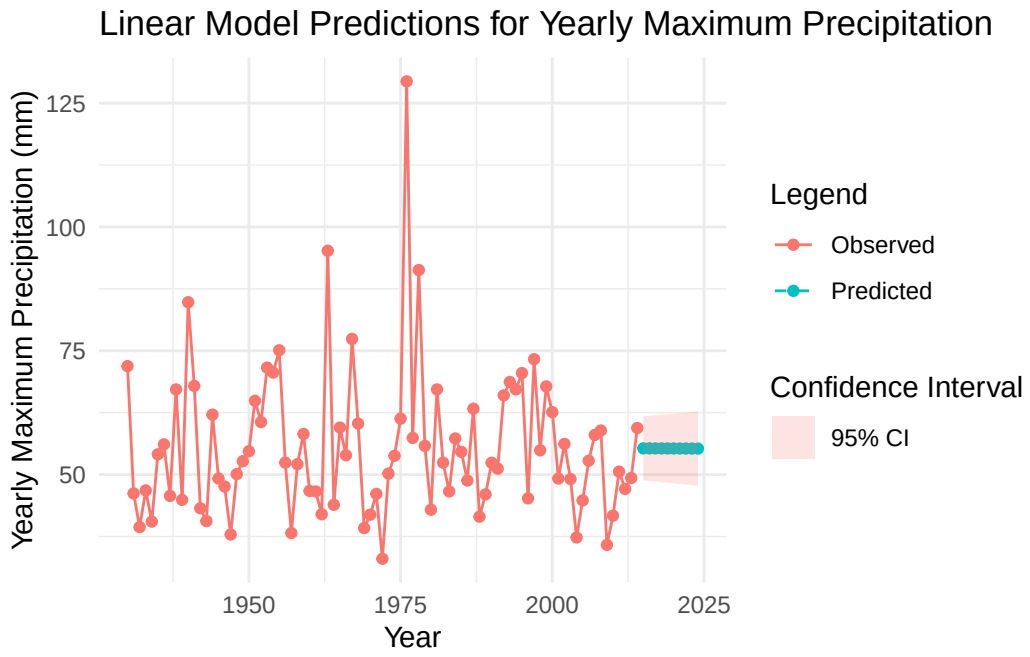
> Question
>
> Fit a linear model to the yearly maximum precipitation values and predict the values for the next 10 years. Provide confidence intervals for your predictions and plot it. Do you think that this a reasonable approach?

```r
# Fit linear model
linear_model <- lm(Precipitation ~ as.numeric(Year), data = yearly_max)
summary(linear_model)
```

```
Call:
lm(formula = Precipitation ~ as.numeric(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
as.numeric(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
```

```r
# Predict for next 10 years
future_years <- data.frame(Year = as.numeric(2015:2024))
predictions <- predict(linear_model, newdata = future_years, interval = "confidence")
```



The very high p-value ($0.9258 < 0.05$) shows that the slope of the regression line is not statistically significant. There is no evidence that the year has a significant effect on yearly maximum precipitation in this dataset. Additionally, the predictions show little change in yearly maximum precipitation over the next 10 years.

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?

```r
# Fit GEV models
gev_model_const <- fevd(yearly_max$Precipitation, type = "GEV", method = "MLE")
gev_model_timevar <- fevd(yearly_max$Precipitation, type = "GEV", method = "MLE", loc = ~as.numer

# Compare models
summary(gev_model_const)
```

```
fevd(x = yearly_max$Precipitation, type = "GEV", method = "MLE")

[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
```

```
summary(gev_model_timevar)
```

```
fevd(x = yearly_max$Precipitation, location.fun = ~as.numeric(yearly_max$Year),
    type = "GEV", method = "MLE")

[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
```

AIC for the constant model (672.9433) is lower compared to the time-varying location model (674.8906). BIC for the constant model is also lower (680.2712) than for the time-varying location model (684.6612). Si nce both AIC and BIC values favor the constant parameter model, this suggest that the GEV model with constant parameters is the preferred model for fitting the yearly maximum precipitation data in Lausanne.

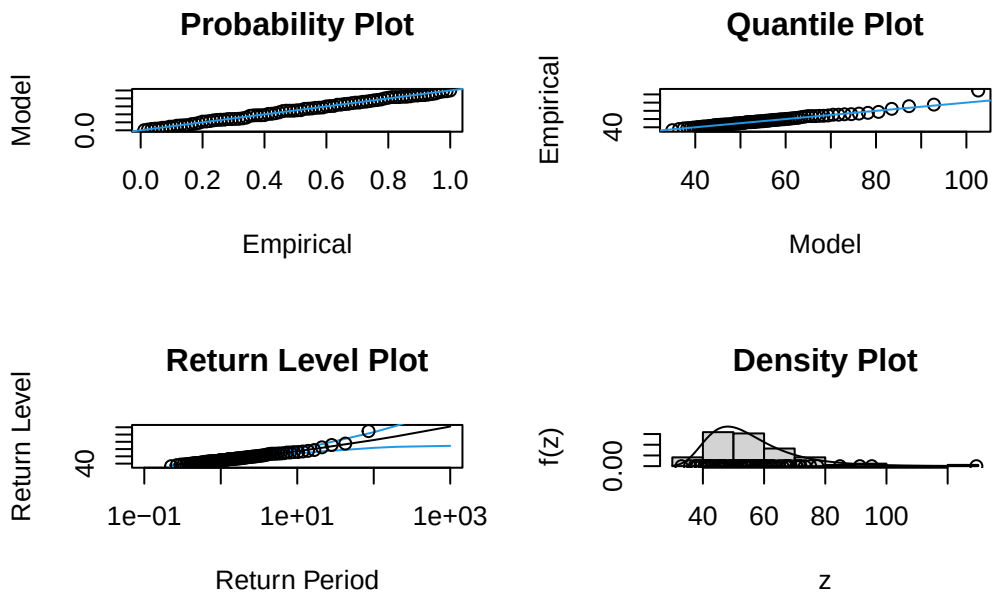
e) Diagnostic plots

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

```
gev_model_const_diag <- gev.fit(yearly_max$Precipitation)
```

```
$conv
[1] 0

$nllh
[1] 333.4716

$mle
[1] 48.92521354  9.97227559  0.08329645

$se
[1] 1.21298615 0.90492024 0.07773763
```

```
gev.diag(gev_model_const_diag)
```



Probability Plot: Since most points lie close to the line, it suggests that the GEV model is reasonably capturing the general behavior of the precipitation extremes.

Quantile Plot: The majority of the points are also close to the diagonal line, it indicates a good fit.

Return Level Plot: The data points generally follow the fitted line, and most fall within the confidence intervals.

Density Plot: The histogram and density overlay indicate that the fitted GEV distribution aligns with the observed data distribution, although there may be sligh deviations at the tail.

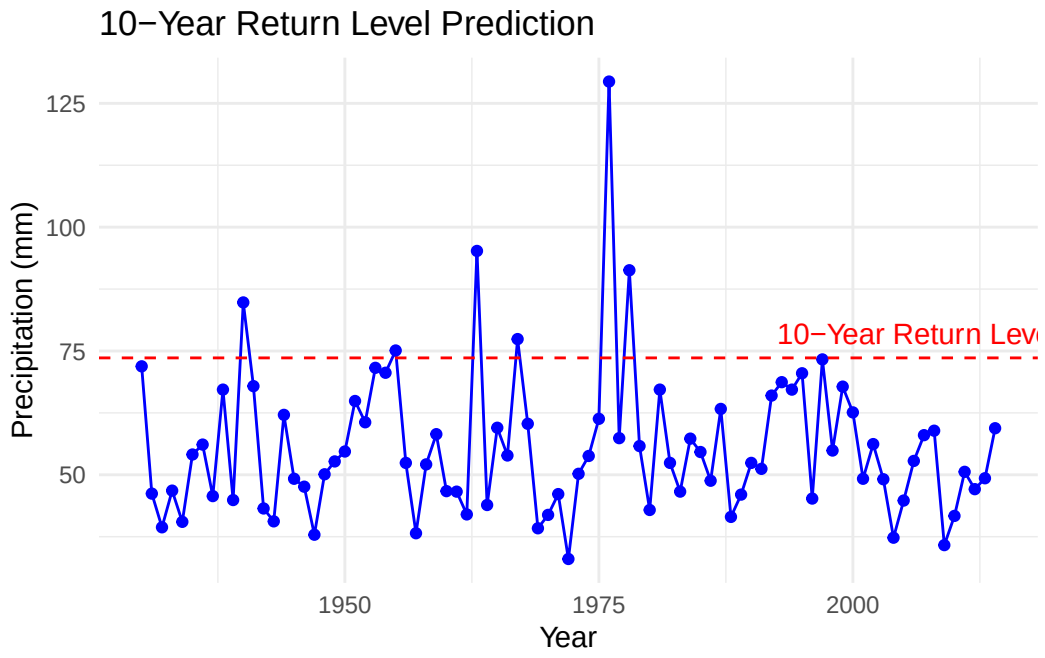
Conclusion: the GEV model with constant parameters provides a reasonably good fit for the yearly maximum precipitation values in Lausanne.

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.

```
# Calculate 10-year return level
return_period <- 10
return_level <- return.level(gev_model_const, return.period = return_period)
```



The plot displays the yearly maximum precipitation in Lausanne over time, with the 10-year return level of 73.61mm marked by a red dashed line. This return level indicates the precipitation level that is expected to be exceeded once every 10 years.

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.

```
# Extract GEV parameters directly from the model
location <- gev_model_const$results$par[1] # Location parameter
```

```r
scale <- gev_model_const$results$par[2]   # Scale parameter
shape <- gev_model_const$results$par[3]   # Shape parameter
```

```r
# Extract GEV parameters from the constant model
location <- gev_model_const$results$par[1]
scale <- gev_model_const$results$par[2]
shape <- gev_model_const$results$par[3]

# Define return periods
return_periods <- c(10, 20, 50, 85)

# Function to calculate return levels for each period
calculate_return_level <- function(return_period) {
  location + (scale / shape) * ((-log(1 - 1 / return_period))^(-shape) - 1)
}

# Calculate return levels
return_levels <- sapply(return_periods, calculate_return_level)

# Display return levels
names(return_levels) <- paste(return_periods, "year return level")
print(return_levels)
```

```
10 year return level 20 year return level 50 year return level
            73.60265             82.52333             94.89274
85 year return level
           102.43790
```

```r
# Count how many yearly maximum values exceed each return level
exceedances <- sapply(return_levels, function(level) sum(yearly_max$Precipitation > level))

# Display exceedance counts
exceedances
```

```
10 year return level 20 year return level 50 year return level
                   6                    4                    2
85 year return level
                   1
```

Each year, there is 1/10 chance of exceeding the 10-year return level of 73.61 mm based on the GEV model, with 6 historical exceedances aligning with this probability.

The GEV approach accurately captures the rarity of extreme events, providing 20, 50, and 85-year return levels (82.53 mm, 94.90 mm, and 102.45 mm, respectively), with 4, 2, and 1 exceedances observed for each.

In contrast, the linear model lacks the specificity to estimate extreme events reliably, as it doesn't account for tail behavior. The GEV model's results match expected exceedance frequencies, validating its effectiveness for extreme precipitation analysis.

h) Return period 100 mm precipitation

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

```
# Desired precipitation level
precipitation_level <- 100

# Calculate the return period for 100 mm of precipitation
return_period_100mm <- 1 / (1 - exp(-((precipitation_level - location) / scale * shape + 1)^(-1 /

cat("The return period for a 100 mm precipitation event is approximately:", return_period_100mm,
```

```
The return period for a 100 mm precipitation event is approximately: 71.77507 years
```

The return period for a 100 mm precipitation event is approximately: 71.70624 years

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.

```
# Desired precipitation threshold
precipitation_threshold <- 150

# Compute the probability of exceeding 150 mm in a given year
prob_exceed_150mm <- 1 - exp(-((precipitation_threshold - location) / scale * shape + 1)^(-1 / sh

cat("The probability of a precipitation event exceeding 150 mm in the next year is approximately:
```

```
The probability of a precipitation event exceeding 150 mm in the next year is approximately: 0.0642(
```

The probability of a precipitation event exceeding 150 mm in the next year is approximately: 0.0643559 %

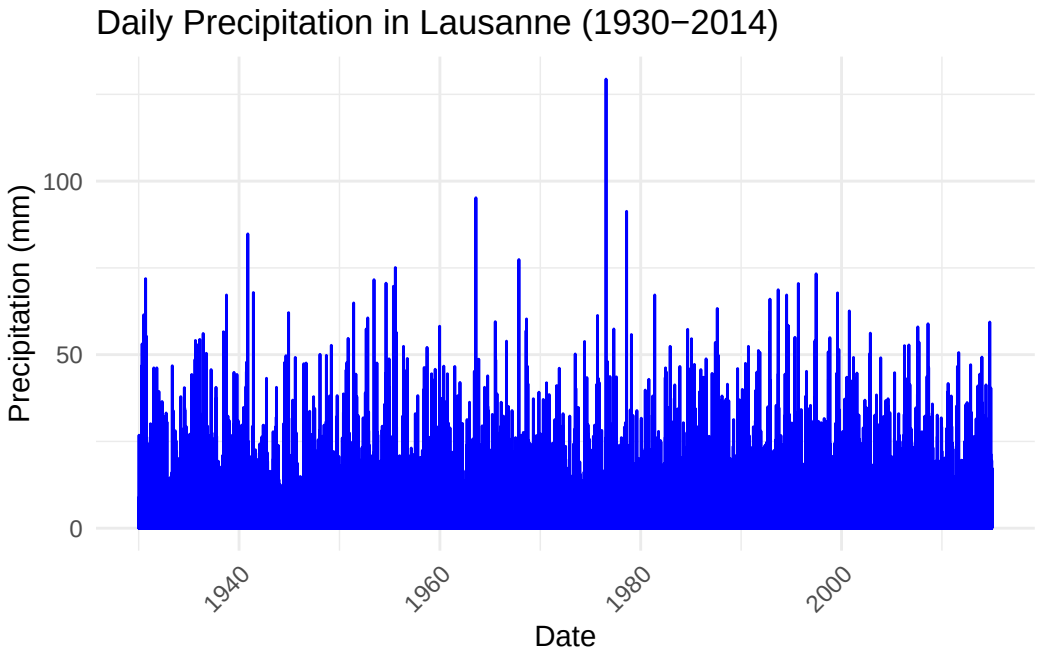
Part 2: Peaks-over-threshold approach

a) Time series plot

Question

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

```
# Load the precipitation data
data_clean_df <- read.csv(here::here("data", "Precipitation_lausanne_full.csv"))

# Ensure that the Date column is in Date format
data_clean_df$Date <- as.Date(data_clean_df$Date, format = "%m/%d/%Y")
```

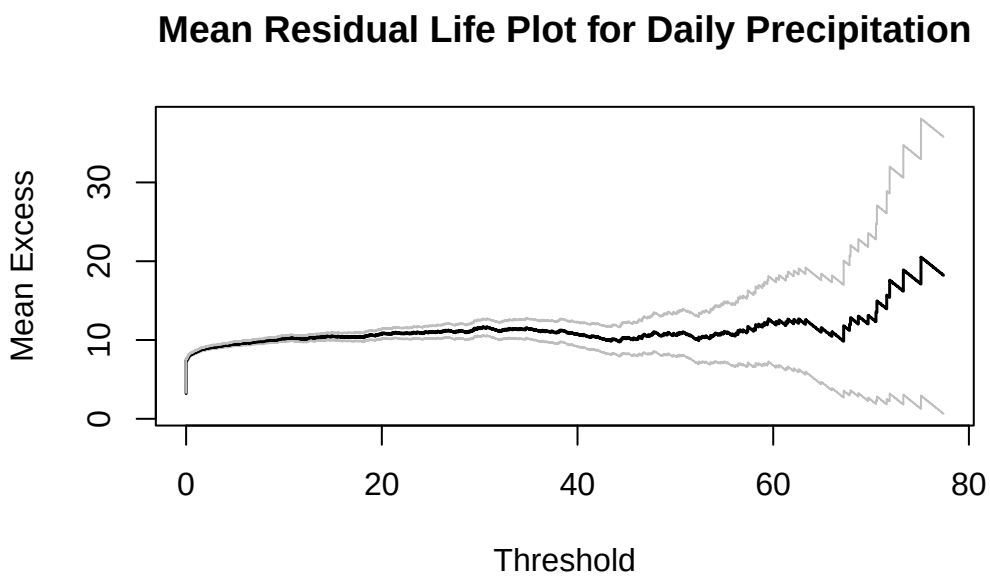


b) Mean Residual Life Plot

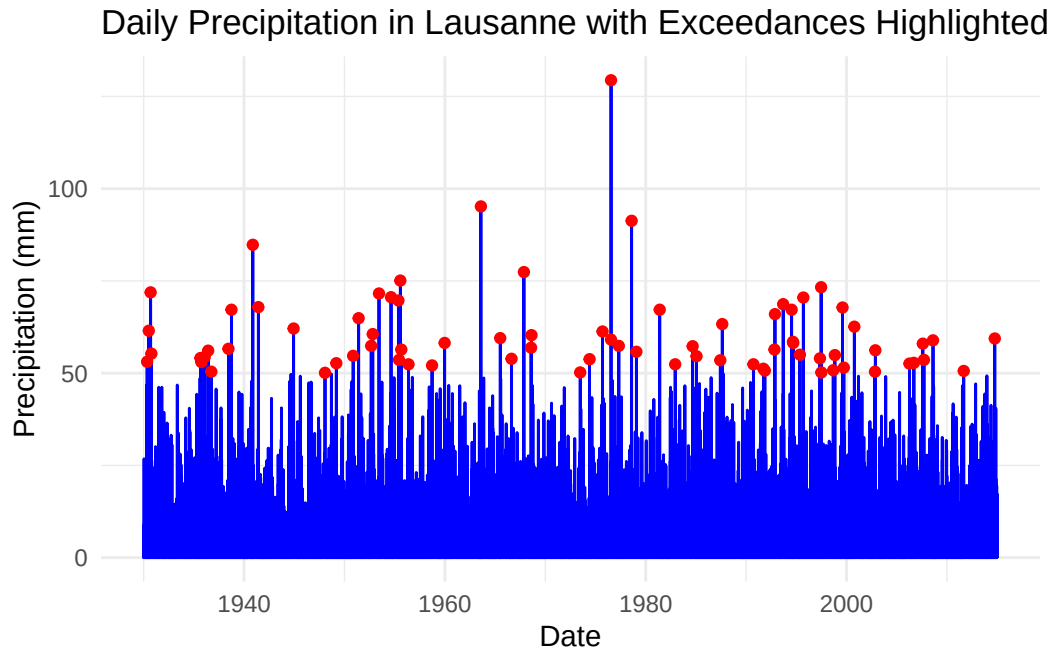
Question

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.

```
# Extract the precipitation values
precipitation_values <- data_clean_df$Precipitation

# Generate the Mean Residual Life Plot
mrlplot(precipitation_values, main = "Mean Residual Life Plot for Daily Precipitation")
```




```
# After observing the MRL plot, let's choose a threshold
# Replace the chosen_threshold value with the one you select after reviewing the plot
chosen_threshold <- 50

# Highlight data exceeding the threshold in the time series plot
data_clean_df$AboveThreshold <- data_clean_df$Precipitation > chosen_threshold
```

