Extreme Value Theory - ENSAI - GS & GR 3A

Practical session 2: The general case

José Gregorio GOMEZ-GARCIA

October 09, 2023

```
library(knitr) # table format in output
library(dplyr) # Load the dplyr library for data manipulation
library(tidyr)
library(ggplot2)
library(cowplot) #to plot multiple graphs on a grid
```

Description

Supporting data for this practical session are:

- Temperature data from Perth, Western Australia, see `Perth_min_temp.csv` and `Perth_max_temp.csv` - Wind data at 12 weather stations in the Republic of Ireland from the R package `gstat`.

Of course, feel free to revisit the datasets used for the first session too. This can help in further justifying that a heavy tail is present in these sets of data.

In each case, it will be important to code the estimators yourself, and to compare your procedures with those of R packages such as `evir`, `evd`, `evt0` and `extRemes`. As a preliminary step, you may therefore want to do the following things:

1. For a fixed sample of data of size n and $2 \leq k \leq n$, implement the Pickands and moment estimators. The result should be a pair of vectors of size $n-1$.

Here is the code for the estimators:

```
my_Pickands <- function(x){
  estimator <- c(NA)
  n = length(x)
  inter.quant <- sort(x, decreasing = TRUE)
  for (k in 2:as.integer(n/4)){
      estimator[k] <- 1/log(2)*log((inter.quant[k]-inter.quant[2*k])/
                                     (inter.quant[2*k]-inter.quant[4*k]))
  }
  return(estimator)
}

my_Moments <- function(x){
  estimator <- c(NA)
  n = length(x)
  inter.quant <- sort(x, decreasing = TRUE)
  for (k in 2:(n-1)){
    M1 <- mean(log(inter.quant[1:(k-1)])) - log(inter.quant[k])
    M2 <- mean((log(inter.quant[1:(k-1)])-log(inter.quant[k]))^2)
```

```
    estimator[k] <- M1 + 1 - 0.5*(1-M1^2/M2)^(-1)
  }
  return(estimator)
}
```

2. Implement likewise the maximum likelihood estimators. This is more difficult but closer to what practitioners typically use. The choice of the optimisation method (gradient descent, Newton-Raphson, Nelder-Mead, brute force…) is up to you, but you should check that you indeed find a maximum of the likelihood. As a starting point to the algorithm, you may want to use probability weighted moment estimators.

Use the likelihood function given in the lecture notes. For the optimization part you can use `optim`.

3. Implement diagnostic tools (QQ-plots…) to judge whether the fittings are reasonable.

- See `plot(fit.pot)` below.

4. Download a few packages. Good choices in this practical session are `evir`, `evd` and `extRemes`. These will help to see if your results make sense (and you might find that one of the packages does something wrong!)

Wind data

The variable of interest is average daily wind speed at weather stations in Ireland. It is sufficient to pick a single station for the analysis, although you may want to compare your results across stations (see final question). Accompanying this variable are the day, month and year of each recording. This data set can be loaded by typing `data(wind)` after having loaded the `gstat` package.

```
library(gstat)
#Attention: The legacy packages maptools, rgdal, and rgeos, underpinning the sp
#package, which was just loaded, were retired in October 2023. Please refer to
#R-spatial evolution reports for details, especially
#https://r-spatial.org/r/2023/05/15/evolution4.html. It may be desirable to make
#the sf package available; package maintainers should consider
#adding sf to Suggests:.
data("wind")
kable(head(wind))
```

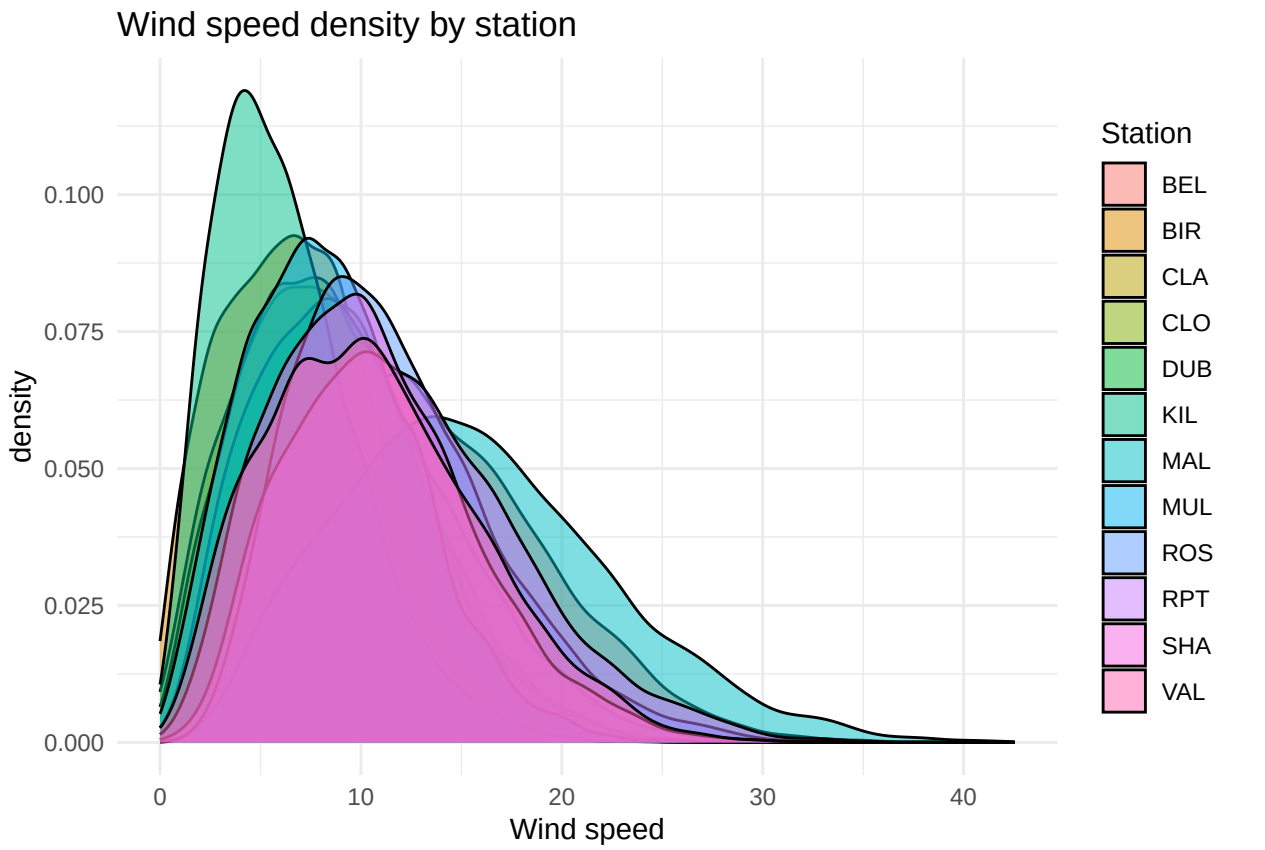
year	month	day	RPT	VAL	ROS	KIL	SHA	BIR	DUB	CLA	MUL	CLO	BEL	MAL
61	1	1	15.04	14.96	13.17	9.29	13.96	9.87	13.67	10.25	10.83	12.58	18.50	15.04
61	1	2	14.71	16.88	10.83	6.50	12.62	7.67	11.50	10.04	9.79	9.67	17.54	13.83
61	1	3	18.50	16.88	12.33	10.13	11.17	6.17	11.25	8.04	8.50	7.67	12.75	12.71
61	1	4	10.58	6.63	11.75	4.58	4.54	2.88	8.63	1.79	5.83	5.88	5.46	10.88
61	1	5	13.33	13.25	11.42	6.17	10.71	8.21	11.92	6.54	10.92	10.34	12.92	11.83
61	1	6	13.21	8.12	9.96	6.67	5.37	4.50	10.67	4.42	7.17	7.50	8.12	13.17

First look at the data table:

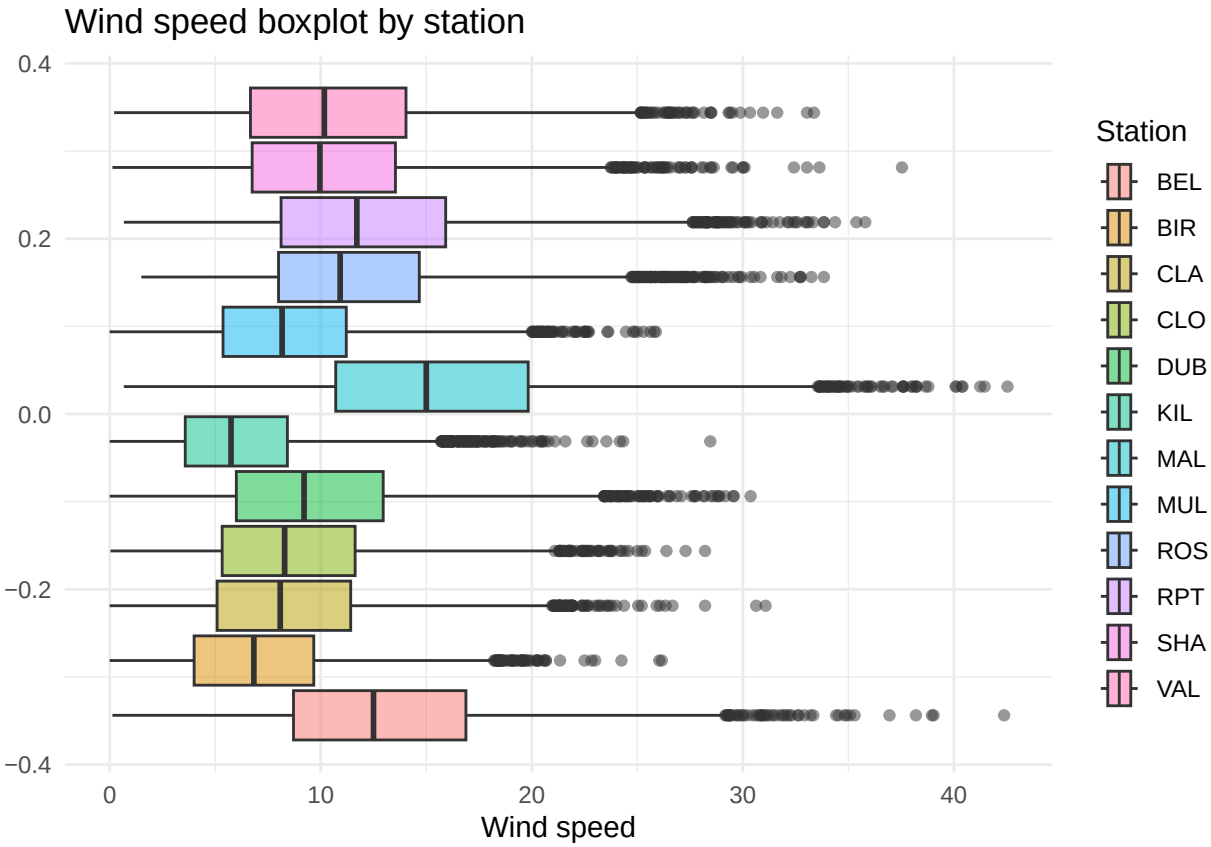
```
str(wind)
summary(wind)
anyNA(wind)
```

The variables are in the correct format, and there are no missing data.

```r
#Select only the wind speed columns (RPT, VAL, ROS, KIL, SHA, BIR, DUB, CLA, MUL, CLO, BEL, MAL)
wind_data <- wind %>%
  select(RPT, VAL, ROS, KIL, SHA, BIR, DUB, CLA, MUL, CLO, BEL, MAL)

# Use the gather() function to group the data into a single "WindSpeed" column
wind_data <- wind_data %>%
  gather(key = "Station", value = "WindSpeed")

ggplot(wind_data, aes(x = WindSpeed, fill = Station)) + #Density for each station
  geom_density(alpha = 0.5) +
  labs(title = "Wind speed density by station", x = "Wind speed") +
  theme_minimal()
```




```r
ggplot(wind_data, aes(x = WindSpeed, fill = Station)) + #Boxplot for each station
  geom_boxplot(alpha = 0.5) +
  labs(title = "Wind speed boxplot by station", x = "Wind speed") +
  theme_minimal()
```

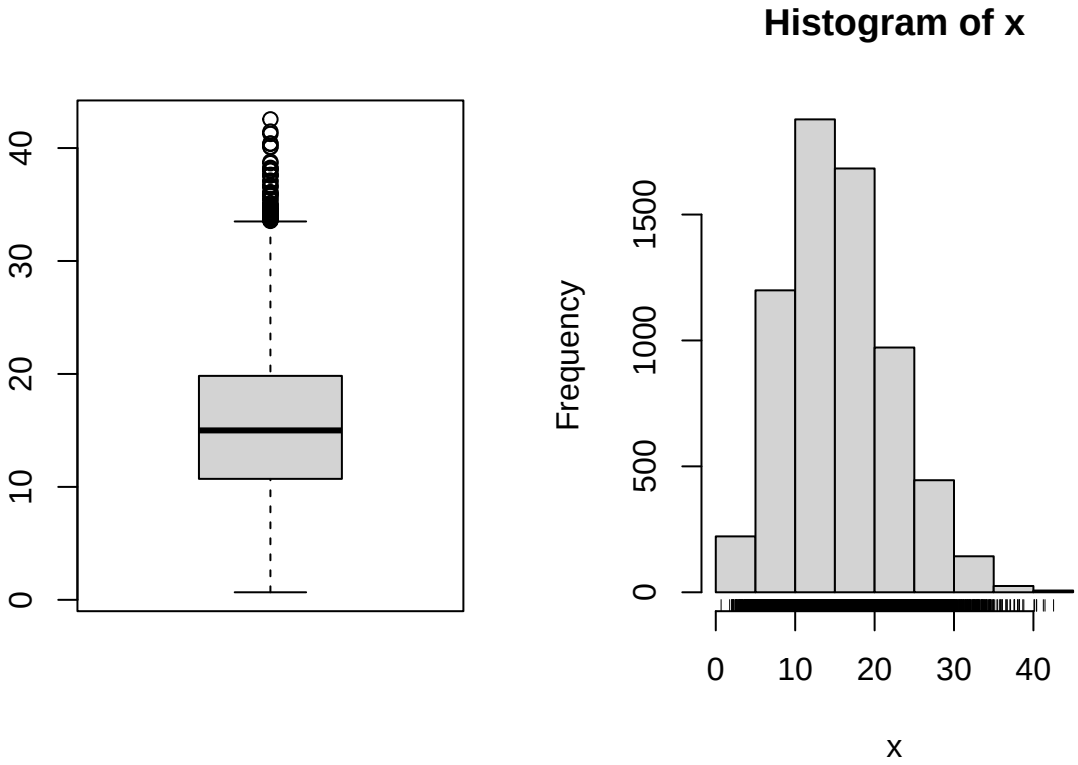


It seems interesting to study the Malin Head station:

```
wind.MAL <- wind[,c(1:3,15)]
x<- wind.MAL$MAL
```

1. Represent the data, first ignoring the time series structure, and then by plotting the data as a time series. Do you notice seasonality, autocorrelation, nonstationarity?

