Extreme Value Theory example

Let's start by uploading the required packages. The `evir` package is one of the many packages to use EVT on data. Another good one is `evd`. But since `evir` is more pedagogical, we stick to it.

```
library(evir)
library(quantmod)
```

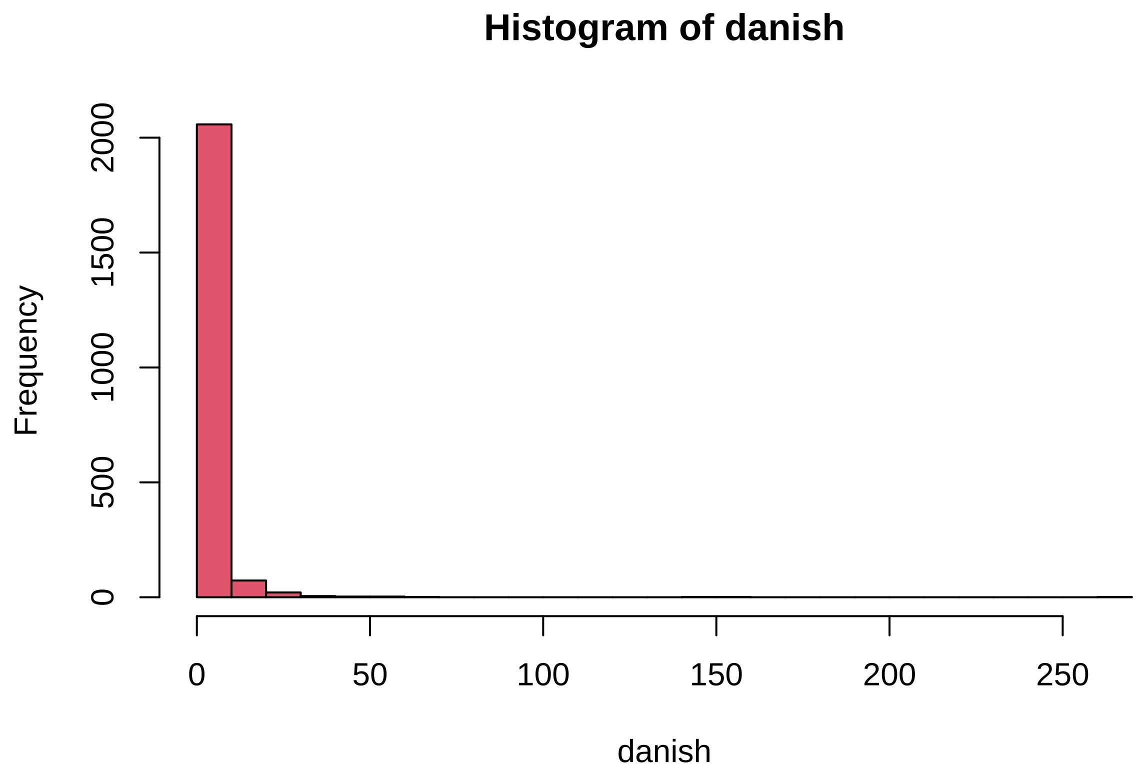
Let us consider the `danish` dataset, containing large fire insurance claims (expressed in 1000) in Denmark from Thursday 3rd January 1980 until Monday 31st December 1990. The data are available in the `evir` package.

```
data(danish)
head(danish)
```

```
## [1] 1.683748 2.093704 1.732581 1.779754 4.612006 8.725274
```

The first thing to do is to have a look at the data and at some basic statistics.

```
hist(danish, 30, col=2) # Try to produce a better plot with ggplot2
```



```
summary(danish)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.321   1.778   3.385   2.967 263.250
```

```
sd(danish)
```

