Pratical 1 and 2 - Retake

Melvil Deleage & Jeff Macaraeg

June 15, 2025

Table of contents

Introduction

This documents is the annex to our final report for the retake project. It contains the full code for the practicals 1 and 2 of the retake with the results and some discussions. This allows anyone to reproduce our results and to understand the code we used.

Practical 1, Part 1 - Financial Returns & Normality

Question a)

Read in the data. Then, assess the stationarity of the (raw) stock indices.

As discussed with the teacher, we will only select 2 indices: SP500 & CAC40.

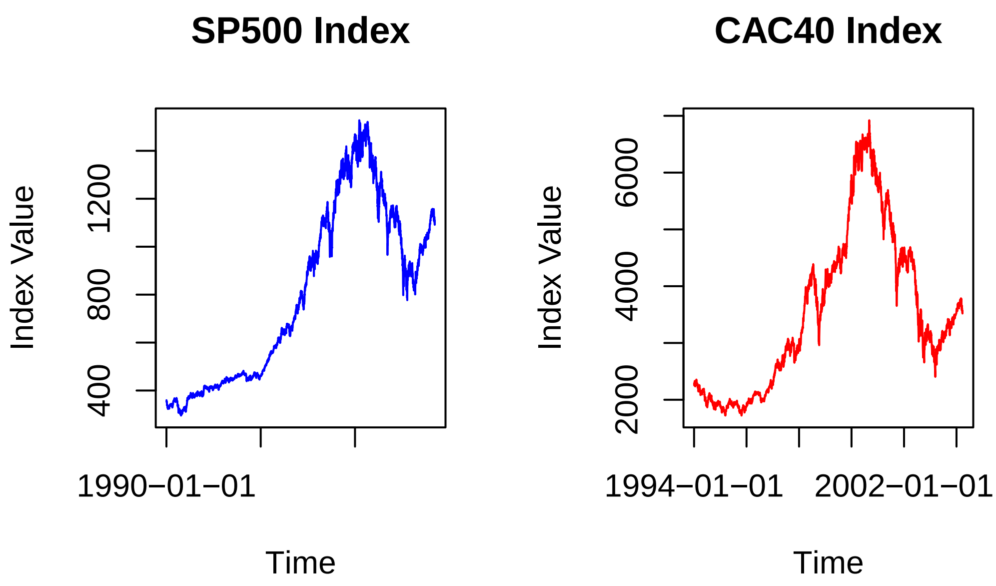
```r
# Load SP500 & CAC40 data
data("sp500")
data("cac40")

# Convert to usable numeric series
sp500_ts <- na.omit(as.numeric(sp500))
cac40_ts <- na.omit(as.numeric(cac40))

# Checking stationarity with Augmented Dickey-Fuller (ADF) test
adf_sp500 <- adf.test(sp500_ts)
adf_cac40 <- adf.test(cac40_ts)

# Print results
print(adf_sp500)
```

```
	Augmented Dickey-Fuller Test

data:  sp500_ts
Dickey-Fuller = -1.2128, Lag order = 15, p-value = 0.9044
alternative hypothesis: stationary
```

```r
print(adf_cac40)
```

```
	Augmented Dickey-Fuller Test

data:  cac40_ts
Dickey-Fuller = -0.7332, Lag order = 13, p-value = 0.9676
alternative hypothesis: stationary
```

```r
par(mfrow = c(1, 2))
# Plot the SP500 index
plot(sp500, type = "l", col = "blue", main = "SP500 Index", ylab = "Index Value")

# Plot the CAC40 index
plot(cac40, type = "l", col = "red", main = "CAC40 Index", ylab = "Index Value")
```

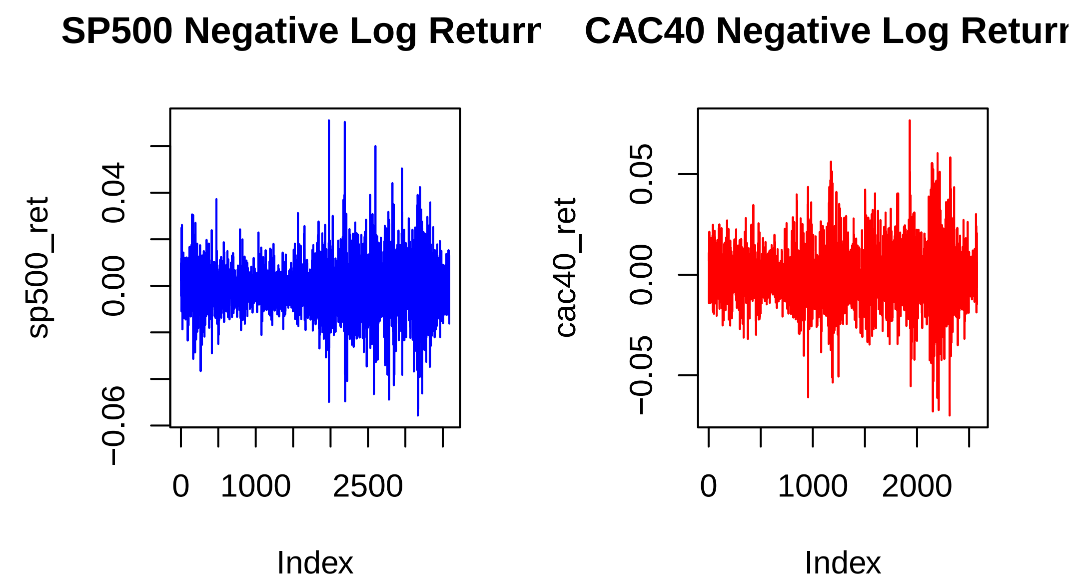


As a result, we can see that for SP500, the p-value is 0.9044 and for CAC40, the p-value is 0.9676. As they are both higher than 0.05, we can conclude that both series are not stationary.

Question b)

Create a function to transform the daily stock indices into their daily 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 to all indices.

```
# Function to compute negative log returns
log_neg_returns <- function(series) {
  ret <- -diff(log(series))
  return(na.omit(ret))  # Remove potential NA values
}

# Compute negative log returns for each index
sp500_ret <- log_neg_returns(sp500_ts)
cac40_ret <- log_neg_returns(cac40_ts)

# Plot individual series of negative log returns
par(mfrow = c(1, 2))
plot(sp500_ret, type="l", main="SP500 Negative Log Returns", col="blue")
plot(cac40_ret, type="l", main="CAC40 Negative Log Returns", col="red")
```




```r
# Assess stationarity with the ADF test
library(tseries)  # Ensure the tseries package is loaded for adf.test()
adf_sp500 <- adf.test(sp500_ret)
adf_cac40 <- adf.test(cac40_ret)

# Print the ADF test results
print("ADF Test for SP500:")
```

```
[1] "ADF Test for SP500:"
```

```r
print(adf_sp500)
```

```

	Augmented Dickey-Fuller Test

data:  sp500_ret
Dickey-Fuller = -14.843, Lag order = 15, p-value = 0.01
alternative hypothesis: stationary
```

```r
print("ADF Test for CAC40:")
```

```
[1] "ADF Test for CAC40:"
```

```r
print(adf_cac40)
```

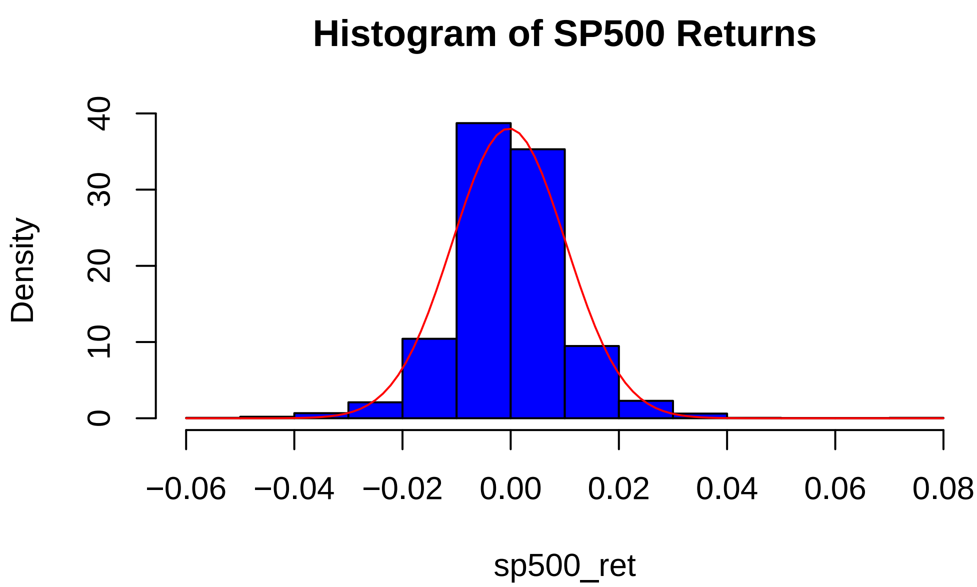
```

	Augmented Dickey-Fuller Test

data:  cac40_ret
Dickey-Fuller = -13.34, Lag order = 13, p-value = 0.01
alternative hypothesis: stationary
```

As both plots seem to not show trends, periodic cycles and a stable variance, the series seem to be stationary. To verify this, the ADF test shows that both p-values (p-value = 0.01) are lower than 0.05, so we reject the hypothesis and thus, both series are stationary.

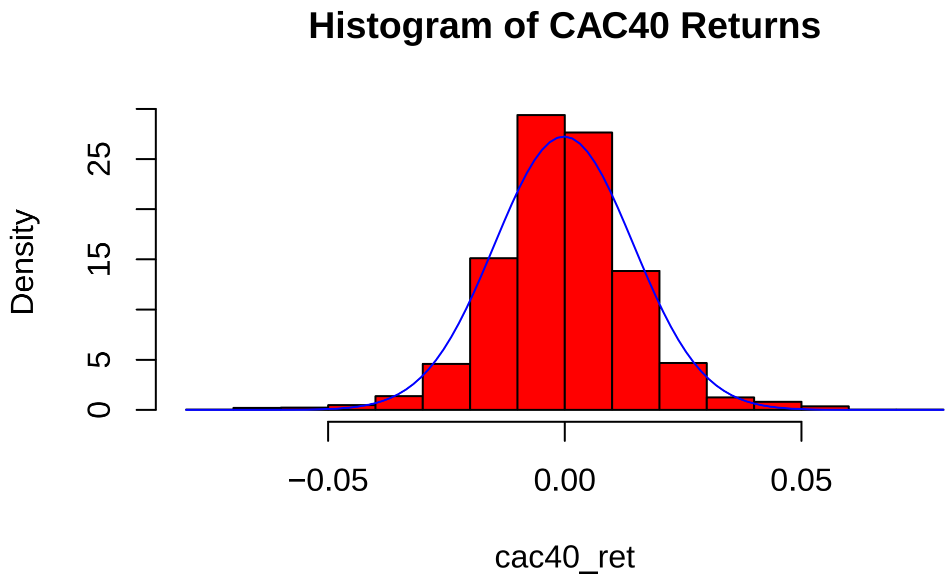
Question c)

> Draw histograms of the negative log returns and compare them to the Normal distribution. What do you observe?

```r
# Histogram of SP500 negative log returns
hist(sp500_ret, probability=TRUE, main="Histogram of SP500 Returns", col="blue")
curve(dnorm(x, mean=mean(sp500_ret), sd=sd(sp500_ret)), col="red", add=TRUE)
```




```r
# Histogram of CAC40 negative log returns
hist(cac40_ret, probability=TRUE, main="Histogram of CAC40 Returns", col="red")
curve(dnorm(x, mean=mean(cac40_ret), sd=sd(cac40_ret)), col="blue", add=TRUE)
```

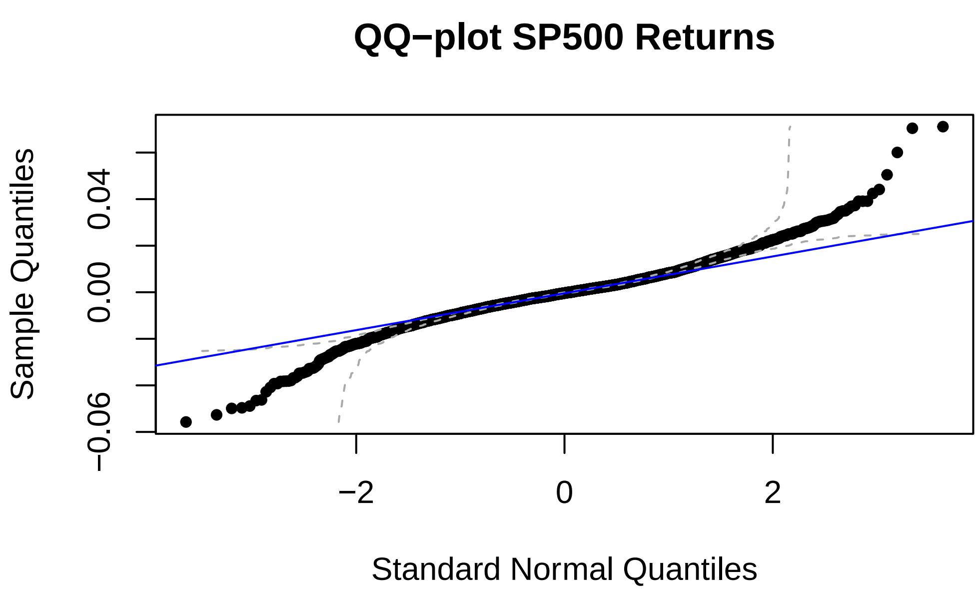


Both histograms have bell-shaped distributions, but are not perfectly aligned with the normal curve. ALso, the tails seeem to go further than what the normal disctribution curve predicts, which can indicate a higher probability of extreme returns than expected in a normal model.

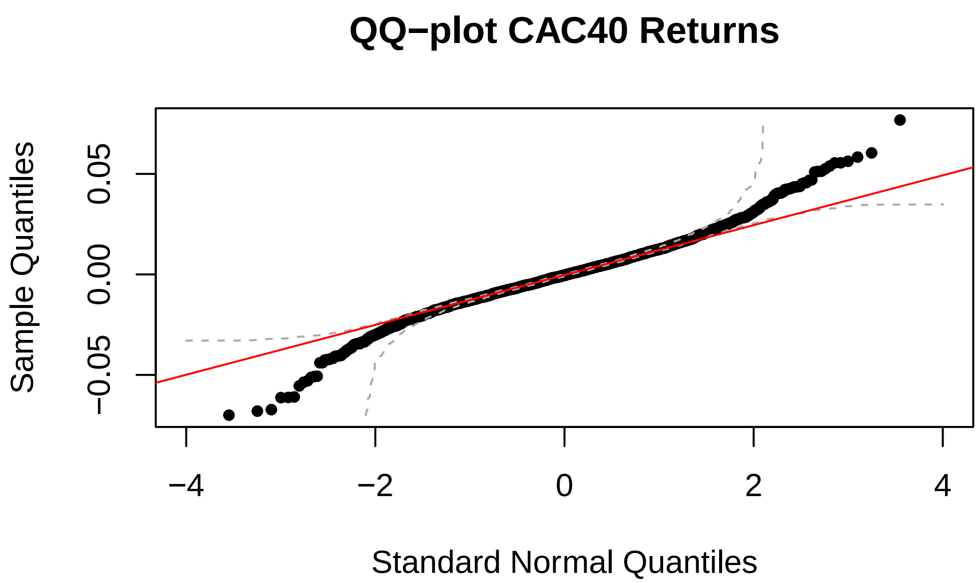
Question d)

> Check the normality assumption of the negative log returns using QQ-plots. What is your conclusion?

```
# QQ-plot SP500
qqnorm(sp500_ret, main="QQ-plot SP500 Returns")
qqline(sp500_ret, col="blue")
```



```r
# QQ-plot CAC40
qqnorm(cac40_ret, main="QQ-plot CAC40 Returns")
qqline(cac40_ret, col="red")
```



On both the lower and upper extremes, the points deviate from the line, showing heavier tails than a normal distribution. So, the negative log returns don't seem to follow normal distribution.

Question e)

> Formally test the normality assumption of the negative log returns using an Anderson-Darling testing procedure. Do you reject the Normal hypothesis?

```r
# ADF test for normality for negative log returns
ad_sp500 <- ad.test(sp500_ret)
ad_cac40 <- ad.test(cac40_ret)

# Print results
print(ad_sp500)
```

```
    Anderson-Darling normality test

data:  sp500_ret
A = 29.24, p-value < 2.2e-16
```

```r
print(ad_cac40)
```

```
    Anderson-Darling normality test

data:  cac40_ret
A = 10.33, p-value < 2.2e-16
```

As both p-values are lower than 0.05 (both at 2.2e-16), we reject the null hypothesis, meaning that the returns are not normally distributed.