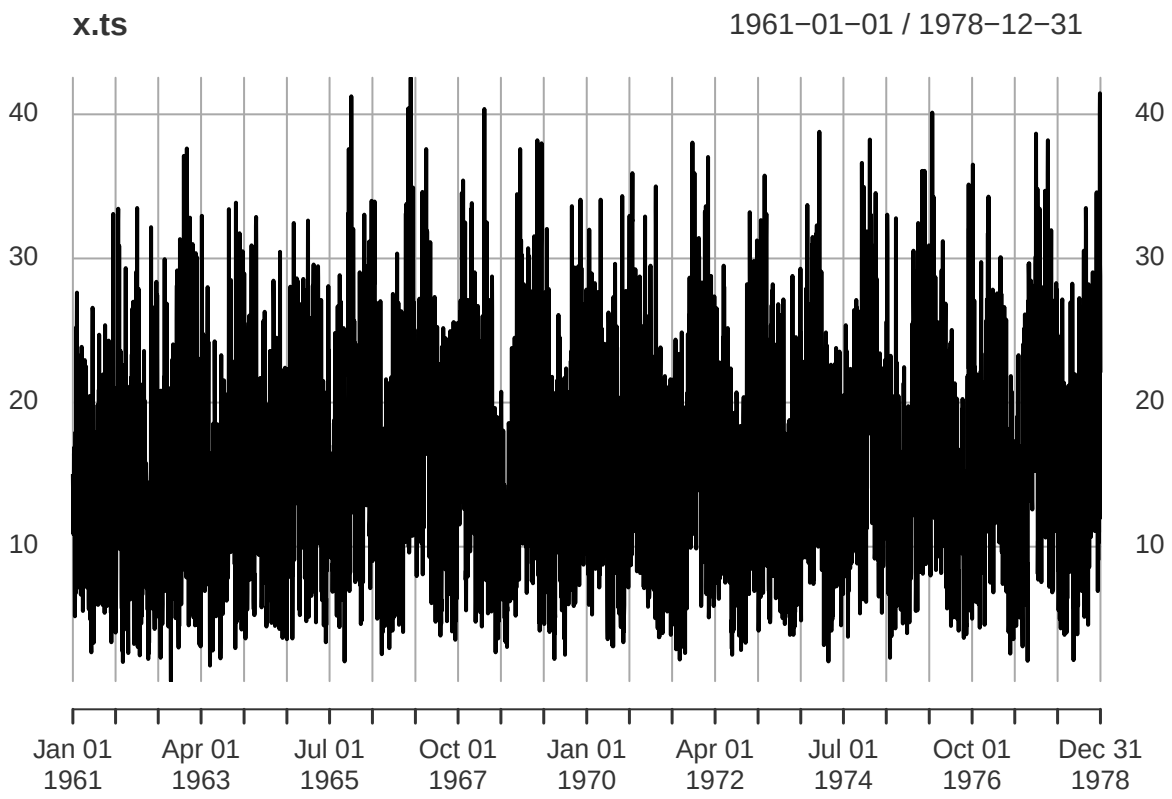
```
par(mfrow=c(1,2))
boxplot(x)
hist(x)
rug(x)
```



The distribution looks short tailed.

- We plot the time series with `xts` format:

```
#clever use of paste to combine date columns
wind$year<-paste(19,wind$year, sep="")
date <- paste(wind$year, wind$month, wind$day ,sep = "-")
date <- as.Date(date)
#You can also simply use a sequence from 1961-1-1 to 1978-12-31. Exercise!

library(xts)
x.ts <- xts(x, as.Date(date, format="%Y-%m-%d"))
par(mfrow=c(1,1))
plot(x.ts)
```



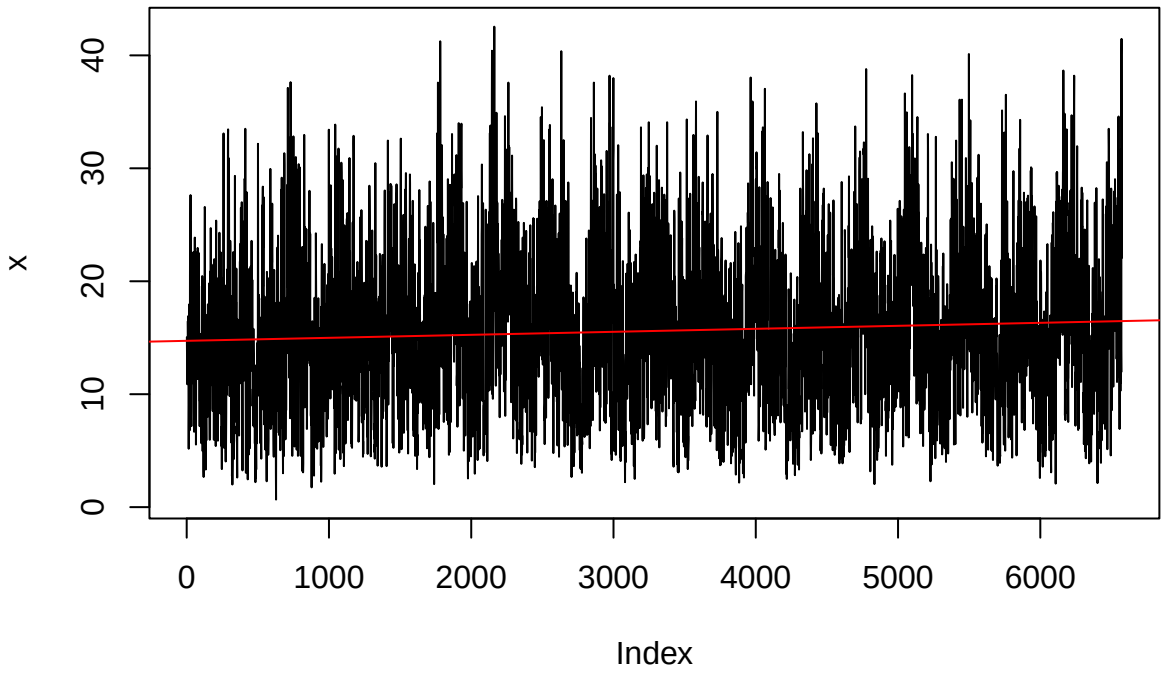
- **Trend**: We check for the existence of a linear trend in the data. To do this, we will use a simple linear regression:

```
days <- 1:length(x)
mod <- lm(x~days)
summary(mod)
```

```
Call:
lm(formula = x ~ days)

Residuals:
    Min      1Q  Median      3Q     Max
-14.297  -4.940  -0.602   4.208  27.240

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 1.473e+01  1.648e-01  89.366  < 2e-16 ***
days        2.658e-04  4.341e-05   6.124 9.63e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.679 on 6572 degrees of freedom
Multiple R-squared:  0.005675,  Adjusted R-squared:  0.005524
F-statistic: 37.51 on 1 and 6572 DF,  p-value: 9.628e-10
```

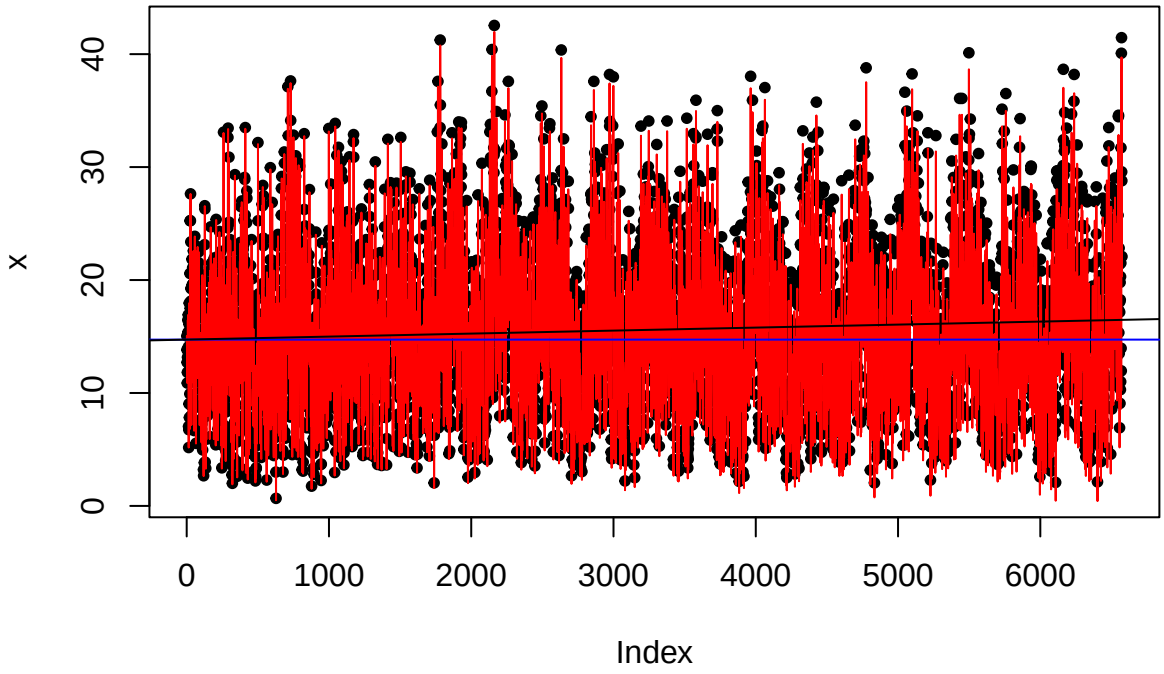
the time (days) is statistically significant.

```
plot(x, type = "l")
abline(mod, col="red")
```



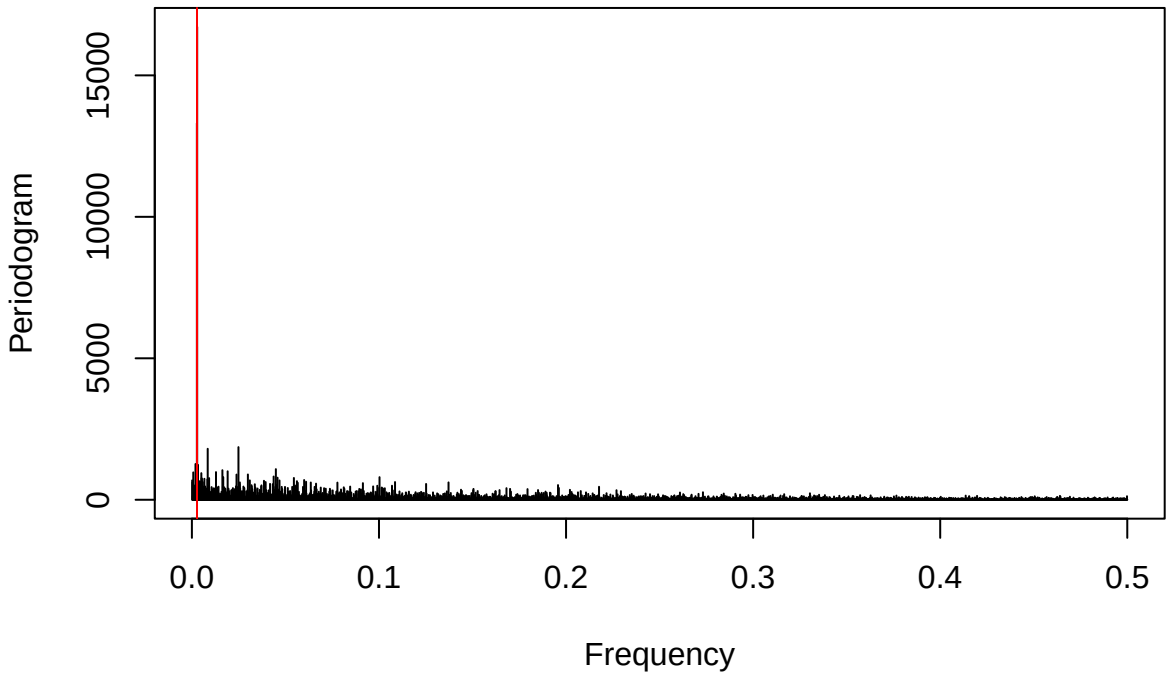
We now remove this statistically significant trend:

```
x.detrended <- x - mod$fitted.values + predict(mod, newdata = data.frame(days=0))
plot(x, pch=20) #the points is the wind.MAL time series
lines(x.detrended, col="red") #red lines is the detrended time series
abline(lm(x.detrended~days), col='blue')
abline(mod)
```



- **Seasonality**: We check for the existence of a seasonality. Since this is wind data, we suspect that we have a yearly seasonality.

```
library(TSA)
per.x.detrended <- TSA::periodogram(x.detrended, lwd = 1)
abline(v=1/365, col="red", lty=1)
```

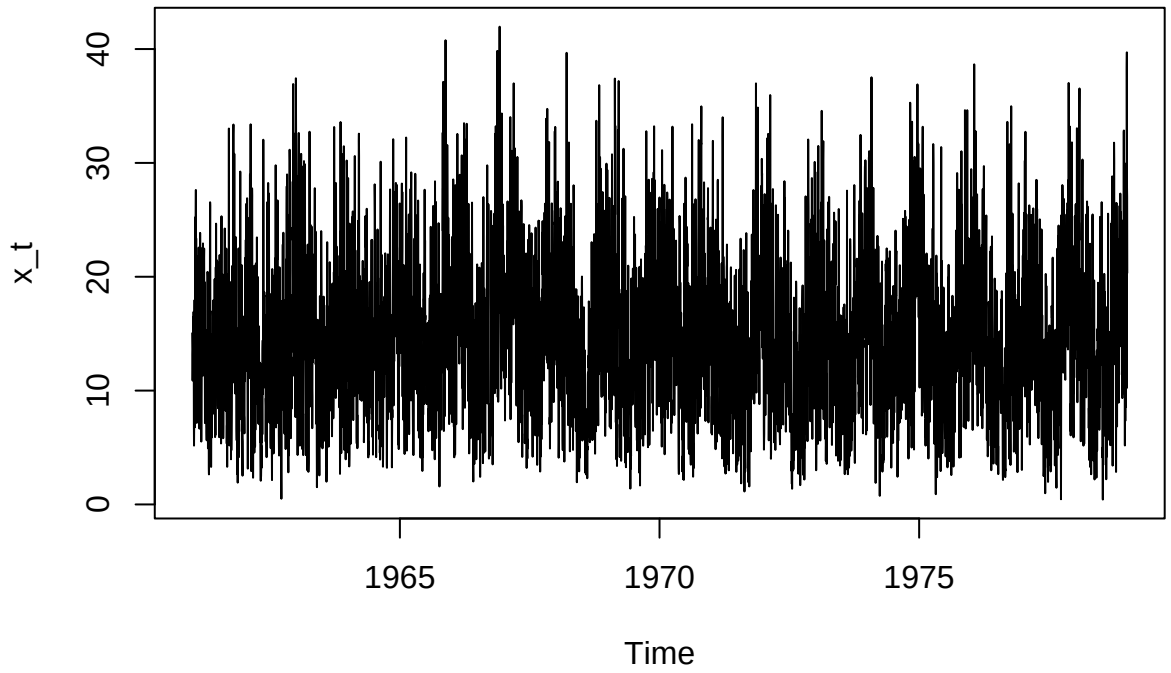



```
detach(package:TSA)
```

Huge peak at 1/365, which corresponds to yearly seasonality.