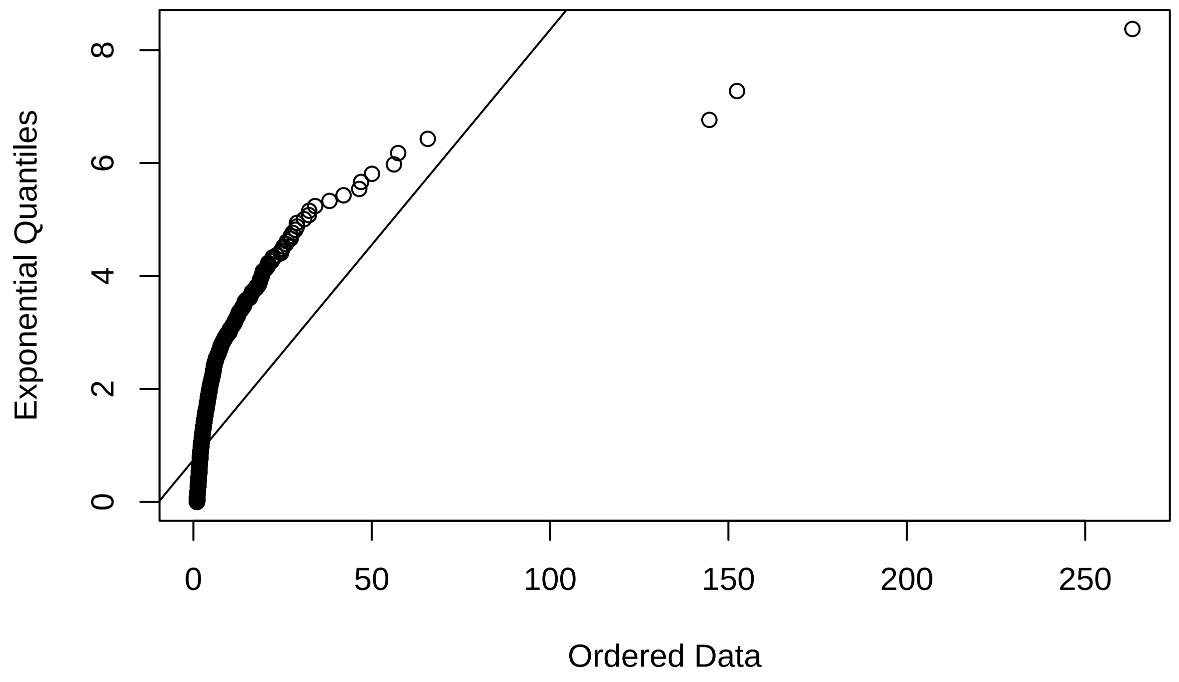
```
## [1] 8.507452
```

As expected (we are considering claims, so only disbursements for us, if we are an insurance company), the data are asymmetric (look at the range and the inter-range) and skewed-to-the-right or positively skewed (the mean is indeed larger than the median). The standard deviation is quite large, compared to the mean, indicating a sensible variability.

With a simple exponential QQ-plot, we can try to understand if heavy tails are present or not. Given the things we have just seen, we would say yes. But let us verify.

The function `qplot` in the `evir` package allows us easily have the plot. The function is built on a GPD, hence the exponential is easily obtained by setting the parameter xi to 0.