Question f)

> Use the fitdistr() function from the MASS package in order to obtain the (maximum-likelihood estimated) parameters of distributions you could imagine for the negative log returns. Try to fit at least two different distributions on the data and, using an information criteria (such as the AIC), decide which distributional framework fits best for each of the series.

```r
# Fitting a normal distribution
fit_norm_sp500 <- fitdistr(sp500_ret, "normal")
fit_norm_cac40 <- fitdistr(cac40_ret, "normal")

# Fitting a Student's t distribution
fit_t_sp500 <- fitdistr(sp500_ret, "t")
fit_t_cac40 <- fitdistr(cac40_ret, "t")

# Print the results
print(fit_norm_sp500)
```

```
      mean              sd
 -0.0003139511    0.0104904994
( 0.0001751582) ( 0.0001238556)
```

```r
print(fit_t_sp500)
```

```
       m                s                df
 -0.0003823438    0.0083396816    6.5020438529
( 0.0001642079) ( 0.0011490812) ( 5.6443703001)
```

```r
print(fit_norm_cac40)
```

```
      mean              sd
 -0.0001723074    0.0146403328
( 0.0002884550) ( 0.0002039685)
```

```r
print(fit_t_cac40)
```

```
       m                s                df
 -0.0003374078    0.0116334321    5.2499036218
( 0.0002632892) ( 0.0002745656) ( 0.5356059623)
```

```r
# Compute AIC values
AIC(fit_norm_sp500)
```

```
[1] -22510.5
```

```r
AIC(fit_t_sp500)
```

```
[1] -22898.7
```

```r
AIC(fit_norm_cac40)
```

```
[1] -14447.55
```

```r
AIC(fit_t_cac40)
```

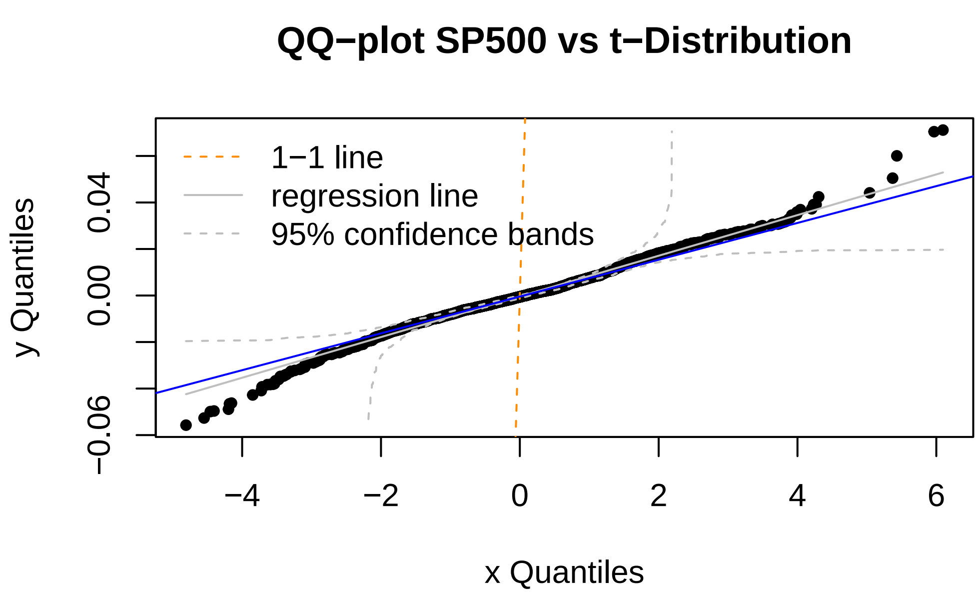
```
[1] -14626
```

As the Student model yields lower AIC than the normal model, it indicates a better balance between the goodness-of-fit and the complexity. It is a more appropriate choice of model.

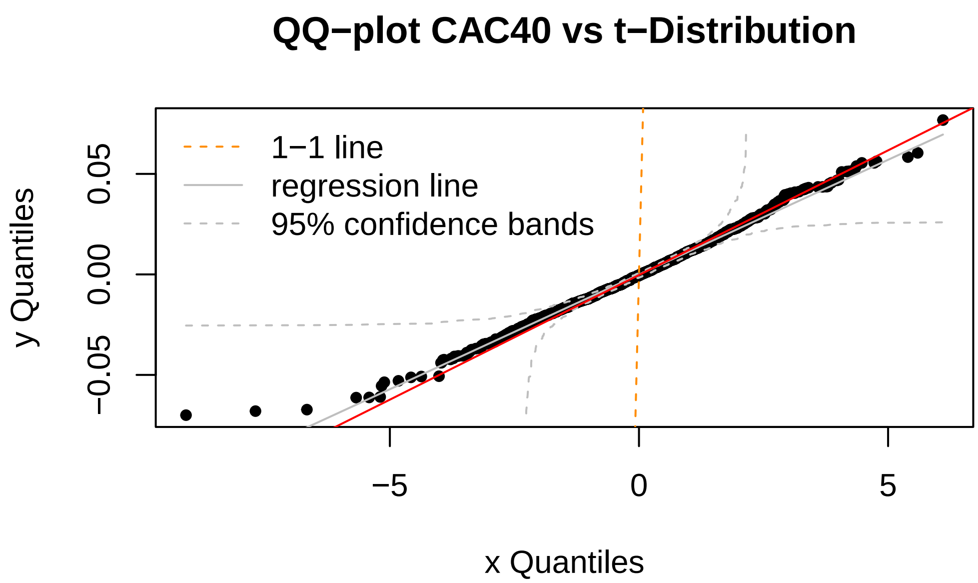
Question g)

> If this has not been done in (f), fit a t-distribution to the negative log returns using fitdistr(). Using a QQ-plot for each of the series, decide whether the fit is better than with a Normal distribution, based on your answer in (d).

```r
# QQ-plot for SP500 using a t-distribution
qqplot(rt(length(sp500_ret), df=fit_t_sp500$estimate["df"]), sp500_ret, main="QQ-plot SP500 vs t-
qqline(sp500_ret, col="blue")
```



```r
# QQ-plot for CAC40 using a t-distribution
qqplot(rt(length(cac40_ret), df=fit_t_cac40$estimate["df"]), cac40_ret, main="QQ-plot CAC40 vs t-
qqline(cac40_ret, col="red")
```



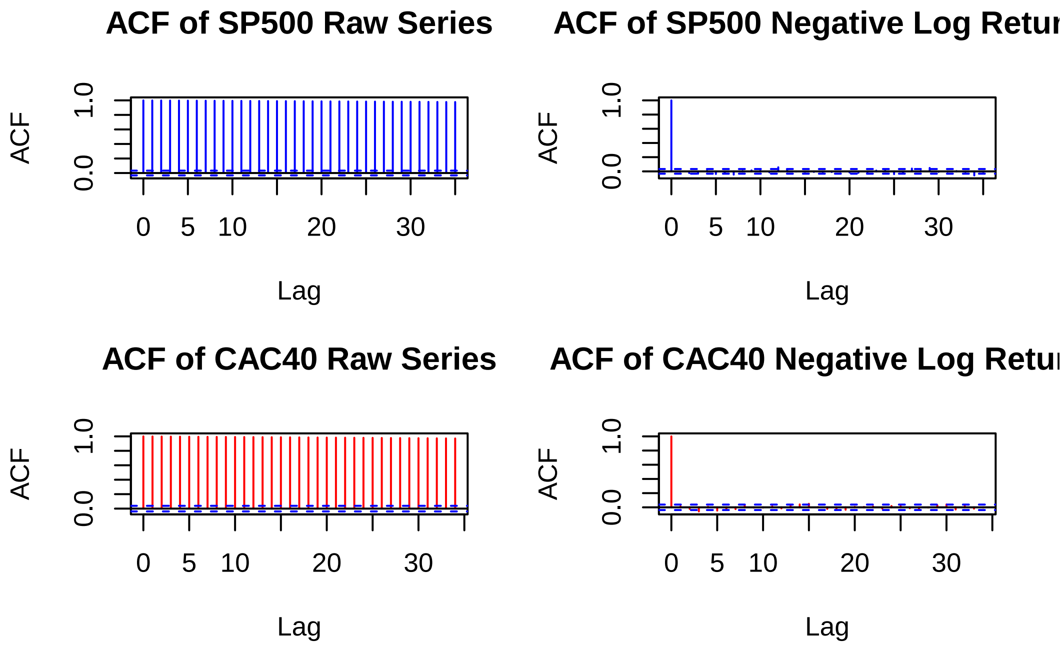
Despite some deviations from points in the extremes, the QQ-plots confirm that a Student's t-distribution provides a better fit for both the SP500 and CAC40 log returns compared to a normal distribution. The heavier tails of the t-distribution captures the extremes better. It is also consistent with question d).

Practical 1, Part 2 - Financial time series, volatility and the random walk hypothesis

Question a)

> Plot the ACF of all the series in Part 1 (i.e. the raw series as well as the negative log returns). What do you observe?

```r
par(mfrow = c(2, 2), mar = c(4, 4, 4, 2))

# ACF for the raw and negative log returns for SP500
acf(sp500, main = "ACF of SP500 Raw Series", col = "blue")
acf(sp500_ret, main = "ACF of SP500 Negative Log Returns", col = "blue")

# ACF for the raw and negative log returns for CAC40
acf(cac40, main = "ACF of CAC40 Raw Series", col = "red")
acf(cac40_ret, main = "ACF of CAC40 Negative Log Returns", col = "red")
```



The raw series show high autocorrelation at all lags, implying non-stationarity. For the negative log returns, it shows little to no autocorrelation, suggesting that the daily returns are close to uncorrelated.

Question b)

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

```r
# Ljung-Box test for SP500
Box.test(sp500, lag=20, type="Ljung-Box")
```

```

	Box-Ljung test

data:  sp500
X-squared = 71035, df = 20, p-value < 2.2e-16
```

```r
Box.test(sp500_ret, lag=20, type="Ljung-Box")
```

```

	Box-Ljung test

data:  sp500_ret
X-squared = 38.931, df = 20, p-value = 0.0068
```

```r
# Ljung-Box test for CAC40
Box.test(cac40, lag=20, type="Ljung-Box")
```

```
    Box-Ljung test

data:  cac40
X-squared = 50889, df = 20, p-value < 2.2e-16
```

```r
Box.test(cac40_ret, lag=20, type="Ljung-Box")
```

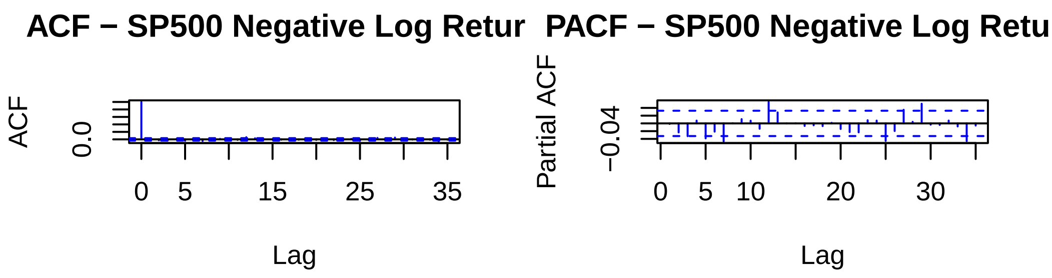
```
    Box-Ljung test

data:  cac40_ret
X-squared = 41.079, df = 20, p-value = 0.003639
```

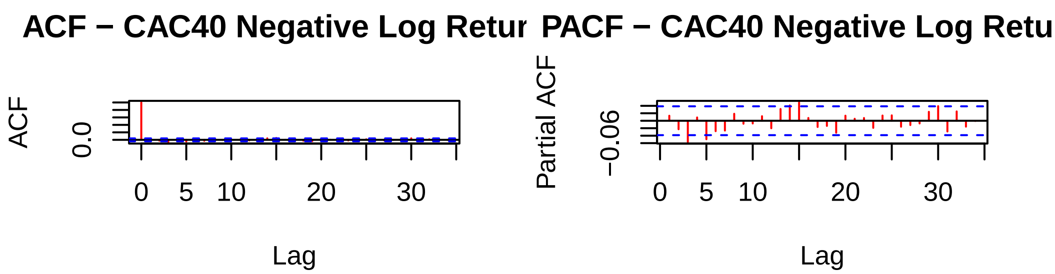
Looking at the raw series for both CAC40 and SP500, we obtain a p-value that is lower than 0.05 (both at 2.2e-16), so we reject the null hypothesis and the raw series show autocorrelation. For both negative log returns, the p-values are higher than the raw series (0.0068 for SP500, 0.003639 for CAC40), but still smaller than the p-value, so again they show autocorrelation.

Question c)

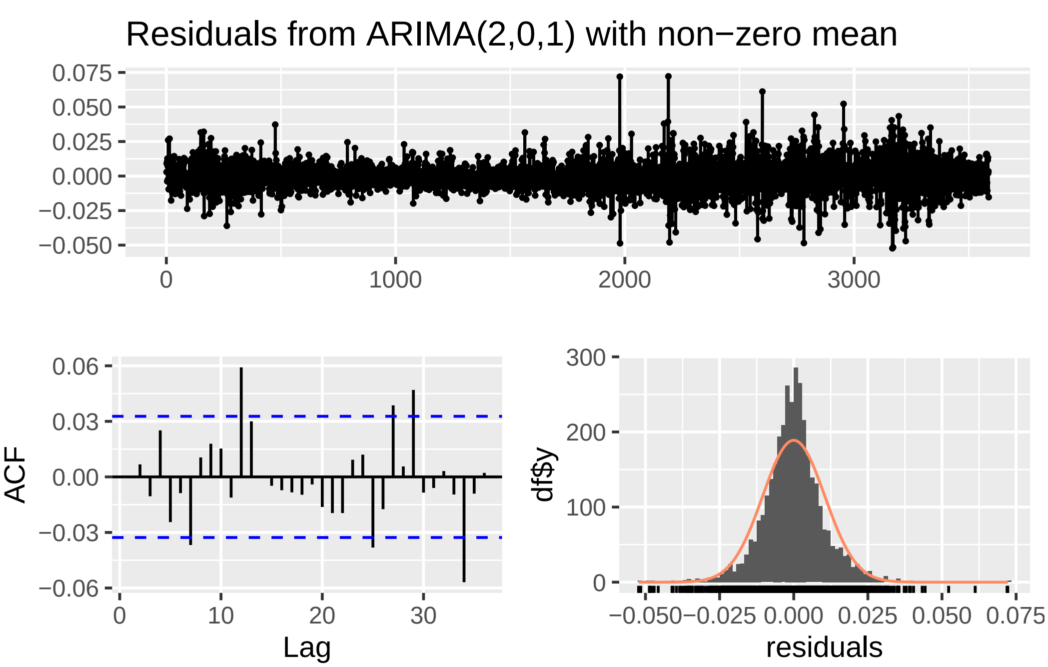
> Propose ARIMA models for each of the negative log returns series, based on visualisation tools (e.g. ACF and PACF). Select an ARIMA model using auto.arima() (forecast package) to each of the negative log returns series. Comment on the difference. Assess the residuals of the resulting models.

```r
par(mfrow = c(1, 2))

# Fit ARIMA models using auto.arima()
arima_sp500 <- auto.arima(sp500_ret)
arima_cac40 <- auto.arima(cac40_ret)

par(mfrow = c(2, 2))

# ACF & PACF for SP500 negative log returns
acf(sp500_ret, main = "ACF - SP500 Negative Log Returns", col = "blue")
pacf(sp500_ret, main = "PACF - SP500 Negative Log Returns", col = "blue")

# ACF & PACF for CAC40 negative log returns
acf(cac40_ret, main = "ACF - CAC40 Negative Log Returns", col = "red")
pacf(cac40_ret, main = "PACF - CAC40 Negative Log Returns", col = "red")
```



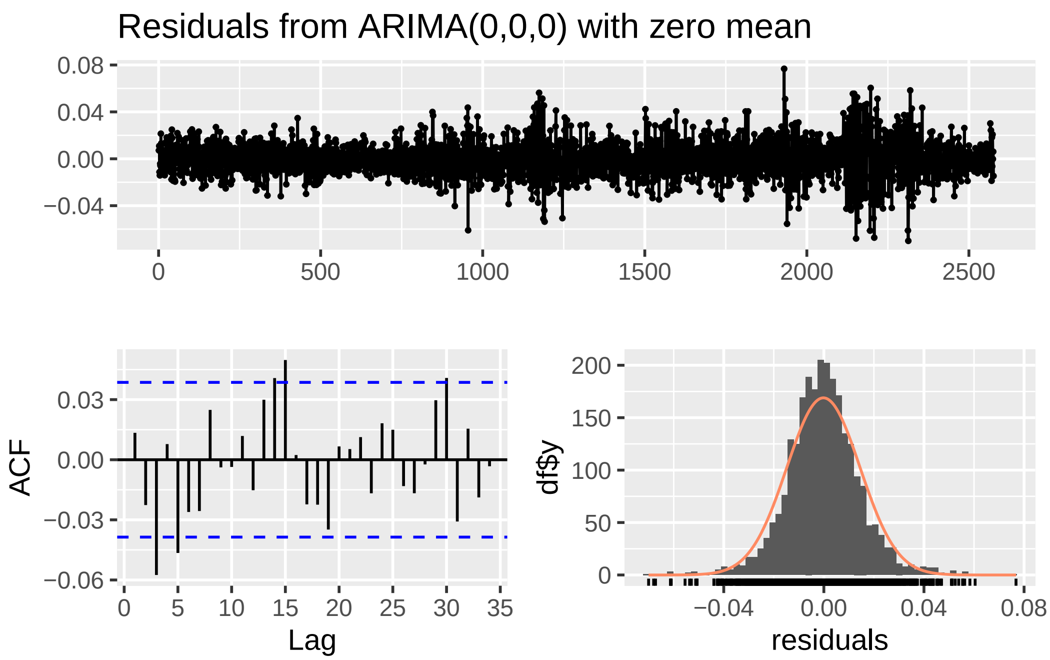



```r
checkresiduals(arima_sp500)
```




```
    Ljung-Box test

data:  Residuals from ARIMA(2,0,1) with non-zero mean
Q* = 12.498, df = 7, p-value = 0.08534

Model df: 3.   Total lags used: 10
```

```r
checkresiduals(arima_cac40)
```




```
    Ljung-Box test

data:  Residuals from ARIMA(0,0,0) with zero mean
Q* = 21.191, df = 10, p-value = 0.0198

Model df: 0.   Total lags used: 10
```

```r
summary(arima_sp500)
```

```
Series: sp500_ret
ARIMA(2,0,1) with non-zero mean

Coefficients:
         ar1      ar2      ma1    mean
      0.7796  -0.0291  -0.7827  -3e-04
s.e.  0.1035   0.0180   0.1024   2e-04

sigma^2 = 0.0001099:  log likelihood = 11261.47
AIC=-22512.94   AICc=-22512.92   BIC=-22482.01

Training set error measures:
                       ME       RMSE        MAE  MPE MAPE      MASE
Training set 2.252009e-06 0.01047816 0.00748605 -Inf  Inf 0.6910122
                      ACF1
Training set -0.0003543668
```

```r
summary(arima_cac40)
```

```
Series: cac40_ret
```

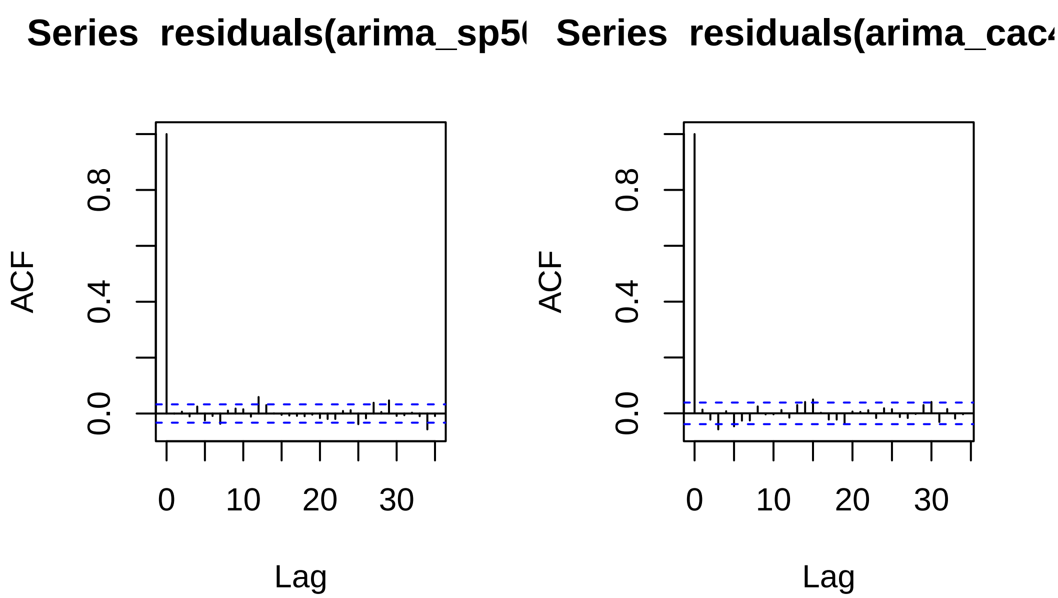
```
ARIMA(0,0,0) with zero mean

sigma^2 = 0.0002144:  log likelihood = 7225.6
AIC=-14449.19   AICc=-14449.19   BIC=-14443.34

Training set error measures:
                        ME       RMSE        MAE MPE MAPE      MASE       ACF1
Training set -0.0001723074 0.01464135 0.01089581 100  100 0.7046982 0.01343285
```

The ACF & PACF for both SP500 & CAC40 negative log returns show almost no significant autocorrelation, indicating that the returns behave in a similar way to white noise.It is also confirmed by checking the residuals. We can also observe that the auto.arima() function selected very simple models.