We use this to specify the frequency in the time series:

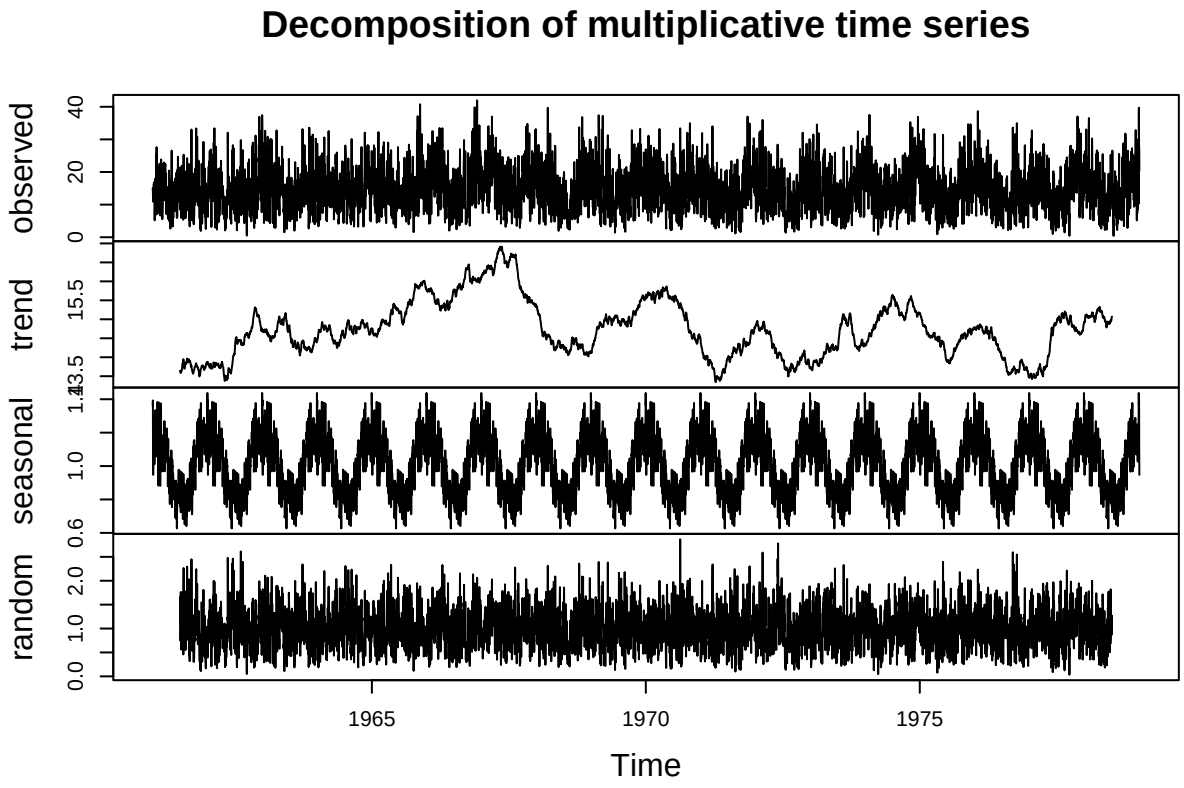
```
x_t <- ts(x.detrended, frequency = 365, start = c(1961,1,1))
plot(x_t)
```




```
#abline(lm(x.detrended~days), col='blue')
```

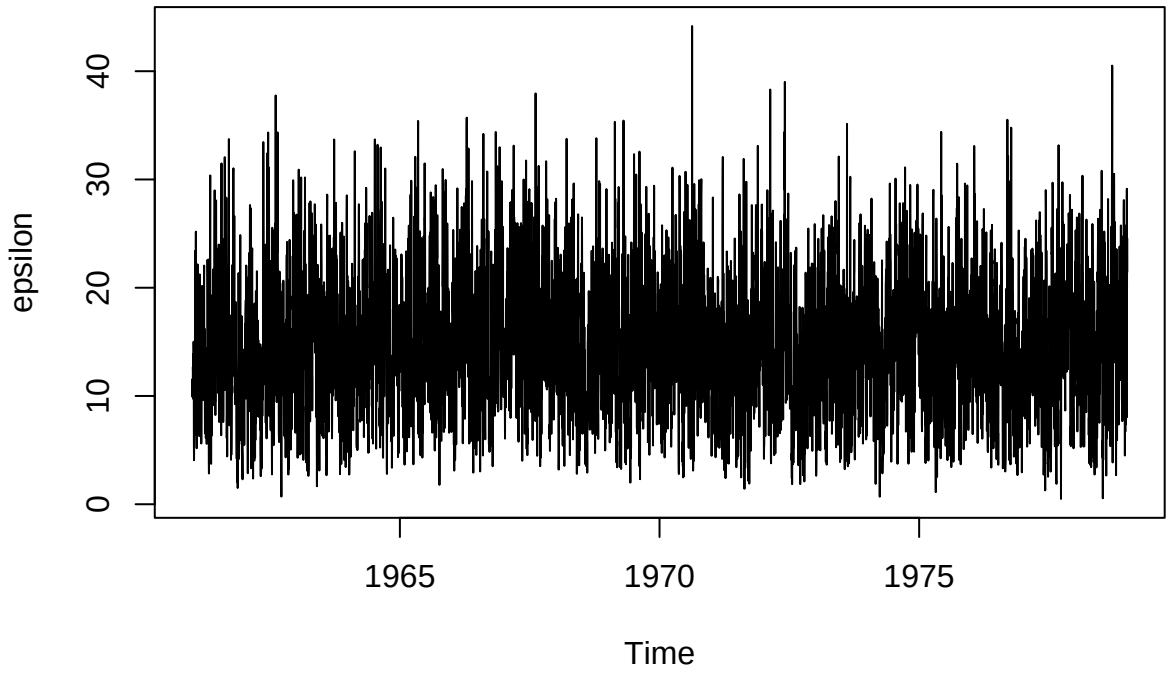
We now use multiplicative decomposition to ensure the data remains positive

```
decomposition <- decompose(x_t, type = 'multiplicative') #
plot(decomposition)
```

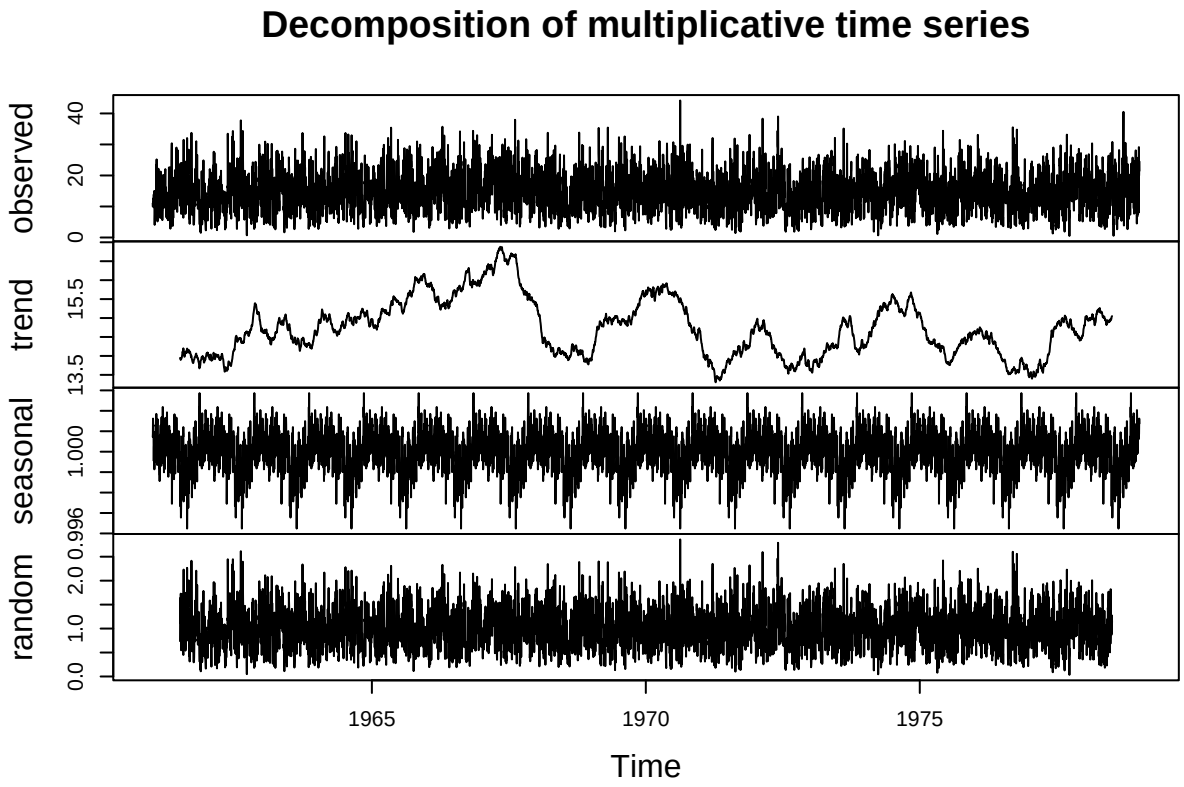


Given that we used multiplicative seasonality, we divide by the seasonal component in order to obtain a deseasoned time series:

```r
epsilon <- x.detrended/decomposition$seasonal
plot(epsilon)
```




```r
decomp1 <- decompose(epsilon, type = 'multiplicative')
plot(decomp1)
```



Here, we observe that the seasonality is now negligible.

- **Stationary test**:

```
library(tseries)
adf.test(epsilon) #Here, the null hypothesis is "non-stationary"
```

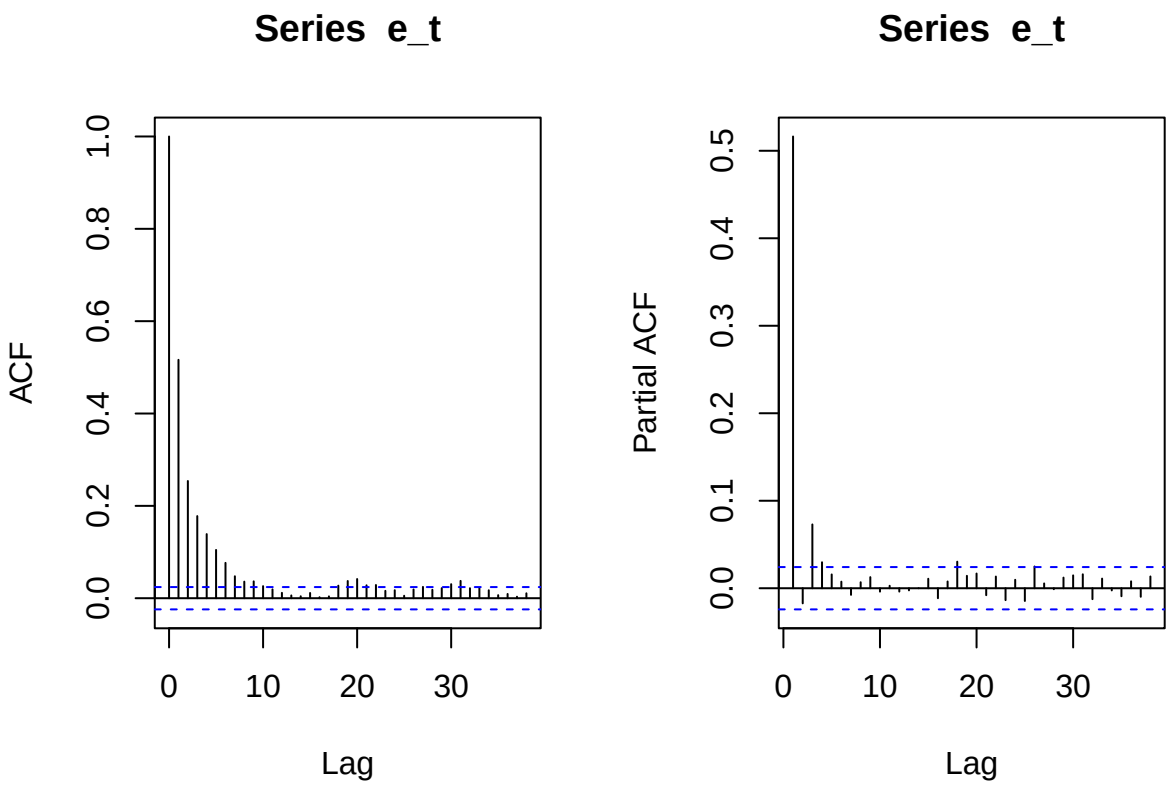
```
    Augmented Dickey-Fuller Test

data:  epsilon
Dickey-Fuller = -16.2, Lag order = 18, p-value = 0.01
alternative hypothesis: stationary
```

The test indicates that our data (detrended and deseasoned) is stationary.

Exercise: Implement other stationarity tests.

- **Autocorrelation**

```
e_t <- as.numeric(epsilon)
par(mfrow=c(1,2))
acf(e_t)
pacf(e_t)
```



It appears to have autocorrelation. We will ensure with an autocorrelation test:

```
library(caschrono)
t(Box.test.2(e_t, nlag = 2:5, type = "Ljung-Box", decim = 2))
```

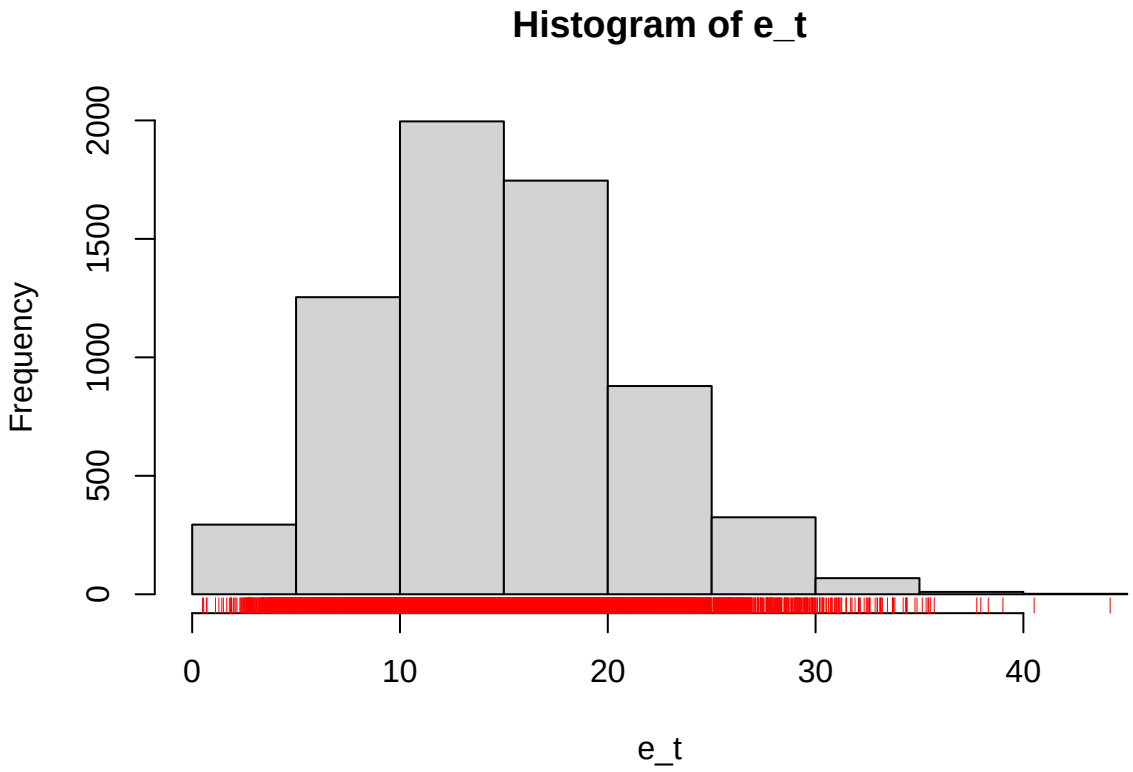
```
        [,1] [,2] [,3] [,4]
Retard     2    3    4    5
p-value    0    0    0    0
```

```
#Here, H0 is "time series is a white noise"
```

For all lags (2,3,4,5), we reject H_0. We therefore have dependence in the data.