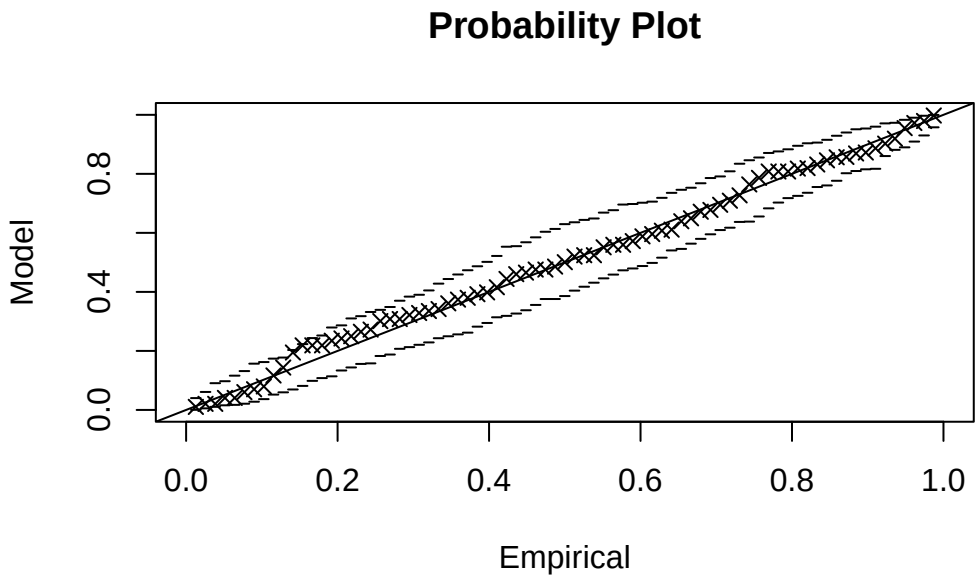


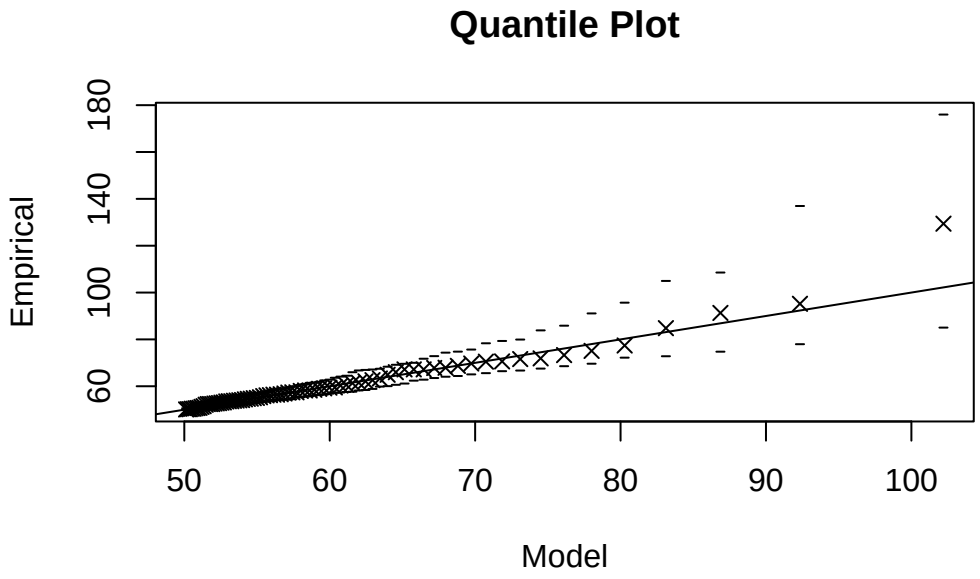
We chose a threshold of 50 mm, which seems appropriate, as it allows for a stable selection of extreme precipitation events base on the MRL plot.

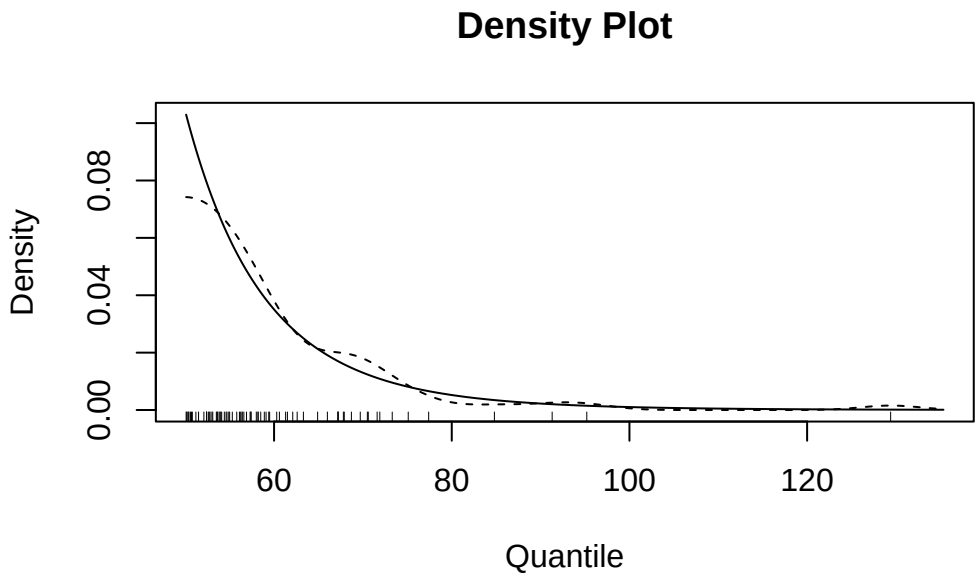
c) Fit a GPD

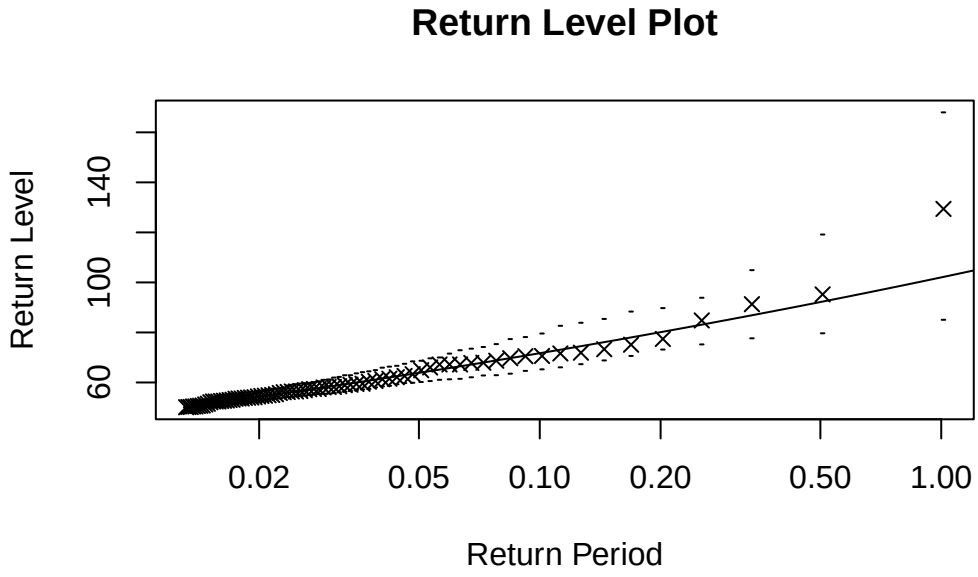
> Question
>
> Fit a GPD for the data exceeding the threshold and draw a diagnostic plot. Is it a reasonable fit? (Hint: if not, you may reconsider the choice of the threshold)

```
# Define the threshold
threshold <- 50

# Fit the GPD model to the exceedances over the threshold
fit_gpd <- fpot(data_clean_df$Precipitation, threshold = threshold)
```









Given the plots, the 50 mm threshold seems to provide an adequate fit without significant deviations. Thus, it is reasonable to proceed with this threshold, as the model sufficiently captures the tail behavior of the date within the confidence intervals.

d) Return levels prediction

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

```r
threshold <- 50

# Extract parameters from the fitted model
shape <- fit_gpd$estimate["shape"]
scale <- fit_gpd$estimate["scale"]

# Calculate the rate of exceedance (lambda)
n_exceedances <- length(fit_gpd$data)  # Number of exceedances
n_total <- nrow(data_clean_df)  # Total number of data points
lambda <- n_exceedances / n_total

# Define return periods
return_periods <- c(10, 20, 50, 85)

# Function to calculate return level
return_level <- function(T) {
  threshold + (scale / shape) * (((T * lambda)^shape) - 1)
}

# Calculate return levels for each return period
return_levels <- sapply(return_periods, return_level)
names(return_levels) <- paste(return_periods, "year return level")

# Display return levels
return_levels
```

```
10 year return level 20 year return level 50 year return level
            74.80835             83.44289             95.79736
85 year return level
           103.47688
```

10-year return level: 74.81 mm, 20-year return level: 83.44, 50-year return level: 95.80, 85-year return level: 103.48

e) Return period 100 mm precipitation

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

```r
# Display both the shape and scale parameters
print(fit_gpd$estimate)
```

```
     scale      shape
9.60205760 0.09818681
```

```r
# Define the target return level and threshold
target_level <- 100
```

```r
threshold <- 50  # Replace if a different threshold was used

# Parameters from the fitted GPD model
shape <- 0.09818681
scale <- 9.60205760

# Calculate the exceedance rate (lambda)
n_exceedances <- length(fit_gpd$data)  # Number of exceedances above threshold
n_total <- nrow(data_clean_df)  # Total number of data points
lambda <- n_exceedances / n_total

# Calculate the return period for the 100 mm event
return_period_100mm <- (1 / lambda) * (1 + (shape * (target_level - threshold) / scale))^(1 / sha

# Display the return period
return_period_100mm
```

```
[1] 67.07623
```

On average, a precipitation event of this magnitude (100 mm) would be expected to occur once every 67 years in Lausanne. The average annual precipitation in Lausanne is around 1150 mm.

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.

```r
# Define the precipitation threshold (150 mm)
precipitation_threshold <- 150

# Extract the parameters of the fitted GPD model
shape <- fit_gpd$estimate["shape"]
scale <- fit_gpd$estimate["scale"]
threshold <- 50  # The threshold you selected for the POT method

# Calculate the CDF for the 150 mm precipitation level
cdf_150mm <- 1 - (1 + (shape / scale) * (precipitation_threshold - threshold))^(-1 / shape)

# Compute the probability of exceeding 150 mm
prob_exceed_150mm <- 1 - cdf_150mm

cat("The probability of a precipitation event exceeding 150 mm in the next year is approximately:
```

```
The probability of a precipitation event exceeding 150 mm in the next year is approximately: 0.07664
```

The probability of a precipitation event exceeding 150 mm in the next year is approximately: 0.07664709%.

```r
# Fit the GEV model to the yearly maximum precipitation (or exceedances)
fit_gev <- fevd(yearly_max$Precipitation, type = "GEV", method = "MLE")

# View the model parameters
summary(fit_gev)
```

```
fevd(x = yearly_max$Precipitation, type = "GEV", method = "MLE")

[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
```

```r
# Extract parameters for the GEV model
location <- fit_gev$results$par[1]  # Location parameter
scale <- fit_gev$results$par[2]     # Scale parameter
shape <- fit_gev$results$par[3]     # Shape parameter
# Calculate the probability of exceeding 150 mm using the GEV model
prob_exceed_150_gev <- 1 - pevd(150, loc = location, scale = scale, shape = shape, type = "GEV")

# Display the probability
prob_exceed_150_gev
```

```
[1] 0.0006420734
```

```r
# Fit the GEV model to the yearly maximum precipitation (or exceedances)
fit_gev <- fevd(yearly_max$Precipitation, type = "GEV", method = "MLE")

# Extract parameters for the GEV model
location <- fit_gev$results$par[1]  # Location parameter
```

```r
scale <- fit_gev$results$par[2]     # Scale parameter
shape <- fit_gev$results$par[3]     # Shape parameter

# Calculate the probability of exceeding 150 mm using the GEV model
prob_exceed_150_gev <- 1 - pevd(150, loc = location, scale = scale, shape = shape, type = "GEV")

# Display the probability
prob_exceed_150_gev
```

```
[1] 0.0006420734
```

```r
# Step 1: Compute the probability of exceeding the threshold (150 mm) for a single day
daily_prob_exceed_150mm <- lambda * (1 + (shape * (150 - threshold) / scale))^(-1 / shape)

# Step 2: Calculate the annual probability of at least one day exceeding 150 mm
annual_prob_exceed_150mm <- 1 - (1 - daily_prob_exceed_150mm)^365

# Display the annual probability
annual_prob_exceed_150mm
```

```
    shape
0.2201784
```

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?

```r
# Fit the GEV model to the yearly maximum precipitation (or exceedances)
fit_gev <- fevd(yearly_max$Precipitation, type = "GEV", method = "MLE")

# View the model parameters
summary(fit_gev)
```

```
fevd(x = yearly_max$Precipitation, type = "GEV", method = "MLE")

[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
```

```r
# Extract parameters for the GEV model
location <- fit_gev$results$par[1]  # Location parameter
scale <- fit_gev$results$par[2]     # Scale parameter
shape <- fit_gev$results$par[3]     # Shape parameter
# Calculate the probability of exceeding 150 mm using the GEV model
prob_exceed_150_gev <- 1 - pevd(150, loc = location, scale = scale, shape = shape, type = "GEV")

# Display the probability
prob_exceed_150_gev
```

```
[1] 0.0006420734
```

```r
# Fit the GEV model to the yearly maximum precipitation (or exceedances)
fit_gev <- fevd(yearly_max$Precipitation, type = "GEV", method = "MLE")

# Extract parameters for the GEV model
location <- fit_gev$results$par[1]  # Location parameter
scale <- fit_gev$results$par[2]     # Scale parameter
shape <- fit_gev$results$par[3]     # Shape parameter

# Calculate the probability of exceeding 150 mm using the GEV model
prob_exceed_150_gev <- 1 - pevd(150, loc = location, scale = scale, shape = shape, type = "GEV")

# Display the probability
prob_exceed_150_gev
```

```
[1] 0.0006420734
```

The block maxima method estimates a 0.064% annual probability of precipitation exceeding 150 mm, while the POT appraoch estimates a slightly higher probability at 0.076%. The block maxima is simpler but loses data by focusing only on yearly maxima, potentially underestimating extremes. In contrast, the POT approach captures all exceedances above a threshold, providing more precise insights into rare events. The POT method is thus preferred for analysing extreme precipitation events.