Question d)

> Assess the residuals of the resulting models from (c), both their raw values and their absolute values, through visual tools (such as the ACF) and formal tests (e.g. Ljung-Box). What do you conclude about the independence assumption?

```r
par(mfrow = c(1, 2))
acf(residuals(arima_sp500))
acf(residuals(arima_cac40))
```

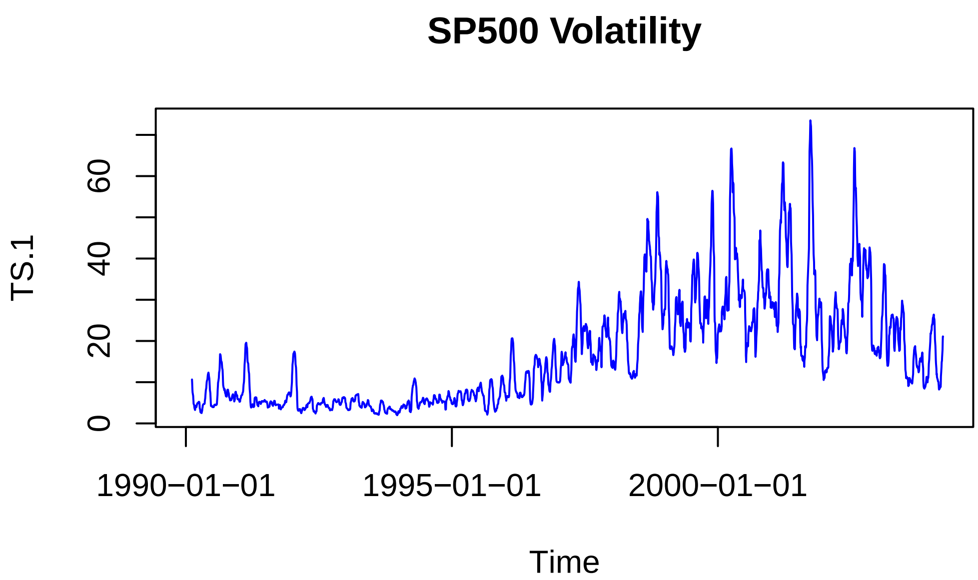


The independence assumption holds for the raw residuals but not for the volatility patterns (autocorrelation in the absolute residuals). So, further modeling could be used to fully capture the dynamics of the residuals.

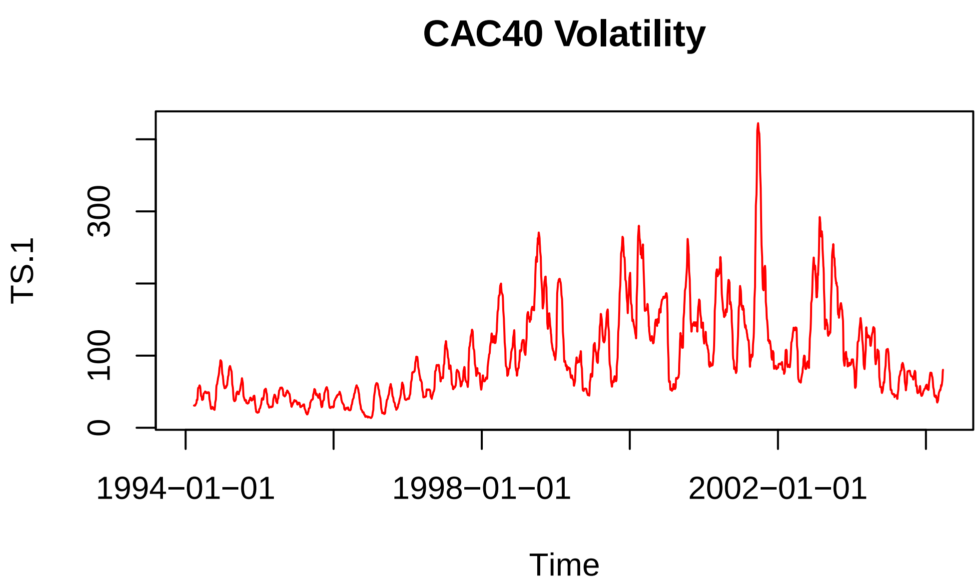
Question e)

> Plot the volatility of the raw series of indices. What is your conclusion on the homoscedasticity assumption?

```r
# Calculate volatility & plots
volatility <- function(series) {
  return(runSD(series, n=30))
}
plot(volatility(sp500), type="l", main="SP500 Volatility", col = "blue")
```




```r
plot(volatility(cac40), type="l", main="CAC40 Volatility", col = "red")
```



The data shows heteroskedasticity, as the volatility (or variance) is not constant over time.

Question f)

> Fit GARCH models to the negative log returns of each series with both standardised and skewed t-distributions, with order (1, 1), using the garchFit() function from the fGarch library. Assess the quality of the fit by evaluating the residuals.

```r
# SP500 & CAC40 standardized t-distribution
garch_sp500_t <- garchFit(~ garch(1, 1), data = sp500_ret, cond.dist = "std", trace = FALSE)
garch_cac40_t <- garchFit(~ garch(1, 1), data = cac40_ret, cond.dist = "std", trace = FALSE)

# SP500 & CAC40 skewed t-distribution
garch_sp500_skt <- garchFit(~ garch(1, 1), data = sp500_ret, cond.dist = "sstd", trace = FALSE)
garch_cac40_skt <- garchFit(~ garch(1, 1), data = cac40_ret, cond.dist = "sstd", trace = FALSE)

summary(garch_sp500_t)
```

```
Title:
 GARCH Modelling

Call:
 garchFit(formula = ~garch(1, 1), data = sp500_ret, cond.dist = "std",
    trace = FALSE)

Mean and Variance Equation:
 data ~ garch(1, 1)
<environment: 0x13ac567c8>
 [data = sp500_ret]

Conditional Distribution:
 std

Coefficient(s):
         mu        omega       alpha1        beta1        shape
-5.6197e-04   3.2345e-07   5.0937e-02   9.4765e-01   7.1692e+00

Std. Errors:
 based on Hessian

Error Analysis:
         Estimate  Std. Error  t value Pr(>|t|)
mu     -5.620e-04   1.250e-04   -4.497 6.90e-06 ***
omega   3.234e-07   1.442e-07    2.242   0.0249 *
alpha1  5.094e-02   7.720e-03    6.598 4.16e-11 ***
beta1   9.476e-01   7.660e-03  123.714  < 2e-16 ***
shape   7.169e+00   8.189e-01    8.755  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Log Likelihood:
 11798.72    normalized:  3.289301
```

```
Description:
 Sun Jun 15 10:11:00 2025 by user: 


Standardised Residuals Tests:
                                  Statistic     p-Value
 Jarque-Bera Test   R    Chi^2  661.3620424 0.000000000
 Shapiro-Wilk Test  R    W        0.9842328 0.000000000
 Ljung-Box Test     R    Q(10)   18.0504009 0.054119378
 Ljung-Box Test     R    Q(15)   34.5937376 0.002808328
 Ljung-Box Test     R    Q(20)   36.2946867 0.014198662
 Ljung-Box Test     R^2  Q(10)    7.7661792 0.651664153
 Ljung-Box Test     R^2  Q(15)   10.7404577 0.770766521
 Ljung-Box Test     R^2  Q(20)   15.4320193 0.751176895
 LM Arch Test       R    TR^2    10.0402112 0.612432829

Information Criterion Statistics:
      AIC       BIC       SIC      HQIC 
-6.575815 -6.567193 -6.575819 -6.572742 
```

```r
summary(garch_sp500_skt)
```

```
Title:
 GARCH Modelling 

Call:
 garchFit(formula = ~garch(1, 1), data = sp500_ret, cond.dist = "sstd", 
    trace = FALSE) 

Mean and Variance Equation:
 data ~ garch(1, 1)
<environment: 0x12f27eb38>
 [data = sp500_ret]

Conditional Distribution:
 sstd 

Coefficient(s):
         mu        omega       alpha1        beta1         skew        shape  
-5.1103e-04   3.3855e-07   5.1496e-02   9.4679e-01   1.0316e+00   7.3173e+00  

Std. Errors:
 based on Hessian 

Error Analysis:
         Estimate  Std. Error  t value Pr(>|t|)    
mu     -5.110e-04   1.308e-04   -3.906  9.4e-05 ***
omega   3.386e-07   1.472e-07    2.299   0.0215 *  
alpha1  5.150e-02   7.770e-03    6.628  3.4e-11 ***
beta1   9.468e-01   7.772e-03  121.816  < 2e-16 ***
skew    1.032e+00   2.406e-02   42.875  < 2e-16 ***
```

```
shape   7.317e+00   8.596e-01    8.512  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Log Likelihood:
 11799.62    normalized:  3.28955

Description:
 Sun Jun 15 10:11:00 2025 by user:


Standardised Residuals Tests:
                                  Statistic     p-Value
 Jarque-Bera Test   R    Chi^2  659.5154771 0.000000000
 Shapiro-Wilk Test  R    W        0.9842608 0.000000000
 Ljung-Box Test     R    Q(10)   17.9330634 0.056103319
 Ljung-Box Test     R    Q(15)   34.5128308 0.002883347
 Ljung-Box Test     R    Q(20)   36.1922613 0.014599763
 Ljung-Box Test     R^2  Q(10)    7.5116696 0.676416753
 Ljung-Box Test     R^2  Q(15)   10.5335043 0.784908436
 Ljung-Box Test     R^2  Q(20)   15.1615192 0.767092017
 LM Arch Test       R    TR^2     9.8488270 0.629221076

Information Criterion Statistics:
      AIC       BIC       SIC      HQIC
-6.575755 -6.565409 -6.575761 -6.572067
```

```r
summary(garch_cac40_t)
```

```
Title:
 GARCH Modelling

Call:
 garchFit(formula = ~garch(1, 1), data = cac40_ret, cond.dist = "std",
    trace = FALSE)

Mean and Variance Equation:
 data ~ garch(1, 1)
<environment: 0x1381432e0>
 [data = cac40_ret]

Conditional Distribution:
 std

Coefficient(s):
         mu        omega       alpha1         beta1        shape
-5.2677e-04   2.0876e-06   6.4925e-02   9.2750e-01   1.0000e+01

Std. Errors:
 based on Hessian
```

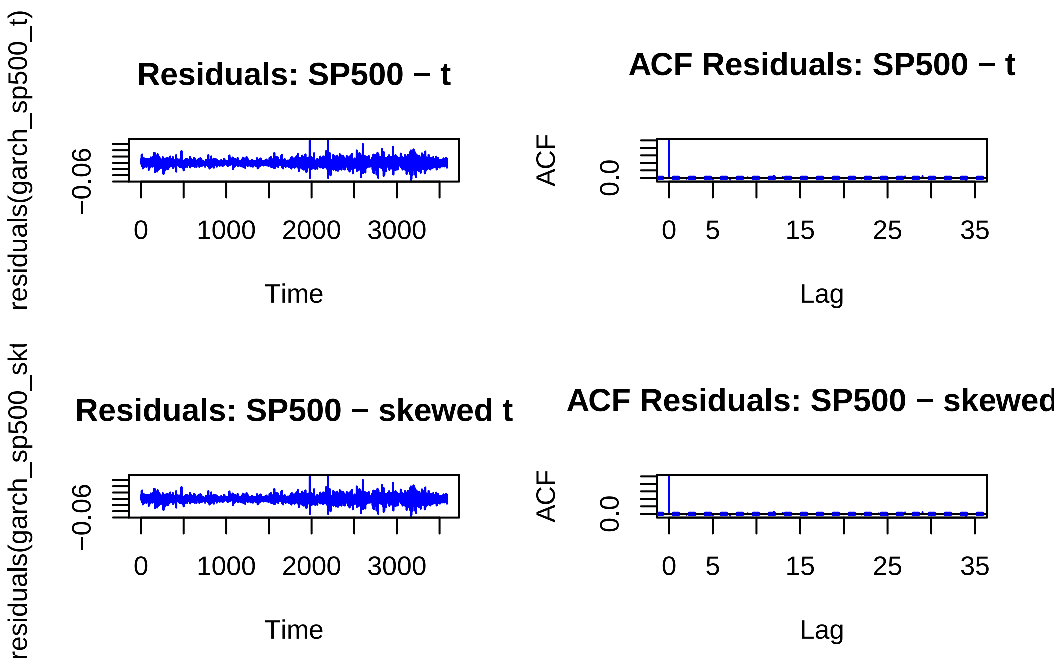
```
Error Analysis:
         Estimate  Std. Error  t value Pr(>|t|)
mu     -5.268e-04   2.343e-04   -2.248  0.02459 *
omega   2.088e-06   7.696e-07    2.712  0.00668 **
alpha1  6.493e-02   9.599e-03    6.764 1.35e-11 ***
beta1   9.275e-01   1.043e-02   88.943  < 2e-16 ***
shape   1.000e+01   1.361e+00    7.350 1.99e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Log Likelihood:
 7496.61    normalized:  2.910175

Description:
 Sun Jun 15 10:11:00 2025 by user:


Standardised Residuals Tests:
                                 Statistic      p-Value
 Jarque-Bera Test   R    Chi^2  36.1606492 1.405448e-08
 Shapiro-Wilk Test  R    W       0.9968239 3.245060e-05
 Ljung-Box Test     R    Q(10)  12.1818800 2.730684e-01
 Ljung-Box Test     R    Q(15)  20.7654478 1.444922e-01
 Ljung-Box Test     R    Q(20)  22.3308637 3.228303e-01
 Ljung-Box Test     R^2  Q(10)  12.6755294 2.423833e-01
 Ljung-Box Test     R^2  Q(15)  13.1950834 5.872327e-01
 Ljung-Box Test     R^2  Q(20)  14.8593966 7.843956e-01
 LM Arch Test       R    TR^2   13.7118254 3.194879e-01

Information Criterion Statistics:
      AIC       BIC       SIC      HQIC
-5.816467 -5.805105 -5.816475 -5.812348
```

```r
summary(garch_cac40_skt)
```