2. Can you find a reasonable statistical model for the whole of the data?

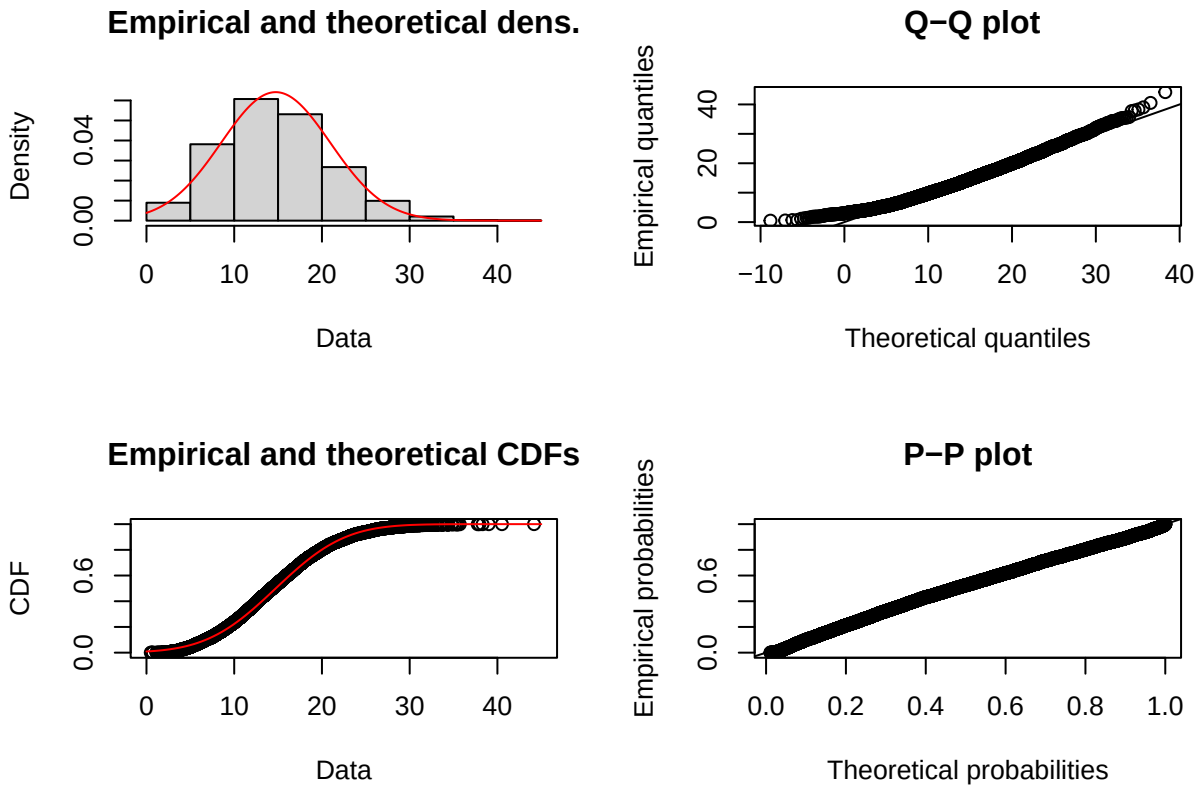
```
par(mfrow=c(1,1))
hist(e_t)
rug(e_t, col="red")
```



Again, the distribution looks short or light tailed.

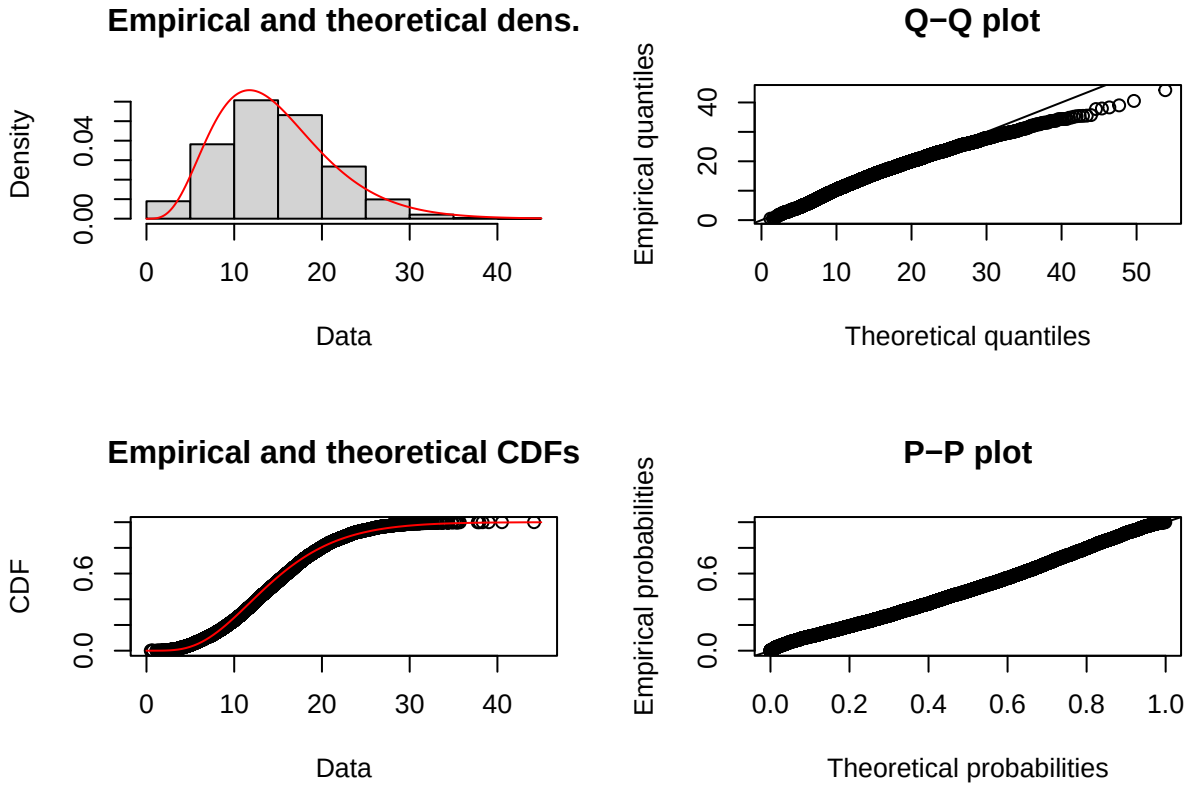
We can (naively) fit a normal distribution:

```
library(fitdistrplus)
norm.fit <- fitdist(e_t, "norm", method = 'mle')
plot(norm.fit)
```



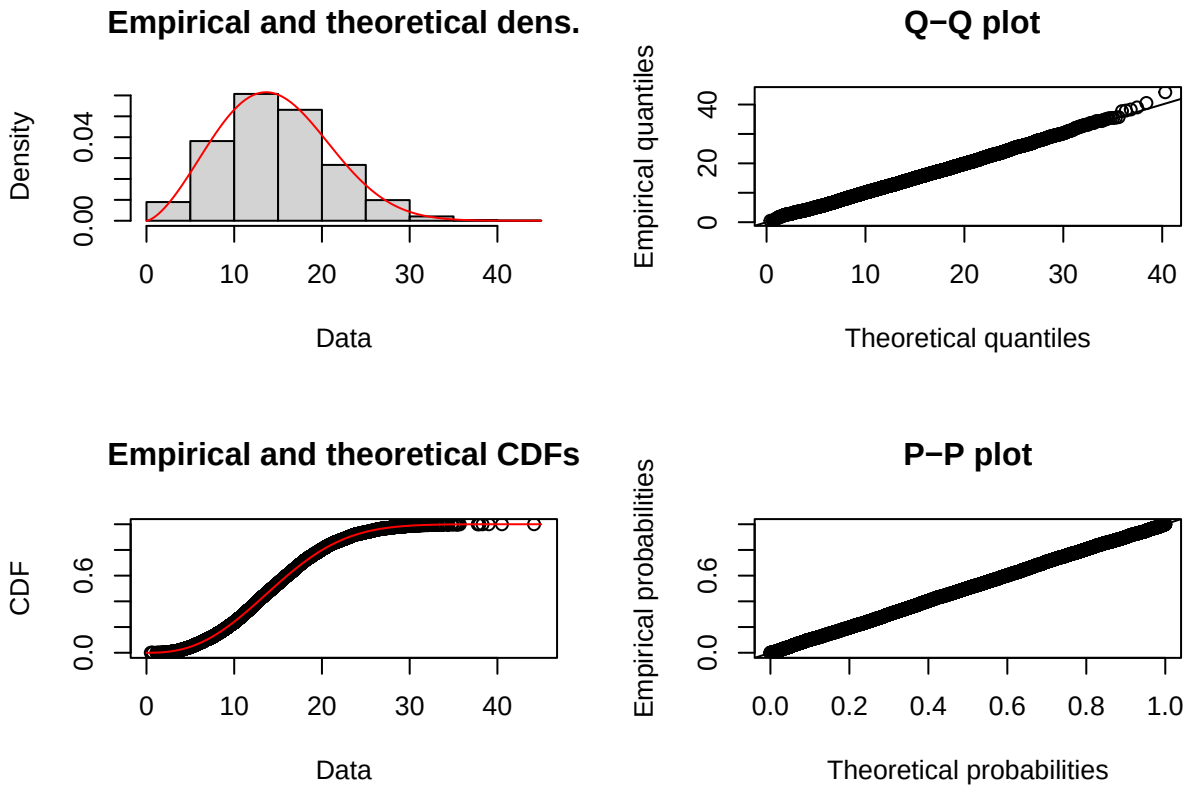
Here we see a poor fit, specially around the tails.

```
gamma.fit <- fitdist(e_t, "gamma", method = 'mle')
plot(gamma.fit)
```



That doesn't work either.

```
weibull.fit <- fitdist(e_t, "weibull", method = 'mle')
plot(weibull.fit)
```




```
weibull.fit
```

```
Fitting of the distribution ' weibull ' by maximum likelihood
Parameters:
       estimate Std. Error
shape  2.536618 0.02418679
scale 16.605223 0.08508612
```

We obtained an almost perfect fit with the Weibull distribution

$$f(x;\kappa,\lambda) := \frac{\kappa}{\lambda}\left(\frac{x}{\lambda}\right)^{\kappa-1}\exp\left(-x/\lambda\right)^{\kappa}$$

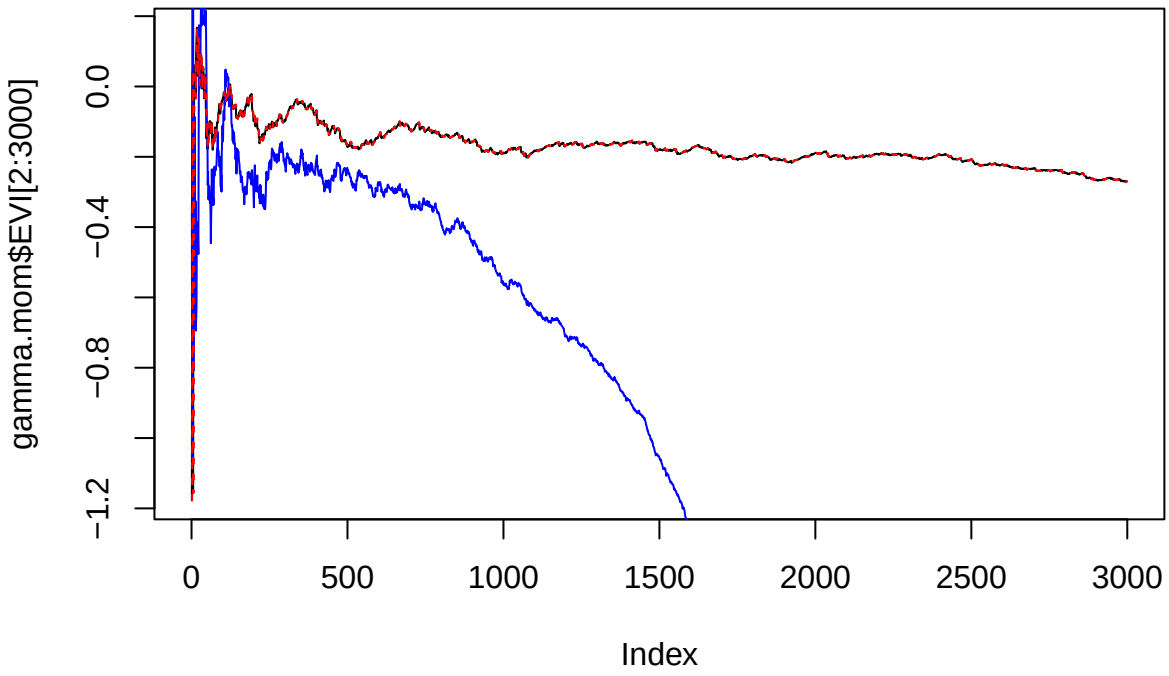
with shape parameter $\kappa = 2.53$ and scale parameter $\lambda = 16.60$.

Remark : We know that the Weibull distribution belongs to the Gumbel domain of attraction, which implies that the extreme value index (EVI) is 0. However, it is too early to declare that the EVI is 0 because we must not forget that we have obtained a distribution that fits the data almost perfectly, but we have no guarantee that the true underlying model F for this data is a Weibull distribution. In fact, we will show that the index of extreme values is rather negative, indicating that the distribution F of our data belongs to the domain of attraction of Weibull $G_\gamma(x) = f(-x;-1/\gamma,1)$, $\gamma < 0$.

3. Calculate an estimate for the extreme value index of the data. Justify your choice of tuning parameters.

- **Semiparametric methods**

```
n <- length(e_t)
gamma.mom <- other.EVI(e_t, 1:(n-1), method = "MO") #Moment estimator
gamma.P <- my_Pickands(e_t) #Our Pickands estimator
gamma.mm <- my_Moments(e_t) #Our Moment estimator
plot(gamma.mom$EVI[2:3000], type="l")
lines(gamma.P[2:3000], col="blue")
lines(gamma.mm[2:3000], col="red", lty = 2)
```



We have coded our moment estimator correctly. We also observe that the moments estimator is more stable than the Pickands estimator.