Part 3: Clustering and Seasonal Variations

```r
# Load Geneva temperature data (assuming the file is now in CSV format)
geneva_data <- read.csv(here::here("data","Geneva_temperature.csv"))

# Inspect the structure of the data to identify the relevant columns
str(geneva_data)
```

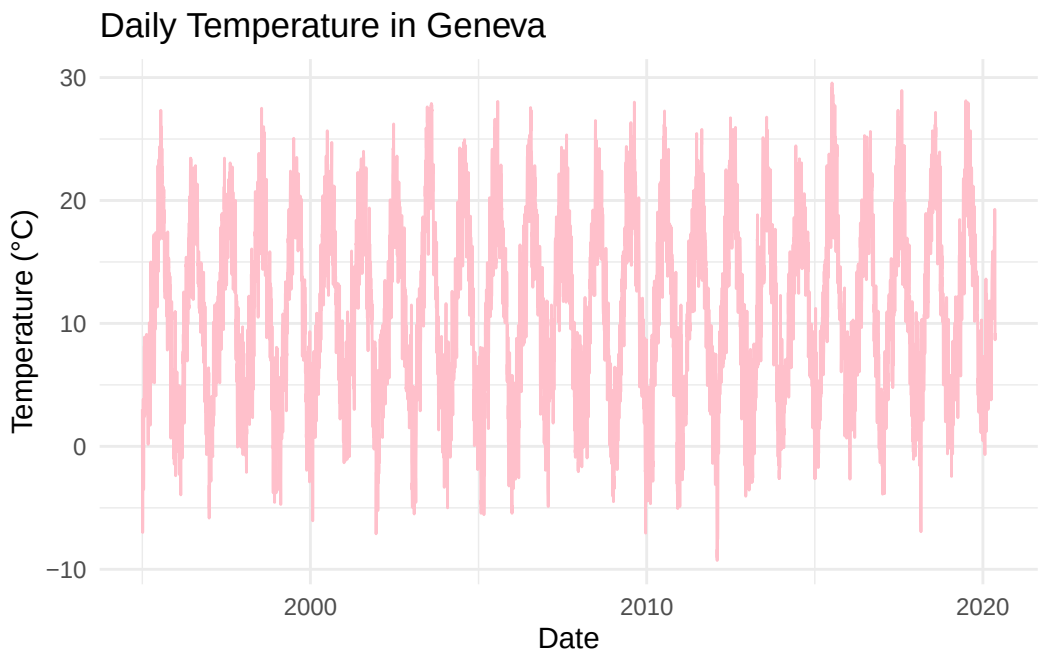
```
'data.frame':   9266 obs. of  9 variables:
 $ X              : int  949772 949773 949774 949775 949776 949777 949778 949779 949780 949781 ...
 $ Region         : chr  "Europe" "Europe" "Europe" "Europe" ...
 $ Country        : chr  "Switzerland" "Switzerland" "Switzerland" "Switzerland" ...
 $ State          : logi  NA NA NA NA NA NA ...
 $ City           : chr  "Geneva" "Geneva" "Geneva" "Geneva" ...
 $ Month          : int  1 1 1 1 1 1 1 1 1 1 ...
 $ Day            : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Year           : int  1995 1995 1995 1995 1995 1995 1995 1995 1995 1995 ...
 $ AvgTemperature: num  3 -1.67 -1.44 -2.89 -7 ...
```

a) Plot the data for Summer

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

```r
# Create the Date column
geneva_data$Date <- as.Date(paste(geneva_data$Year, geneva_data$Month, geneva_data$Day, sep = "-"

# Check if Date column was created correctly
str(geneva_data)
```

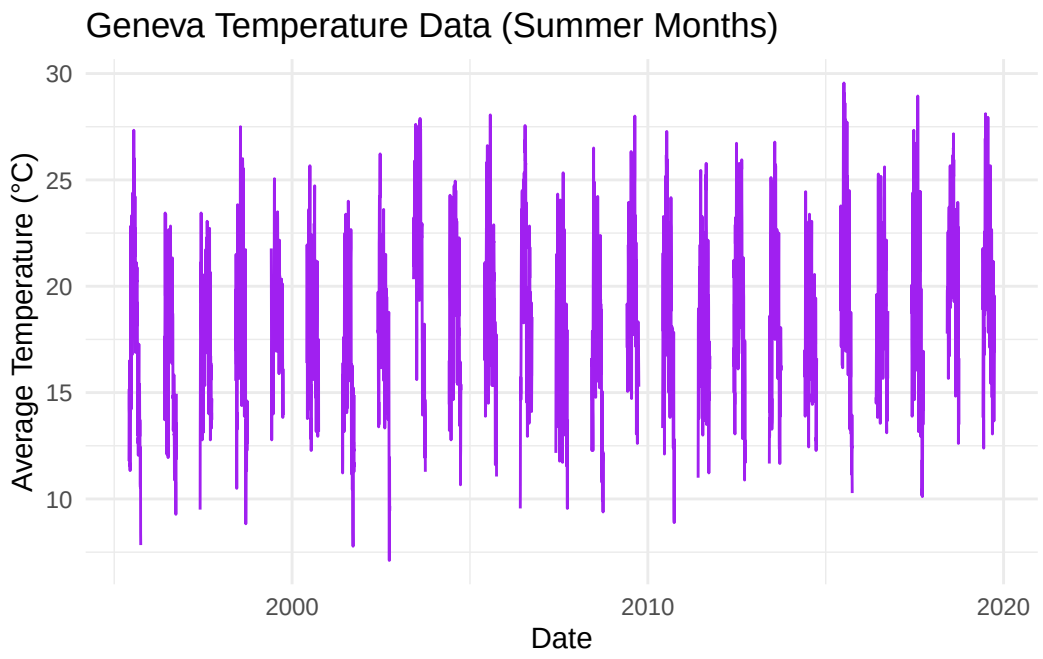
```
'data.frame':   9266 obs. of  10 variables:
 $ X              : int  949772 949773 949774 949775 949776 949777 949778 949779 949780 949781 ...
 $ Region         : chr  "Europe" "Europe" "Europe" "Europe" ...
 $ Country        : chr  "Switzerland" "Switzerland" "Switzerland" "Switzerland" ...
 $ State          : logi  NA NA NA NA NA NA ...
 $ City           : chr  "Geneva" "Geneva" "Geneva" "Geneva" ...
 $ Month          : int  1 1 1 1 1 1 1 1 1 1 ...
 $ Day            : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Year           : int  1995 1995 1995 1995 1995 1995 1995 1995 1995 1995 ...
 $ AvgTemperature: num  3 -1.67 -1.44 -2.89 -7 ...
 $ Date           : Date, format: "1995-01-01" "1995-01-02" ...
```




```r
# Subset the data for summer months (June to September)
summer_data <- subset(geneva_data, Month >= 6 & Month <= 9)

# Check the first few rows of the summer data
head(summer_data)
```

```
        X Region     Country State   City Month Day Year AvgTemperature
152 949923 Europe Switzerland    NA Geneva     6   1 1995       11.77778
153 949924 Europe Switzerland    NA Geneva     6   2 1995       12.55556
154 949925 Europe Switzerland    NA Geneva     6   3 1995       14.66667
155 949926 Europe Switzerland    NA Geneva     6   4 1995       15.16667
156 949927 Europe Switzerland    NA Geneva     6   5 1995       14.22222
157 949928 Europe Switzerland    NA Geneva     6   6 1995       14.00000
          Date
152 1995-06-01
153 1995-06-02
154 1995-06-03
155 1995-06-04
156 1995-06-05
157 1995-06-06
```

```r
# Add a Year column to distinguish each summer period
summer_data$Year <- format(summer_data$Date, "%Y")
```



b) Compare extremal index

> Question
>
> 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?

```
# Determine a threshold for extreme temperatures (e.g., the 95th percentile)
threshold <- quantile(summer_data$AvgTemperature, 0.95, na.rm = TRUE)
cat("Chosen threshold for extreme temperatures:", threshold, "°C\n")
```

```
Chosen threshold for extreme temperatures: 24.94444 °C
```


```
# Determine a threshold for extreme temperatures (e.g., the 95th percentile)
threshold <- quantile(summer_data$AvgTemperature, 0.95, na.rm = TRUE)
cat("Chosen threshold for extreme temperatures:", threshold, "°C\n")
```

```
Chosen threshold for extreme temperatures: 24.94444 °C
```


```
# Identify the days where the temperature exceeds the threshold
extreme_days <- summer_data$AvgTemperature > threshold

# Compute the extremal index
extremal_index <- extremalindex(summer_data$AvgTemperature, threshold = threshold)

# Display the extremal index
```

```r
cat("The extremal index is:", extremal_index, "\n")
```

```
The extremal index is: 0.2612517 40 9
```

```r
# Calculate the probability that an extreme day is followed by another extreme day
probability_following_extreme <- 1 - extremal_index

# Display the probability
cat("The probability that if today is extreme, tomorrow will also be extreme is:", probability_fo
```

```
The probability that if today is extreme, tomorrow will also be extreme is: 0.7387483 -39 -8
```

The extremal index of the subsetted series, with a threshold of 24.94°C, is 0.2613 (this value is closer to 0 than 1). Therefore, if the temperature today is extreme, there is a 26.13 % probability that tomorrow's temperature will also be extreme. While there is some clustering of extreme events, the persistence is not high, suggesting that extreme temperature days are relatively isolated in time.

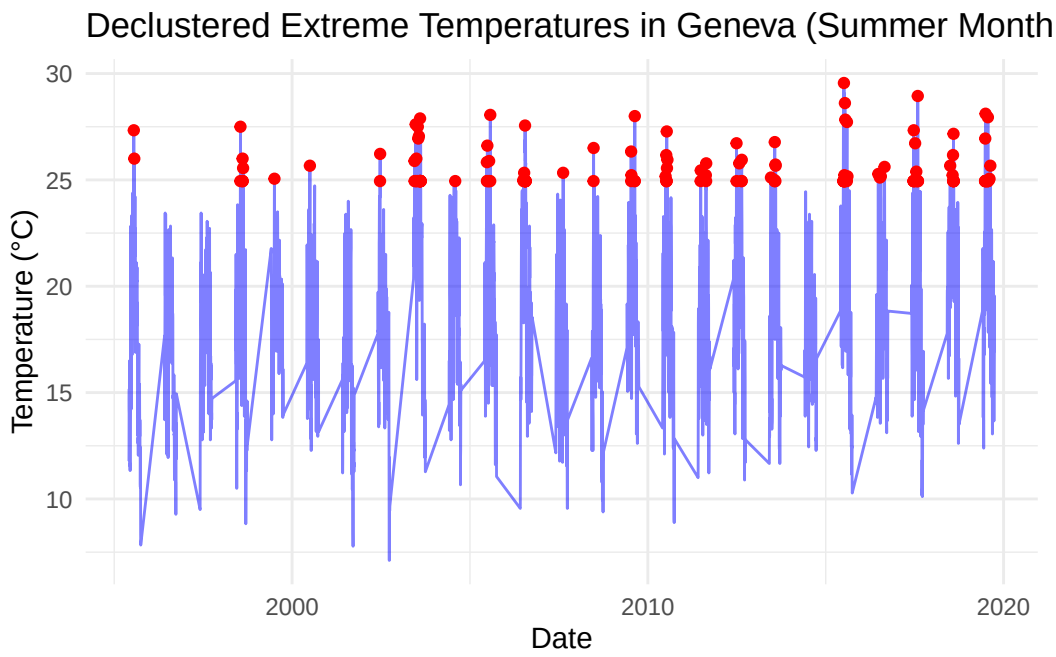
c) Declustering the data

> Question
>
> Decluster the data using a suitable threshold. Plot the resulting declustered data. (Hint: you may want to use `decluster` function in the `extRemes` package.)

```r
# Determine a threshold for extreme temperatures (e.g., the 95th percentile)
threshold <- quantile(summer_data$AvgTemperature, 0.95, na.rm = TRUE)
cat("Chosen threshold for extreme temperatures:", threshold, "°C\n")
```

```
Chosen threshold for extreme temperatures: 24.94444 °C
```

```r
# Decluster the data using the chosen threshold
declustered_data <- decluster(summer_data$AvgTemperature, threshold = threshold, run.length = 1)

# Update the data frame to include only declustered values (setting non-extreme values to NA for
summer_data$Declustered <- ifelse(summer_data$AvgTemperature >= threshold, declustered_data, NA)
```



The declustered data highlights extreme temperature events (above 24.94°C) as red dots. These events occur sporadically, with occasional clusters, indicating that while extremes can group together, they are generally distinct. The threshold effectively identifies significant temperature spikes without over-clustering.

d) Fit a GPD

Question

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

```
# Step 1: Fit the GPD model to the raw data (data above the chosen threshold)
raw_data <- summer_data$AvgTemperature[summer_data$AvgTemperature >= threshold]  # Extreme values
fit_raw <- fpot(raw_data, threshold = threshold)  # Fit the GPD to the raw data

# Step 2: Fit the GPD model to the declustered data (data above the chosen threshold)
declustered_data <- summer_data$Declustered[!is.na(summer_data$Declustered)]  # Declustered extre
fit_declustered <- fpot(declustered_data, threshold = threshold)  # Fit the GPD to the declustere

# Step 3: Extract GPD model parameters for both raw and declustered data
shape_raw <- fit_raw$estimate["shape"]  # Shape parameter (xi) for raw data
scale_raw <- fit_raw$estimate["scale"]  # Scale parameter (sigma) for raw data

shape_declustered <- fit_declustered$estimate["shape"]  # Shape parameter (xi) for declustered da
scale_declustered <- fit_declustered$estimate["scale"]  # Scale parameter (sigma) for declustered

# Step 4: Define a function to calculate the return level using the GPD formula
return_level <- function(shape, scale, threshold, return_period) {
  return(threshold + (scale / shape) * (((return_period * length(raw_data) / length(summer_data$A
}
```

```r
# Step 5: Calculate the 10-year return level for both raw and declustered data
return_period <- 10  # 10-year return level

# 10-year return level for raw data
return_level_raw <- return_level(shape_raw, scale_raw, threshold, return_period)

# 10-year return level for declustered data
return_level_declustered <- return_level(shape_declustered, scale_declustered, threshold, return_

# Step 6: Print the results for both models
cat("10-year return level for raw data:", return_level_raw, "°C\n")
```

```
10-year return level for raw data: 23.69478 °C
```

```r
cat("10-year return level for declustered data:", return_level_declustered, "°C\n")
```

```
10-year return level for declustered data: 23.523 °C
```

In this analysis, we fitted a Generalized Pareto Distribution (GPD) to both the raw and declustered temperature data for the summer months in Geneva. The 10-year return level for the raw data is 23.69°C, while for the declustered data it is slightly lower at 23.52°C. The declustered model, which accounts for isolated extreme events without clustering, provides a slightly more conservative estimate of extreme temperatures compared to the raw model.

Practical 3: Heat wave in Switzerland

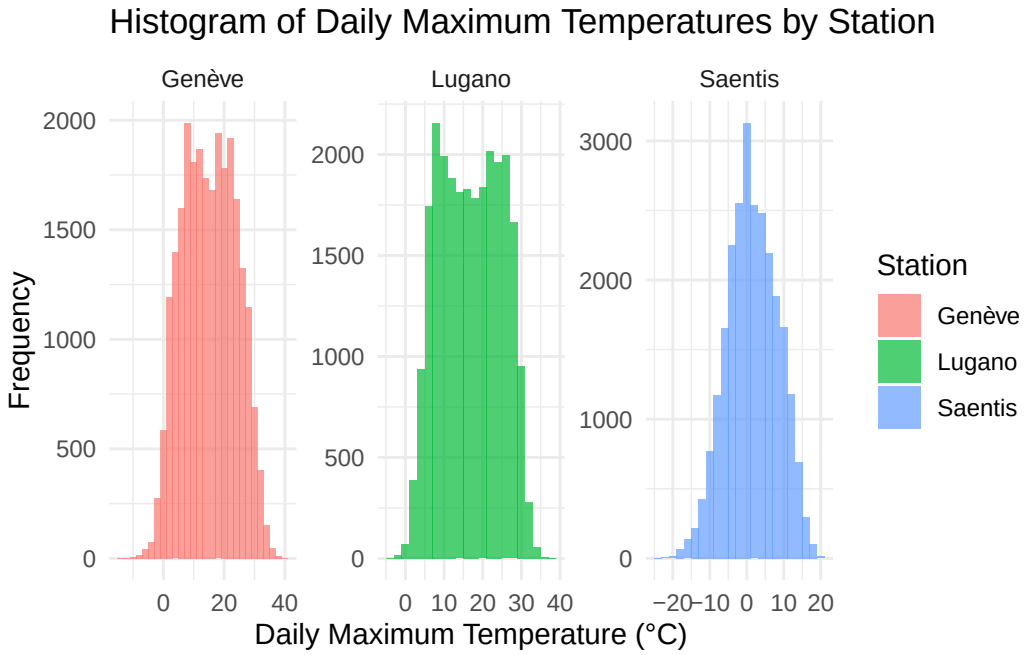
Part 1: Extreme Temperature Events Analysis

Question

- What are the trends and patterns of extreme heat events in Switzerland over the past century?
- How has the frequency of heatwaves changed over time, and can we model future occurrences?

Draw an histogram of the daily maximum temperatures values.

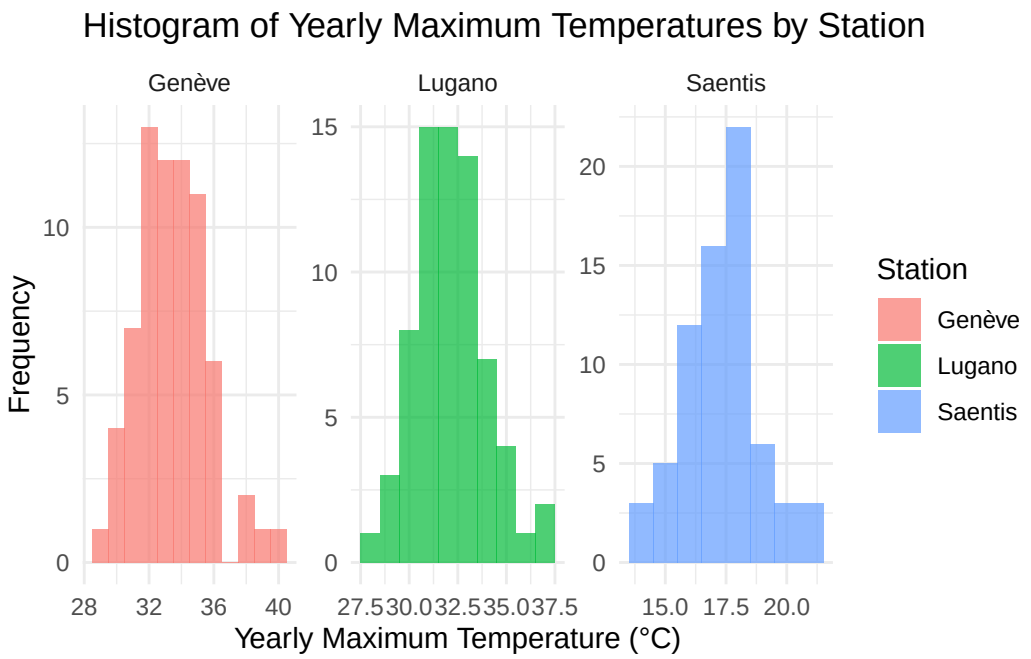
```
# Filter out rows with non-finite values in TMAX
data_filtered <- data %>%
  filter(!is.na(TMAX))  # Use the correct column name for maximum temperature
```



The histograms of daily maximum temperatures reveal clear climatic differences between the three stations. Geneva has a temperate climate, with most temperatures between 0°C and 30°C. Lugano has warmer conditions, reflecting its southern location. Säntis, a high-altitude resort, has much colder temperatures, ranging from -20°C to 20°C, with a peak around 0°C. These differences highlight the influence of altitude and geography on temperature distribution.