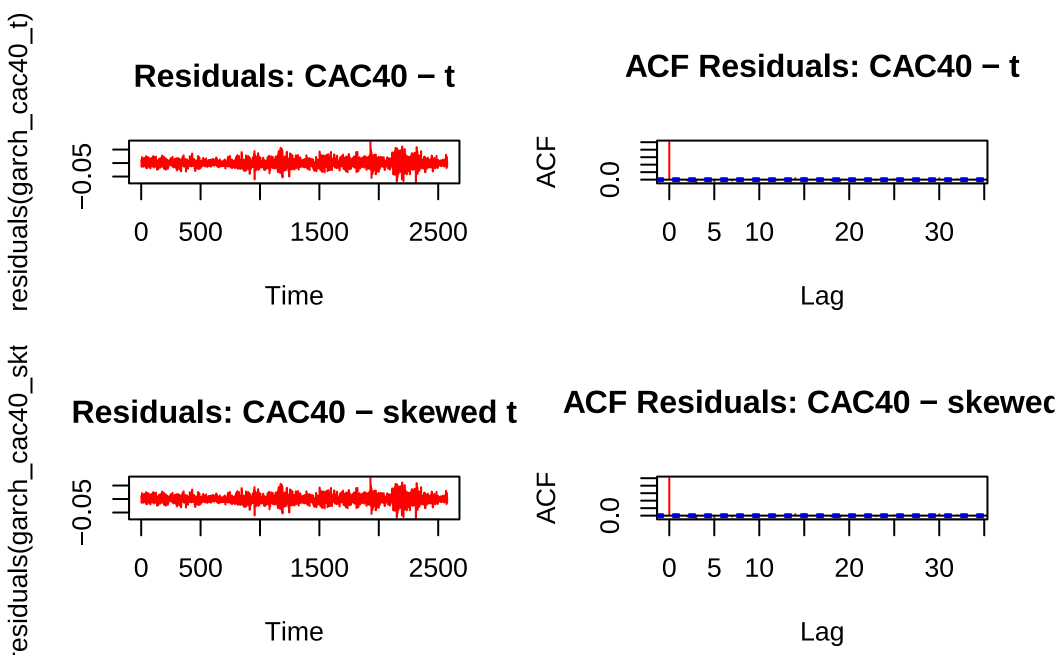
```
Title:
 GARCH Modelling

Call:
 garchFit(formula = ~garch(1, 1), data = cac40_ret, cond.dist = "sstd",
    trace = FALSE)

Mean and Variance Equation:
 data ~ garch(1, 1)
<environment: 0x138241f58>
 [data = cac40_ret]

Conditional Distribution:
 sstd

Coefficient(s):
```

```
         mu        omega       alpha1        beta1         skew        shape
-4.1284e-04   2.0402e-06   6.5401e-02   9.2730e-01   1.0840e+00   1.0000e+01

Std. Errors:
 based on Hessian

Error Analysis:
         Estimate  Std. Error  t value Pr(>|t|)
mu     -4.128e-04   2.377e-04   -1.737  0.08245 .
omega   2.040e-06   7.576e-07    2.693  0.00708 **
alpha1  6.540e-02   9.543e-03    6.853 7.23e-12 ***
beta1   9.273e-01   1.030e-02   90.004  < 2e-16 ***
skew    1.084e+00   3.327e-02   32.581  < 2e-16 ***
shape   1.000e+01   1.342e+00    7.452 9.24e-14 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Log Likelihood:
 7500.141    normalized:  2.911545

Description:
 Sun Jun 15 10:11:01 2025 by user:


Standardised Residuals Tests:
                                 Statistic      p-Value
 Jarque-Bera Test   R    Chi^2  36.6432206 1.104144e-08
 Shapiro-Wilk Test  R    W       0.9967943 2.934125e-05
 Ljung-Box Test     R    Q(10)  12.0943005 2.787949e-01
 Ljung-Box Test     R    Q(15)  20.7596486 1.446860e-01
 Ljung-Box Test     R    Q(20)  22.3409037 3.223024e-01
 Ljung-Box Test     R^2  Q(10)  12.3893095 2.598459e-01
 Ljung-Box Test     R^2  Q(15)  12.9276394 6.078871e-01
 Ljung-Box Test     R^2  Q(20)  14.5756952 8.001500e-01
 LM Arch Test       R    TR^2   13.4251438 3.389111e-01

Information Criterion Statistics:
      AIC       BIC       SIC      HQIC
-5.818432 -5.804797 -5.818443 -5.813490
```

```r
# Residual plots SP500
par(mfrow = c(2, 2))
ts.plot(residuals(garch_sp500_t), main = "Residuals: SP500 - t", col = "blue")
acf(residuals(garch_sp500_t), main = "ACF Residuals: SP500 - t", col = "blue")

ts.plot(residuals(garch_sp500_skt), main = "Residuals: SP500 - skewed t", col = "blue")
acf(residuals(garch_sp500_skt), main = "ACF Residuals: SP500 - skewed t", col = "blue")
```




```
# Residual plots CAC40
par(mfrow = c(2, 2))
ts.plot(residuals(garch_cac40_t), main = "Residuals: CAC40 - t", col = "red")
acf(residuals(garch_cac40_t), main = "ACF Residuals: CAC40 - t", col = "red")

ts.plot(residuals(garch_cac40_skt), main = "Residuals: CAC40 - skewed t", col = "red")
acf(residuals(garch_cac40_skt), main = "ACF Residuals: CAC40 - skewed t", col = "red")
```




```
# Ljung-Box Test on Garch residuals
Box.test(residuals(garch_sp500_t), lag = 20, type = "Ljung-Box")
```

```
	Box-Ljung test
```

```
data:  residuals(garch_sp500_t)
X-squared = 38.931, df = 20, p-value = 0.0068
```

```r
Box.test(residuals(garch_sp500_skt), lag = 20, type = "Ljung-Box")
```

```
    Box-Ljung test

data:  residuals(garch_sp500_skt)
X-squared = 38.931, df = 20, p-value = 0.0068
```

```r
Box.test(residuals(garch_cac40_t), lag = 20, type = "Ljung-Box")
```

```
    Box-Ljung test

data:  residuals(garch_cac40_t)
X-squared = 41.079, df = 20, p-value = 0.003639
```

```r
Box.test(residuals(garch_cac40_skt), lag = 20, type = "Ljung-Box")
```

```
    Box-Ljung test

data:  residuals(garch_cac40_skt)
X-squared = 41.079, df = 20, p-value = 0.003639
```

By evaluating the residuals, we observe that for both SP500 and CAC40, the p-values are below 0.05, meaning there is still autocorrelation and residuals are not white noise. Looking at the AIC/BIC and the log-likelihood, it seems that the skewed distribution is slightly better for both models especially for CAC40. Thus, the Garch models are a good start but can be improved.

Question g)

> 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 ARMA(p,q) on the negative log returns; • fit a GARCH(1,1) on the residuals of the ARMA(p,q) fit. Proceed with the above recipe. Assess the quality of the above fit.

To fit an ARMA(p,q) on the negative log returns:

```r
# Select best ARMA model
arma_sp500 <- auto.arima(sp500_ret, max.p=5, max.q=5, seasonal=FALSE, ic="aic")
arma_cac40 <- auto.arima(cac40_ret, max.p=5, max.q=5, seasonal=FALSE, ic="aic")

# Get residuals
res_sp500 <- residuals(arma_sp500)
res_cac40 <- residuals(arma_cac40)
```

To fit GARCH(1,1) on the ARMA residuals:

```r
garch_sp500_arma_res <- garchFit(~ garch(1, 1), data = res_sp500, cond.dist = "sstd", trace = FAL
garch_cac40_arma_res <- garchFit(~ garch(1, 1), data = res_cac40, cond.dist = "sstd", trace = FAL
```

To assess the fit quality:

```r
# Show summaries
summary(garch_sp500_arma_res)
```

```
Title:
 GARCH Modelling

Call:
 garchFit(formula = ~garch(1, 1), data = res_sp500, cond.dist = "sstd",
    trace = FALSE)

Mean and Variance Equation:
 data ~ garch(1, 1)
<environment: 0x1396bf7f0>
 [data = res_sp500]

Conditional Distribution:
 sstd

Coefficient(s):
         mu        omega       alpha1        beta1         skew        shape
-2.2520e-05   3.3032e-07   5.0301e-02   9.4808e-01   1.0551e+00   7.1935e+00

Std. Errors:
 based on Hessian

Error Analysis:
         Estimate  Std. Error  t value Pr(>|t|)
mu     -2.252e-05   1.325e-04   -0.170   0.8651
omega   3.303e-07   1.438e-07    2.298   0.0216 *
alpha1  5.030e-02   7.540e-03    6.671 2.53e-11 ***
beta1   9.481e-01   7.530e-03  125.907  < 2e-16 ***
skew    1.055e+00   2.446e-02   43.141  < 2e-16 ***
shape   7.193e+00   8.408e-01    8.556  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Log Likelihood:
 11806.58    normalized:  3.291491

Description:
 Sun Jun 15 10:11:02 2025 by user:


Standardised Residuals Tests:
                                  Statistic    p-Value
 Jarque-Bera Test   R    Chi^2  725.5180693 0.00000000
```

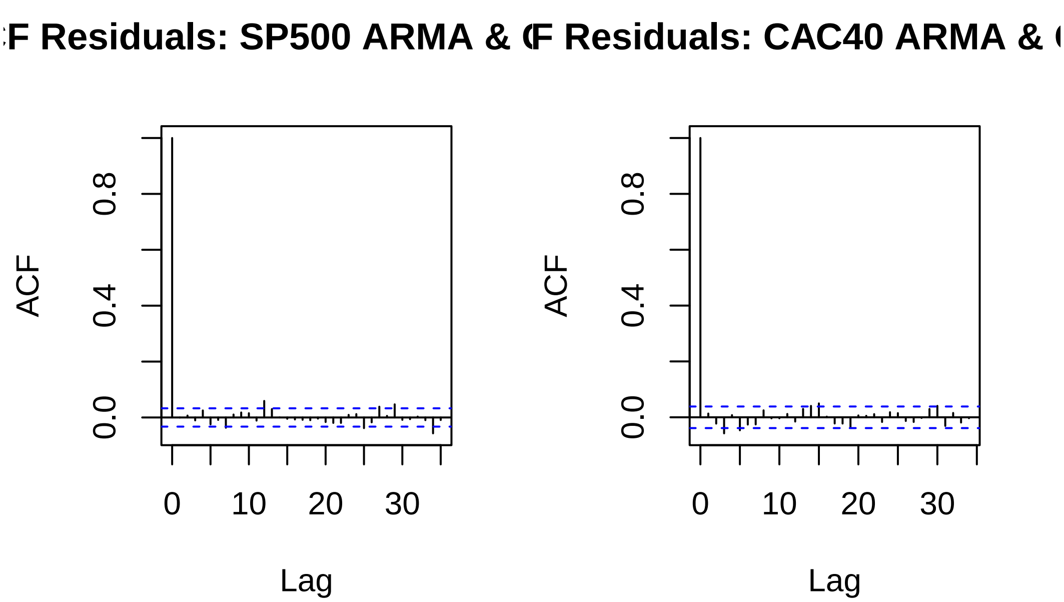
```
 Shapiro-Wilk Test  R    W        0.9831582 0.00000000
 Ljung-Box Test     R    Q(10)   10.4216838 0.40430817
 Ljung-Box Test     R    Q(15)   27.7618601 0.02310814
 Ljung-Box Test     R    Q(20)   29.1478746 0.08488825
 Ljung-Box Test     R^2  Q(10)    8.0543302 0.62352994
 Ljung-Box Test     R^2  Q(15)   11.0084514 0.75199504
 Ljung-Box Test     R^2  Q(20)   15.6980861 0.73516897
 LM Arch Test       R    TR^2    10.2583123 0.59331040

Information Criterion Statistics:
      AIC       BIC       SIC      HQIC
-6.579637 -6.569291 -6.579642 -6.575949
```

```r
summary(garch_cac40_arma_res)
```

```
Title:
 GARCH Modelling

Call:
 garchFit(formula = ~garch(1, 1), data = res_cac40, cond.dist = "sstd",
    trace = FALSE)

Mean and Variance Equation:
 data ~ garch(1, 1)
<environment: 0x149246eb8>
 [data = res_cac40]

Conditional Distribution:
 sstd

Coefficient(s):
        mu        omega       alpha1        beta1         skew        shape
-4.1284e-04   2.0402e-06   6.5401e-02   9.2730e-01   1.0840e+00   1.0000e+01

Std. Errors:
 based on Hessian

Error Analysis:
         Estimate  Std. Error  t value Pr(>|t|)
mu     -4.128e-04   2.377e-04   -1.737  0.08245 .
omega   2.040e-06   7.576e-07    2.693  0.00708 **
alpha1  6.540e-02   9.543e-03    6.853 7.23e-12 ***
beta1   9.273e-01   1.030e-02   90.004  < 2e-16 ***
skew    1.084e+00   3.327e-02   32.581  < 2e-16 ***
shape   1.000e+01   1.342e+00    7.452 9.24e-14 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Log Likelihood:
 7500.141    normalized:  2.911545
```

```
Description:
 Sun Jun 15 10:11:02 2025 by user:


Standardised Residuals Tests:
                                 Statistic      p-Value
 Jarque-Bera Test   R    Chi^2  36.6432206 1.104144e-08
 Shapiro-Wilk Test  R    W       0.9967943 2.934125e-05
 Ljung-Box Test     R    Q(10)  12.0943005 2.787949e-01
 Ljung-Box Test     R    Q(15)  20.7596486 1.446860e-01
 Ljung-Box Test     R    Q(20)  22.3409037 3.223024e-01
 Ljung-Box Test     R^2  Q(10)  12.3893095 2.598459e-01
 Ljung-Box Test     R^2  Q(15)  12.9276394 6.078871e-01
 Ljung-Box Test     R^2  Q(20)  14.5756952 8.001500e-01
 LM Arch Test       R    TR^2   13.4251438 3.389111e-01

Information Criterion Statistics:
      AIC       BIC       SIC      HQIC
-5.818432 -5.804797 -5.818443 -5.813490
```

```r
# Residual plots
par(mfrow = c(1, 2))
acf(residuals(garch_sp500_arma_res), main = "ACF Residuals: SP500 ARMA & GARCH")
acf(residuals(garch_cac40_arma_res), main = "ACF Residuals: CAC40 ARMA & GARCH")
```




```r
# Ljung-Box tests on residuals
Box.test(residuals(garch_sp500_arma_res), lag = 20, type = "Ljung-Box")
```

```
    Box-Ljung test

data:  residuals(garch_sp500_arma_res)
```

```
X-squared = 30.657, df = 20, p-value = 0.05989
```

```r
Box.test(residuals(garch_cac40_arma_res), lag = 20, type = "Ljung-Box")
```

```
	Box-Ljung test

data:  residuals(garch_cac40_arma_res)
X-squared = 41.079, df = 20, p-value = 0.003639
```

For SP500, we observe that the ARMA + GARCH fit improves the model quality quite clearly: higher log-likelihood, lower AIC and no autocorrelation

On the contrary, CAC40 has an identical log-likelihood and AIC as before, with a p-value still too low (so still some significant autocorrelation). It could be good to tune ARMA better or to try another GARCH variant.

Question h)

> Use the garchAuto.R script in order to fit a GARCH on the residuals of the ARMA(p,q) from (g). Assess the quality of the fit.

```r
# Source the garchAuto.R file
source("data/Practical1/garchAuto.R")

# Fit best ARMA+GARCH model on SP500 residuals
best_garch_sp500 <- garchAuto(res_sp500, trace=TRUE)
```

```
Loading required package: parallel

   Analyzing (0,0,1,1) with sged distribution done.Good model. AIC = -6.577199, forecast: 0
   Analyzing (0,1,1,1) with sged distribution done.Good model. AIC = -6.576938, forecast: 2e-04
   Analyzing (0,2,1,1) with sged distribution done.Good model. AIC = -6.576852, forecast: 2e-04
   Analyzing (0,3,1,1) with sged distribution done.Good model. AIC = -6.577534, forecast: 1e-04
   Analyzing (0,4,1,1) with sged distribution done.Good model. AIC = -6.577077, forecast: 1e-04
   Analyzing (0,5,1,1) with sged distribution done.Good model. AIC = -6.578182, forecast: -2e-04
   Analyzing (1,0,1,1) with sged distribution done.Good model. AIC = -6.576932, forecast: 2e-04
   Analyzing (1,1,1,1) with sged distribution done.Bad model.
   Analyzing (1,2,1,1) with sged distribution done.Bad model.
   Analyzing (1,3,1,1) with sged distribution done.Good model. AIC = -6.577338, forecast: 2e-04
   Analyzing (1,4,1,1) with sged distribution done.Good model. AIC = -6.576873, forecast: 2e-04
   Analyzing (1,5,1,1) with sged distribution done.Good model. AIC = -6.578753, forecast: -1e-04
   Analyzing (2,0,1,1) with sged distribution done.Good model. AIC = -6.576831, forecast: 2e-04
   Analyzing (2,1,1,1) with sged distribution done.Bad model.
   Analyzing (2,2,1,1) with sged distribution done.Bad model.
   Analyzing (2,3,1,1) with sged distribution done.Good model. AIC = -6.578384, forecast: -4e-04
   Analyzing (2,4,1,1) with sged distribution done.Bad model.
   Analyzing (2,5,1,1) with sged distribution done.Bad model.
   Analyzing (3,0,1,1) with sged distribution done.Good model. AIC = -6.577463, forecast: 1e-04
   Analyzing (3,1,1,1) with sged distribution done.Good model. AIC = -6.577279, forecast: 2e-04
   Analyzing (3,2,1,1) with sged distribution done.Good model. AIC = -6.578381, forecast: -4e-04
```

```
Analyzing (3,3,1,1) with sged distribution done.Bad model.
Analyzing (3,4,1,1) with sged distribution done.Bad model.
Analyzing (3,5,1,1) with sged distribution done.Bad model.
Analyzing (4,0,1,1) with sged distribution done.Good model. AIC = -6.577022, forecast: 2e-04
Analyzing (4,1,1,1) with sged distribution done.Good model. AIC = -6.576817, forecast: 2e-04
Analyzing (4,2,1,1) with sged distribution done.Bad model.
Analyzing (4,3,1,1) with sged distribution done.Bad model.
Analyzing (4,4,1,1) with sged distribution done.Bad model.
Analyzing (4,5,1,1) with sged distribution done.Bad model.
Analyzing (5,0,1,1) with sged distribution done.Good model. AIC = -6.578058, forecast: -2e-04
Analyzing (5,1,1,1) with sged distribution done.Good model. AIC = -6.578805, forecast: -2e-04
Analyzing (5,2,1,1) with sged distribution done.Bad model.
Analyzing (5,3,1,1) with sged distribution done.Bad model.
Analyzing (5,4,1,1) with sged distribution done.Bad model.
Analyzing (5,5,1,1) with sged distribution done.Bad model.
```

```r
# Fit best ARMA+GARCH model on CAC40 residuals
best_garch_cac40 <- garchAuto(res_cac40, trace=TRUE)
```

```
Analyzing (0,0,1,1) with sged distribution done.Good model. AIC = -5.820957, forecast: -4e-04
Analyzing (0,1,1,1) with sged distribution done.Good model. AIC = -5.820384, forecast: -4e-04
Analyzing (0,2,1,1) with sged distribution done.Good model. AIC = -5.820133, forecast: -5e-04
Analyzing (0,3,1,1) with sged distribution done.Good model. AIC = -5.822429, forecast: -6e-04
Analyzing (0,4,1,1) with sged distribution done.Good model. AIC = -5.82211, forecast: -7e-04
Analyzing (0,5,1,1) with sged distribution done.Good model. AIC = -5.824134, forecast: -4e-04
Analyzing (1,0,1,1) with sged distribution done.Good model. AIC = -5.820385, forecast: -4e-04
Analyzing (1,1,1,1) with sged distribution done.Bad model.
Analyzing (1,2,1,1) with sged distribution done.Good model. AIC = -5.821597, forecast: -0.0014
Analyzing (1,3,1,1) with sged distribution done.Bad model.
Analyzing (1,4,1,1) with sged distribution done.Bad model.
Analyzing (1,5,1,1) with sged distribution done.Good model. AIC = -5.824193, forecast: -8e-04
Analyzing (2,0,1,1) with sged distribution done.Good model. AIC = -5.820152, forecast: -5e-04
Analyzing (2,1,1,1) with sged distribution done.Good model. AIC = -5.821675, forecast: -0.0015
Analyzing (2,2,1,1) with sged distribution done.Bad model.
Analyzing (2,3,1,1) with sged distribution done.Good model. AIC = -5.823722, forecast: -7e-04
Analyzing (2,4,1,1) with sged distribution done.Good model. AIC = -5.823997, forecast: -0.001
Analyzing (2,5,1,1) with sged distribution done.Good model. AIC = -5.824055, forecast: -2e-04
Analyzing (3,0,1,1) with sged distribution done.Good model. AIC = -5.822217, forecast: -5e-04
Analyzing (3,1,1,1) with sged distribution done.Good model. AIC = -5.821607, forecast: 4e-04
Analyzing (3,2,1,1) with sged distribution done.Good model. AIC = -5.823868, forecast: -8e-04
Analyzing (3,3,1,1) with sged distribution done.Bad model.
Analyzing (3,4,1,1) with sged distribution done.Bad model.
Analyzing (3,5,1,1) with sged distribution done.Bad model.
Analyzing (4,0,1,1) with sged distribution done.Good model. AIC = -5.821934, forecast: -5e-04
Analyzing (4,1,1,1) with sged distribution done.Good model. AIC = -5.822103, forecast: 3e-04
Analyzing (4,2,1,1) with sged distribution done.Bad model.
Analyzing (4,3,1,1) with sged distribution done.Bad model.
Analyzing (4,4,1,1) with sged distribution done.Bad model.
Analyzing (4,5,1,1) with sged distribution done.Bad model.
Analyzing (5,0,1,1) with sged distribution done.Good model. AIC = -5.823899, forecast: -2e-04
Analyzing (5,1,1,1) with sged distribution done.Good model. AIC = -5.824684, forecast: -7e-04
```

```
Analyzing (5,2,1,1) with sged distribution done.Bad model.
Analyzing (5,3,1,1) with sged distribution done.Bad model.
Analyzing (5,4,1,1) with sged distribution done.Bad model.
Analyzing (5,5,1,1) with sged distribution done.Bad model.
```

```r
# View model summaries
summary(best_garch_sp500)
```

```
Title:
 GARCH Modelling

Call:
 garchFit(formula = formula, data = data, cond.dist = ll$dist,
    trace = FALSE)

Mean and Variance Equation:
 data ~ arma(5, 1) + garch(1, 1)
<environment: 0x13ec26158>
 [data = data]

Conditional Distribution:
 sged

Coefficient(s):
         mu          ar1          ar2          ar3          ar4          ar5
-2.2520e-05   6.1560e-01  -5.1324e-05  -1.9771e-02   2.1914e-02  -3.3774e-02
        ma1        omega       alpha1        beta1         skew        shape
-6.3161e-01   3.8994e-07   5.2238e-02   9.4520e-01   1.0729e+00   1.3954e+00

Std. Errors:
 based on Hessian

Error Analysis:
         Estimate  Std. Error  t value Pr(>|t|)
mu     -2.252e-05   5.003e-05   -0.450 0.652641
ar1     6.156e-01   1.885e-01    3.265 0.001093 **
ar2    -5.132e-05   2.043e-02   -0.003 0.997996
ar3    -1.977e-02   1.974e-02   -1.001 0.316667
ar4     2.191e-02   2.052e-02    1.068 0.285454
ar5    -3.377e-02   1.918e-02   -1.761 0.078316 .
ma1    -6.316e-01   1.888e-01   -3.345 0.000823 ***
omega   3.899e-07   1.556e-07    2.506 0.012194 *
alpha1  5.224e-02   7.840e-03    6.663 2.69e-11 ***
beta1   9.452e-01   7.986e-03  118.360  < 2e-16 ***
skew    1.073e+00   2.364e-02   45.383  < 2e-16 ***
shape   1.395e+00   4.584e-02   30.442  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Log Likelihood:
 11811.09    normalized:  3.292748
```