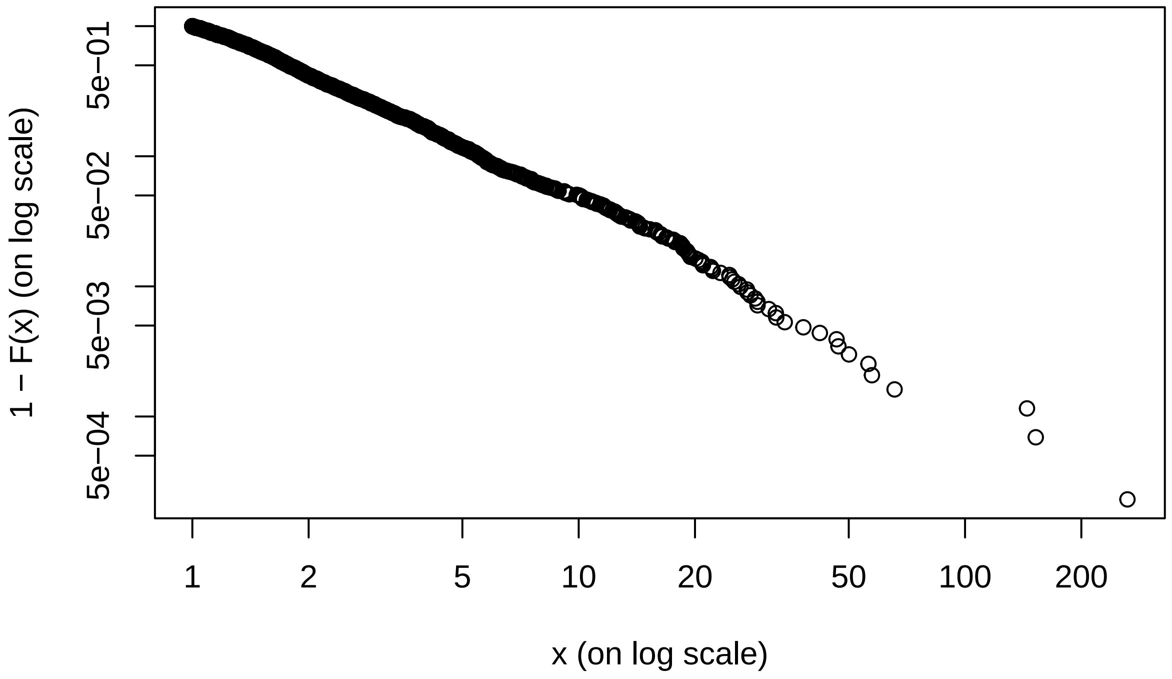
```
qplot(danish,xi=0)
```



The clear concavity in the plot is a strong signal of the presence of heavy tails.

What about a Zipf plot, to look for the behavior of the survival function? The plot is easily made with the function `emplot` (empirical plot). It is important to specifiy the option *xy* to have a log-log representation.

```
emplot(danish,'xy')
```