Extract the yearly maximum values for each station and draw their histogram:

```
# Extract yearly maximum values for each station
yearly_max <- data %>%
  group_by(NAME, Year = format(Date, "%Y")) %>%
  summarise(Yearly_Max = max(TMAX, na.rm = TRUE), .groups = "drop")
```



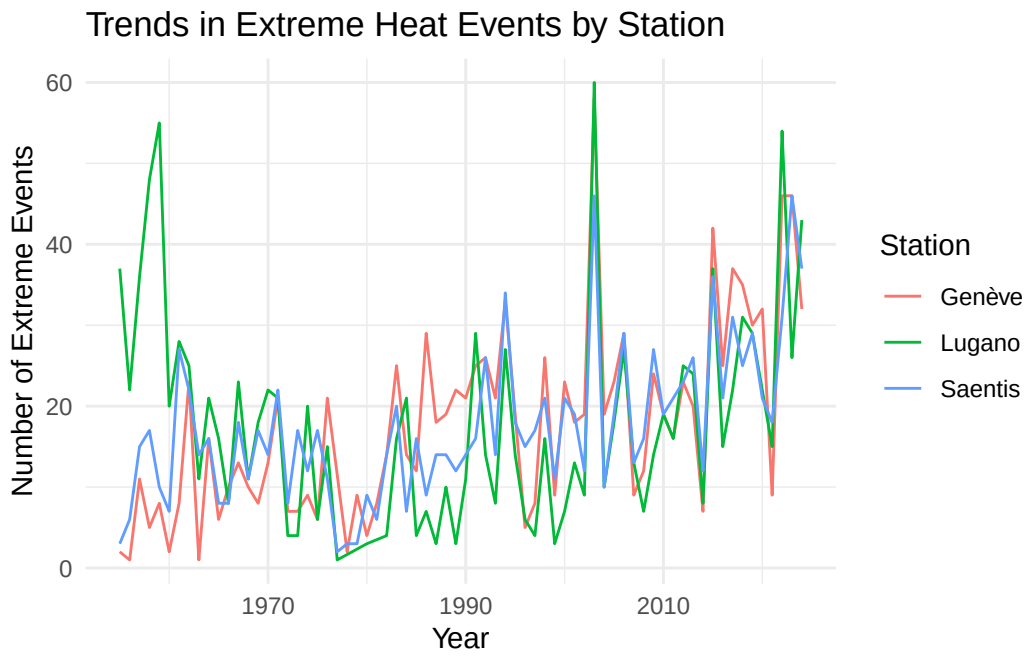
Analyse trends and patterns of extreme heat events for each station

To analyse the trends in extreme heat events for each station, I first calculated the 95th percentile of maximum temperatures (TMAX) for each station to define the threshold for extreme heat events. Next, I counted the number of extreme heat events for each station and year, grouping the data by station name and year. Finally, I visualised the trends by plotting the number of extreme heat events over time for each station.

```
# Calculate 95th percentile thresholds for each station
extreme_thresholds <- data %>%
  group_by(NAME) %>%
  summarise(Threshold_95 = quantile(TMAX, 0.95, na.rm = TRUE))

# Identify extreme heat events per station
data_extreme <- data %>%
  inner_join(extreme_thresholds, by = "NAME") %>%
  mutate(Extreme = TMAX > Threshold_95)

# Count the total number of extreme events per station
extreme_event_counts <- data_extreme %>%
  filter(Extreme) %>%
  group_by(NAME) %>%
  summarise(Total_Extreme_Events = n(), .groups = "drop")

# Print the total number of extreme events per station
print(extreme_event_counts)
```

```
# A tibble: 3 x 2
  NAME    Total_Extreme_Events
  <chr>                  <int>
1 Genève                  1230
2 Lugano                  1239
3 Saentis                 1223
```

```r
# Count extreme events by year and station
extreme_counts_by_year <- data_extreme %>%
  filter(Extreme) %>%
  mutate(Year = format(Date, "%Y")) %>%
  group_by(NAME, Year) %>%
  summarise(Count = n(), .groups = "drop")
```



The graph shows the trends in extreme heat events for the three stations: Geneva, Lugano and Säntis. All stations exhibit fluctuations, with notable peaks around 2003, 2015 and 2022-2023. The total number of extreme heat events above the 95th percentile threshold for each station shows similar magnitudes. Lugano has the highest count of extreme events (1,239), closely followed by Geneva (1,230), with Säntis having the fewest (1,223). This similarity may indicate that the frequency of extreme heat events is comparable across the stations despite their varying geographical and climatic characteristics. However, differences in the thresholds for each station and regional climatic patterns should be considered. For instance, Geneva and Lugano experience higher maximum temperatures, which suggest more frequent heat extremes due to their lower altitudes and warmer climates. On the other hand, Säntis has a significantly lower threshold, which indicate that extreme events are relative to the local climate conditions.

```r
# Merge station names into the thresholds data using 'NAME' column
thresholds_with_names <- extreme_thresholds %>%
  left_join(data %>% select(NAME) %>% distinct(), by = "NAME")

# Display the thresholds with station names
thresholds_with_names %>%
  arrange(desc(Threshold_95)) %>%
  print()
```

```
# A tibble: 3 x 2
  NAME    Threshold_95
  <chr>          <dbl>
1 Genève          29.2
2 Lugano          29.1
```

```
3 Saentis         12.7
```

The thresholds: Geneva: 29.2ºC, Lugano: 29,1ºC, and Säntis 12,7ºC.

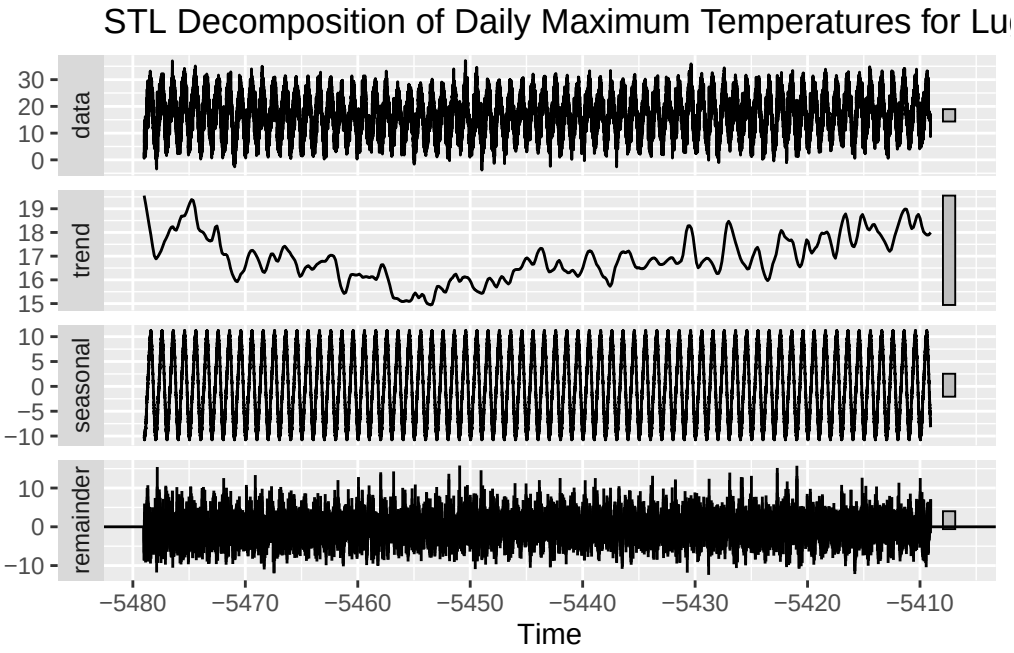
In this part, we'll analyze daily maximum temperatures by applying STL decomposition to separate the data into trend, seasonal, and irregular components. We will then use the trend component to forecast future temperatures for the next 10 years using an ARIMA model.

Yearly maximum temperature trends and predictions for the next 10 years for Lugano.

```r
# Filter the data for Lugano
lugano_data <- data %>%
  filter(NAME == "Lugano")

# Check for missing values
sum(is.na(lugano_data$TMAX))
```

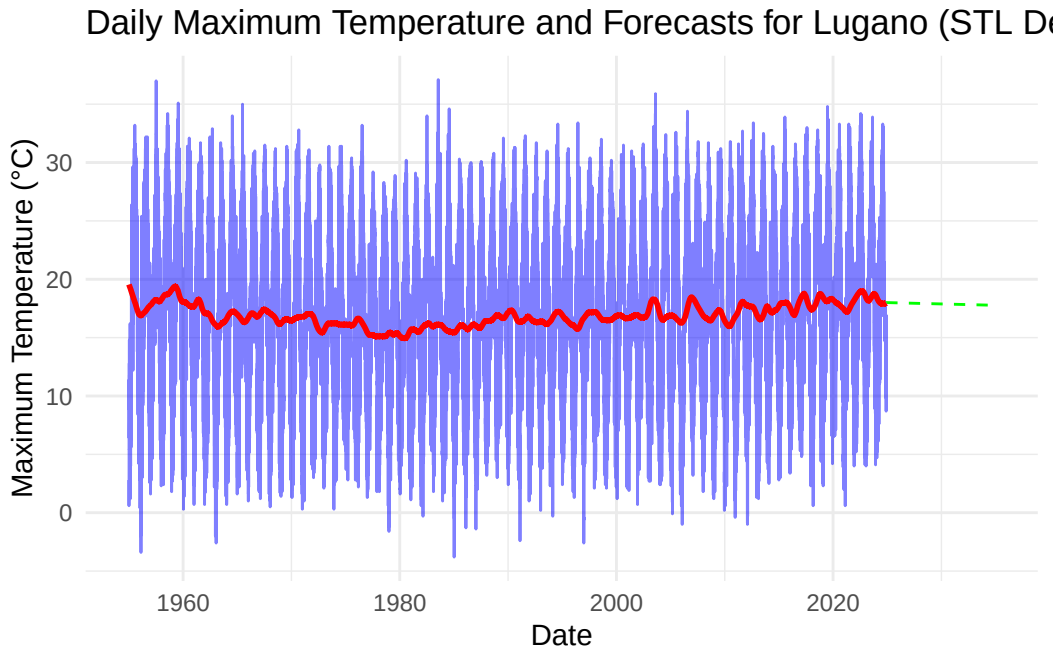
```
[1] 135
```

```r
# Replace NAs with the average temperature
lugano_data$TMAX[is.na(lugano_data$TMAX)] <- mean(lugano_data$TMAX, na.rm = TRUE)

# Create a time series of daily maximum temperatures
ts_daily_max <- ts(lugano_data$TMAX, frequency = 365, start = c(min(lugano_data$Date), 1))

# Apply STL decomposition (with seasonality)
stl_decomp <- stl(ts_daily_max, s.window = "periodic")

# Extract the trend component from the STL decomposition
trend_component <- stl_decomp$time.series[, "trend"]
```



```r
# Apply an ARIMA model to the trend component
arima_model <- auto.arima(trend_component, seasonal = FALSE, stepwise = TRUE)

# Forecast the next 10 years (365 days per year)
forecast_steps <- 365 * 10  # Forecast for 10 years
arima_forecast <- forecast(arima_model, h = forecast_steps)

# Extract the forecasted values
forecast_values <- data.frame(Year = seq(max(lugano_data$Date) + 1, by = "day", length.out = fore
                              Forecast_Max = as.numeric(arima_forecast$mean))
```

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```



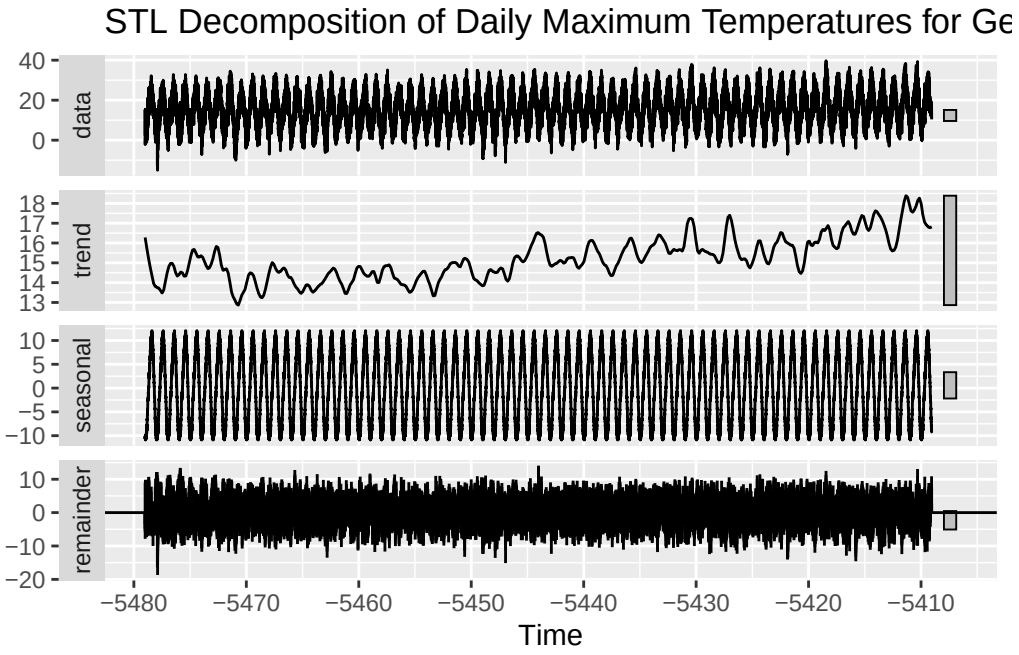
The analysis of daily maximum temperatures for Lugano shows a small decrease over time, as seen in the red trend line from the STL decomposition. While the drop is not drastic, there is a slight downward trend. The forecast, based on the ARIMA model applied to the trend, suggests that this trend will continue with minimal changes over the next 10 years, indicating a period of stability in temperature patterns for Lugano.

Yearly maximum temperature trends and predictions for the next 10 years for Geneva.

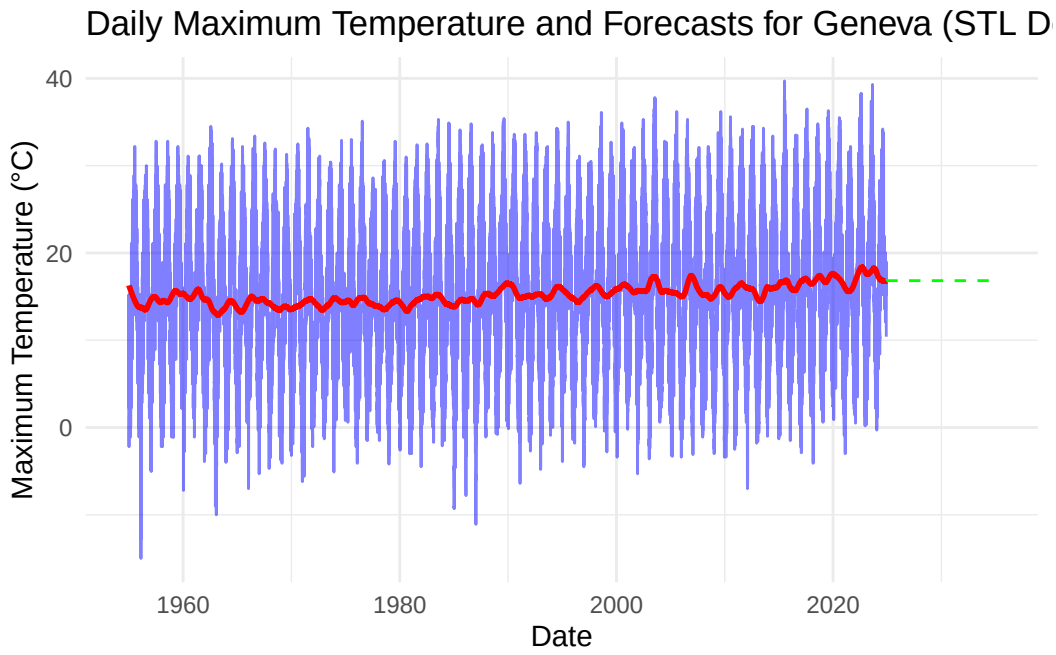
```r
# Filter the data for Geneva
geneva_data <- data %>%
  filter(NAME == "Genève")

# Check for missing values
sum(is.na(geneva_data$TMAX))
```

```
[1] 197
```

```r
# Replace NAs with the average temperature
geneva_data$TMAX[is.na(geneva_data$TMAX)] <- mean(geneva_data$TMAX, na.rm = TRUE)

# Create a time series of daily maximum temperatures
ts_daily_max_geneva <- ts(geneva_data$TMAX, frequency = 365, start = c(min(geneva_data$Date), 1))

# Apply STL decomposition (with seasonality)
stl_decomp_geneva <- stl(ts_daily_max_geneva, s.window = "periodic")

# Extract the trend component from the STL decomposition
trend_component_geneva <- stl_decomp_geneva$time.series[, "trend"]
```




```r
# Apply an ARIMA model to the trend component
arima_model_geneva <- auto.arima(trend_component_geneva, seasonal = FALSE, stepwise = TRUE)

# Forecast the next 10 years (365 days per year)
forecast_steps <- 365 * 10  # Forecast for 10 years
arima_forecast_geneva <- forecast(arima_model_geneva, h = forecast_steps)

# Extract the forecasted values
forecast_values_geneva <- data.frame(Year = seq(max(geneva_data$Date) + 1, by = "day", length.out
                                     Forecast_Max = as.numeric(arima_forecast_geneva$mean))
```