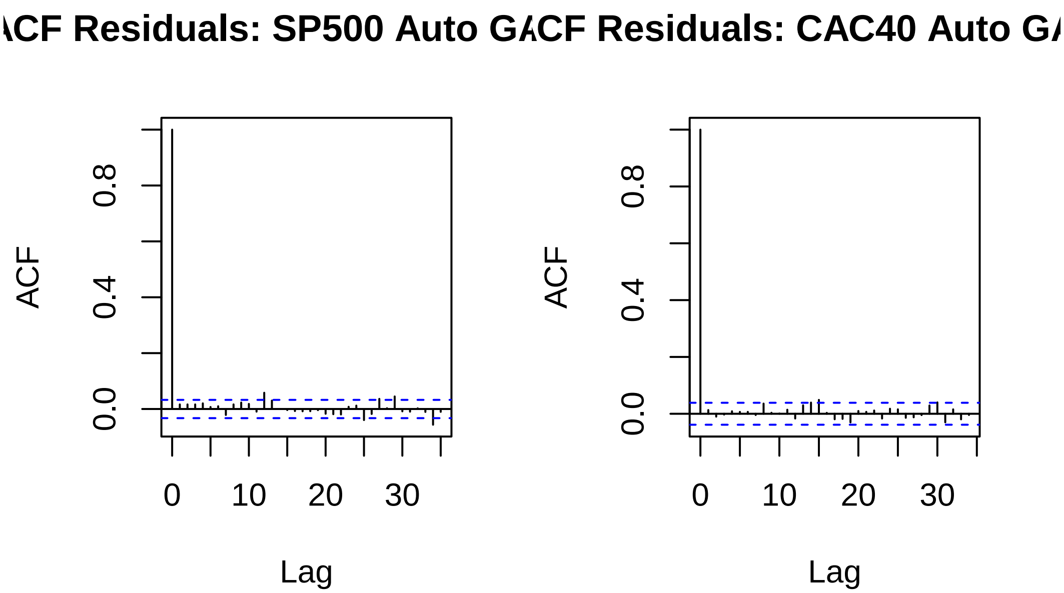
```
Description:
 Sun Jun 15 10:12:03 2025 by user:


Standardised Residuals Tests:
                                  Statistic     p-Value
 Jarque-Bera Test   R    Chi^2  766.0267687 0.000000000
 Shapiro-Wilk Test  R    W        0.9822839 0.000000000
 Ljung-Box Test     R    Q(10)   14.5209369 0.150528097
 Ljung-Box Test     R    Q(15)   31.5502756 0.007408813
 Ljung-Box Test     R    Q(20)   32.8242579 0.035269235
 Ljung-Box Test     R^2  Q(10)    7.8181492 0.646594554
 Ljung-Box Test     R^2  Q(15)   11.0400638 0.749749175
 Ljung-Box Test     R^2  Q(20)   15.6369602 0.738875811
 LM Arch Test       R    TR^2    10.1047177 0.606774224

Information Criterion Statistics:
      AIC       BIC       SIC      HQIC
-6.578805 -6.558113 -6.578827 -6.571430
```

```
summary(best_garch_cac40)
```

```
Title:
 GARCH Modelling

Call:
 garchFit(formula = formula, data = data, cond.dist = ll$dist,
    trace = FALSE)

Mean and Variance Equation:
 data ~ arma(5, 1) + garch(1, 1)
<environment: 0x13957b9e8>
 [data = data]

Conditional Distribution:
 sged

Coefficient(s):
         mu          ar1          ar2          ar3          ar4          ar5
-1.8718e-04   6.2888e-01  -1.3514e-02  -4.6209e-02   3.3830e-02  -5.3382e-02
        ma1        omega       alpha1        beta1         skew        shape
-6.3026e-01   2.0199e-06   6.5264e-02   9.2512e-01   1.1129e+00   1.7928e+00

Std. Errors:
 based on Hessian

Error Analysis:
         Estimate  Std. Error  t value Pr(>|t|)
mu     -1.872e-04   1.063e-04   -1.761  0.07831 .
ar1     6.289e-01   1.329e-01    4.731 2.23e-06 ***
```

```
ar2    -1.351e-02   2.371e-02   -0.570  0.56869
ar3    -4.621e-02   2.397e-02   -1.928  0.05390 .
ar4     3.383e-02   2.492e-02    1.357  0.17469
ar5    -5.338e-02   2.107e-02   -2.534  0.01127 *
ma1    -6.303e-01   1.317e-01   -4.784 1.72e-06 ***
omega   2.020e-06   7.125e-07    2.835  0.00458 **
alpha1  6.526e-02   8.804e-03    7.413 1.23e-13 ***
beta1   9.251e-01   9.948e-03   92.996  < 2e-16 ***
skew    1.113e+00   3.255e-02   34.186  < 2e-16 ***
shape   1.793e+00   7.472e-02   23.993  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Log Likelihood:
 7514.193    normalized:  2.917

Description:
 Sun Jun 15 10:12:49 2025 by user:


Standardised Residuals Tests:
                                 Statistic      p-Value
 Jarque-Bera Test   R    Chi^2  42.7453156 5.223633e-10
 Shapiro-Wilk Test  R    W       0.9961469 3.556087e-06
 Ljung-Box Test     R    Q(10)   8.3033881 5.992288e-01
 Ljung-Box Test     R    Q(15)  17.6818769 2.797593e-01
 Ljung-Box Test     R    Q(20)  19.6727478 4.785610e-01
 Ljung-Box Test     R^2  Q(10)  13.5760142 1.932238e-01
 Ljung-Box Test     R^2  Q(15)  14.0134143 5.245120e-01
 Ljung-Box Test     R^2  Q(20)  16.4207731 6.902011e-01
 LM Arch Test       R    TR^2   14.2477321 2.851692e-01

Information Criterion Statistics:
      AIC       BIC       SIC      HQIC
-5.824684 -5.797414 -5.824727 -5.814799
```

```r
# Check residual diagnostics
par(mfrow = c(1, 2))
acf(residuals(best_garch_sp500), main = "ACF Residuals: SP500 Auto GARCH")
acf(residuals(best_garch_cac40), main = "ACF Residuals: CAC40 Auto GARCH")
```




```r
# Ljung-Box test
Box.test(residuals(best_garch_sp500), lag=20, type="Ljung-Box")
```

```
	Box-Ljung test

data:  residuals(best_garch_sp500)
X-squared = 28.262, df = 20, p-value = 0.1033
```

```r
Box.test(residuals(best_garch_cac40), lag=20, type="Ljung-Box")
```

```
	Box-Ljung test

data:  residuals(best_garch_cac40)
X-squared = 22.765, df = 20, p-value = 0.3005
```

As a result, both SP500 and CAC40 now have the best fit compared to the previous fits we have tried. Residuals are now white noise for both indices (so no more autocorrelation) and volatility are well-captured.

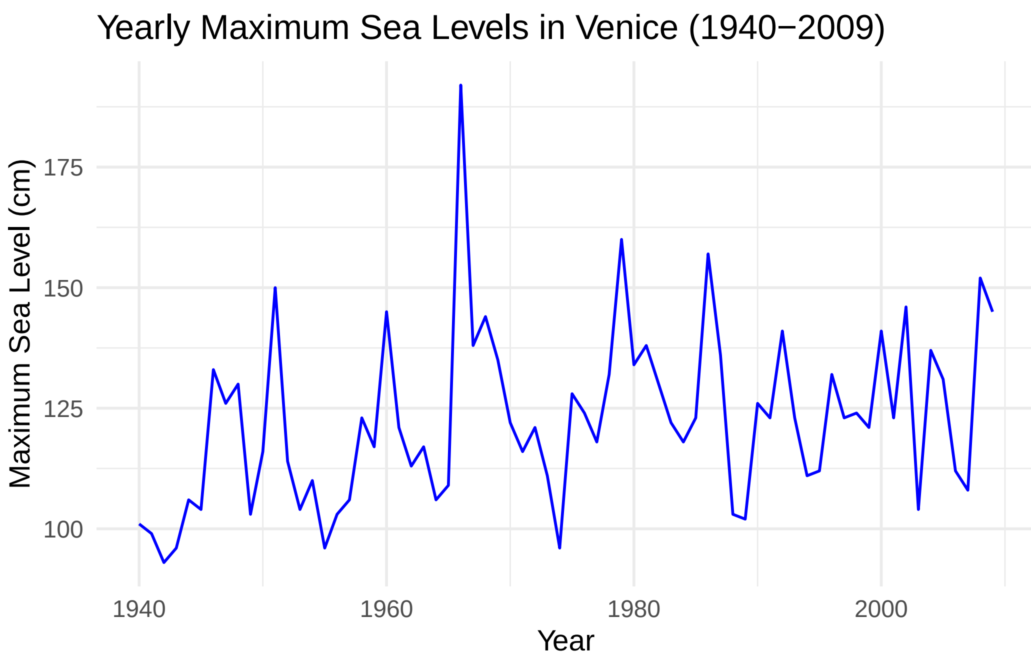
Practical 2, Part 1 - Venice

The `venice90` dataset can be found in the `VGAM` package.

Question a)

> Read in the data. Extract and represent the yearly max values from 1940 to 2009. What do you observe ?

```r
library(VGAM)
data(venice90)

# Transform venice90 into a data frame
venice90_df <- as.data.frame(venice90)

# Group by year and extract the maximum sea level per year between 1940 to 2009
yearly_max <- venice90_df %>%
  group_by(year) %>%
  summarise(max_sealevel = max(sealevel))

# Plot the yearly maximum sea levels
ggplot(yearly_max, aes(x = year, y = max_sealevel)) +
  geom_line(color = "blue") +
  labs(
    x = "Year",
    y = "Maximum Sea Level (cm)",
    title = "Yearly Maximum Sea Levels in Venice (1940-2009)"
  ) +
  theme_minimal()
```



We can observe some variability over the years and a slight upward trend, so the maximum levels in Venice seem to be increasing.

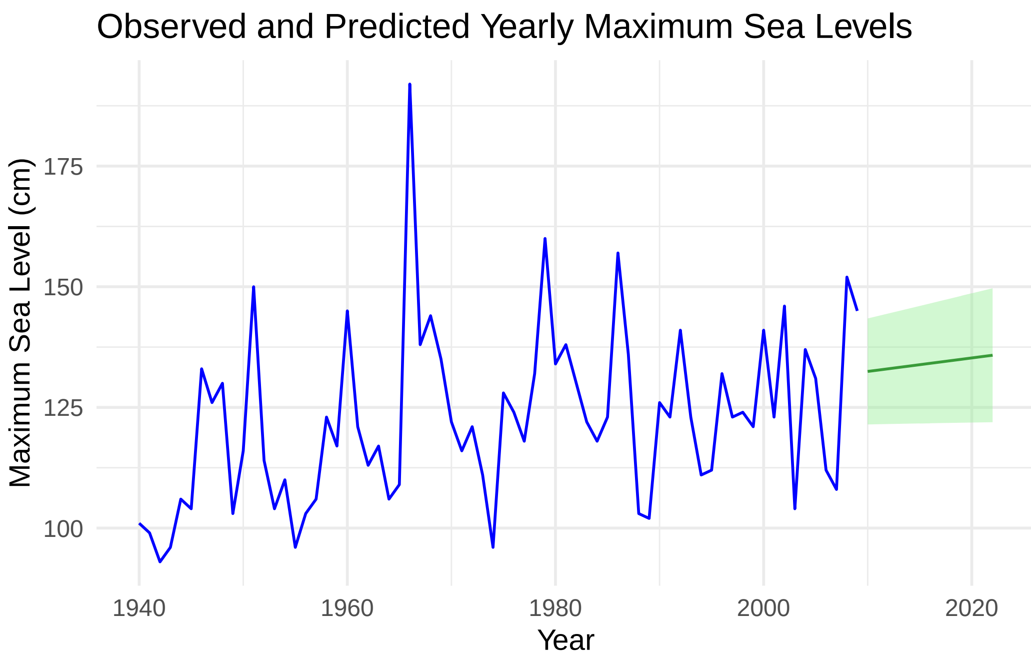
Question b)

> We are end of 2009 and would like to predict the yearly maximum values over the next 13 years (from 2010 to 2022). A naive approach consists of fitting a linear model on the observed yearly maxima and predict their values for 2010–2022. Proceed to this prediction and provide confidence intervals.

```r
# Fit linear model
model <- lm(max_sealevel ~ year, data = yearly_max)

# Predict for 2010-2022 with confidence intervals
future_years <- data.frame(year = 2010:2022)
pred <- predict(model, newdata = future_years, interval = "confidence", level = 0.99)

# Show predictions
cbind(future_years, pred)
```

```
   year      fit      lwr      upr
1  2010 132.4522 121.4683 143.4361
2  2011 132.7321 121.5137 143.9505
3  2012 133.0121 121.5576 144.4665
4  2013 133.2920 121.6002 144.9838
5  2014 133.5719 121.6414 145.5025
6  2015 133.8519 121.6813 146.0225
7  2016 134.1318 121.7200 146.5436
8  2017 134.4118 121.7576 147.0659
9  2018 134.6917 121.7942 147.5892
10 2019 134.9716 121.8298 148.1135
11 2020 135.2516 121.8644 148.6387
12 2021 135.5315 121.8982 149.1649
13 2022 135.8115 121.9311 149.6919
```

```r
# Combine predictions with years
pred_df <- cbind(future_years, as.data.frame(pred))

# Plot observed and predicted with confidence intervals
ggplot() +
  geom_line(data = yearly_max, aes(x = year, y = max_sealevel), color = "blue") +
  geom_line(data = pred_df, aes(x = year, y = fit), color = "darkgreen") +
  geom_ribbon(data = pred_df, aes(x = year, ymin = lwr, ymax = upr), fill = "lightgreen", alpha =
  labs(x = "Year", y = "Maximum Sea Level (cm)",
       title = "Observed and Predicted Yearly Maximum Sea Levels") +
  theme_minimal()
```

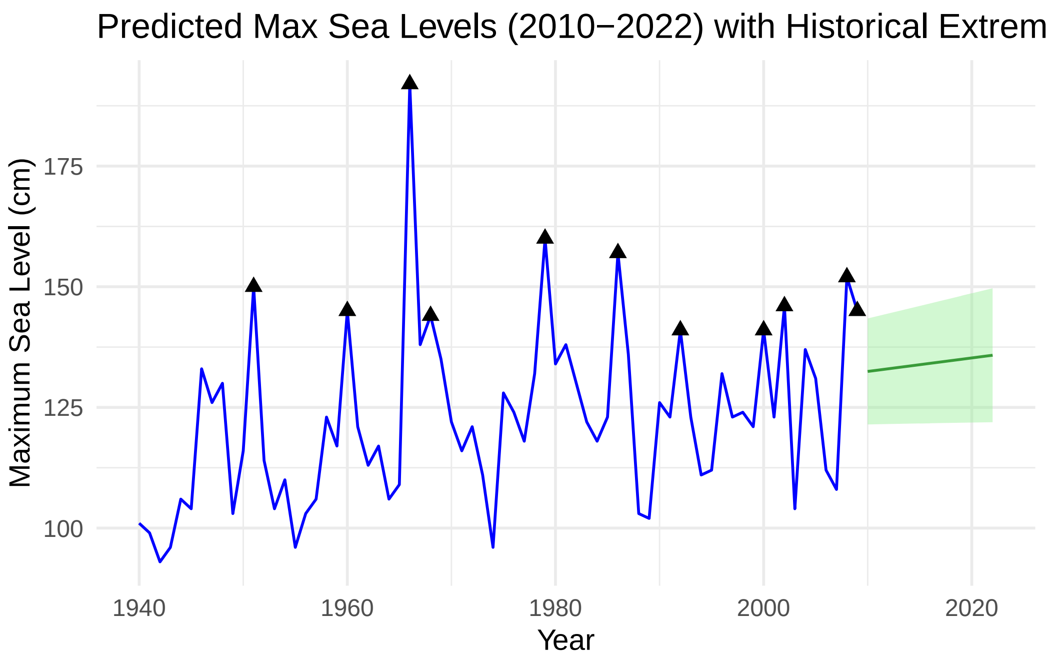


We used a confidence interval of 99% to predict for the years 2010 to 2022.