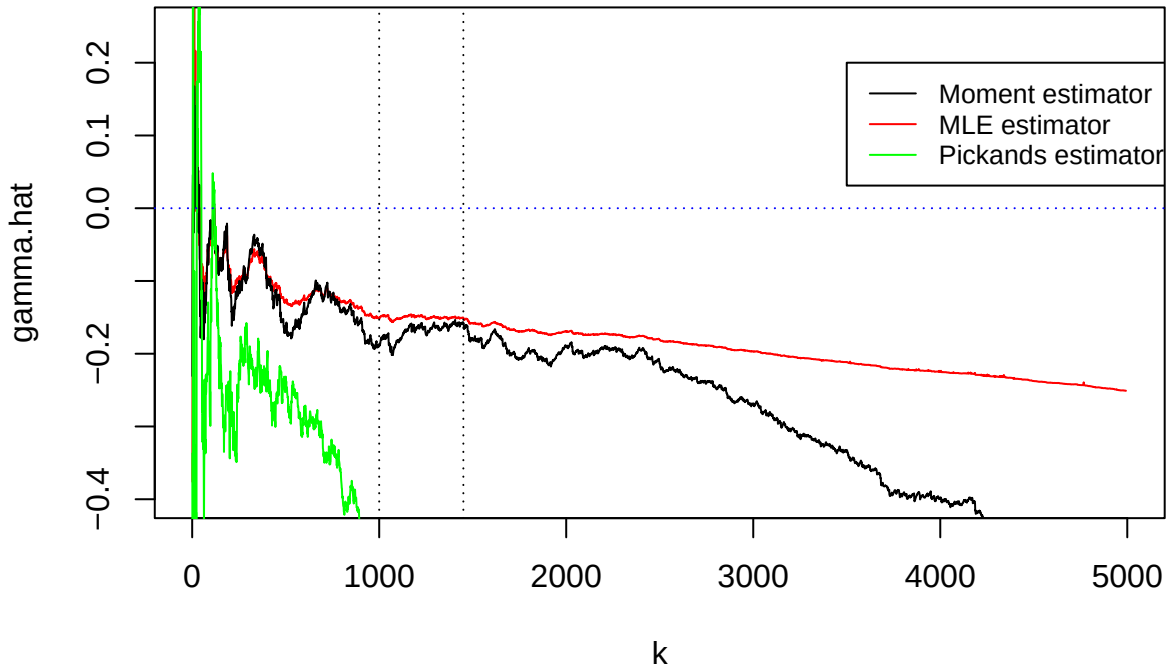
Exercise: Check the `method="GH"` (Generalized Hill) in `other.EVI`.

- **Parametric method**

```
N = 5000 #Up to 5000 because the execution time is too long
intermediate_quantile <- sort(e_t, decreasing = TRUE)
gamma.ml <- c(NA)
for (k in 2:N){
  fit <- fevd(e_t, threshold = intermediate_quantile[k+1], type = "GP")
  gamma.ml[k]<-fit$results$par[[2]]
  #print(k)
}
```

```
plot(gamma.ml[8:N], type="l", ylab = "gamma.hat", xlab = "k", col="red",
     ylim = c(-0.4, 0.25), xlim = c(0,5000))
lines(gamma.mom$EVI[8:N])
lines(gamma.P, col="green")
# Add a legend
legend(3500, 0.2, legend=c("Moment estimator", "MLE estimator", "Pickands estimator"),
       col=c("black", "red", "green"), lty=1, cex=0.8)
abline(v=1450, lty=3)
abline(h=0, col="blue", lty=3)
```

```
abline(v=1000, lty=3)
```



The data is clearly short-tailed according to these estimators.

- **Extrapolation**: We choose to work with the MLE estimator because it looks more stable.

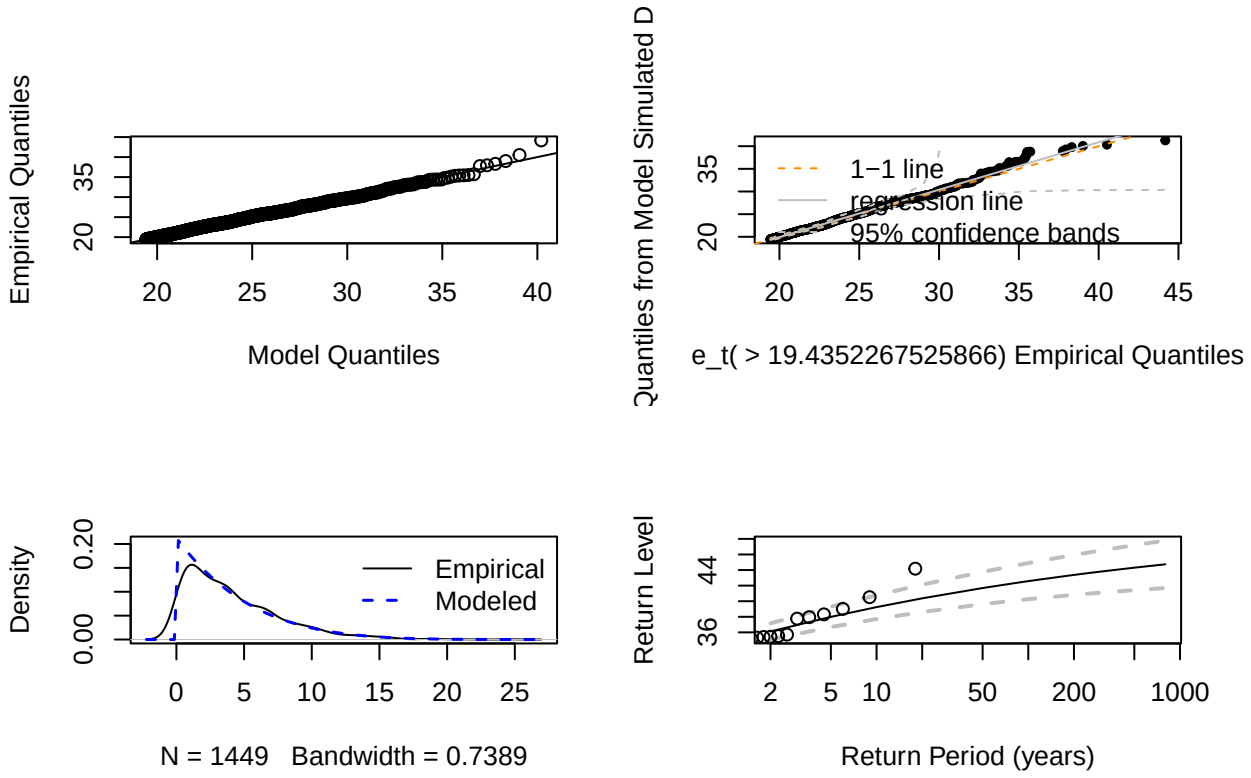
$k = 1450$ seems like a decent spot (the end of the first stable region for the red curve)

```
k = 1450
gamma.hat = gamma.ml[k]
gamma.hat
```

```
[1] -0.1505832
```

We re-run the fitting function at the specified k to get the parameters we need:

```
u <- intermediate_quantile[k] #point from which we need to extrapolate (n-k order statistic)
fit.pot <- fevd(e_t, threshold =  u, type = "GP")
plot(fit.pot)
```

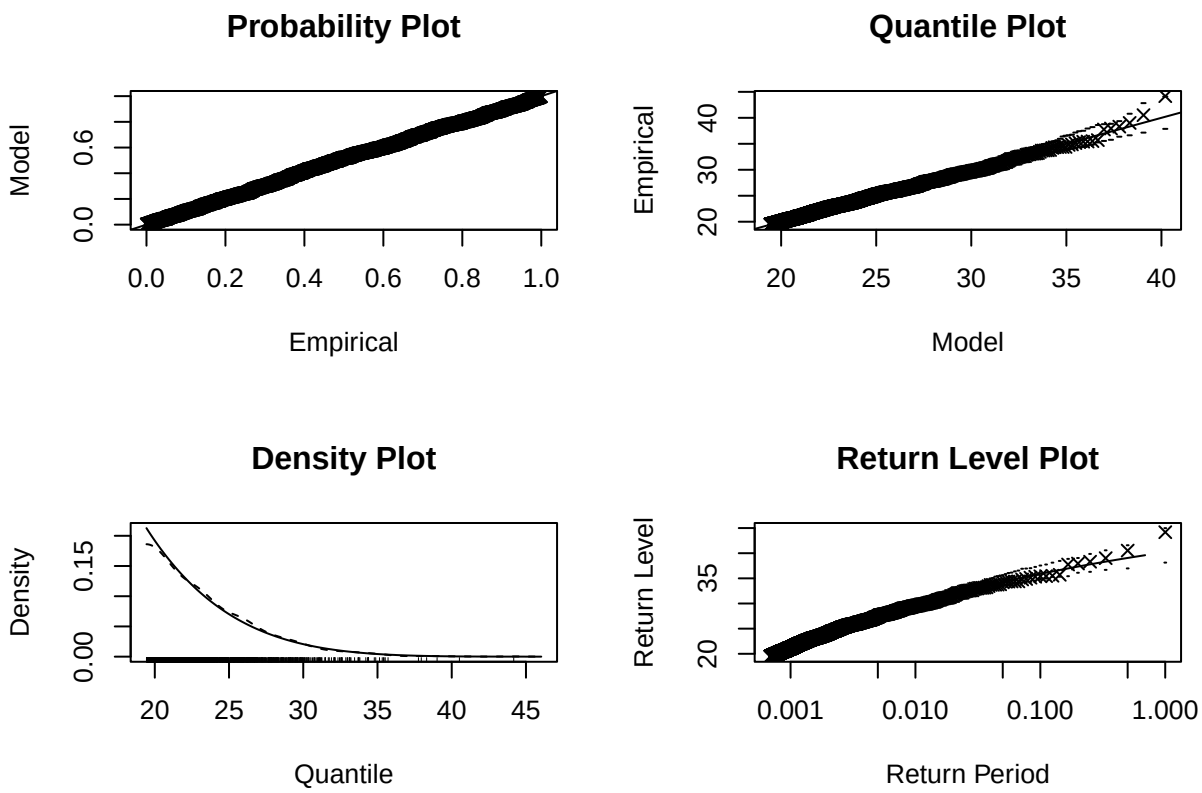



```
fit.pot$results$par
```

```
     scale      shape
 4.6995777 -0.1507364
```

Remark: You can also use the `fpot` instead of `fevd`.

```
fit.pot2 <- fpot(e_t, u, 'gpd')
par(mfrow=c(2,2))
plot(fit.pot2)
```



With this $k = 1450$, we obtained a good adjustment for the exceedances.

Remark: We cannot apply the QQ-exponential plot (see Lecture notes, page 25) to verify if the estimated EVI parameter is correct because this tool works effectively when we are certain that the EVI is positive. For similar tools, you can see for example Beirlant et al.

4. Calculate an estimate for the extreme quantiles at level 0.995, 0.999 and $1 - 1/n$. Do these make sense?

Now we will construct the extreme quantile estimator which takes as argument the estimators of the scale and shape (evi), see Lecture notes, page 48.

```
extreme.quantile <- function(threshold,scale,shape,beta,k,n){
  threshold + (sigma/shape)*((n*(1-beta)/k)^(-shape)-1)
}
```

```
beta1 = 0.995
beta2 = 0.999
beta3 = 1-1/n
sigma = fit.pot$results$par[[1]]
gamma = fit.pot$results$par[[2]]

qbeta1.ml <- extreme.quantile(threshold = u,
                              scale = sigma,
                              shape = gamma,
                              beta=beta1,
                              k,n)

qbeta2.ml <- extreme.quantile(threshold = u,
                              scale = sigma,
                              shape = gamma,
```

```
                              beta=beta2,
                              k,n)

qbeta3.ml <- extreme.quantile(threshold = u,
                              scale = sigma,
                              shape = gamma,
                              beta=beta3,
                              k,n)

print(c(qbeta1.ml,quantile(e_t, beta1)))
```

```
            99.5%
32.99518 32.87724
```

```
print(c(qbeta2.ml,quantile(e_t, beta2)))
```

```
            99.9%
36.79026 35.60203
```

```
print(c(qbeta3.ml,quantile(e_t, beta3)))
```

```
          99.98479%
 40.20612  40.51815
```

We can observe that extrapolation in this scenario doesn't yield significantly different outcomes when compared to the empirical quantile. This is due to the favorable conditions we have: data with a short tail and a substantial sample size. The combination of a large sample size and a distribution with a short tail indicates that there won't be a significant lack of data at the tail, thus the empirical quantiles won't face significant challenges. However, in cases where the sample size is limited, you will observe notable distinctions between the extrapolated and empirical quantiles.

5. Can you provide a confidence interval for the extreme value index and for these extreme quantiles? If so, why? If not, can you suggest a method that would allow you to do so?

- Confidence intervals are meaningless because of dependence in the data. However, you can calculate "naive" confidence intervals as in First Practical Session.

6. Can you think of a way to assess the correlation between extreme values at different stations? If so, can you use this in order to analyze/infer joint extremes of wind speed?

- We could use the following measure:

$$\rho(h, \|s_i - s_j\|) := \lim_{x \to \theta} \mathbf{P}(X_{t+h}(s_i) > x | X_t(s_j) > x)$$

where s_i, $i = 1, 2, \dots, m$ are the positions of stations and $X_t(s)$ is the wind speed in position s at time t. Here, we suppose that the marginal distributions F_{s_i} have the same right endpoint θ. This is a simple version of the spatial extremegram of YONG BUM CHO et al, 2015, which is a generalization of the extremogram (a correlogram for extreme events) defined by Davis and Mikosch, 2009.

- **Estimation**

For estimation, without assuming spatial stationarity for the moment, you can estimate these values with

$$\hat{\rho}_k(h, i, j) = \frac{\sum_{t=1}^{n-h} \mathbb{1}\{X_{t+h}(s_i) > U_k \ , \ X_t(s_j) > U_k\}}{\sum_{t=1}^{n} \mathbb{1}\{X_t(s_j) > U_k\}}$$

where U_k is the k-th highest value of all the data $\{x_t(s_i), t = 1, \dots, n, i = 1, \dots, m\}$. $\hat{\rho}_k(h, i, j)$ is a estimator of the pre-asymptotic extremogram $\mathbf{P}(X_{t+h}(s_i) > u_n | X_t(s_j) > u_n)$.

Exercise:

- Implement the code for each $h = 0, 1, 2, .., H$ and each $i, j = 1, 2, ..., m$ (m =total number of stations).

- Create graphs with h on the abscissa and $\hat{\rho}$ on the ordinate with a color for each (i, j).

Temperature data

The variables of interest are average monthly minimum temperature and maximum temperature in Perth, Western Australia, between 1944 and 2019. Accompanying these variables are the month and year of each recording.