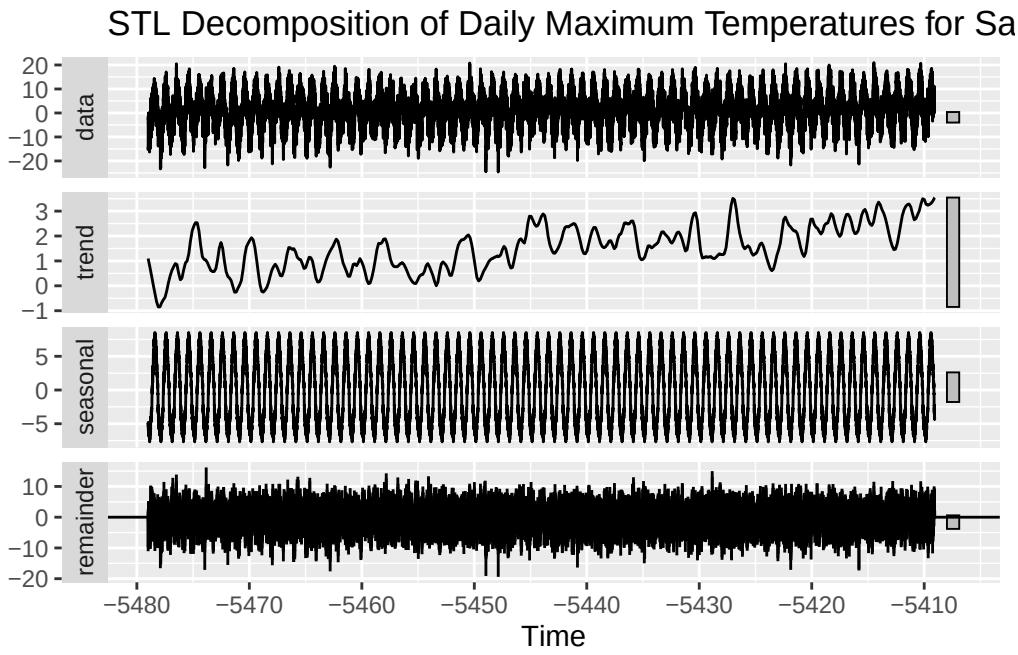
The analysis of daily maximum temperatures for **Geneva** shows a small increase over time, as seen in the red trend line from the STL decomposition. However, this increase is not large, and the temperatures stay fairly stable. The forecast, based on the ARIMA model applied to the trend, suggests that this stable pattern will continue over the next 10 years, with only slight changes, indicating little variation in temperature for Geneva.

Yearly maximum temperature trends and predictions for the next 10 years for Säntis.

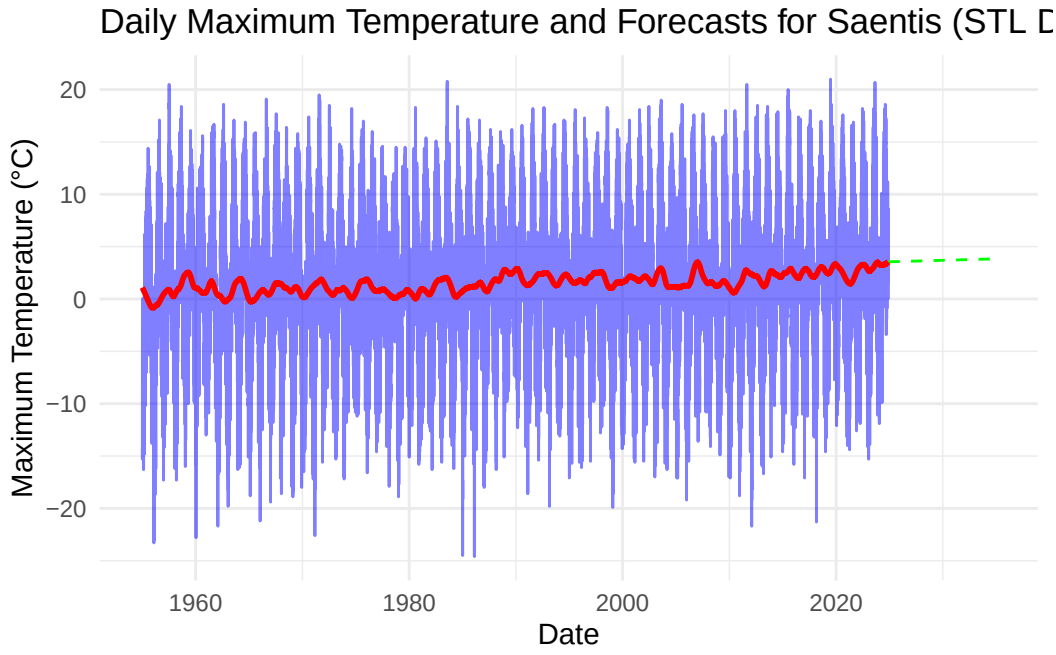
```
# Filter the data for Saentis
saentis_data <- data %>%
  filter(NAME == "Saentis")

# Check for missing values
sum(is.na(saentis_data$TMAX))
```

```
[1] 63
```

```
# Replace NAs with the average temperature
saentis_data$TMAX[is.na(saentis_data$TMAX)] <- mean(saentis_data$TMAX, na.rm = TRUE)

# Create a time series of daily maximum temperatures
ts_daily_max_saentis <- ts(saentis_data$TMAX, frequency = 365, start = c(min(saentis_data$Date),

# Apply STL decomposition (with seasonality)
stl_decomp_saentis <- stl(ts_daily_max_saentis, s.window = "periodic")

# Extract the trend component from the STL decomposition
trend_component_saentis <- stl_decomp_saentis$time.series[, "trend"]
```




```
# Apply an ARIMA model to the trend component
arima_model_saentis <- auto.arima(trend_component_saentis, seasonal = FALSE, stepwise = TRUE)

# Forecast the next 10 years (365 days per year)
forecast_steps <- 365 * 10  # Forecast for 10 years
arima_forecast_saentis <- forecast(arima_model_saentis, h = forecast_steps)

# Extract the forecasted values
forecast_values_saentis <- data.frame(Year = seq(max(saentis_data$Date) + 1, by = "day", length.o
                                      Forecast_Max = as.numeric(arima_forecast_saentis$mean))
```



The analysis of daily maximum temperatures for Säntis shows a slight upward trend, as indicated by the red trend line from the STL decomposition. While the increase is modest, it suggests a gradual rise in temperatures

over time. The forecast, based on the ARIMA model applied to the trend component, predicts that this upward trend will continue over the next 10 years, with minimal change, indicating a period of slow warming for Saentis.

Part 2: Seasonal and Clustering Analysis

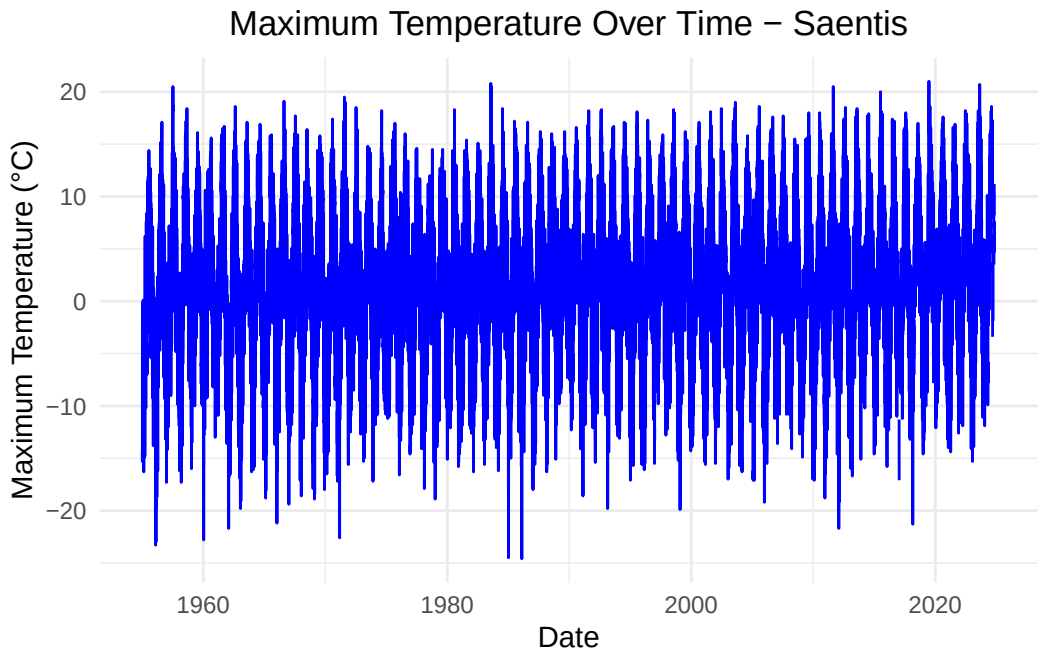
Question

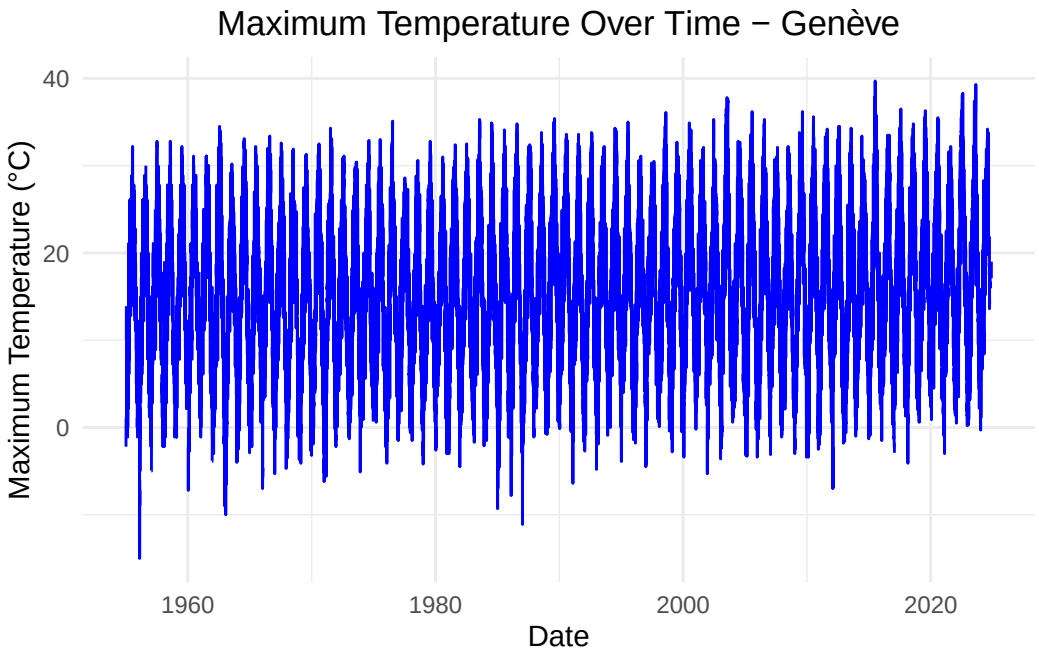
- Do extreme temperature events show clustering behavior during specific seasons (e.g., summer months)?
- What is the extreme index for temperature data in Switzerland, and how does it inform the clustering of extreme events?

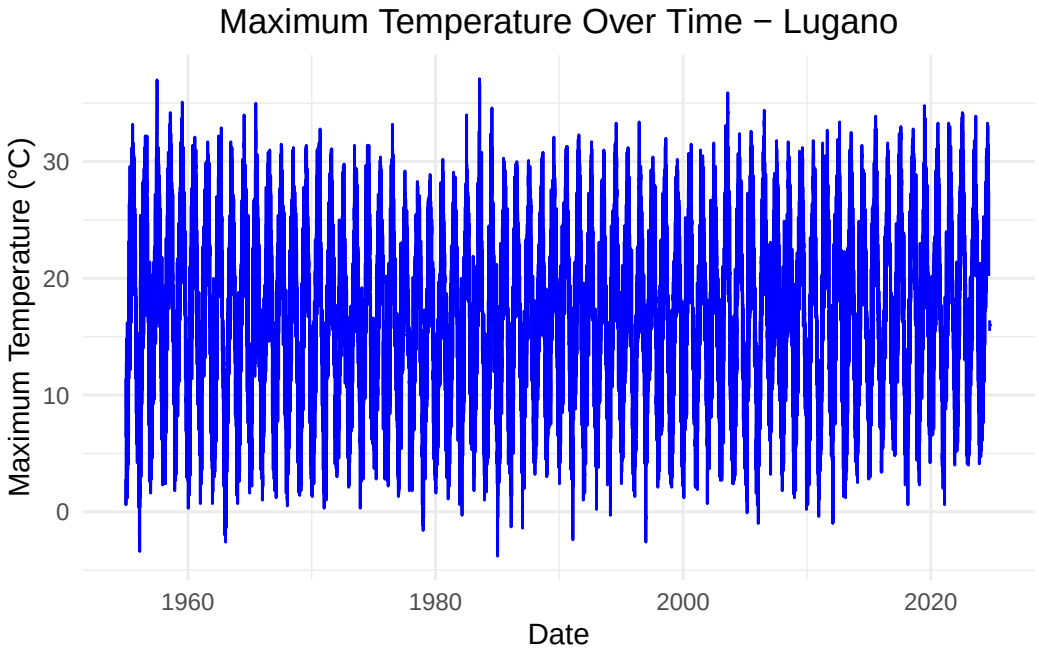
First we will explore the data and visualize the maximum temperatures over time for each station.

```
# Convert the date column to Date format
stations_data$DATE <- as.Date(stations_data$DATE, format = "%Y-%m-%d")

# Filter the data starting from the date "1955-01-01"
stations_data <- subset(stations_data, DATE >= as.Date("1955-01-01"))
```

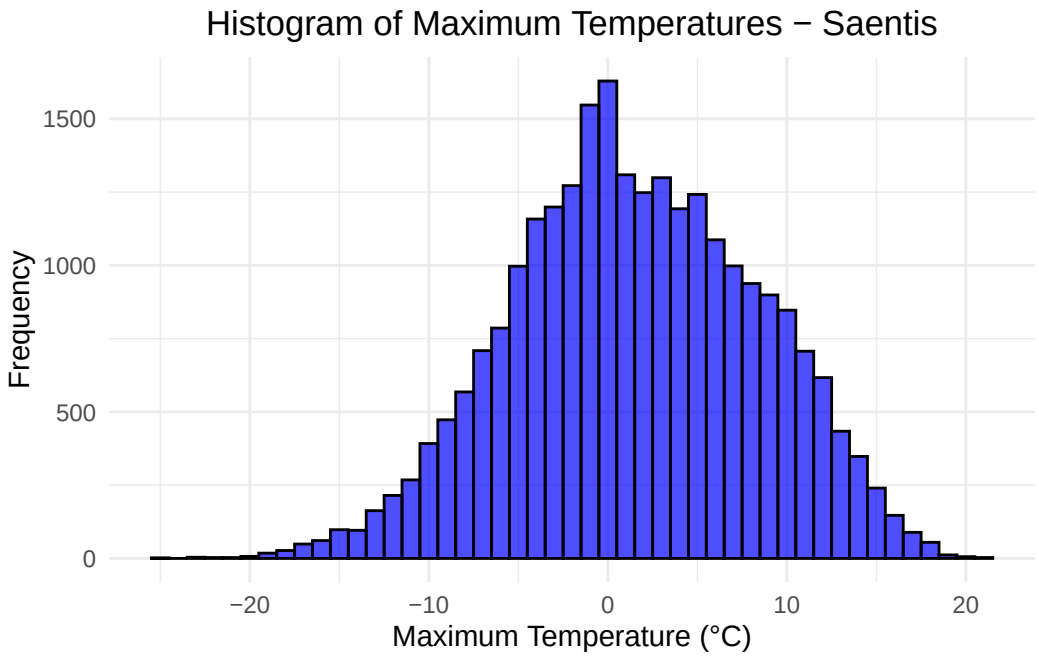
We can then create plots to visualize the maximum temperatures over time for each station.