Question c)

Represent in the same graph the predicted yearly max values for the period 2010–2022, their pointwise confidence bounds and the observed values greater than 140 cm from the table below.

```r
# Observed values > 140 cm
extreme_vals <- yearly_max %>% filter(max_sealevel > 140)

# Plot everything together
ggplot() +
  # Historical data
  geom_line(data = yearly_max, aes(x = year, y = max_sealevel), color = "blue") +

  # Predictions
  geom_line(data = pred_df, aes(x = year, y = fit), color = "darkgreen") +
  geom_ribbon(data = pred_df, aes(x = year, ymin = lwr, ymax = upr),
              fill = "lightgreen", alpha = 0.4) +

  # Highlight points > 140 cm
  geom_point(data = extreme_vals, aes(x = year, y = max_sealevel),
             color = "black", size = 2, shape = 17) +  # triangle shape

  labs(x = "Year", y = "Maximum Sea Level (cm)",
       title = "Predicted Max Sea Levels (2010-2022) with Historical Extremes (>140 cm)") +
  theme_minimal()
```



This plot provides all the necessary information, from the historical data in the blue line, to the yearly maximum values with the red points, the dark green line being the prediction for 2010 to 2022, the light green area being the confidence intervals and finally, the black triangles being the values greater than 140cm.

Now we perform a risk analysis and because we are interested in the period 2010–2022, we want to calculate the 13-years return level., for each year.

Question d)

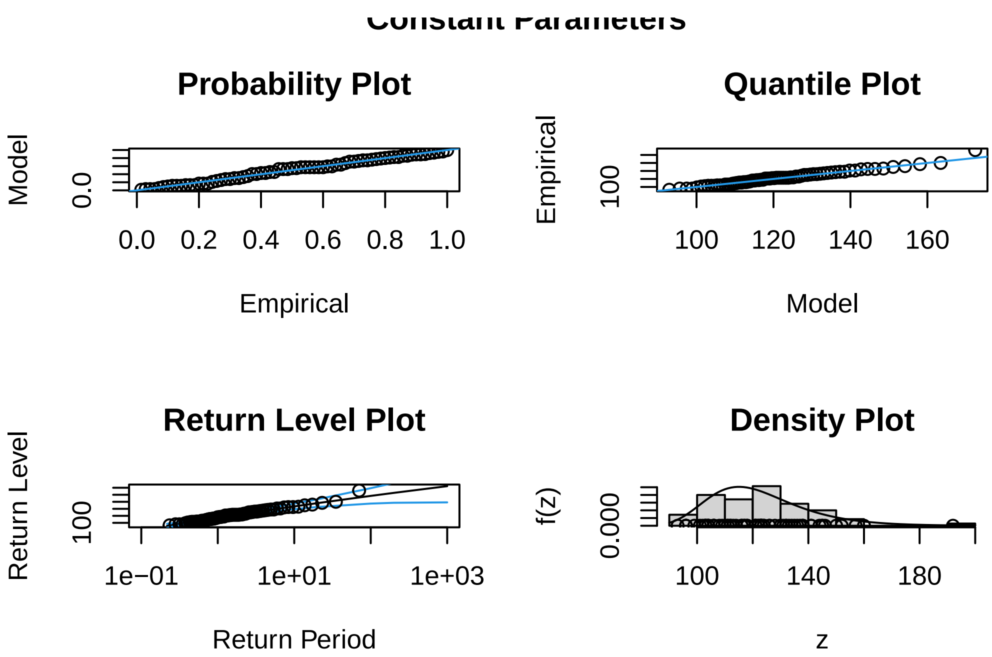
Fit a GEV a with constant parameters to the historical yearly max values. Fit a GEV with time varying location parameter. Compare the two embedded models using likelihood ratio test (LRT). Show diagnostic plots.

```
# Prepare the data (1940-2009), extract yearly maxima and center the year
venice90_df <- as_tibble(venice90) %>%
  filter(year >= 1940, year <= 2009) %>%
  group_by(year) %>%
  summarise(max_sealevel = max(sealevel), .groups = "drop") %>%
  mutate(year_centered = year - mean(year))  # mean-centred year

sea_levels <- venice90_df$max_sealevel                      # response
year_covariate <- matrix(venice90_df$year_centered, ncol = 1)  # 1-column covariate matrix

# Helper to rename GEV parameter estimates
name_gev_par <- function(fit, location_trend = FALSE, scale_trend = FALSE, shape_trend = FALSE) {
  names_vec <- c("location0")
  if (location_trend) names_vec <- c(names_vec, "location1")
  names_vec <- c(names_vec, "scale0")
  if (scale_trend) names_vec <- c(names_vec, "scale1")
  names_vec <- c(names_vec, "shape0")
  if (shape_trend) names_vec <- c(names_vec, "shape1")
```

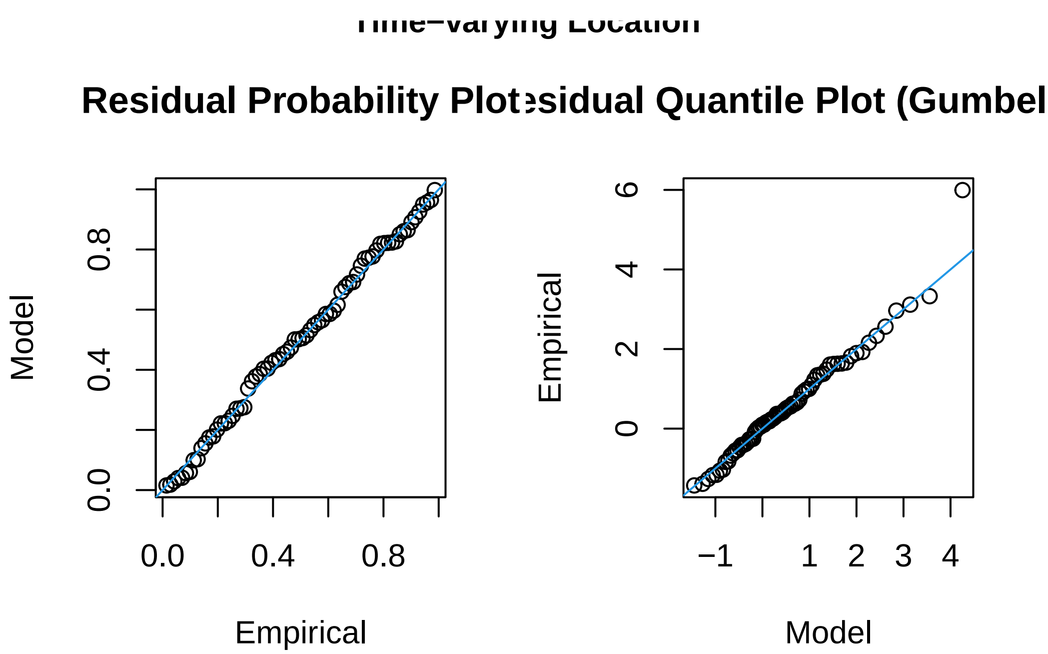
```r
  stopifnot(length(names_vec) == length(fit$mle))
  names(fit$mle) <- names(fit$se) <- names_vec
  fit
}

# Fit GEV with constant parameters
fit_const <- gev.fit(sea_levels, show = FALSE) |> name_gev_par()

# Fit GEV with time-varying location (trend on location)
fit_trend <- gev.fit(sea_levels, ydat = year_covariate, mul = 1, show = FALSE) |> name_gev_par(lo

# Likelihood Ratio Test
LRT <- 2 * (-fit_trend$nllh + fit_const$nllh)
pval <- pchisq(LRT, df = 1, lower.tail = FALSE)

# Select best model
if (pval < 0.05) {
  best_fit <- fit_trend
  best_model <- "Time-varying Location"
} else {
  best_fit <- fit_const
  best_model <- "Constant Parameters"
}
cat(sprintf("\n--- d) Likelihood-ratio test ---\nLRT = %.2f,  p = %.3f\nSelected model: %s\n",
            LRT, pval, best_model))
```

```
--- d) Likelihood-ratio test ---
LRT = 11.62,  p = 0.001
Selected model: Time-varying Location
```

```r
# Diagnostic plots
par(mfrow = c(2,2)); gev.diag(fit_const); title("Constant Parameters", outer = TRUE)
```




```r
par(mfrow = c(2,2)); gev.diag(fit_trend); title("Time-varying Location", outer = TRUE)
```



We fitted a constant and a time-varying model. The latter is better thanks to the low p-value and the log-likelihood of 11.62.The model looks overall okay, despite having some outliers which might influence it. There are no major pattern nor heteroskedasticity.

Question e)

> Add if necessary a time varying scale and or shape GEV parameter. Select the best model according to LRT.

```r
# Fit GEV models with additional time-varying parameters
fit_loc_scale <- gev.fit(sea_levels, ydat = year_covariate, mul = 1, sigl = 1,
                        show = FALSE) |> name_gev_par(location_trend = TRUE, scale_trend = TRUE)

fit_loc_shape <- gev.fit(sea_levels, ydat = year_covariate, mul = 1, shl = 1,
                        show = FALSE) |> name_gev_par(location_trend = TRUE, shape_trend = TRUE)

fit_loc_scale_shape <- gev.fit(sea_levels, ydat = year_covariate, mul = 1, sigl = 1, shl = 1,
                              show = FALSE) |> name_gev_par(location_trend = TRUE, scale_trend =

# Collect log-likelihoods for comparison
log_likelihoods <- purrr::map_dbl(
  list(fit_const, fit_trend, fit_loc_scale, fit_loc_shape, fit_loc_scale_shape),
  \(fit) -fit$nllh
)

names(log_likelihoods) <- c("const", "location", "location+scale", "location+shape", "location+sc

# Likelihood Ratio Test Table
lrt_tbl <- tibble(
  comparison = c("location vs const",
                 "location+scale vs location",
                 "location+shape vs location",
                 "location+scale+shape vs location+scale"),
  LR = c(2 * (log_likelihoods["location"] - log_likelihoods["const"]),
         2 * (log_likelihoods["location+scale"] - log_likelihoods["location"]),
         2 * (log_likelihoods["location+shape"] - log_likelihoods["location"]),
         2 * (log_likelihoods["location+scale+shape"] - log_likelihoods["location+scale"])),
  df = 1,
  p = pchisq(LR, df, lower.tail = FALSE)
)

print(lrt_tbl, digits = 3)
```

```
# A tibble: 4 x 4
  comparison                                LR    df        p
  <chr>                                  <dbl> <dbl>    <dbl>
1 location vs const                     11.6       1 0.000651
2 location+scale vs location             0.892     1 0.345
3 location+shape vs location             5.03      1 0.0250
4 location+scale+shape vs location+scale 5.94      1 0.0148
```

```r
# Start with location trend model as baseline
best_fit <- fit_trend
best_model_name <- "location"

# Check if adding scale improves the model significantly
if (lrt_tbl$p[2] < 0.05) {
  best_fit <- fit_loc_scale
  best_model_name <- "location+scale"
```

```r
}

# If scale was not added, check if shape improves the model
if (best_model_name == "location" && lrt_tbl$p[3] < 0.05) {
  best_fit <- fit_loc_shape
  best_model_name <- "location+shape"
}

# If both location and scale are in the model, check if adding shape improves it further
if (best_model_name == "location+scale" && lrt_tbl$p[4] < 0.05) {
  best_fit <- fit_loc_scale_shape
  best_model_name <- "location+scale+shape"
}

cat("\nSelected model:", best_model_name, "\n")
```

```
Selected model: location+shape
```

The best model includes time-varying location and shape parameters. The addition of a time-varying scale is not necessary based on the LRT. This model provides the best fit and should be used for further analysis or prediction.

Question f) + g)

f) Predict the 13-years return level, each year from 2010 to 2022.
g) Calculate confidence bands for these predictions.

```r
# Extract parameter value by name (return 0 if not found)
get_param <- function(params, name) {
  ifelse(name %in% names(params), params[[name]], 0)
}

# Compute model parameters at covariate value z
get_gev_parameters <- function(fit, z) {
  p <- fit$mle
  location0 <- get_param(p, "location0"); location1 <- get_param(p, "location1")
  scale0    <- get_param(p, "scale0");    scale1    <- get_param(p, "scale1")
  shape0    <- get_param(p, "shape0");    shape1    <- get_param(p, "shape1")

  list(
    location = location0 + location1 * z,
    scale    = scale0    + scale1    * z,
    shape    = shape0    + shape1    * z
  )
}

# Compute 13-year return level using GEV parameters
return_level <- function(location, scale, shape, m = 13) {
  p <- 1 - 1/m
  if (abs(shape) < 1e-6) {
```

```r
    location - scale * log(-log(p))  # Gumbel case
  } else {
    location + (scale / shape) * ((-log(p))^(-shape) - 1)
  }
}

# Delta method standard error for return level at covariate z
return_level_se <- function(fit, z, m = 13) {
  params <- get_gev_parameters(fit, z)
  location <- params$location
  scale    <- params$scale
  shape    <- params$shape
  p        <- 1 - 1/m

  if (abs(shape) < 1e-6) {
    dloc <- 1
    dsca <- -log(-log(p))
    dshp <- 0
  } else {
    A    <- (-log(p))^(-shape) - 1
    dloc <- 1
    dsca <- A / shape
    dshp <- -scale / shape^2 * A + scale / shape * (-log(p))^(-shape) * log(-log(p))
  }

  # Gradient vector
  grad <- setNames(numeric(length(fit$mle)), names(fit$mle))
  grad["location0"] <- dloc
  if ("location1" %in% names(grad)) grad["location1"] <- dloc * z
  grad["scale0"]    <- dsca
  if ("scale1" %in% names(grad)) grad["scale1"] <- dsca * z
  grad["shape0"]    <- dshp
  if ("shape1" %in% names(grad)) grad["shape1"] <- dshp * z

  # Standard error via delta method
  sqrt(as.numeric(t(grad) %*% fit$cov %*% grad))
}

# Predictions for 2010 to 2022
years_future <- 2010:2022
z_future     <- years_future - mean(venice90_df$year)  # center future years

predicted_return_levels <- map2_dfr(years_future, z_future, \(year, z) {
  params <- get_gev_parameters(best_fit, z)
  rl     <- return_level(params$location, params$scale, params$shape, m = 13)
  se     <- return_level_se(best_fit, z, m = 13)

  tibble(
    year = year,
    return_level = rl,
    lower_bound  = rl - qnorm(0.975) * se,
```

```
    upper_bound  = rl + qnorm(0.975) * se
  )
})

# Print the predictions
print(as.data.frame(predicted_return_levels), digits = 5)
```

```
   year return_level lower_bound upper_bound
1  2010       147.78      137.25      158.32
2  2011       147.87      137.72      158.02
3  2012       147.96      138.17      157.75
4  2013       148.06      138.60      157.52
5  2014       148.17      139.00      157.33
6  2015       148.27      139.38      157.17
7  2016       148.39      139.73      157.04
8  2017       148.51      140.07      156.94
9  2018       148.63      140.39      156.87
10 2019       148.75      140.68      156.82
11 2020       148.88      140.96      156.81
12 2021       149.02      141.22      156.82
13 2022       149.16      141.47      156.85
```

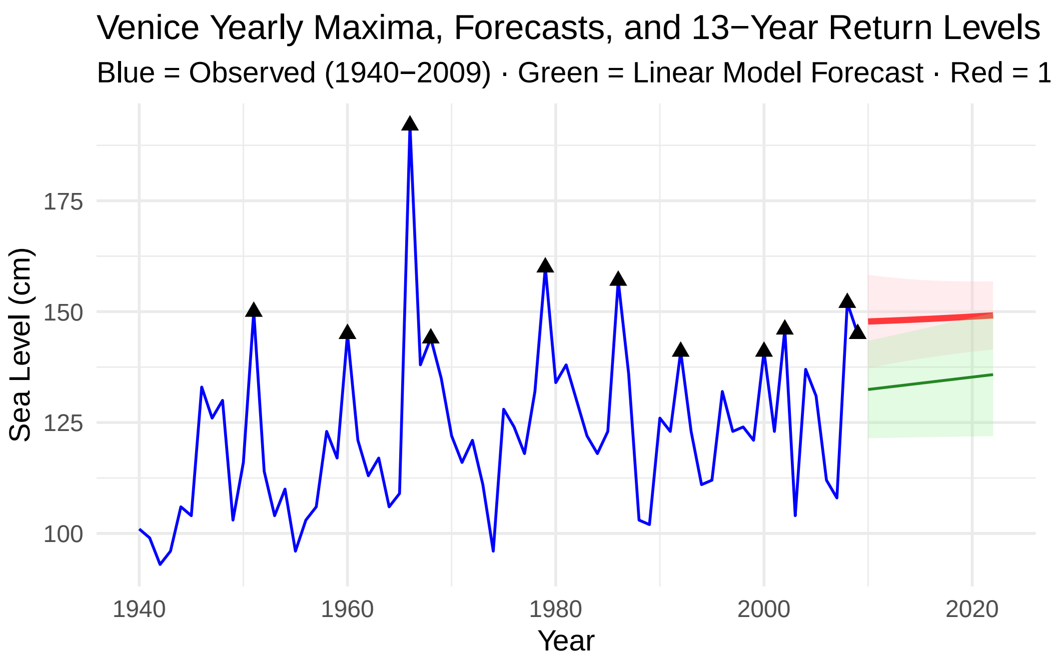
For each year from 2010 to 2022, the estimated 13-year return level gradually increases from approximately 147.78 cm to 149.16 cm. This indicates a slight upward trend in extreme sea level risk over time. The 95% confidence intervals range from about 137–158 cm in 2010 to 141–157 cm in 2022, showing that while uncertainty remains, the expected extremes are becoming higher. This trend supports the idea that extreme sea level events in Venice are becoming more likely and potentially more severe over time.