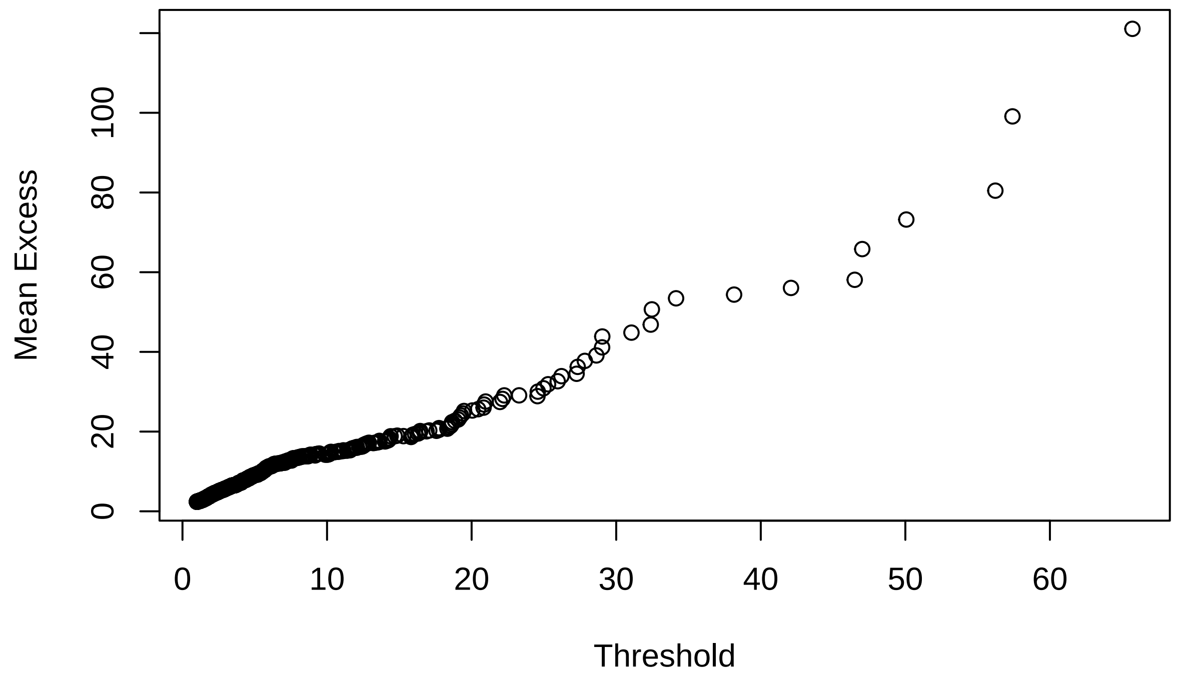


A clear negative linear slope is present. This is a first signal of the fat tailed nature of the data. But remember: a Zipf plot verifies a necessary yet not sufficient condition! Looking at the range of the plot, the credibility of the plot seems ok. The mean and the variance are below 5, while we observe a maximum which is two orders of magnitude larger.

Given that linearity appears from the very beginning, we can think that our Danish claims actually follow a pure power law.

But a Zipf plot is not enough. Let us also consider a Meplot, using the homonymous function `meplot`.

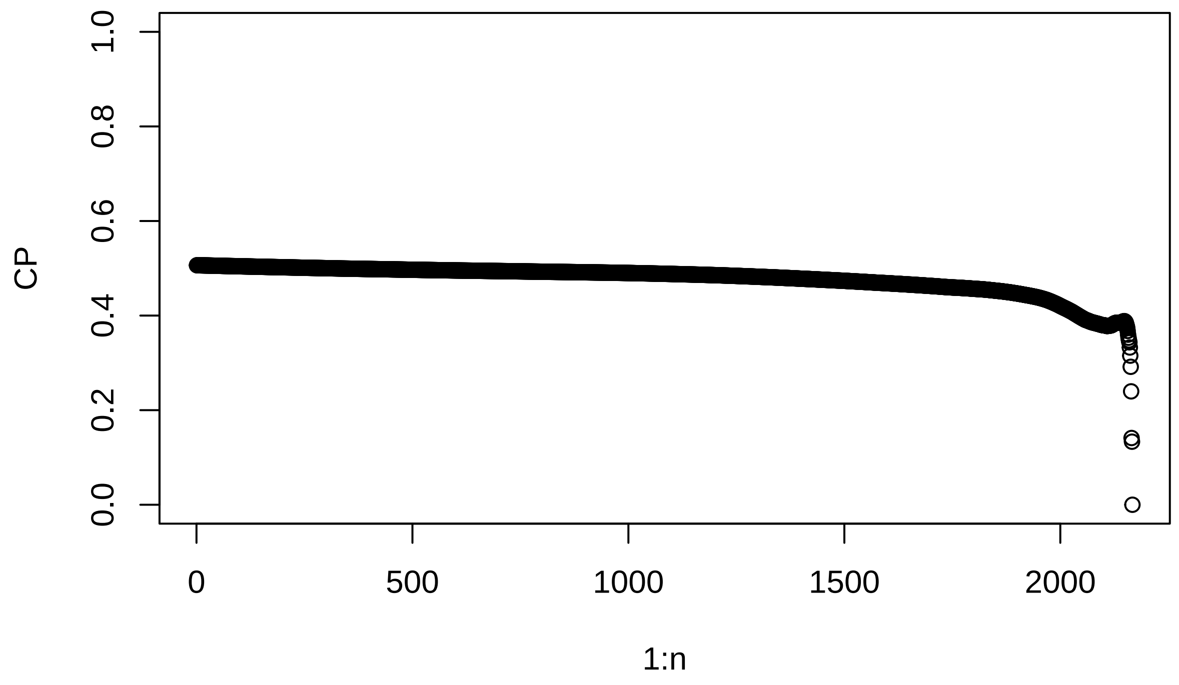
```
meplot(danish)
```



The plot is consistent with van der Wijk's law. Another signal of the presence of a fat tail.