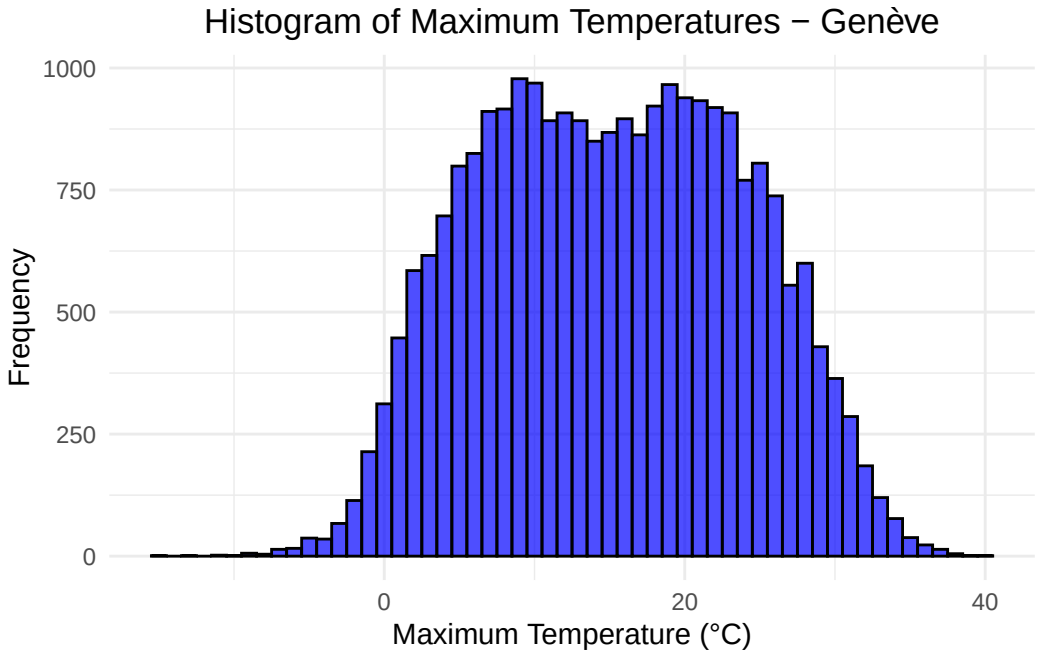


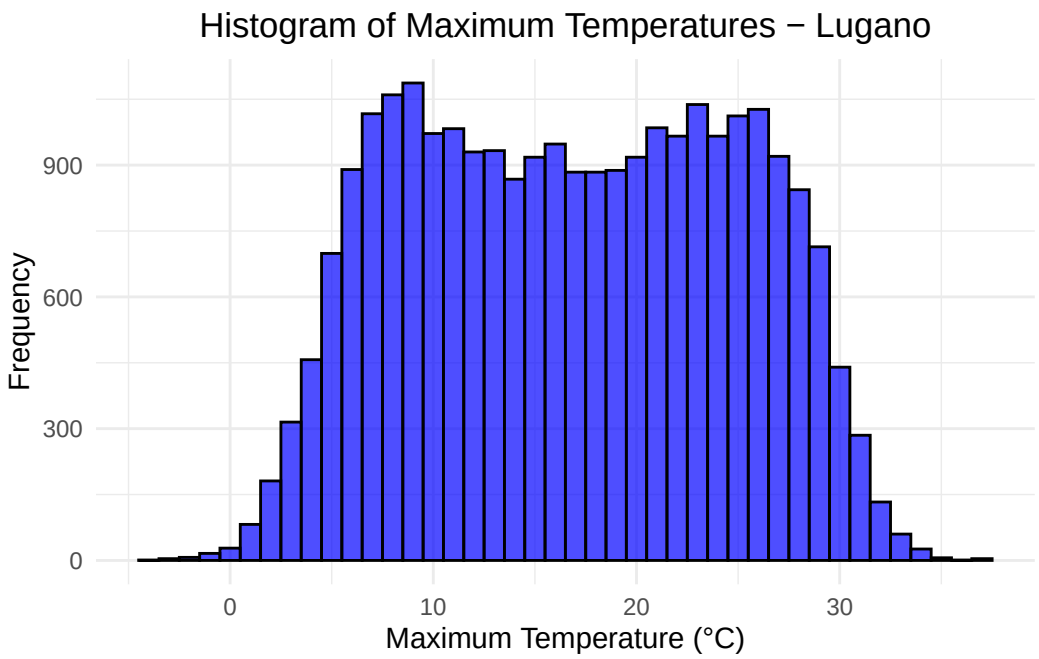




We can also create histograms to visualize the distribution of maximum temperatures for each station.


Histogram of Maximum Temperatures − Saentis
Frequency
1500
1000
500
0
−20
−10
0
10
20
Maximum Temperature (°C)


Histogram of Maximum Temperatures − Genève
Frequency
1000
750
500
250
0
0
20
40
Maximum Temperature (°C)



This part clean the data by removing rows with missing values for the TMAX column and then split the data by station.

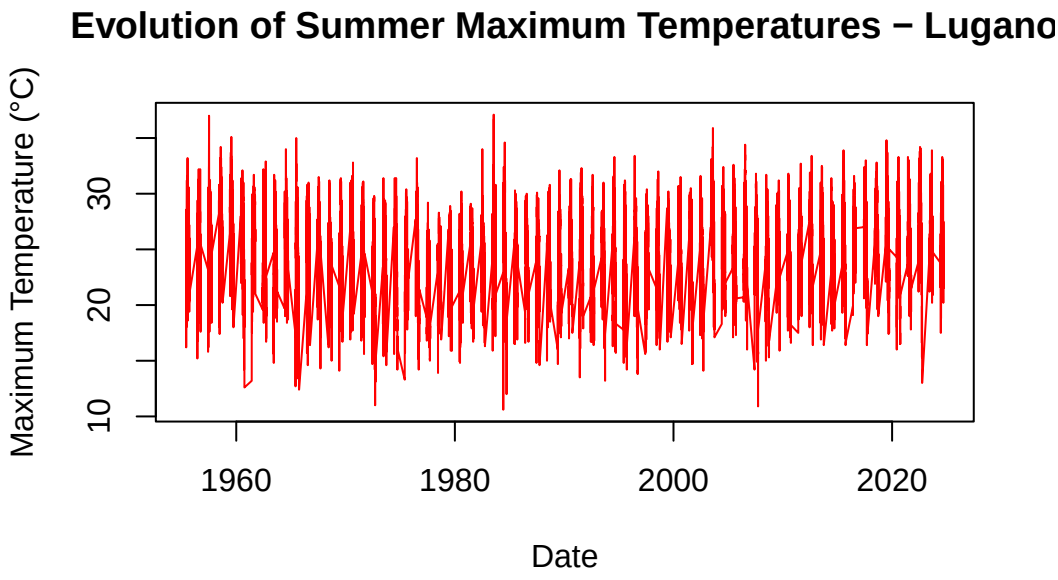
```r
# Remove only rows where TMAX is NA
stations_data <- stations_data[!is.na(stations_data$TMAX), ]

# Split the data by station
lugano_data <- subset(stations_data, NAME == "Lugano")
genève_data <- subset(stations_data, NAME == "Genève")
saentis_data <- subset(stations_data, NAME == "Saentis")

# Number of rows for each station
rows_l <- nrow(lugano_data)
rows_g <- nrow(genève_data)
rows_s <- nrow(saentis_data)
```

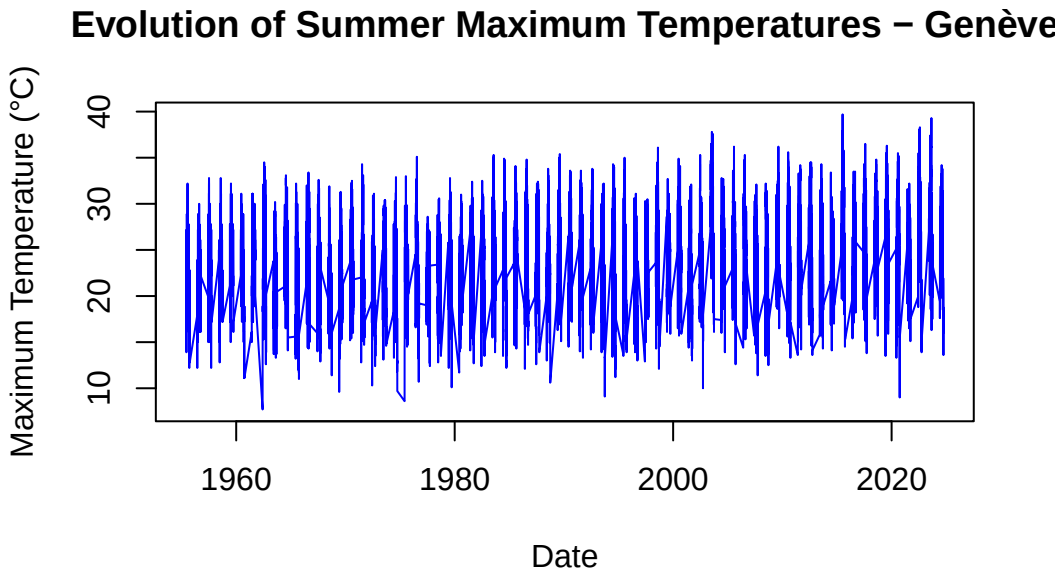
We see that we have 25397 rows for Lugano, 25334 rows for Genève, and 25462 rows for Säntis.

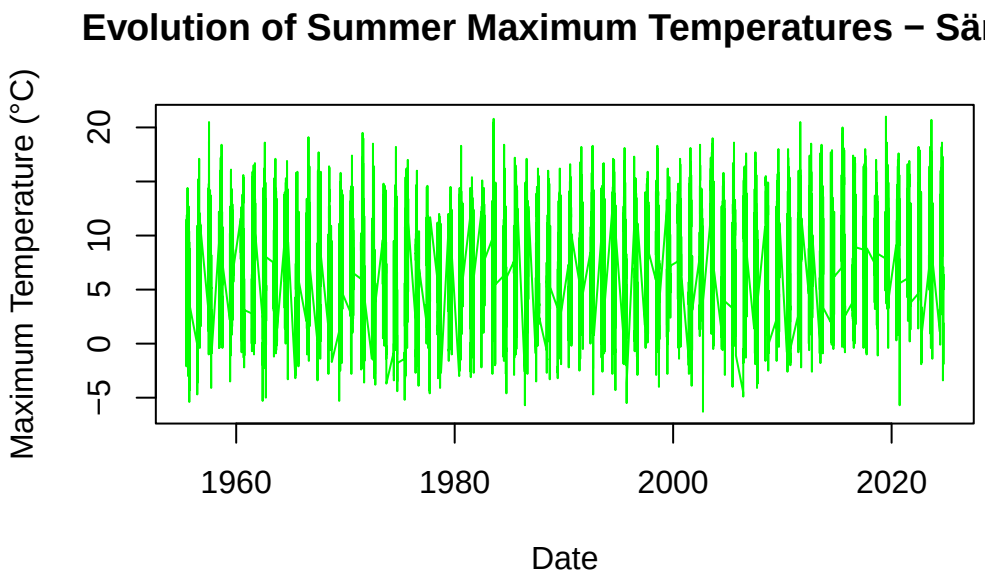
As we are interested at heatwaves, we'll only focus on the summer period. We'll filter the data for the months of June, July, August, and September.

```r
# Filter summer data per station
lugano_summer <- subset(lugano_data, format(DATE, "%m") %in% c("06", "07", "08", "09"))
genève_summer <- subset(genève_data, format(DATE, "%m") %in% c("06", "07", "08", "09"))
saentis_summer <- subset(saentis_data, format(DATE, "%m") %in% c("06", "07", "08", "09"))
```

Let's visualize the evolution of summer maximum temperatures for each station.


Evolution of Summer Maximum Temperatures − Lugano
Maximum Temperature (°C)
10
20
30
1960
1980
2000
2020
Date


Evolution of Summer Maximum Temperatures − Genève
Maximum Temperature (°C)
10
20
30
40
1960
1980
2000
2020
Date

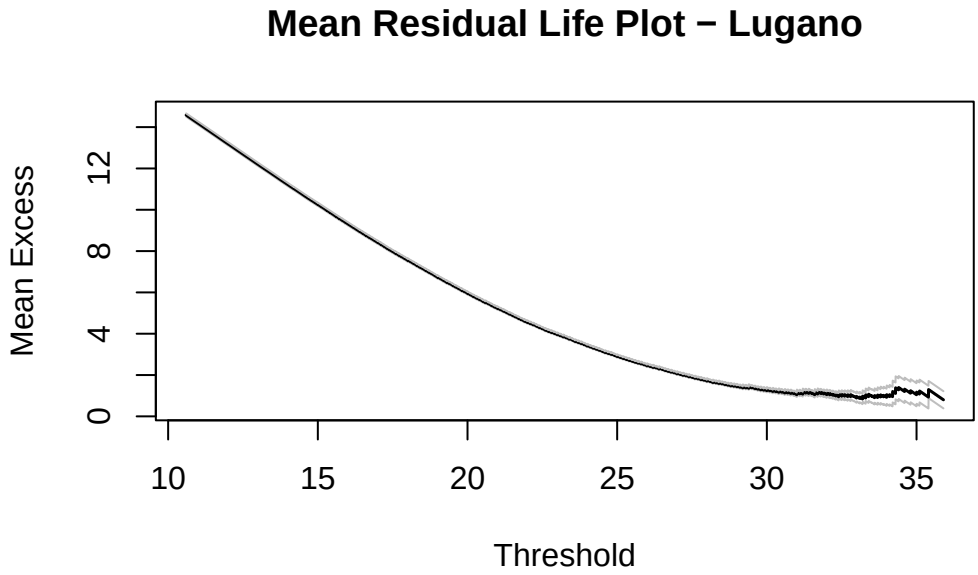


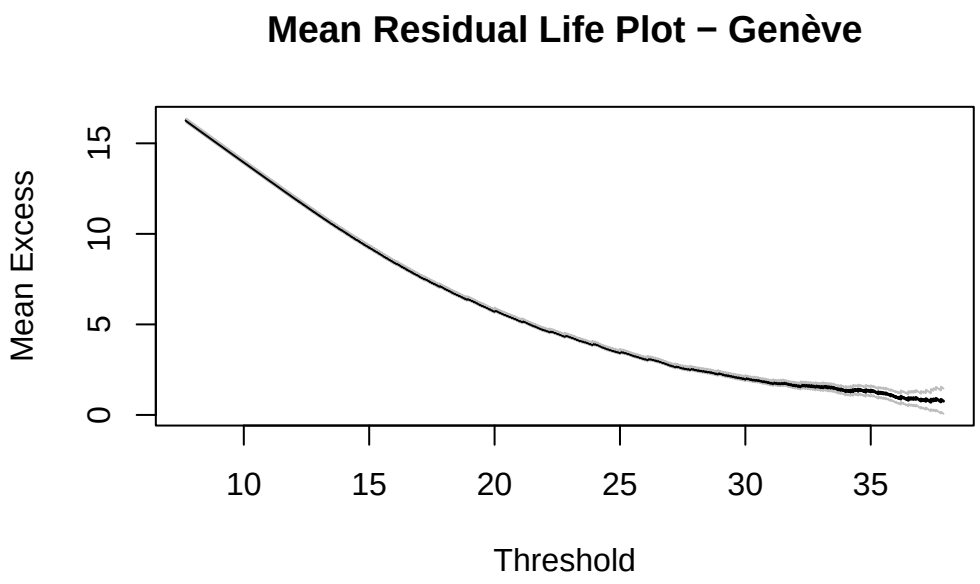
Cluster analysis

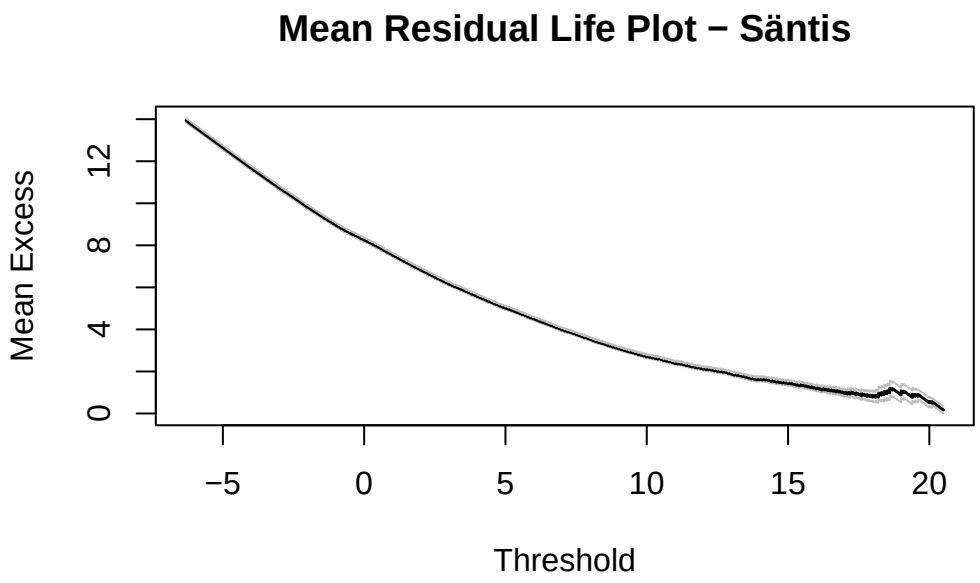
We can now focus on the clustering analysis of extreme temperature events. We will define a threshold for extreme temperatures based on the 95th percentile of summer maximum temperatures for each station.

```
# Calculate thresholds for each station
threshold_lugano <- quantile(lugano_summer$TMAX, 0.95, na.rm = TRUE)
threshold_genève <- quantile(genève_summer$TMAX, 0.95, na.rm = TRUE)
threshold_saentis <- quantile(saentis_summer$TMAX, 0.95, na.rm = TRUE)
```

The thresholds for Lugano is 30.8, for Genève is 31.8, and for Säntis is 15. We can now validate with Mean Residual Life (MRL) plots this results.