Question h)

> Represent in the same graph your predictions of the 13-years return levels, their pointwise confidence intervals, the predicted yearly max values from the linear model and the observed values greater than 140 cm from the table below.

```
# Linear model forecasts
linear_forecast_df <- pred_df %>%
  rename(lower_ci_linear = lwr, upper_ci_linear = upr, predicted_linear = fit)

# Observed extremes > 140 cm
extreme_values <- venice90_df %>% filter(max_sealevel > 140)

# Plot observed, linear forecast, and GEV return levels
ggplot() +
  geom_line(data = venice90_df, aes(x = year, y = max_sealevel), color = "blue") +
  geom_line(data = predicted_return_levels, aes(x = year, y = return_level), color = "red", linew
  geom_ribbon(data = predicted_return_levels, aes(x = year, ymin = lower_bound, ymax = upper_boun
  geom_line(data = linear_forecast_df, aes(x = year, y = predicted_linear), color = "darkgreen")
  geom_ribbon(data = linear_forecast_df, aes(x = year, ymin = lower_ci_linear, ymax = upper_ci_li
  geom_point(data = extreme_values, aes(x = year, y = max_sealevel), shape = 17, size = 2) +
  labs(
```

```
    x = "Year", y = "Sea Level (cm)",
    title = "Venice Yearly Maxima, Forecasts, and 13-Year Return Levels",
    subtitle = "Blue = Observed (1940-2009) · Green = Linear Model Forecast · Red = 13-Year Retur
  ) +
  theme_minimal()
```



Question i)

> Broadly speaking, each year, there is a chance of 1/13 that the observed value is above the 13-years return level. Comment the results for both the linear model prediction and GEV approach. Note that 12 of the 20 events occurred in the 21st century.

While both models provide useful insights, the linear model clearly underestimates extremes and provides overly narrow confidence intervals. The GEV approach, especially with time-varying parameters, is more suited for modeling extremes and gives a more realistic picture of sea level risk. However, even the GEV predictions fall short of the most recent high events, such as 2.04m in 2022, indicating that the system is non-stationary and that risk is increasing over time. This shift is emphasized by the concentration of extreme events in the 21st century, suggesting that return periods are shortening and that what was once a 13-year event may now be happening more frequently.

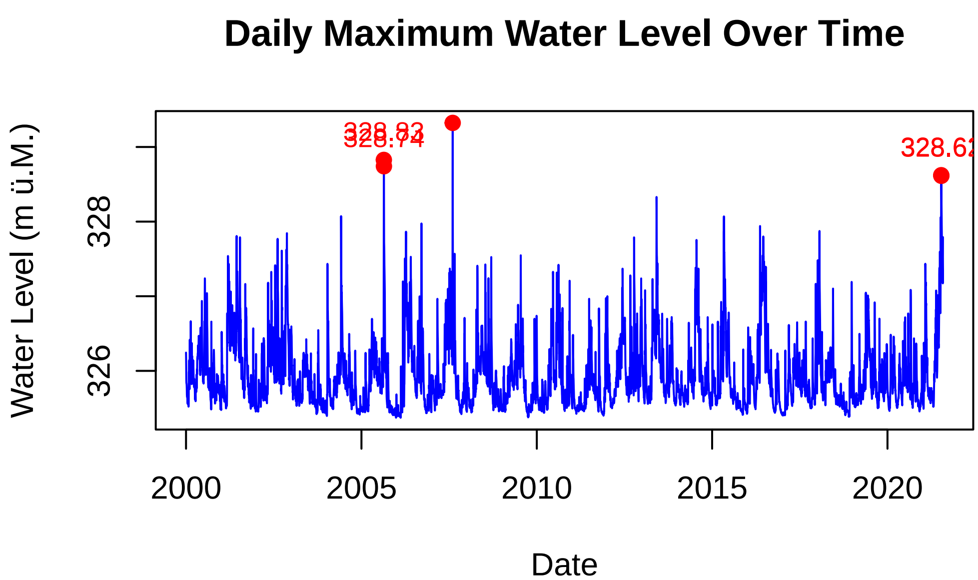
Practical 2, Part 2 - Nuclear Reactors

Question a)

> Read in the data. Display a time series plot of the water level across the data range and try to identify times of highest levels.

```
# Load the Rdata file
load("data/Practical2/niveau.Rdata")
```

```r
# Convert Zeitstempel to Date
niveau$Zeitstempel <- as.Date(niveau$Zeitstempel)

par(mfrow = c(1,1))
# Plot the time series
plot(niveau$Zeitstempel, niveau$Wert, type = "l", col = "blue",
     xlab = "Date", ylab = "Water Level (m ü.M.)",
     main = "Daily Maximum Water Level Over Time")

# Highlight top 5 highest water levels
top5 <- niveau[order(-niveau$Wert), ][1:5, ]
points(top5$Zeitstempel, top5$Wert, col = "red", pch = 19)
text(top5$Zeitstempel, top5$Wert, labels = round(top5$Wert, 2),
     pos = 3, cex = 0.8, col = "red")
```




```r
print(top5)
```

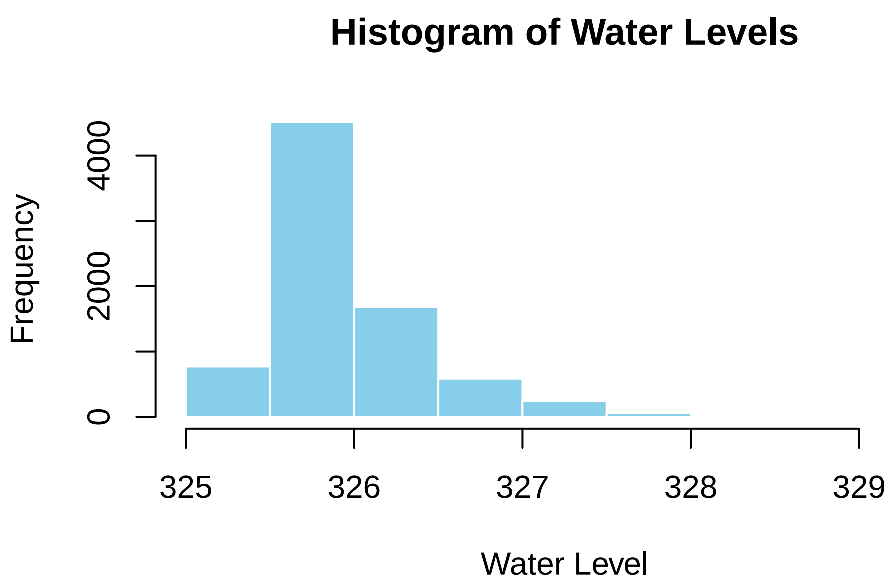
```
                  Stationsname Stationsnummer Parameter   Zeitreihe
2778 Untersiggenthal, Stilli           2205     Pegel Tagesmaxima
2061 Untersiggenthal, Stilli           2205     Pegel Tagesmaxima
2062 Untersiggenthal, Stilli           2205     Pegel Tagesmaxima
7865 Untersiggenthal, Stilli           2205     Pegel Tagesmaxima
7866 Untersiggenthal, Stilli           2205     Pegel Tagesmaxima
     Parametereinheit Gewässer Zeitstempel Zeitpunkt_des_Auftretens    Wert
2778           m ü.M.     Aare  2007-08-09      2007-08-09 11:15:00 329.323
2061           m ü.M.     Aare  2005-08-22      2005-08-22 18:05:00 328.827
2062           m ü.M.     Aare  2005-08-23      2005-08-23 08:35:00 328.742
7865           m ü.M.     Aare  2021-07-14      2021-07-14 07:45:00 328.622
7866           m ü.M.     Aare  2021-07-15      2021-07-15 15:05:00 328.614
                         Freigabestatus
2778    Freigegeben, validierte Daten
```

```
2061    Freigegeben, validierte Daten
2062    Freigegeben, validierte Daten
7865 Freigegeben, provisorische Daten
7866 Freigegeben, provisorische Daten
```

Question b)

Now display a histogram of the water levels. What do you observe about the distribution?

```r
# Simple histogram of water levels
hist(niveau$Wert,
     main = "Histogram of Water Levels",
     xlab = "Water Level",
     col = "skyblue",
     border = "white")
```



The distribution is right-skewed. Most levels are concentrated between 325 and 326. Extreme levels such as above 327 are rare yet still present. These can represent potential flood events or unusual conditions.

The FOEN plans for several degrees of risk. In this assignment, we focus on two risk levels: 50-year events and 100-year events.

Question c)

Explain how you would model the high water levels using a peaks-over-threshold approach.

```r
# Calculate the 95th percentile threshold
threshold <- quantile(niveau$Wert, 0.99)

# Extract exceedances above the threshold
exceedances <- niveau$Wert[niveau$Wert > threshold]
```

```r
# Print threshold and number of exceedances
cat("99% threshold:", threshold, "\n")
```

```
99% threshold: 327.5054
```

```r
cat("Number of exceedances:", length(exceedances), "\n")
```

```
Number of exceedances: 79
```

Using a Peaks-over-Threshold approach, we set a threshold above which the values are considered extreme. This threshold should be high enough to focus only on rare exceedences, but not to high to avoid having too few exceedences. Here, the threshold is set at the 99th percentile, which is at 327.5054 (so around 327.51) meters. The exceedances are modeled using the Generalized Pareto Distribution, suitable for a skewed distribution. In this case, the POT approach is useful as there are a lot of non-extreme values. This approach thus focuses only on the extreme events to assess a better statistical efficiency, especially with daily data over many years.

Question d)

> Comment on the aspect of clustering of extremes. How do you propose to measure and deal with clustering of the daily water levels?

```r
# Set time gap for declustering (e.g., 3 days)
run_length <- 3

# Identify dates of exceedances
exceed_dates <- niveau$Zeitstempel[niveau$Wert > threshold]

# Sort dates
exceed_dates <- sort(exceed_dates)

# Initialize clusters
clusters <- list()
current_cluster <- c(exceed_dates[1])

for (i in 2:length(exceed_dates)) {
  if (as.numeric(exceed_dates[i] - tail(current_cluster, 1)) <= run_length) {
    current_cluster <- c(current_cluster, exceed_dates[i])
  } else {
    clusters <- append(clusters, list(current_cluster))
    current_cluster <- c(exceed_dates[i])
  }
}
clusters <- append(clusters, list(current_cluster))

# Extract cluster maxima
cluster_maxima <- sapply(clusters, function(cluster_dates) {
  max(niveau$Wert[niveau$Zeitstempel %in% cluster_dates])
})
```

```r
# Results
cat("Number of exceedances before declustering:", length(exceedances), "\n")
```

```
Number of exceedances before declustering: 79
```

```r
cat("Number of cluster maxima (after declustering):", length(cluster_maxima), "\n")
```

```
Number of cluster maxima (after declustering): 28
```

Clustering extremes uses runs methods. We keep only one peak per cluster, which makes the exceedances more independent and suitable for modelling.

Question e)

> Perform the analysis you suggest in c) and d) and compute the 50- and 100-year return levels. Explain your choice of threshold and provide an estimate of uncertainty for the return levels. Note: take care to compute the return level in yearly terms.

Using the POT approach:

```r
# Fit GPD model
fit <- gpd.fit(cluster_maxima, threshold)
```

```
$threshold
     99%
327.5054

$nexc
[1] 28

$conv
[1] 0

$nllh
[1] 0.8853974

$mle
[1] 0.3366714 0.1202787

$rate
[1] 1

$se
[1] 0.1009116 0.2337588
```

```r
# Basic info
sigma <- fit$mle[1]
xi <- fit$mle[2]
cov_mat <- fit$cov

# Estimate exceedance rate per year
years <- as.numeric(difftime(max(niveau$Zeitstempel), min(niveau$Zeitstempel), units = "days")) /
lambda <- length(cluster_maxima) / years

# Return level function
return_level <- function(T, sigma, xi) {
  threshold + (sigma / xi) * ((T * lambda)^xi - 1)
}

# Simulate 1000 sets of parameters
set.seed(1)
params <- mvrnorm(1000, mu = c(sigma, xi), Sigma = cov_mat)

# Calculate return levels
rl_50 <- apply(params, 1, function(p) return_level(50, p[1], p[2]))
rl_100 <- apply(params, 1, function(p) return_level(100, p[1], p[2]))

# Get point estimates
rl_50_est <- return_level(50, sigma, xi)
rl_100_est <- return_level(100, sigma, xi)

# Confidence intervals
ci_50   <- quantile(rl_50,  c(0.025, 0.975))
ci_100  <- quantile(rl_100, c(0.025, 0.975))

# Print nicely
cat("50-year return level:", round(rl_50_est, 2), "\n")
```

```
50-year return level: 329.33
```

```r
cat("95% CI:", round(ci_50[1], 2), "-", round(ci_50[2], 2), "\n\n")
```

```
95% CI: 328.37 - 331.62
```

```r
cat("100-year return level:", round(rl_100_est, 2), "\n")
```

```
100-year return level: 329.73
```

```r
cat("95% CI:", round(ci_100[1], 2), "-", round(ci_100[2], 2), "\n")
```

```
95% CI: 328.43 - 333.53
```

The threshold is the 99th percentile to capture the extremes, have a balance between bias and variance and to have an adequate sample size to fit a GPD

Question f)

> Explain the drawbacks and advantages of using a block maxima method instead of the one used in c)-e).

The Block Maxima method selects the maximum observation from a given time interval, but uses only one observation per block which leads to an inefficient use of the data. The POT approach uses all the values above the given threshold and handles clustering well via declustering. It is more efficient and flexible especially when extreme events happen in clusters. Thus, the POT approach is more precise and provides more information on the behavior of extreme events.

Practical 2, Part 3 - Night temperatures in Lausanne

Question a)

> Read in the data for the daily night maximum temperatures in Lausanne. Subset the summer months (June to September).

```r
# Read the CSV files
nightmax <- read_csv("data/Practical2/nightmax.csv", show_col_types = FALSE) %>%
  select(-1) %>%                                    # drop left-hand index col
  rename(tmax = `night.max`) %>%
  mutate(date = ymd(date))
```

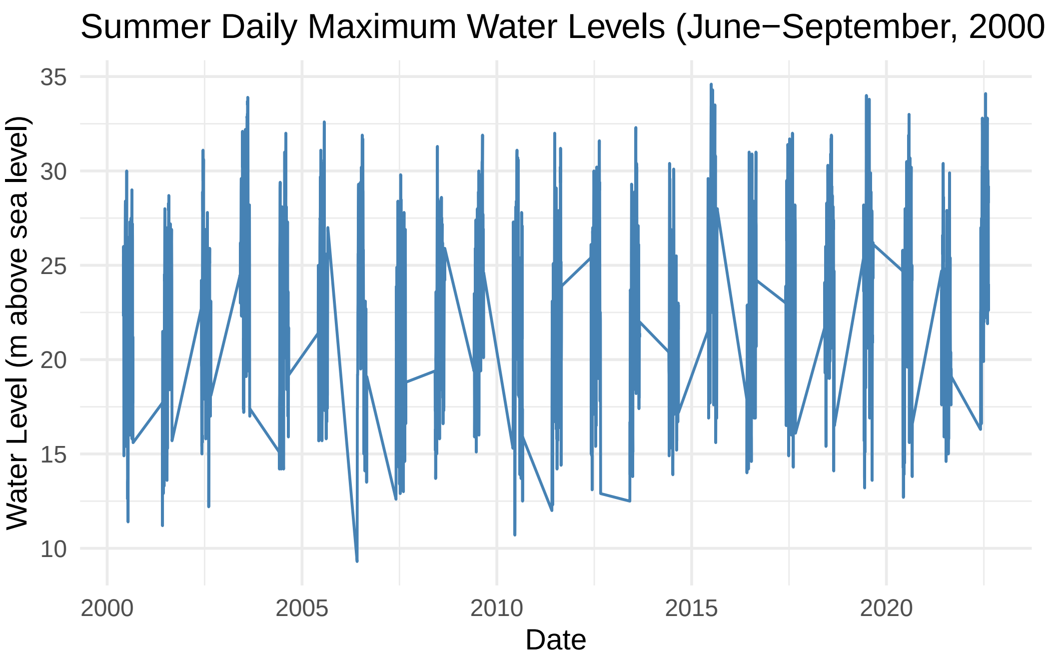
```
New names:
* `` -> `...1`
```

```r
nightmin <- read_csv("data/Practical2/nightmin.csv", show_col_types = FALSE) %>%
  select(-1) %>%
  rename(tmin = `night.min`) %>%
  mutate(date = ymd(date))
```

```
New names:
* `` -> `...1`
```

```r
# Summer (June-August) maxima and winter (Dec-Feb) minima
summer_max <- nightmax %>%
  filter(month(date) %in% 6:8, !is.na(tmax))

winter_min <- nightmin %>%
  filter(month(date) %in% c(12, 1, 2), !is.na(tmin))

# Plot summer daily maximum water levels
ggplot(summer_max, aes(x = date, y = tmax)) +
  geom_line(color = "steelblue") +
  labs(title = "Summer Daily Maximum Water Levels (June-September, 2000-2021)",
       x = "Date",
       y = "Water Level (m above sea level)") +
```

```
  theme_minimal()
```



We are doing the same process for minimum to asnwer question e.