A concentration profile (CP) is another reliable tool to better understand the tail. To build a CP, we can use the functions in the `ineq` library.

```r
library(ineq)
sort_danish=sort(danish) # We sort the data
n=length(danish)
CP=c() #Empty array for storage
for (i in 1:n) {
  CP[i]=ineq(sort_danish[i:n],type="Gini") # Truncated Gini
}
plot(1:n,CP,ylim=c(0,1))
```



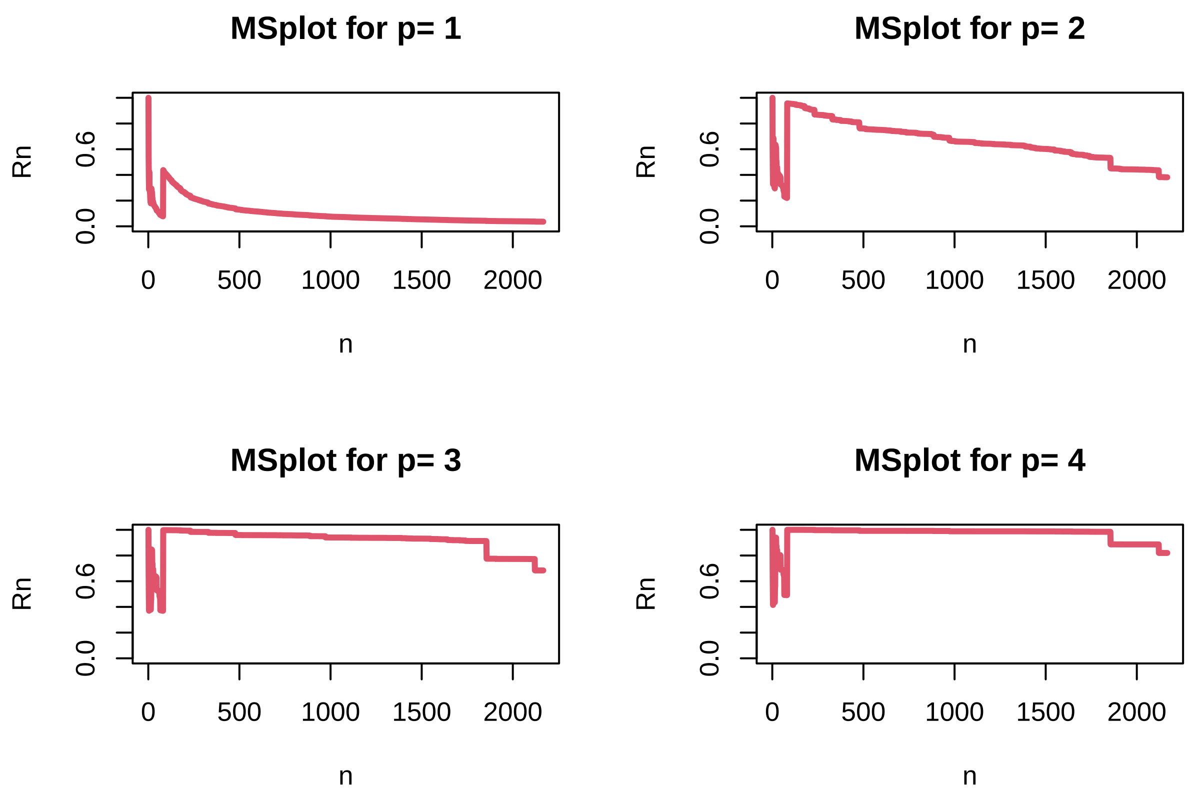
The nearly horizontal behavior we observe (the last part of the plot is to be ignored for the limited amount of points considered) can be seen as a further signal of Paretianity.

Let us try to say something about moments using the MS plot. Here below a simple code to check for the first 4 moments.

```
MSplot <- function(data,p=4) {
  par(mfrow = c(2, 2))
  x=abs(data)
  for (i in 1:p) {
    y=x^i
    S=cumsum(y)
    M=cummax(y)
    R=M/S
    plot(1:length(x),R,type='l', col=2, lwd=3, ylim=c(0,1),xlab='n', ylab='Rn',
         main=paste("MSplot for p=",i))
  }
  par(mfrow = c(1, 1))
 # return(R)
}
```

Let us run it.