The Mean Residual Life (MRL) plot is used to validate the choice of thresholds for defining extreme events. It shows the mean excess, or the average amount by which temperatures exceed a threshold, as the threshold increases. For Geneva for example, in the plot, the curve decreases steeply at lower thresholds, indicating the inclusion of non-extreme values. Around 31°C, the curve stabilizes, suggesting this is a suitable threshold for defining extremes. The stability and narrow confidence bands confirm the reliability of this threshold. At higher thresholds, wider confidence intervals indicate less reliable estimates, further supporting the choice of 31°C as appropriate for the analysis.

```
# Identify extreme days for Lugano
lugano_extreme_days <- lugano_summer[lugano_summer$TMAX > threshold_lugano, ]

# Identify extreme days for Genève
geneve_extreme_days <- genève_summer[genève_summer$TMAX > threshold_genève, ]
```

```r
# Identify extreme days for Säntis
saentis_extreme_days <- saentis_summer[saentis_summer$TMAX > threshold_saentis, ]
```

The number of extreme days above the chosen threshold provides a measure of the frequency of extreme events at each station. For example, Lugano has 406 extreme days, Geneva has 413, and Säntis has 413. These counts indicate the occurrence of extreme temperatures during the summer months, with Säntis experiencing the highest number of extreme days among the three stations.

```r
# Calculation of the extremal index for each station
extremal_index_lugano <- extremalindex(lugano_summer$TMAX, threshold = threshold_lugano)
extremal_index_geneve <- extremalindex(genève_summer$TMAX, threshold = threshold_genève)
extremal_index_saentis <- extremalindex(saentis_summer$TMAX, threshold = threshold_saentis)

cat("Extremal index for Lugano:", extremal_index_lugano, "\n")
```

```
Extremal index for Lugano: 0.2007176 80 17
```

```r
cat("Extremal index for Genève:", extremal_index_geneve, "\n")
```

```
Extremal index for Genève: 0.2699455 112 8
```

```r
cat("Extremal index for Säntis:", extremal_index_saentis, "\n")
```

```
Extremal index for Säntis: 0.3572708 148 8
```

These results represent the extremal index calculations for each station, which measure the tendency of extreme events to occur in clusters. Here's what the numbers mean:

Extremal Index (First Number): For Lugano: 0.2, for Geneva 0.27, and for Säntis 0.36. The extremal index ranges between 0 and 1. A value close to 0 indicates that extreme events are strongly clustered. A value closer to 1 suggests that extreme events are more isolated and independent. In this case, Lugano shows the most clustering of extremes (lowest index), while Säntis has more independent extreme events (highest index).

Number of Clusters (Second Number): For Lugano, we get 80, for Geneva 112, and for Säntis 148. This indicates the number of distinct clusters of extreme events detected for each station. More clusters suggest a higher frequency of extremes occurring over the studied period. This indicates the number of distinct clusters of extreme events detected for each station. More clusters suggest a higher frequency of extremes occurring over the studied period.

Max Run Length (Third Number): For Lugano: 17, for Geneva 8, and for Säntis 8. This indicates the maximum run length of a cluster, which is the largest number of consecutive days within one cluster where extreme events occurred.

Interpretation: Lugano has the strongest clustering (lowest extremal index) and also experiences longer extreme event clusters (17 days). This suggests that once an extreme event starts in Lugano, it is more likely to persist for a prolonged period. Genève has moderate clustering and more clusters (112), but the clusters are shorter (8 days). This reflects a higher frequency of extremes but with shorter durations. Säntis has the weakest clustering (highest extremal index) and the most clusters (148), but like Genève, the clusters have a maximum duration of 8 days. This suggests that extreme events in Säntis are relatively isolated and occur more

independently. Overall, these results show differences in the frequency, persistence, and clustering of extreme temperature events across the three stations.

We can now proceed to decluster the data to remove dependencies between extreme events.

```
# Decluster the data using the chosen threshold for each station
lugano_declustered <- decluster(lugano_summer$TMAX, threshold = threshold_lugano)
genève_declustered <- decluster(genève_summer$TMAX, threshold = threshold_genève)
saentis_declustered <- decluster(saentis_summer$TMAX, threshold = threshold_saentis)

# Add the declustered data to the corresponding datasets
lugano_summer$Declustered <- ifelse(lugano_summer$TMAX >= threshold_lugano, lugano_declustered, N
genève_summer$Declustered <- ifelse(genève_summer$TMAX >= threshold_genève, genève_declustered, N
saentis_summer$Declustered <- ifelse(saentis_summer$TMAX >= threshold_saentis, saentis_declustere
```

Purpose of Declustering: Declustering aims to remove temporal dependence within clusters of extreme values. Extreme temperature events often occur consecutively (e.g., during heatwaves), but statistical models like the Generalized Pareto Distribution (GPD) require independent extreme events. Declustering ensures we retain only the most representative extreme events within a cluster.

Threshold Selection: The process begins with a chosen threshold for each station (30.8, 31.8, 15). Only values exceeding these thresholds are considered as "extreme."

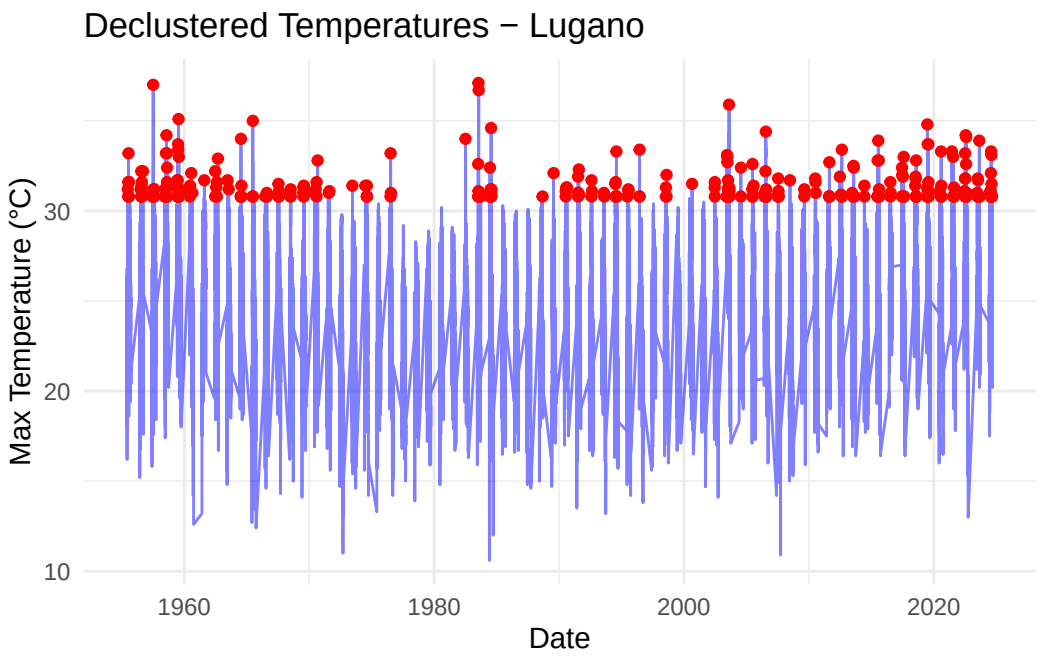
decluster Function:

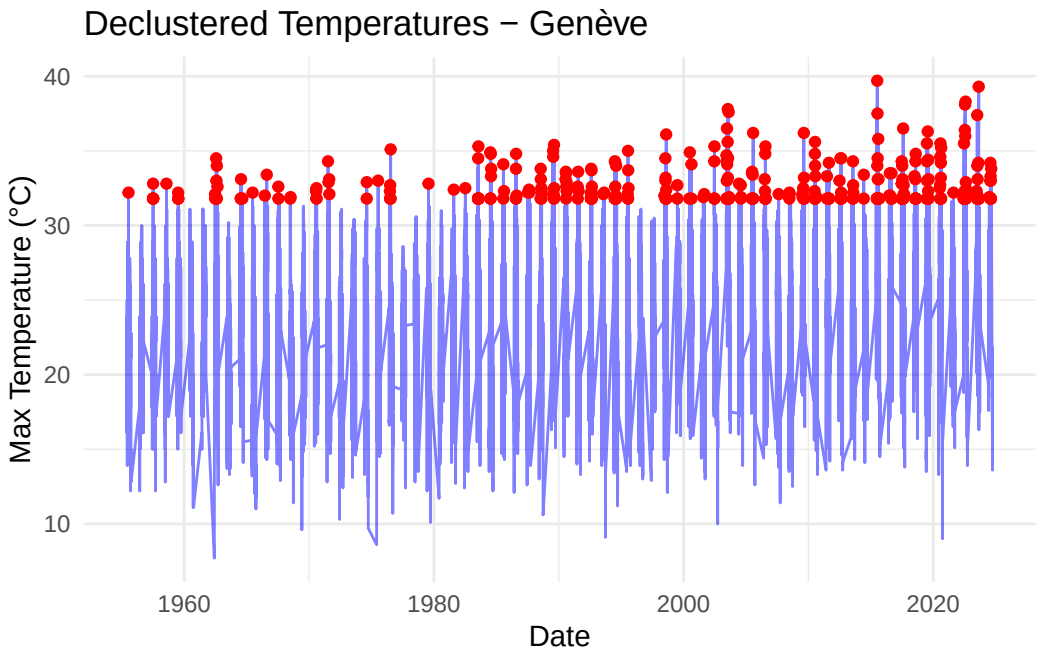
- The decluster function scans through the extreme temperature values and identifies clusters of consecutive exceedances above the threshold.
- The run.length parameter defines the minimum gap (in days) between two clusters. Here, by default, run.length = 1 means that at least one day below the threshold is needed to separate two clusters.
- The function outputs a declustered series where only one representative extreme value (usually the maximum) is retained per cluster.

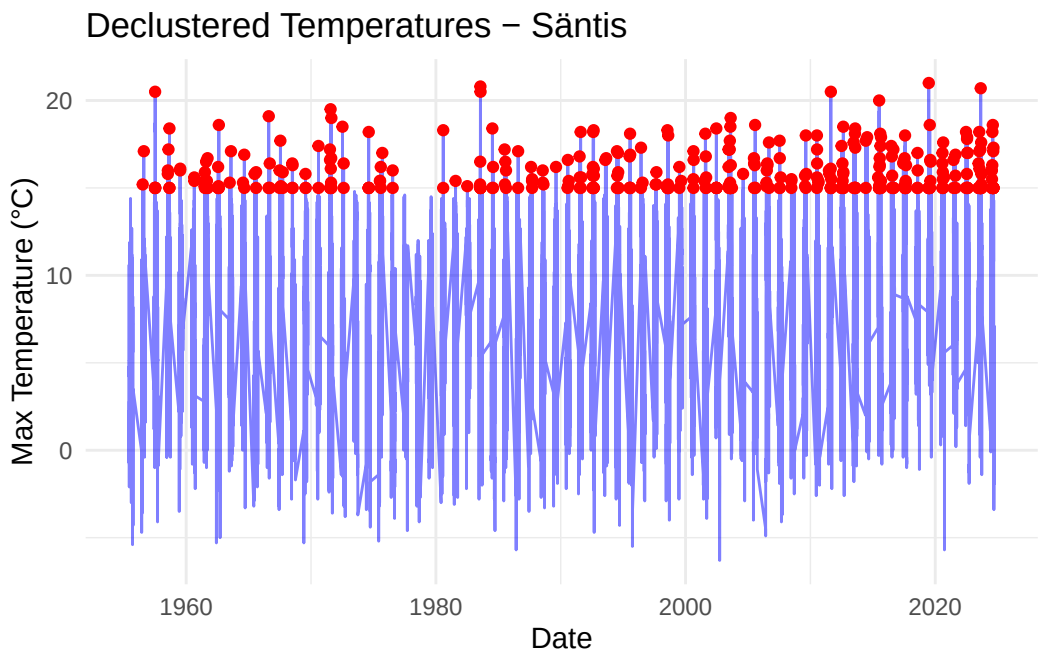
Adding Declustered Data to the Dataset: After declustering, we match the declustered extreme values to their respective dates in the original datasets (lugano_summer, genève_summer, saentis_summer). If a value in the original dataset is part of the declustered series, it is retained in the new column Declustered. Otherwise, it is set to NA.

Outcome:

The datasets now have a Declustered column, which contains only the independent extreme events identified after declustering. This column can be used for further statistical analyses, such as fitting extreme value models (e.g., GPD) or estimating return levels.


Declustered Temperatures − Lugano
Max Temperature (°C)
10
20
30
1960
1980
2000
2020
Date


Declustered Temperatures − Genève
Max Temperature (°C)
10
20
30
40
1960
1980
2000
2020
Date



In this step, we visualize the declustered extreme temperature data alongside the original temperature series for each station (Lugano, Genève, and Säntis).

Purpose:

After applying the declustering process, we want to identify and isolate independent extreme events. The declustered data removes consecutive exceedances above a defined threshold, retaining only one representative extreme temperature per cluster.

What we do:

The blue line (geom_line) represents the original time series of maximum temperatures for the summer months. The red points (geom_point) represent the declustered extreme temperatures, which correspond to independent extreme events above the defined threshold. These declustered values are extracted using the decluster function earlier and are stored in a new column (Declustered).

Why it is useful:

The visual comparison between the original data and the declustered extremes allows us to verify the effectiveness of the declustering process. The declustered points (red dots) highlight when extreme events occur independently, removing short-term dependencies. This cleaned version of the data is crucial for accurate statistical modeling of extreme events, such as fitting a Generalized Pareto Distribution (GPD).

How to interpret:

Overall, the declustered temperature graphs show an increasing trend in the frequency and intensity of extreme events, particularly in Genève, where the rise is most pronounced since the 1980s. At Säntis, the trend is less clear, with frequent but lower-intensity extremes remaining relatively stable over time. For Lugano, the pattern is less distinct, making it harder to confirm a clear upward trend, though some higher peaks appear more frequent in recent decades.

Then, to calculate the probability that an extreme day is followed by another extreme day for each station, we use the extremal index, which measures the degree of clustering of extremes. The extremal index can be interpreted as the inverse of the average cluster size.

```r
# Compute the probability of consecutive extreme days for each station
compute_extreme_following_prob <- function(extremal_index) {
  return(1 - extremal_index) # Probability of consecutive extremes
}

# For Lugano
extremal_index_lugano <- extremalindex(lugano_summer$TMAX, threshold = threshold_lugano)
prob_consecutive_extremes_lugano <- compute_extreme_following_prob(extremal_index_lugano[1])
cat("Probability of consecutive extremes for Lugano:", prob_consecutive_extremes_lugano, "\n")
```

```
Probability of consecutive extremes for Lugano: 0.7992824
```

```r
# For Genève
extremal_index_genève <- extremalindex(genève_summer$TMAX, threshold = threshold_genève)
prob_consecutive_extremes_genève <- compute_extreme_following_prob(extremal_index_genève[1])
cat("Probability of consecutive extremes for Genève:", prob_consecutive_extremes_genève, "\n")
```

```
Probability of consecutive extremes for Genève: 0.7300545
```

```r
# For Saentis
extremal_index_saentis <- extremalindex(saentis_summer$TMAX, threshold = threshold_saentis)
prob_consecutive_extremes_saentis <- compute_extreme_following_prob(extremal_index_saentis[1])
cat("Probability of consecutive extremes for Saentis:", prob_consecutive_extremes_saentis, "\n")
```

```
Probability of consecutive extremes for Saentis: 0.6427292
```

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))

# Santïs
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.

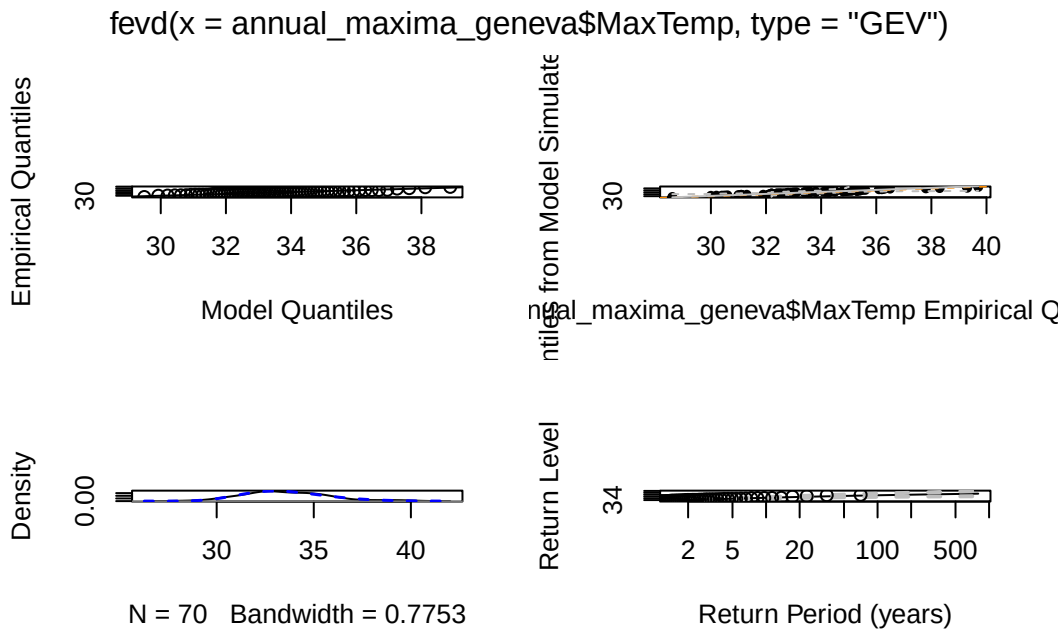
We calculate the annual maximum temperatures for Geneva, Santïs and Lugano and fit a GEV distribution to these maxima.