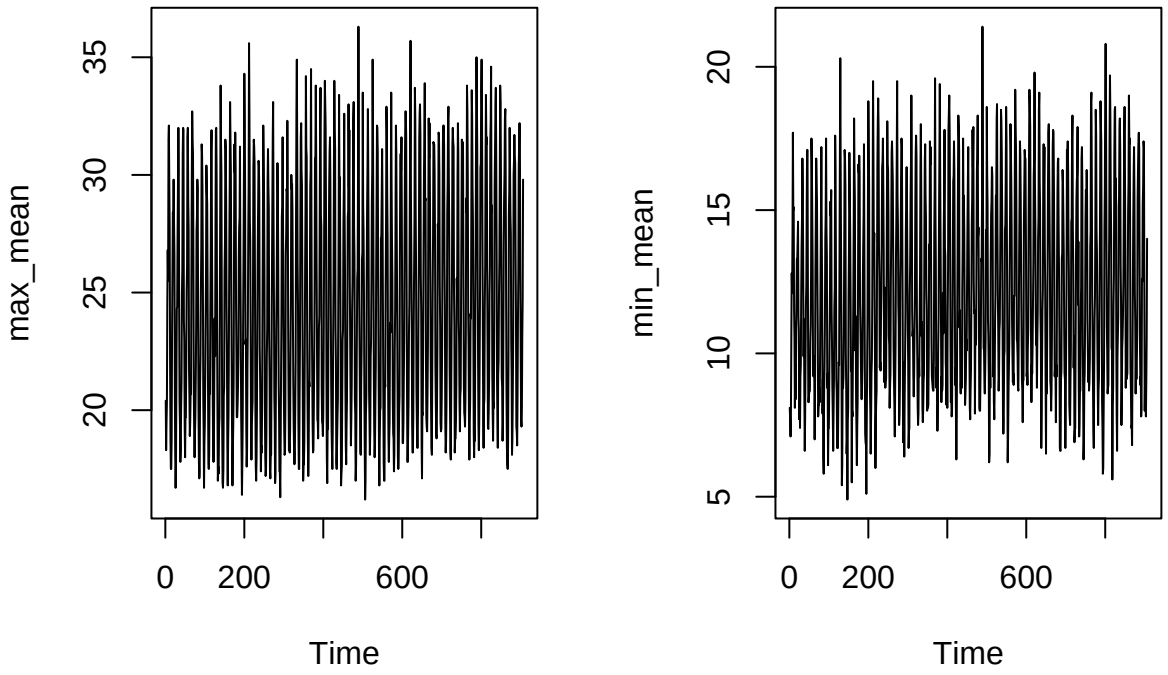
```
Perth_max_temp <- read.csv("~/Documents/GIT/teaching/extreme value theory/data sets for practical sessi
Perth_min_temp <- read.csv("~/Documents/GIT/teaching/extreme value theory/data sets for practical sessi
kable(head(Perth_max_temp))
```

Product.code	Bureau.of.Meteorology.station.number	Year	Month	Mean.maximum.temperature	Quality
IDCJAC0002	9021	1944	6	20.4	Y
IDCJAC0002	9021	1944	7	18.3	Y
IDCJAC0002	9021	1944	8	19.5	Y
IDCJAC0002	9021	1944	9	20.3	Y
IDCJAC0002	9021	1944	10	24.6	Y
IDCJAC0002	9021	1944	11	26.8	Y

```
kable(head(Perth_min_temp))
```

Product.code	Bureau.of.Meteorology.station.number	Year	Month	Mean.minimum.temperature	Quality
IDCJAC0004	9021	1944	6	8.1	Y
IDCJAC0004	9021	1944	7	7.7	Y
IDCJAC0004	9021	1944	8	7.1	Y
IDCJAC0004	9021	1944	9	7.9	Y
IDCJAC0004	9021	1944	10	10.5	Y
IDCJAC0004	9021	1944	11	12.8	Y

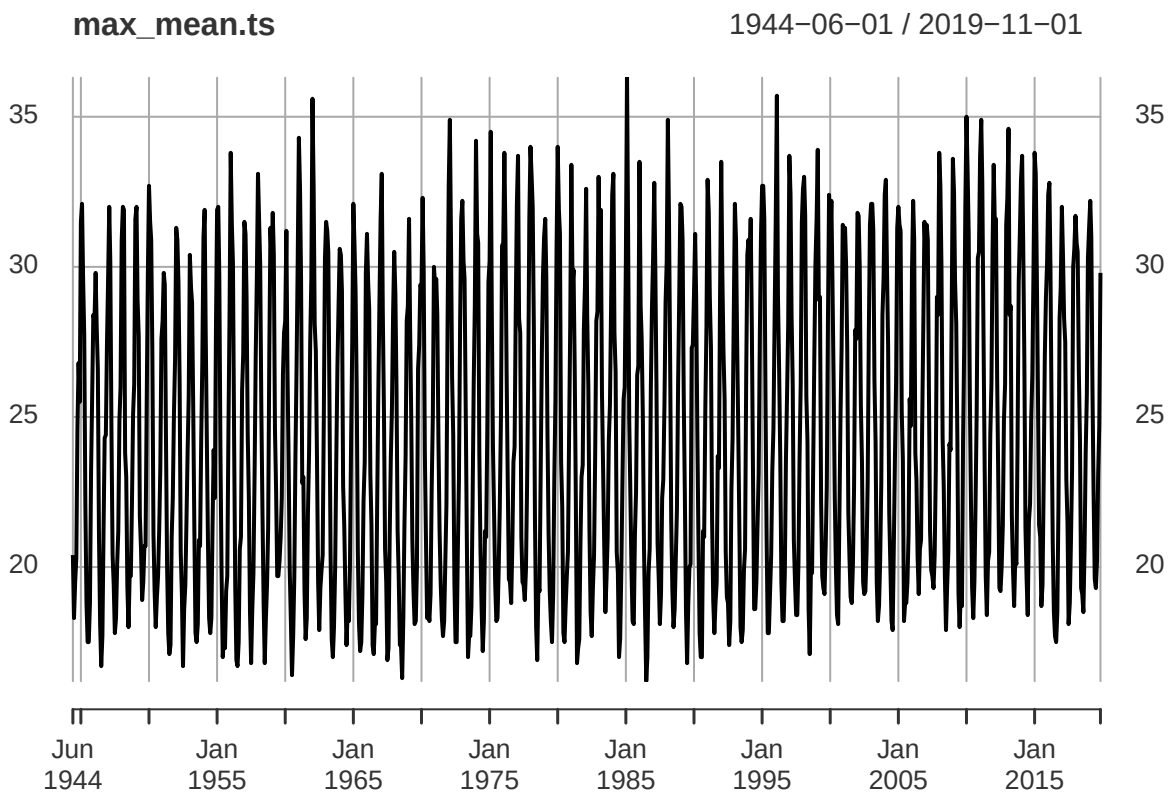
```
max_mean <- Perth_max_temp$Mean.maximum.temperature
min_mean <- Perth_min_temp$Mean.minimum.temperature
par(mfrow=c(1,2))
plot.ts(max_mean)
plot.ts(min_mean)
```



We will work with the max

1. Represent the data, first ignoring the time series structure, and then by plotting the data as a time series. Do you notice seasonality, autocorrelation, nonstationarity?

```
date <- paste(Perth_max_temp$Year, Perth_max_temp$Month, 1, sep="-")
max_mean.ts<-xts(max_mean, as.Date(date,format = "%Y-%m-%d"))
par(mfrow=c(1,1))
plot(max_mean.ts)
```



- **Trend**

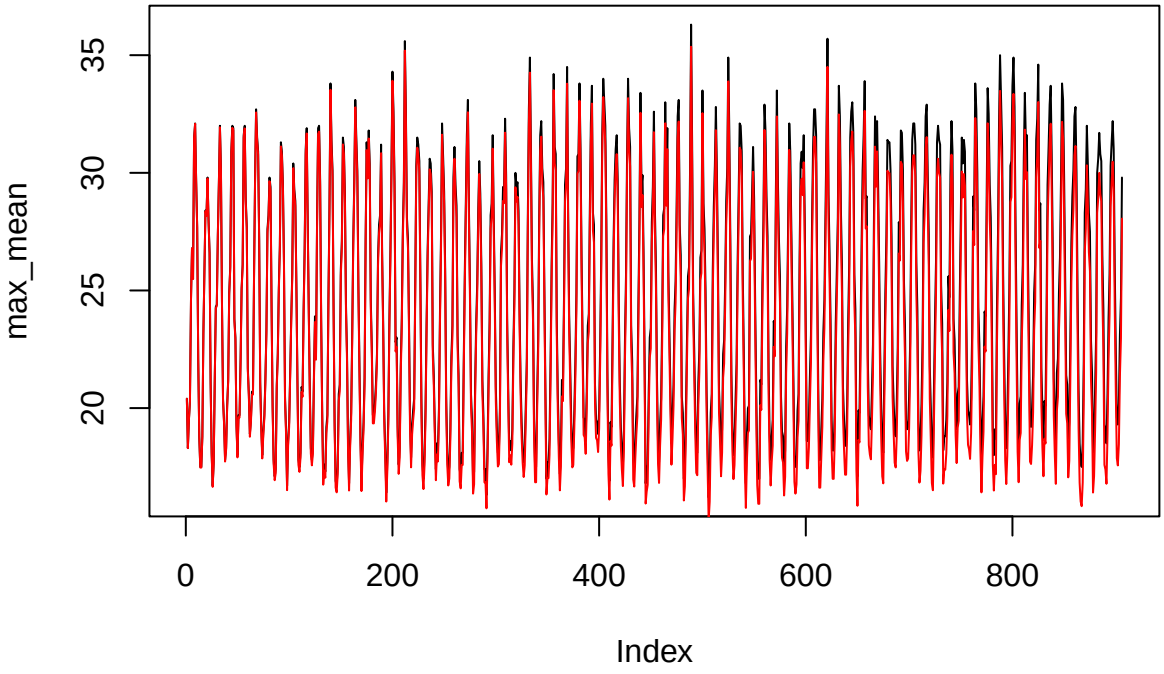
```
t <- 1:length(max_mean)
mod0 <- lm(max_mean~t)
summary(mod0)
```

```
Call:
lm(formula = max_mean ~ t)

Residuals:
    Min      1Q  Median      3Q     Max
-8.4200 -4.7840 -0.5036  4.6847 11.7127

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 2.365e+01  3.417e-01  69.205  < 2e-16 ***
t           1.926e-03  6.526e-04   2.951  0.00325 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.138 on 904 degrees of freedom
Multiple R-squared:  0.009543,  Adjusted R-squared:  0.008447
F-statistic:  8.71 on 1 and 904 DF,  p-value: 0.003247
```

Small statistically significant trend. We remove it.

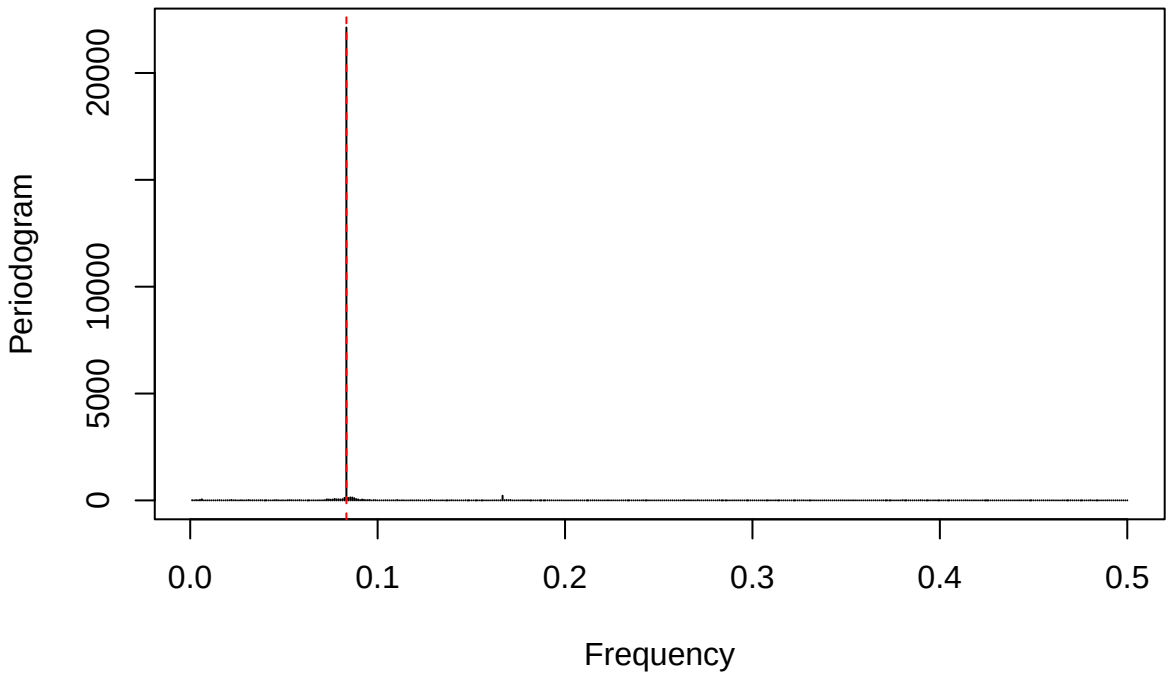
```
max_mean_detrended <- max_mean - mod0$fitted.values + predict(mod0, newdata = data.frame(t=0))
#We leave it at its first reference value to avoid truncating to positive values
#when implementing the EVI calculation.
```

```
plot(max_mean, type="l")
lines(max_mean_detrended, col="red")
```



- **Seasonality**

```
library(TSA)
per.max.detrended <- TSA::periodogram(max_mean_detrended, lwd = 1)
abline(v=1/12, col="red", lty=2)
```

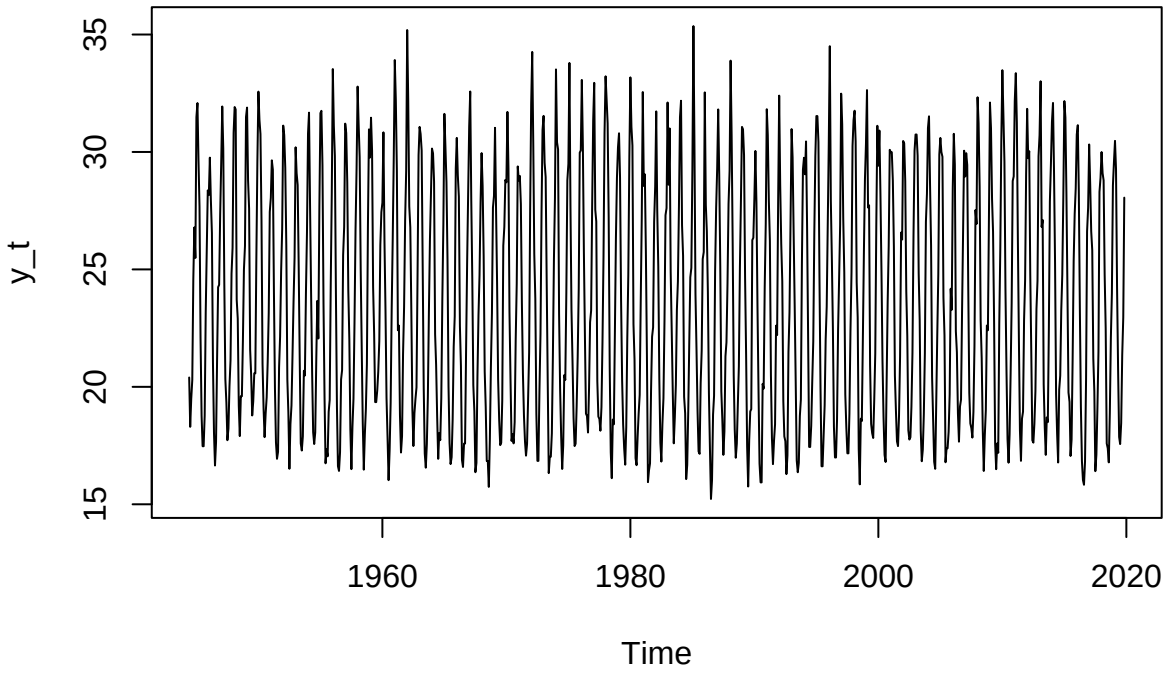



```
detach(package:TSA)
```

Huge peak at 1/12, which corresponds to 12-month seasonality.