Question b)

Assess whether extremes of the subsetted series in (a) occur in cluster.

```
# 95th-percentile thresholds
u_max <- quantile(summer_max$tmax, 0.95, na.rm = TRUE)
u_min <- quantile(-winter_min$tmin, 0.95, na.rm = TRUE)  # negate minima

ei_max <- extremalindex(summer_max$tmax,
                        threshold  = u_max,
                        method     = "runs",
                        run.length = 3)

ei_min <- extremalindex(-winter_min$tmin,
                        threshold  = u_min,
                        method     = "runs",
                        run.length = 3)

print(ei_max)
```

```
Runs Estimator for the Extremal Index
   extremal.index number.of.clusters         run.length
        0.4019608         41.0000000          3.0000000
```

```
print(ei_min)
```

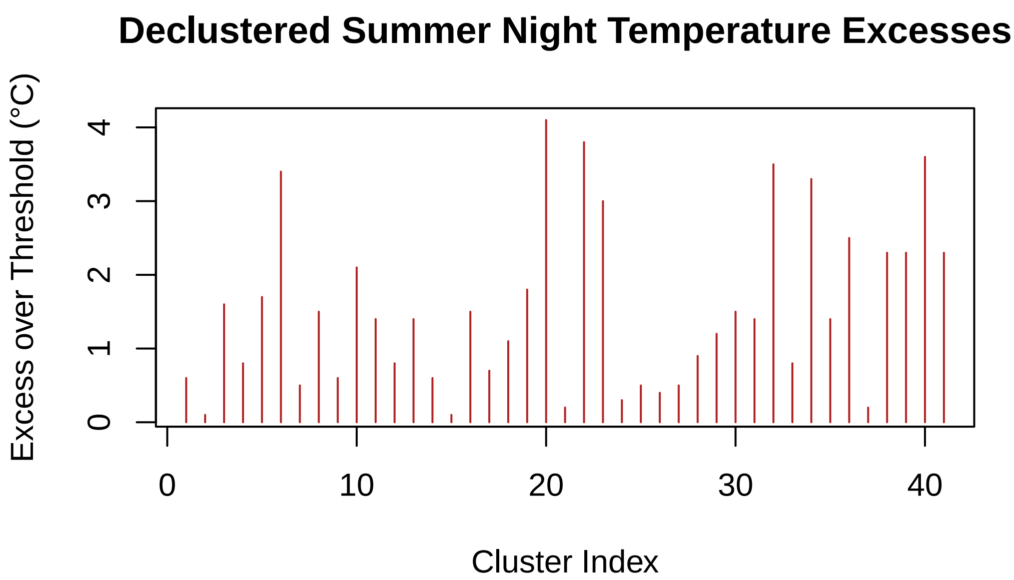
```
Runs Estimator for the Extremal Index
   extremal.index number.of.clusters         run.length
        0.3673469         36.0000000          3.0000000
```

The obtained extremal index is 0.402, which is lower than 1. This suggests that extreme night temperatures during summer in Lausanne tend to occur in clusters rather than being isolated. This means that if you observe one extremely hot night, there is a higher chance that other extreme nights will follow shortly, such as during a heatwave for example.

Question c)

> Decluster the data from (a) using a suitable threshold. Plot the resulting declustered data. (Hint: you may want to use the extRemes package.)

```r
# Retain cluster peaks only (r = 3)
dc_max <- decluster(summer_max$tmax,
                    threshold = u_max,
                    method    = "runs",
                    r         = 3)

dc_min <- decluster(-winter_min$tmin,
                    threshold = u_min,
                    method    = "runs",
                    r         = 3)

# Excesses over threshold
excess_max <- dc_max[dc_max > u_max] - u_max
excess_min <- dc_min[dc_min > u_min] - u_min

plot(excess_max, type = "h", col = "firebrick",
     main = "Declustered Summer Night Temperature Excesses",
     xlab = "Cluster Index", ylab = "Excess over Threshold (°C)")
```

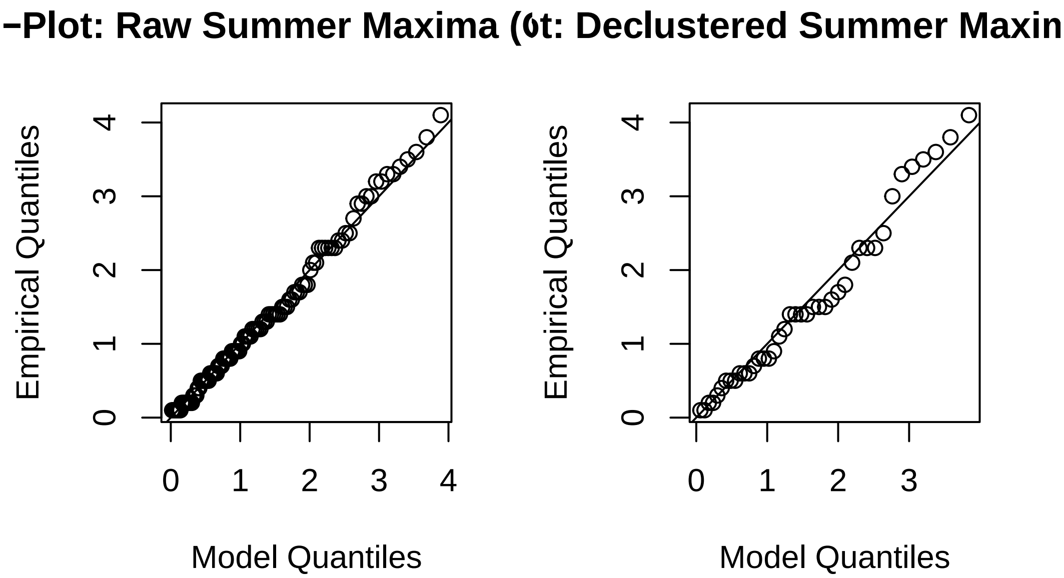


After declustering the extreme summer night temperatures using the 95th percentile threshold and a 3-day run length, we isolated 42 independent exceedances above the threshold. The resulting plot of declustered excesses reveals a wide range of magnitudes, with some cluster peaks exceeding 4°C above the threshold. This confirms the presence of significant and varied extreme temperature events, now stripped of temporal dependence.

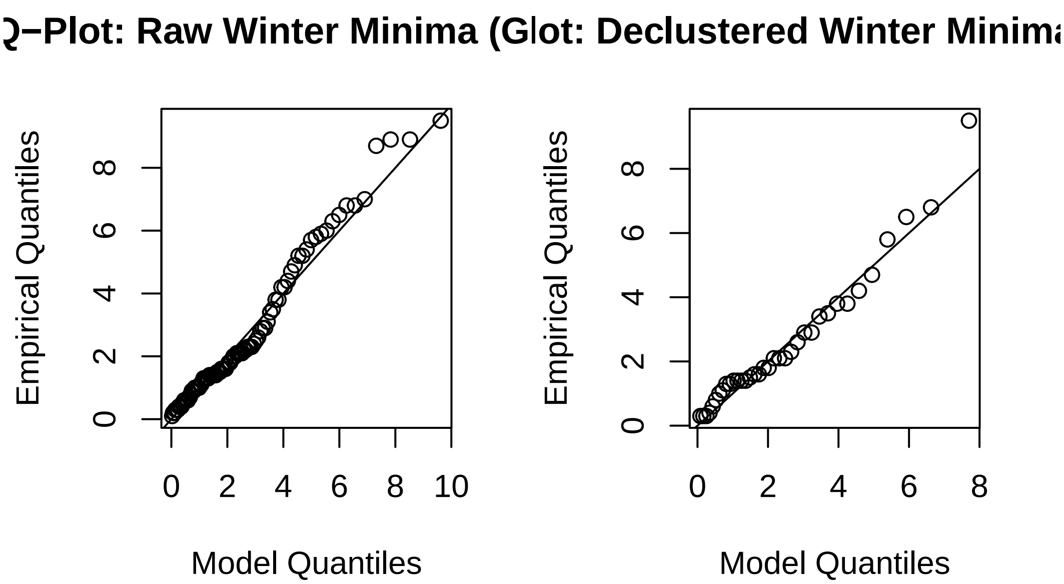
Question d)

Fit a GPD to the data, both raw and declustered. Assess the quality of the fit.

```
fit_max_raw <- fevd(summer_max$tmax[summer_max$tmax > u_max] - u_max,
                    type = "GP", threshold = 0, method = "MLE")

fit_max_dc  <- fevd(excess_max, type = "GP", threshold = 0, method = "MLE")

fit_min_raw <- fevd((-winter_min$tmin)[-winter_min$tmin > u_min] - u_min,
                    type = "GP", threshold = 0, method = "MLE")

fit_min_dc  <- fevd(excess_min, type = "GP", threshold = 0, method = "MLE")

par(mfrow = c(1,2))
plot(fit_max_raw, type = "qq", main = "QQ-Plot: Raw Summer Maxima (GPD Fit)")
plot(fit_max_dc, type = "qq", main = "QQ-Plot: Declustered Summer Maxima (GPD Fit)")
```




```r
plot(fit_min_raw, type = "qq", main = "QQ-Plot: Raw Winter Minima (GPD Fit)")
plot(fit_min_dc, type = "qq", main = "QQ-Plot: Declustered Winter Minima (GPD Fit)")
```



Despite the raw model having points more aligned in the QQ-plot, the declustered model is theoretically better due to the lower AIC. The deviations in the QQ-plot for the declustered can be explained due to the smaller sample than the raw data.

Question e)

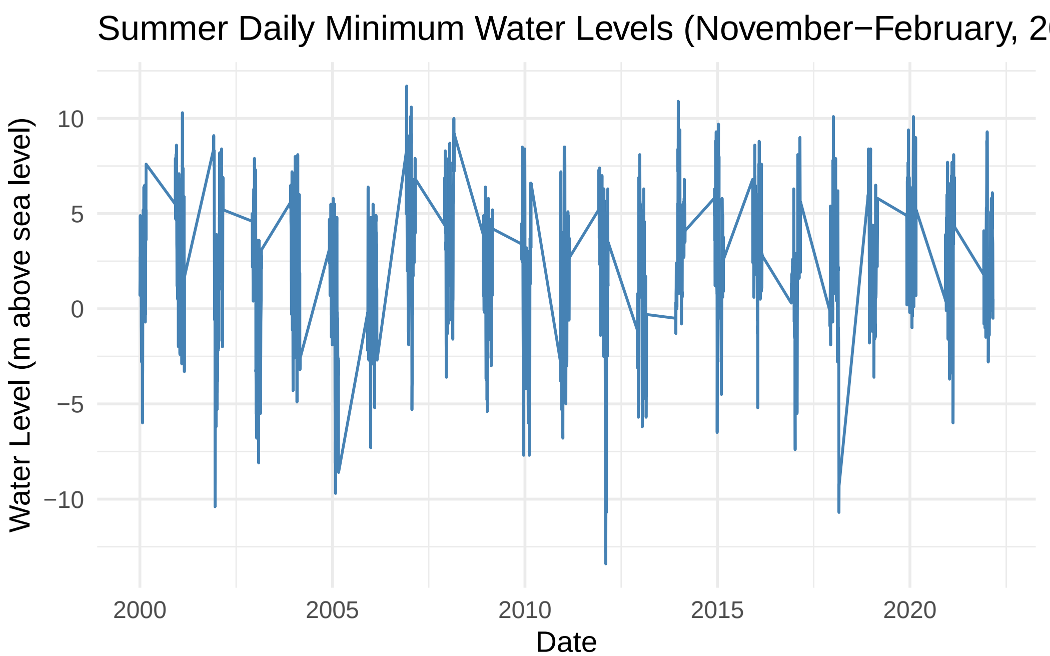
> Repeat the above analysis for the negatives of the daily nightly minimum temperatures for the winter months (November-February).

```r
summary_list <- list(
  winter_extremal_index = ei_min,
  gpd_winter_raw        = fit_min_raw,
  gpd_winter_dc         = fit_min_dc
)
print(summary_list)
```

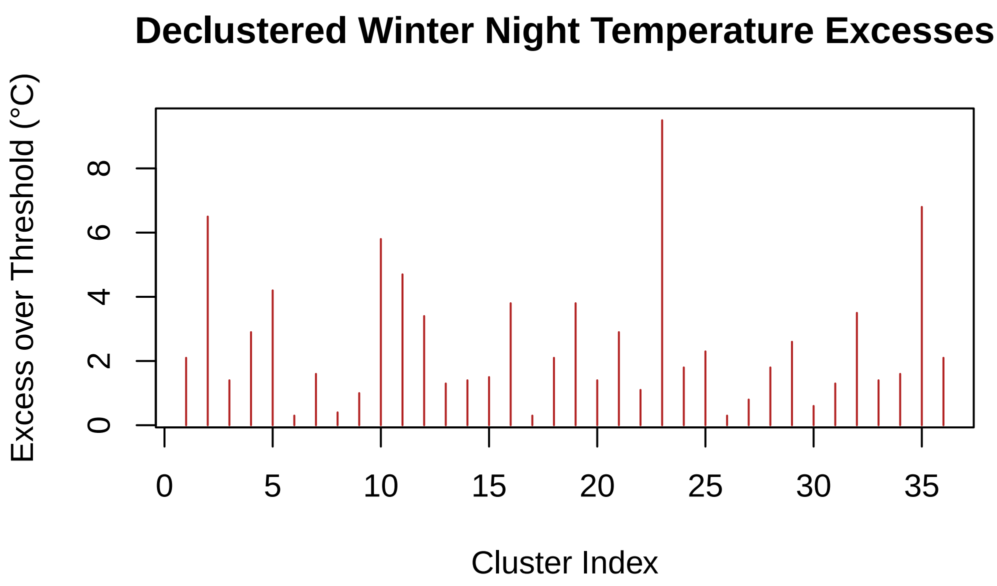
```
$winter_extremal_index

 Runs Estimator for the Extremal Index
    extremal.index number.of.clusters          run.length
         0.3673469         36.0000000           3.0000000

$gpd_winter_raw

fevd(x = (-winter_min$tmin)[-winter_min$tmin > u_min] - u_min,
    threshold = 0, type = "GP", method = "MLE")

[1] "Estimation Method used: MLE"


 Negative Log-Likelihood Value:  186.0393


 Estimated parameters:
     scale      shape
 2.8113093 -0.1352909

 Standard Error Estimates:
    scale     shape
0.4203671 0.1110884

 Estimated parameter covariance matrix.
            scale       shape
scale  0.17670849 -0.03804989
shape -0.03804989  0.01234063

 AIC = 376.0786

 BIC = 381.2486

$gpd_winter_dc

fevd(x = excess_min, threshold = 0, type = "GP", method = "MLE")

[1] "Estimation Method used: MLE"


 Negative Log-Likelihood Value:  68.36057
```

```
 Estimated parameters:
     scale      shape
 3.0113921 -0.2034978

 Standard Error Estimates:
    scale     shape
0.6471591 0.1398393

 Estimated parameter covariance matrix.
            scale       shape
scale  0.41881484 -0.07244737
shape -0.07244737  0.01955502

 AIC = 140.7211

 BIC = 143.8882
```

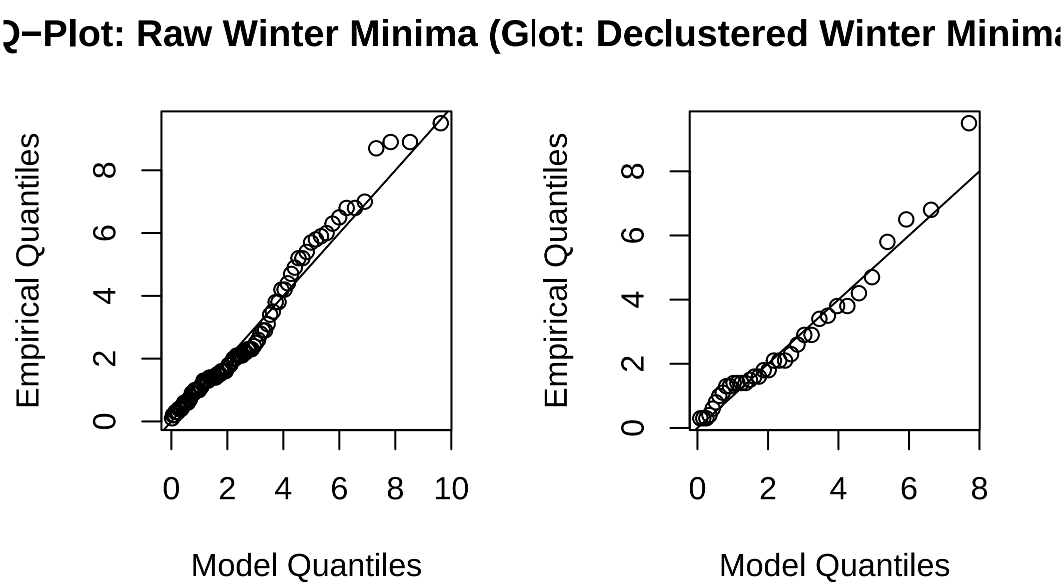
```r
par(mfrow = c(1,1))
# Plot summer daily maximum water levels
ggplot(winter_min, aes(x = date, y = tmin)) +
  geom_line(color = "steelblue") +
  labs(title = "Summer Daily Minimum Water Levels (November-February, 2000-2021)",
       x = "Date",
       y = "Water Level (m above sea level)") +
  theme_minimal()
```




```r
plot(excess_min, type = "h", col = "firebrick",
     main = "Declustered Winter Night Temperature Excesses",
     xlab = "Cluster Index", ylab = "Excess over Threshold (°C)")
```




```r
par(mfrow = c(1,2))
plot(fit_min_raw, type = "qq", main = "QQ-Plot: Raw Winter Minima (GPD Fit)")
plot(fit_min_dc, type = "qq", main = "QQ-Plot: Declustered Winter Minima (GPD Fit)")
```



We apply the negative to the winter values to treat the extremely low values as high values for modelling purposes. We then do an extremal index and we obtain 0.367, lower than 1, indicating clustering. We then declustered using the 95th percentile threshold to the negated temperatures and the plot shows that some peaks go even above 6 degrees. Fitting the model using GPD shows that the AIC for the declustered is again lower than raw, so a better fit.