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

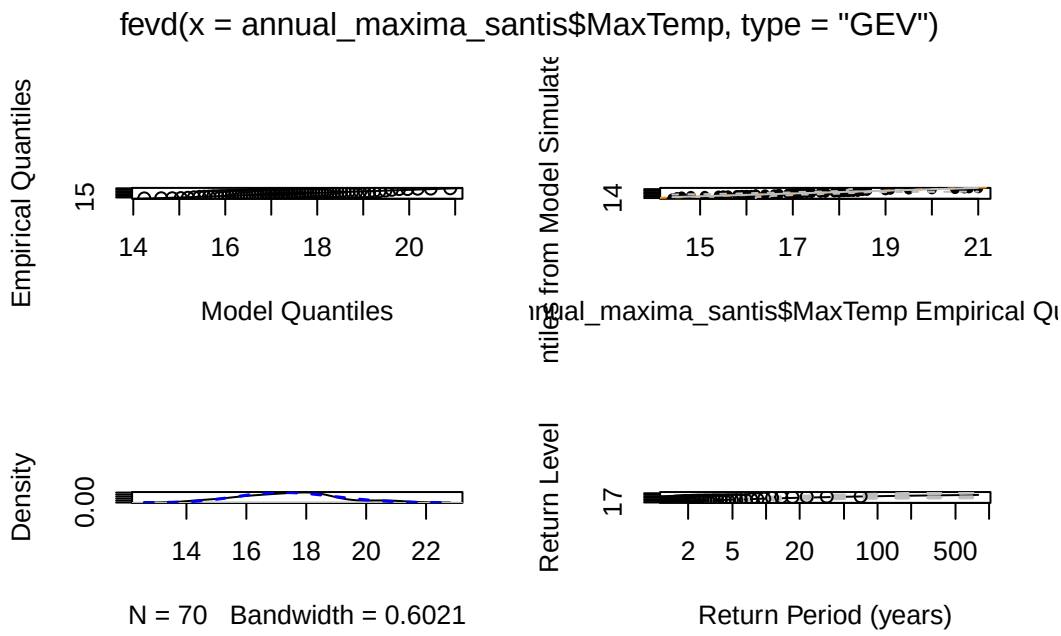
```r
annual_maxima_geneva <- data_geneva %>%
  mutate(Year = year(DATE)) %>%
  group_by(Year) %>%
  summarize(MaxTemp = max(TMAX), .groups = 'drop')

# Santïs
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')
```

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

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

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

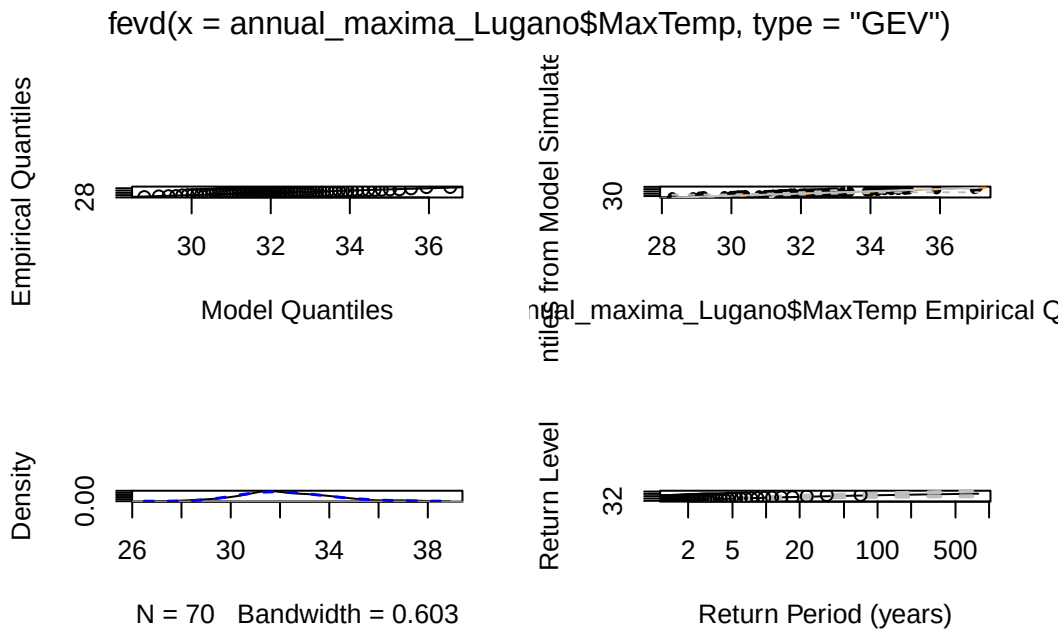
```r
# Diagnostic plots for the GEV fit
# Geneva
par(mfrow = c(2, 2))
plot(gev_fit_geneva)
```



```r
# Santïs
par(mfrow = c(2, 2))
plot(gev_fit_santis)
```




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



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 overal the fit is good.

```r
# 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))

# Santïs
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

```r
# 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"))
```

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

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

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

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

# Santïs
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 Santïs, and 406 for Lugano.

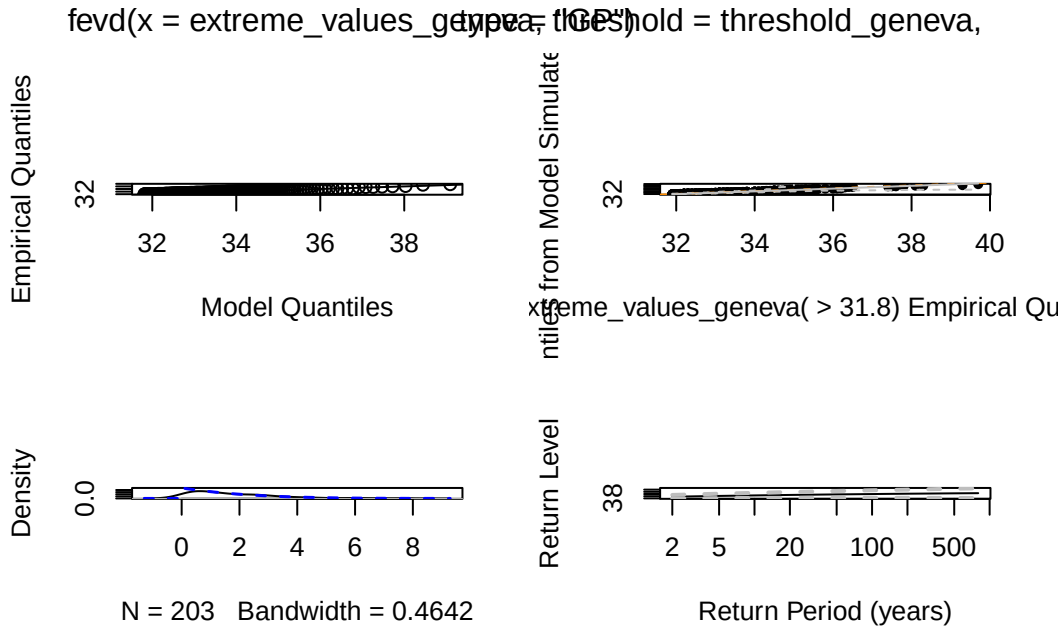
```
# Decluster the data using the chosen threshold for each station
# Genava
geneva_declustered <- extRemes::decluster(geneva_summer$TMAX, threshold = threshold_geneva, run.l

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

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

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

# Santïs
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
```

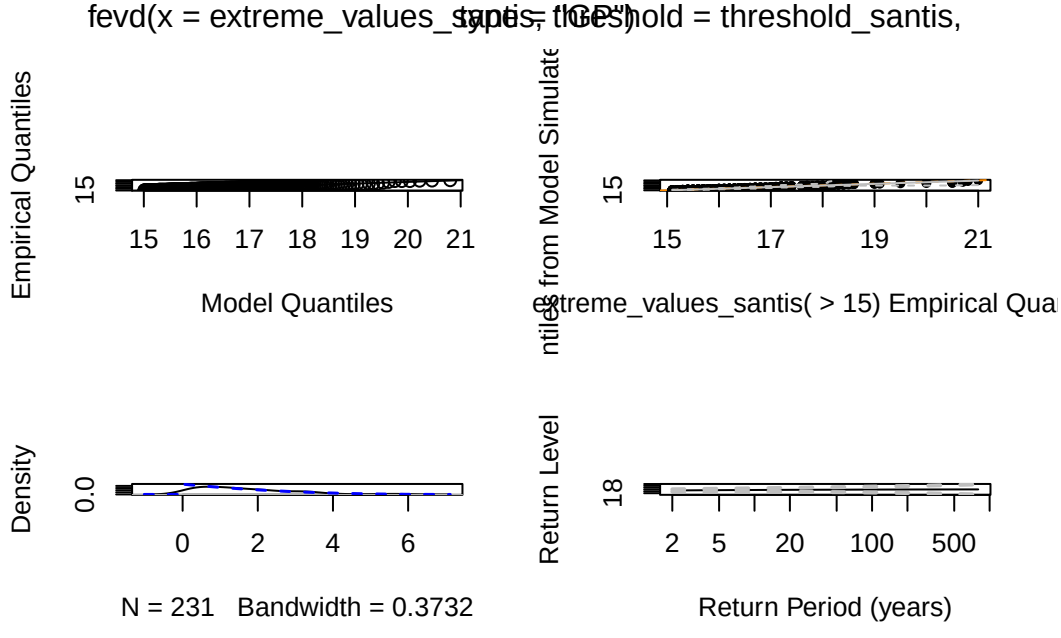
```
# Extract declustered extreme values
# Geneva
extreme_values_geneva <- geneva_summer[["declustered"]][!is.na(geneva_summer[["declustered"]])]

# Santïs
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")

# Santïs
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")
```

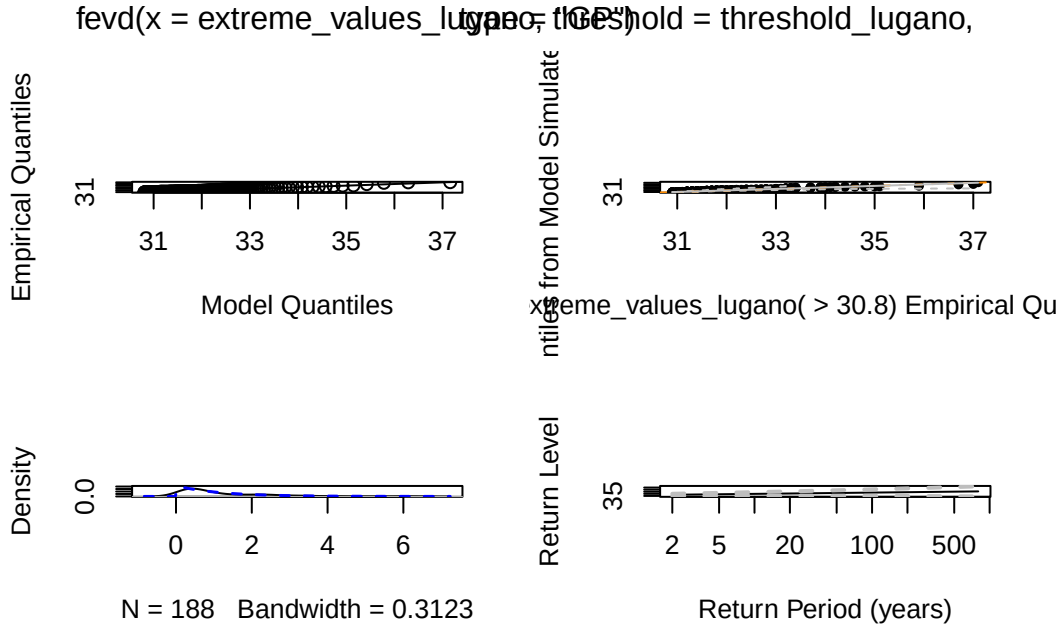
```
# Diagnostic plots for the GPD fit
# Genève
par(mfrow = c(2, 2))
plot(gpd_fit_geneva)
```




```
# Santïs
par(mfrow = c(2, 2))
plot(gpd_fit_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 overal 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))

# Santïs
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
Santïs	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
Santïs	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 Santïs.

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 Santïs.

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
Santïs	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.

Return Period

> Question
>
> What is the return period of extreme temperature events exceeding a specific threshold?

Now, we will analyse the return period for each city. The return period is the average interval between occurrences of extreme temperatures (exceeding the threshold, here at the 95th percentile) at each station. Here, we will test for a threshold set at the 95th percentile, which we can consider as extreme.

```
# Compute the 95th percentile threshold for each city
threshold_geneva <- quantile(annual_maxima_geneva$MaxTemp, 0.95, na.rm = TRUE)
threshold_santis <- quantile(annual_maxima_santis$MaxTemp, 0.95, na.rm = TRUE)
threshold_lugano <- quantile(annual_maxima_Lugano$MaxTemp, 0.95, na.rm = TRUE)

# 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"]

# Function to calculate return period
calculate_return_period <- function(location, scale, shape, threshold) {
  cdf_value <- pevd(threshold, loc = location, scale = scale, shape = shape, type = "GEV")
  return_period <- 1 / (1 - cdf_value)
  return(return_period)
}

# Calculate return periods using the 95th percentile thresholds
```

```
return_period_geneva <- calculate_return_period(location_geneva, scale_geneva, shape_geneva, thre
return_period_santis <- calculate_return_period(location_santis, scale_santis, shape_santis, thre
return_period_lugano <- calculate_return_period(location_lugano, scale_lugano, shape_lugano, thre
```

As a result, we can observe the following return periods:

Station	Elevation	Threshold (°C)	Return period (years)
Geneva	375m	37.21	17.23
Säntis	2500m	20.5	37.22
Lugano	275m	35.06	14.76

On average, Geneva is expected to experience temperatures exceeding their 95th percentile approximately once every 17.23 years, once every 37.22 for Säntis and once every 14.76 for Lugano. Looking at the results, we can observe that Geneva and Lugano, which both are at a lower elevation than Säntis, experience extreme heats more frequently, indicating a higher vulnerability to heat waves compared to Säntis. For Geneva, which is a city influenced by the lake and continental climate, this suggests that these extremes are not annual, but they are plausible. For Lugano, it faces those heats a bit more frequently due to its lower elevation and warmer southern climate, so both cities need to prepare some adaptation strategies to mitigate the risk of issues such as for the environment or public health in the event of extreme temperatures. On the other hand, Säntis faces these extremes way less often thanks to its higher altitude and cooler overall temperatures. However, increasing temperatures can also be disruptive for its ecosystems and infrastructure.

The differences in return periods highlight the variation in elevation, geography and micro-climates of regions in Switzerland and their susceptibility to extreme heat waves. Although not similar across regions, increasing temperatures poses serious issues for all regions. As return periods are calculated in a given time using historical data, in a context of climate change with average temperature in Switzerland rising, the same threshold will be exceeded more frequently in the future than what our current models suggest. In reality, the return period are lower than what the model suggest. These results highlights our understanding of the current risk and the importance of setting clear strategies to mitigate any arising risks.

Probability of New Record High Temperature

> Question
>
> What is the probability of observing a record high temperature in Switzerland within the next year? 3 years? Decade?

Using the previously fitted model using the GEV distribution, let's calculate the probability of reaching a new record for these cities in Switzerland.

First, we calculate the highest temperature records for the 3 cities:

```
# Historical record highs for each city
record_high_geneva <- max(annual_maxima_geneva$MaxTemp)
record_high_santis <- max(annual_maxima_santis$MaxTemp)
record_high_lugano <- max(annual_maxima_Lugano$MaxTemp)

# Historical record dates for each city
record_date_geneva <- annual_maxima_geneva$Year[which.max(annual_maxima_geneva$MaxTemp)]
record_date_santis <- annual_maxima_santis$Year[which.max(annual_maxima_santis$MaxTemp)]
```

```
record_date_lugano <- annual_maxima_Lugano$Year[which.max(annual_maxima_Lugano$MaxTemp)]
```

As a result, we obtain the following records:

Station	Elevation	Highest recorded temperature (°C)	Recorded year of record
Geneva	375m	39.7	2015
Säntis	2500m	21	2019
Lugano	275m	37.1	1983

Now, we can calculate the probabilities of having a new record for each city within 1, 3 and 10 years. We are calculating the probability of exceeding this record in the given time horizon.

```
# 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 Säntis
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
```

As a result, we can observe the following probabilities for each city:

Station	Elevation	1-year probability (%)	3-year probability (%)	10-year probability (%)
Geneva	375m	0.61	1.83	5.96
Säntis	2500m	1.16	3.43	10.98
Lugano	275m	0.68	2.01	6.55

We observe that Säntis has the highest probabilities of reaching a new record. This can act as a warning indicator of shifting climate patterns. Indeed, high altitude regions are sensitive to global warming and increasing heat waves risks here could signal broader changes that might affect other regions. Geneva and Lugano which have lower elevations reflect milder extremes, though still present.

As we calculate the probabilities at a given period using historical data, we obtain the above probabilities. However, the reality might be different. Indeed, as we can see for Geneva and Säntis for example, the last record high temperatures where less than 10 years ago, so we are not safe from having a new record. So, probabilities might be different and higher in the future than the ones given in the table, also given the rising trend at the beginning of the practical.

Now, let's interpret the risks on the short, mid and long term.

- In the short-term, even though heat waves might occur, they are less likely to reach unprecedented levels. It doesn't mean that they won't happen, but just that the records might stay within the historical ranges.
- In the mid-term, record-breaking temperatures remain quite rare, but the risk of extreme events is growing, indicating potential stress on ecosystems, agriculture and infrastructures in case these events are prolonged.
- On the longer term, new records of temperatures are plausible as the probabilities increase. There is a risk of growing vulnerability depending on the frequency and intensity of the heat waves. This indicates higher risks that heat waves might impact infrastructures and health, such as higher risk of heat-related health issues, energy demands or disruptions on agriculture or water resources.

As the time horizon increases, so does the probability of seeing a record high temperature for each station. Although the records are low in the short term, the likelihood clearly grows significantly over longer periods and the reality might even be worst than our models.