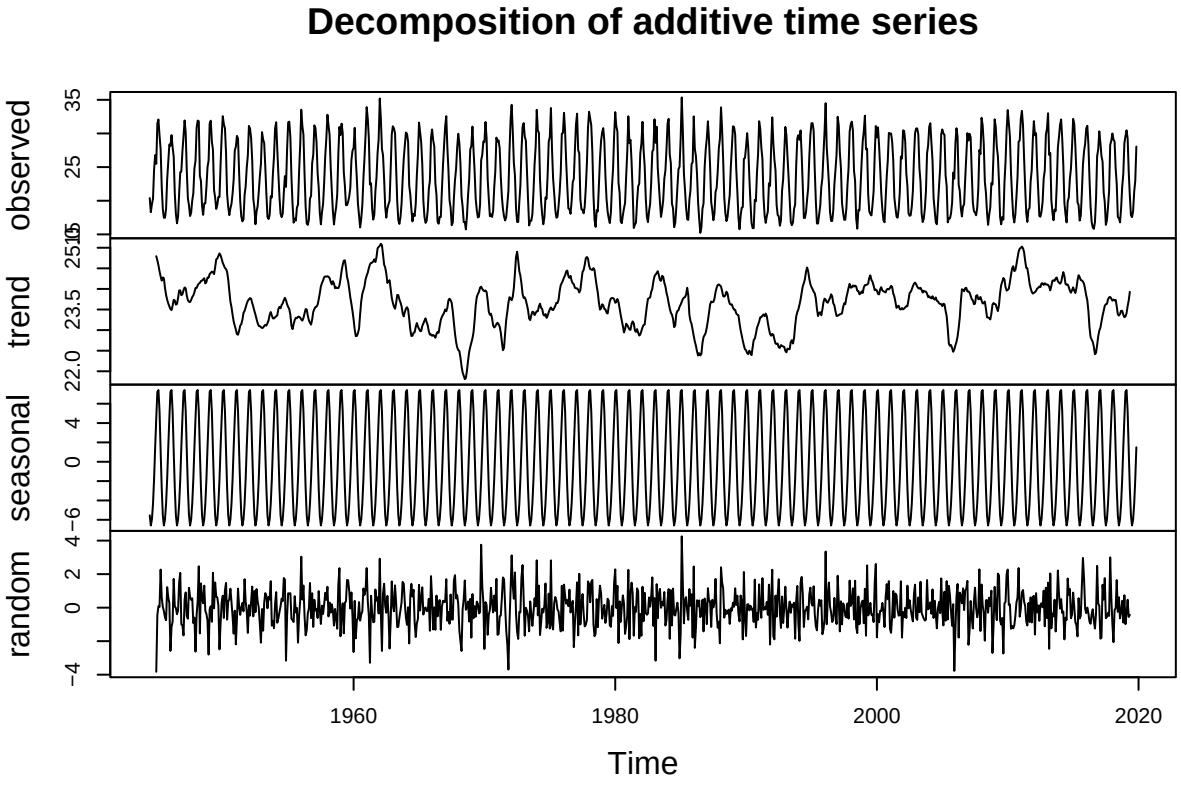
We use this to specify the frequency in the time series:

```
y_t <- ts(max_mean_detrended, frequency = 12, start = c(1944,6))
plot(y_t)
```



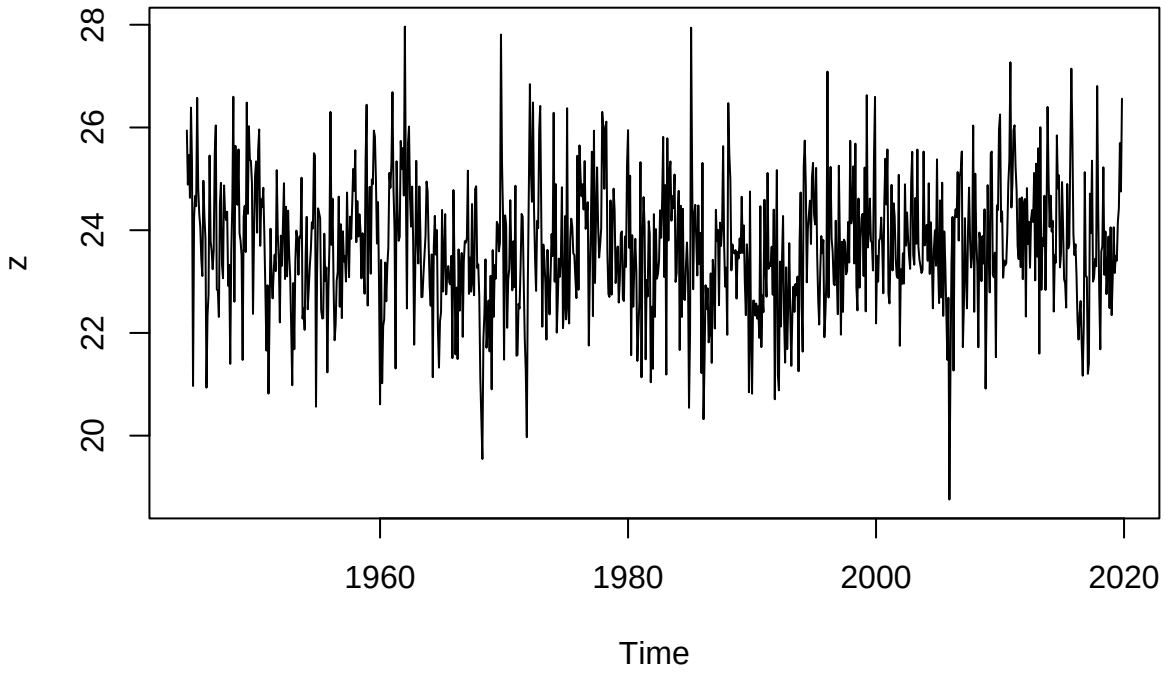
We now use additive decomposition which is adapted for temperature data

```
decomposition2 <- decompose(y_t, type = 'additive')
plot(decomposition2)
```



Given that we used additive seasonality, we subtract the seasonal component in order to obtain a deseasoned time series:

```
z <- max_mean_detrended - decomposition2$seasonal
plot(z)
```



- **Stationary test**:

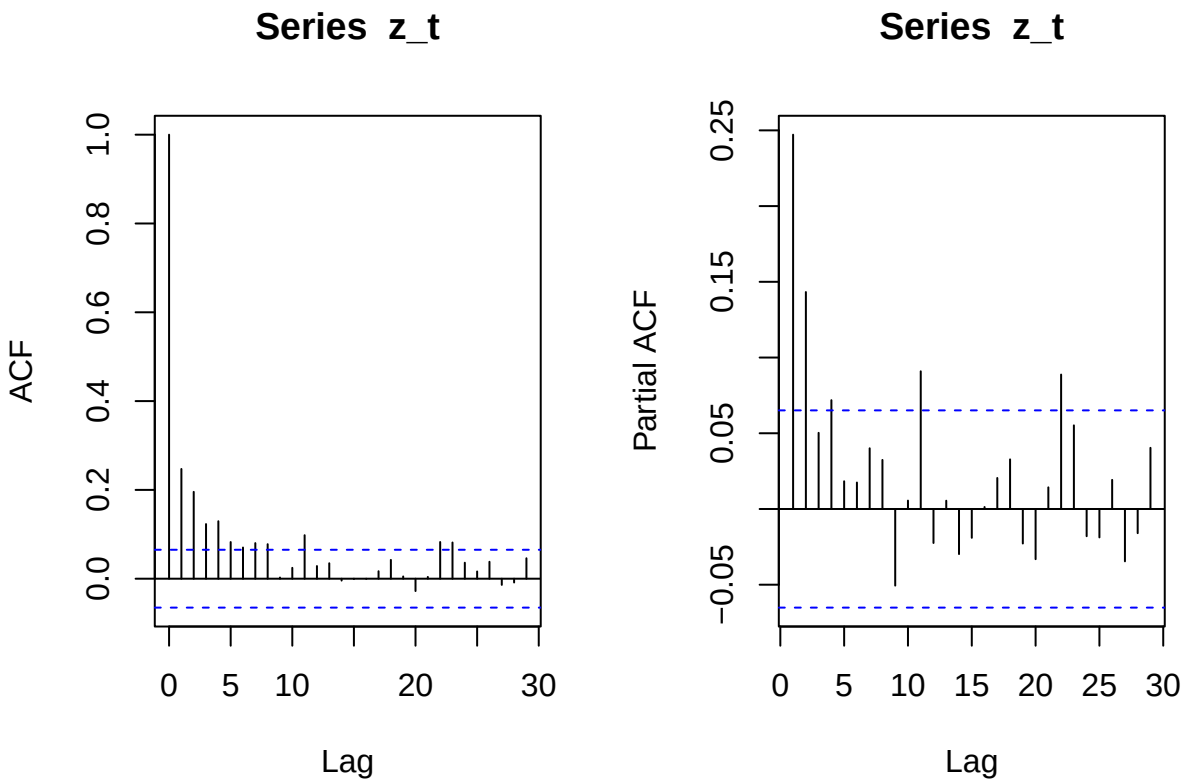
```
library(tseries)
adf.test(z) #Here, the null hypothesis is "non-stationary"
```

```
    Augmented Dickey-Fuller Test

data:  z
Dickey-Fuller = -7.8884, Lag order = 9, p-value = 0.01
alternative hypothesis: stationary
```

The test indicates that our data (detrended and deseasoned) is stationary.

- **Autocorrelation**

```
z_t <- as.numeric(z)
par(mfrow=c(1,2))
acf(z_t)
pacf(z_t)
```



It appears to have autocorrelation. We will ensure with an autocorrelation test:

```
t(Box.test.2(e_t, nlag = 1:7, type = "Ljung-Box", decim = 2))
```

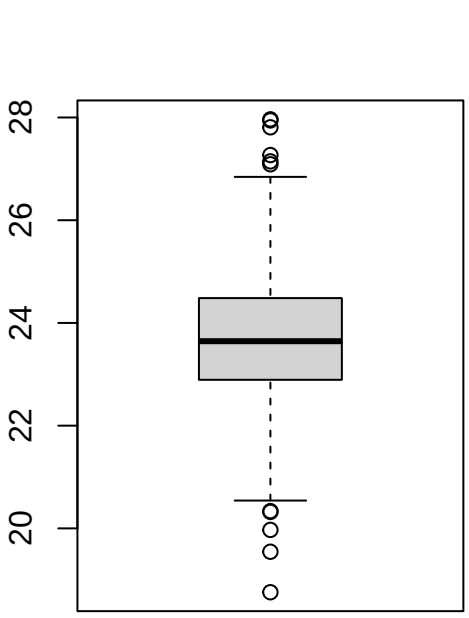
```
        [,1] [,2] [,3] [,4] [,5] [,6] [,7]
Retard     1    2    3    4    5    6    7
p-value    0    0    0    0    0    0    0
```

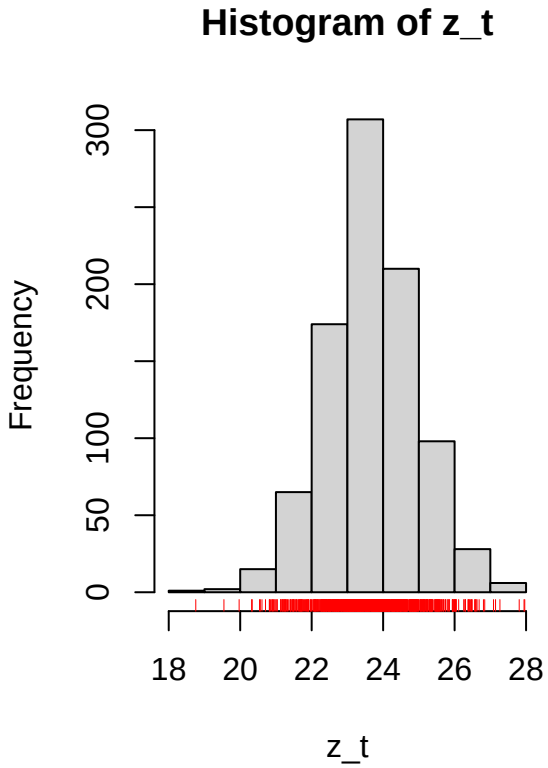
```
#Here, H0 is "time series is a white noise"
```

For all lags 1:7, we reject H_0. We therefore have dependence in the data.

2. Can you find a reasonable statistical model for the whole of the data?

```
par(mfrow=c(1,2))
boxplot(z_t)
hist(z_t)
rug(z_t, col="red")
```

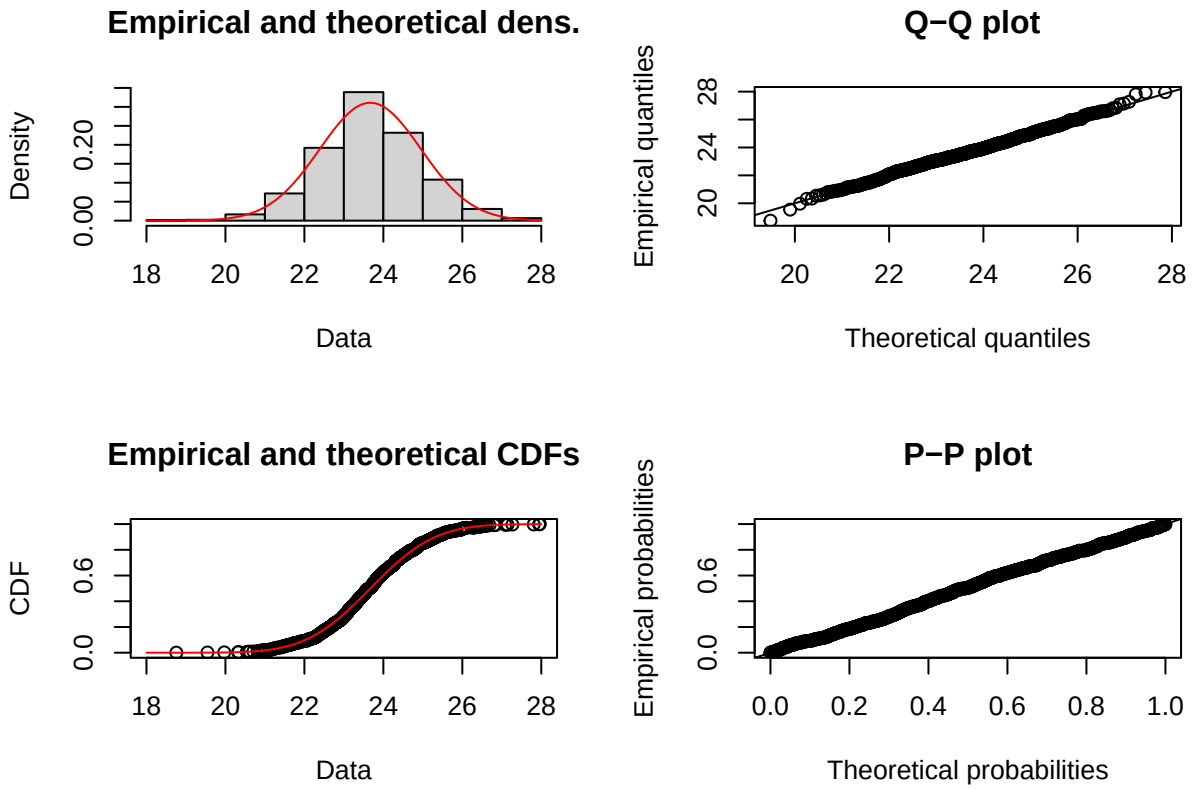




The distribution looks short or light tailed.

We can fit a normal distribution:

```
library(fitdistrplus)
norm.fit2 <- fitdist(z_t, "norm", method = 'mle')
plot(norm.fit2)
```

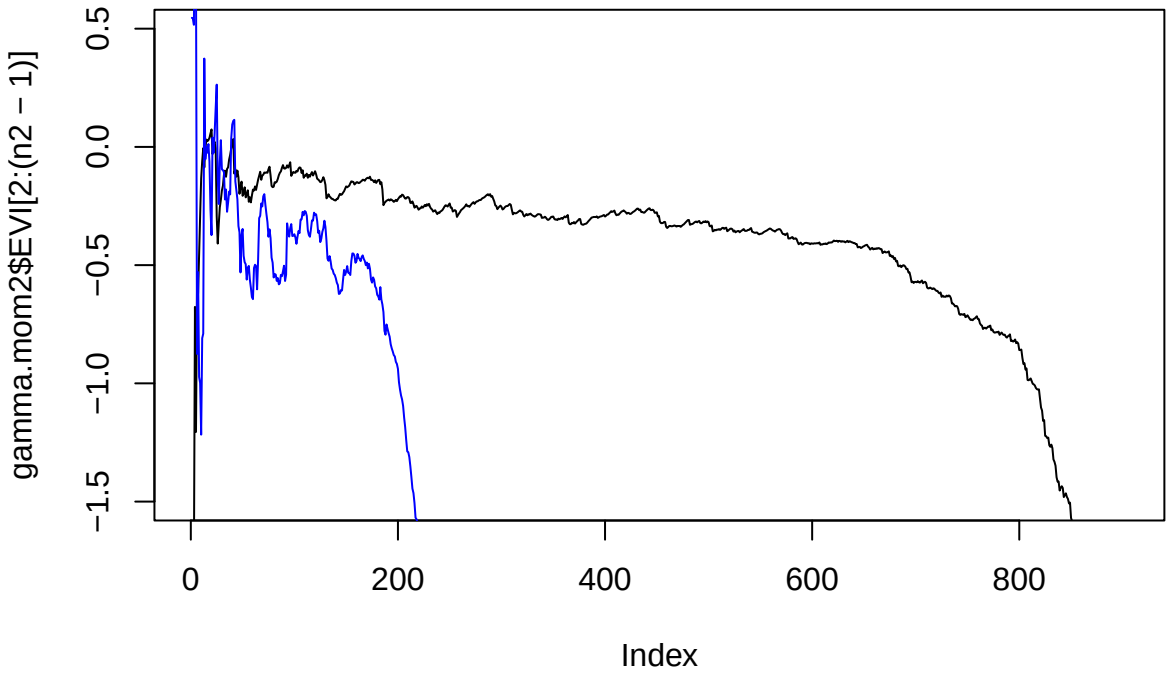


As expected, at first glance we obtained a good fit with a normal distribution. If the true distribution is normal, then we obtain that it belongs to the domain of attraction of a Gumbel ($\gamma = 0$). However, for the same reason of the previous part, we will look at this with the EVI estimators.

3. Calculate an estimate for the extreme value index of the data. Justify your choice of tuning parameters.

- Semiparametric methods

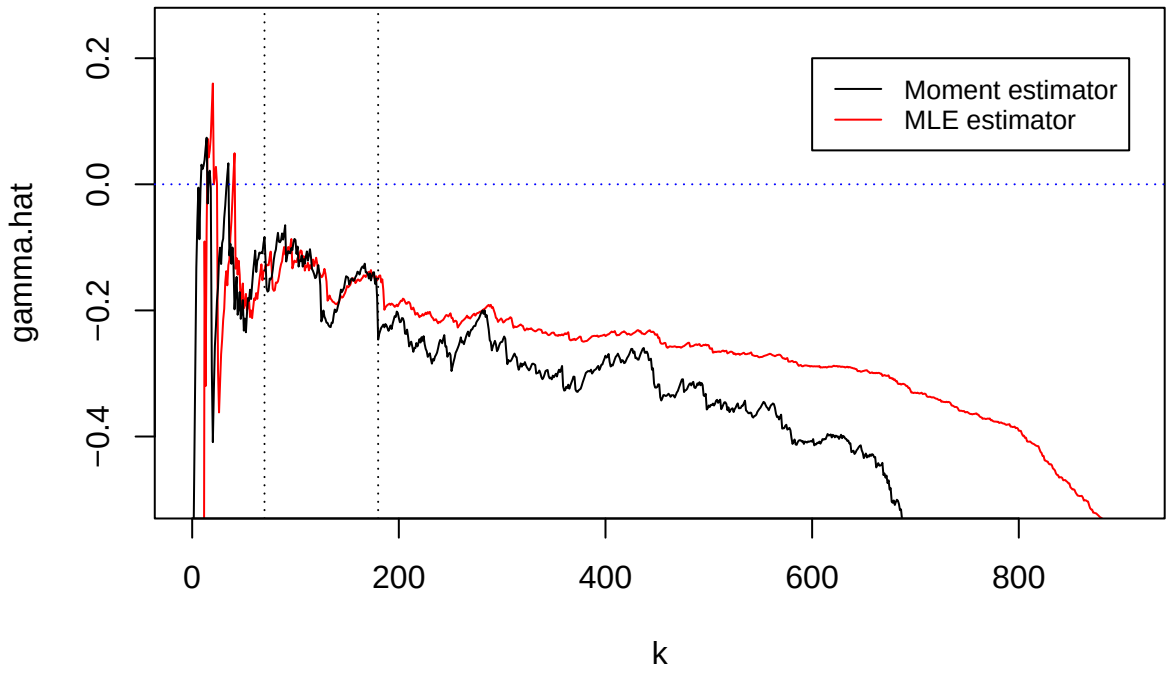
```
n2 <- length(z_t)
gamma.mom2 <- other.EVI(z_t, 1:(n2-1), method = "MO") #Moment estimator
gamma.P2 <- my_Pickands(z_t) #Our Pickands estimator
par(mfrow=c(1,1))
plot(gamma.mom2$EVI[2:(n2-1)], type="l", ylim = c(-1.5,0.5))
lines(gamma.P2[2:(n2-1)], col="blue")
```



- **Parametric method**

```
intermediate_quantile2 <- sort(z_t, decreasing = TRUE)
gamma.ml2 <- c(NA)
for (k in 2:(n2-1)){
  fit <- fevd(z_t, threshold = intermediate_quantile2[k+1], type = "GP")
  gamma.ml2[k]<-fit$results$par[[2]]
}
```

```
plot(gamma.ml2[2:(n2-1)], type="l", ylab = "gamma.hat", xlab = "k", col="red",
     ylim = c(-0.5, 0.25), xlim = c(0,(n2-1)))
lines(gamma.mom2$EVI[8:(n2-2)])
#lines(gamma.P2, col="green")
# Add a legend
legend(600, 0.2, legend=c("Moment estimator", "MLE estimator"),
       col=c("black", "red"), lty=1, cex=0.8)
abline(v=70, lty=3)
abline(h=0, col="blue", lty=3)
abline(v=180, lty=3)
```



As for the wind data, we obtain here that the data is clearly short-tailed according to these estimators.

- **Extrapolation**: We choose to work with the MLE estimator because it looks more stable.

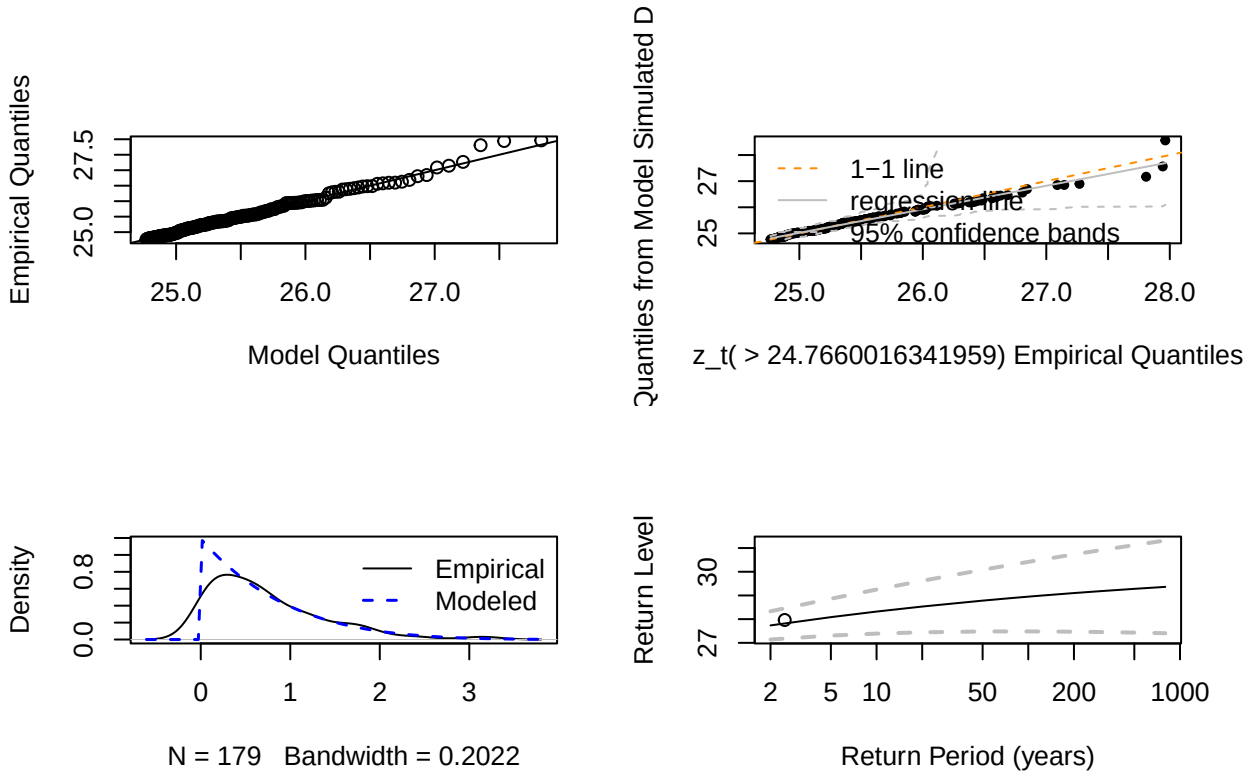
$k = 180$ seems like a decent spot (the end of the first stable region for the red curve)

```
k = 180
gamma.hat2 = gamma.ml2[k]
gamma.hat2
```

```
[1] -0.152457
```

We re-run the fitting function at the specified k to get the parameters we need:

```
set.seed(123)
u2 <- intermediate_quantile2[k] #point from which we need to extrapolate (n-k order statistic)
fit.pot2 <- fevd(z_t, threshold =  u2, type = "GP")
#Exercise: implement the command fevd with type = "GEV"
plot(fit.pot2)
```



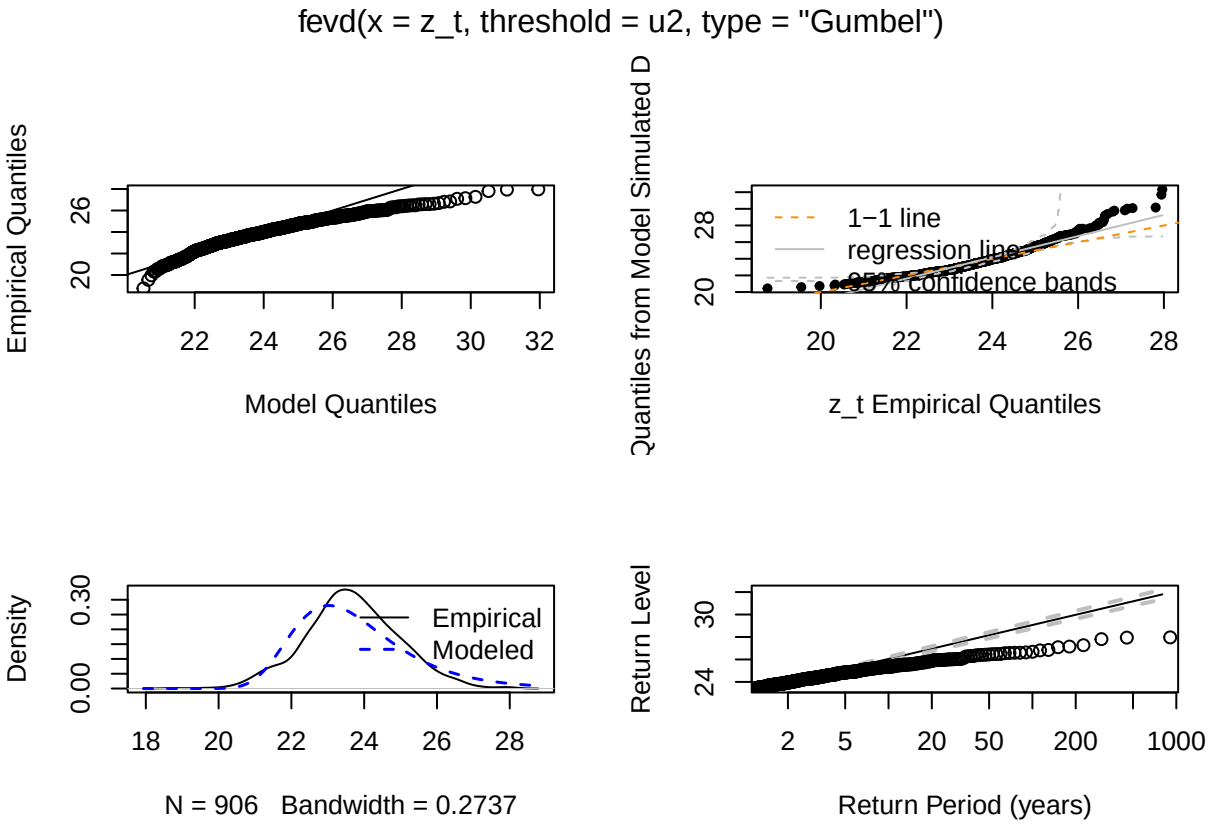
This seems a good choice of k to model the extremes to the right.

```
fit.pot2$results$par
```

```
     scale      shape
 0.8396377 -0.1457370
```

We can check if a Gumbel adjustment would work:

```
set.seed(123)
fit.pot3 <- fevd(z_t, threshold = u2, type = "Gumbel")
plot(fit.pot3)
```



We can clearly see that this does not work at all, which shows that in fact, the EVI is negative. That is to say, we are in the case of a short-tailed distribution.

4. Calculate an estimate for the extreme quantiles at level 0.99, 0.995 and 0.999. Do these make sense?

```
beta1 = 0.995
beta2 = 0.999
beta3 = 1-1/n
sigma2 = fit.pot2$results$par[[1]]
gamma2 = fit.pot2$results$par[[2]]

qbeta1.ml2 <- extreme.quantile(threshold = u2,
                               scale = sigma2,
                               shape = gamma2,
                               beta=beta1,
                               k,n2)

qbeta2.ml2 <- extreme.quantile(threshold = u2,
                               scale = sigma2,
                               shape = gamma2,
                               beta=beta2,
                               k,n2)

qbeta3.ml2 <- extreme.quantile(threshold = u2,
                               scale = sigma2,
                               shape = gamma2,
                               beta=beta3,
                               k,n2)
```

```
print(c(qbeta1.ml2,quantile(z_t, beta1)))
```

```
           99.5%
38.15782 27.11564
```

```
print(c(qbeta2.ml2,quantile(z_t, beta2)))
```

```
           99.9%
42.10000 27.94509
```

```
print(c(qbeta3.ml2,quantile(z_t, beta3)))
```

```
          99.98479%
 45.67916  27.96090
```

Unlike what we observed in the wind data, here we see that the extreme quantiles are higher than the empirical quantiles. This is due to the fact that we have much less data here (in contrast to the case of wind data), which justifies the use of interpolation to calculate the extreme quantiles (because empirical quantile estimators are limited in these situations).

5. Can you provide a confidence interval for the extreme value index and for these extreme quantiles? If so, why? If not, can you suggest a method that would allow you to do so?

- Confidence intervals are meaningless because of dependence in the data. However, you can calculate "naive" confidence intervals as in First Practical Session.