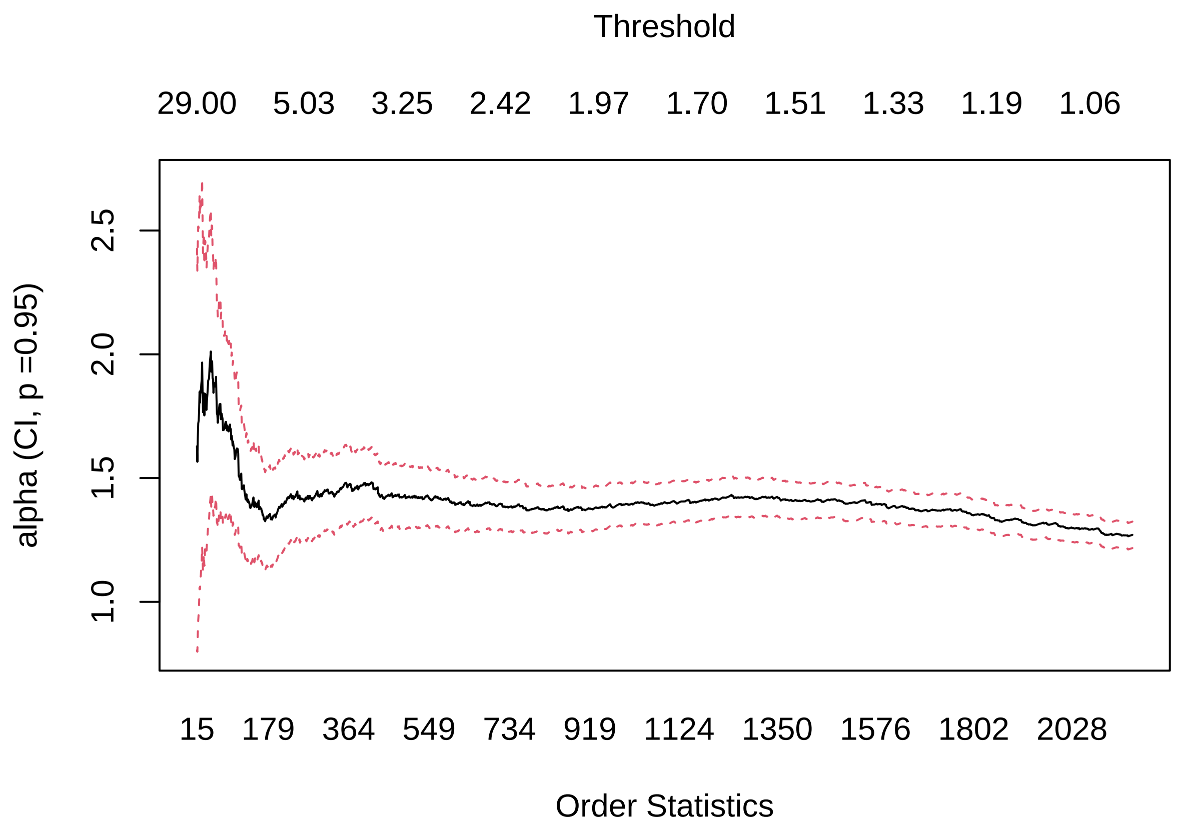
```
MSplot(danish)
```



We can see that for the first moment convergence is clear, while for the others (starting from the second) we can suspect that they are not defined. If confirmed, a similar finding would tell us that the standard deviation we have computed above is useless fo inference.

A Hill plot is an excellent way to give extra substance to what we have just said about the moments. If our guesses are correct, we expect an alpha between 1 and 2, or a xi smaller than 1 but larger than 0.5

```
hill(danish)
```



This is indeed the case. A value of alpha around 1.5 is highly plausible looking at the plot. The stability seems to kick in around a threshold of 5.5 (look at the numbers on top). In terms if xi we expect about 1/1.5, that is 0.6 or so. In EVT, my suggestion is not to waste your time too much with second or third decimals. The first one is more than enough.

The 5.5 threshold seems compatible with both the Zipf plot and the meplot. About 10% of all the claims lie above the threshold.

Let us fit a GPD above such a threshold. If the fit is reliable, the tail parameter should be stable under higher thresholds.

```
fit=gpd(danish,5.5)
tail(fit)
```

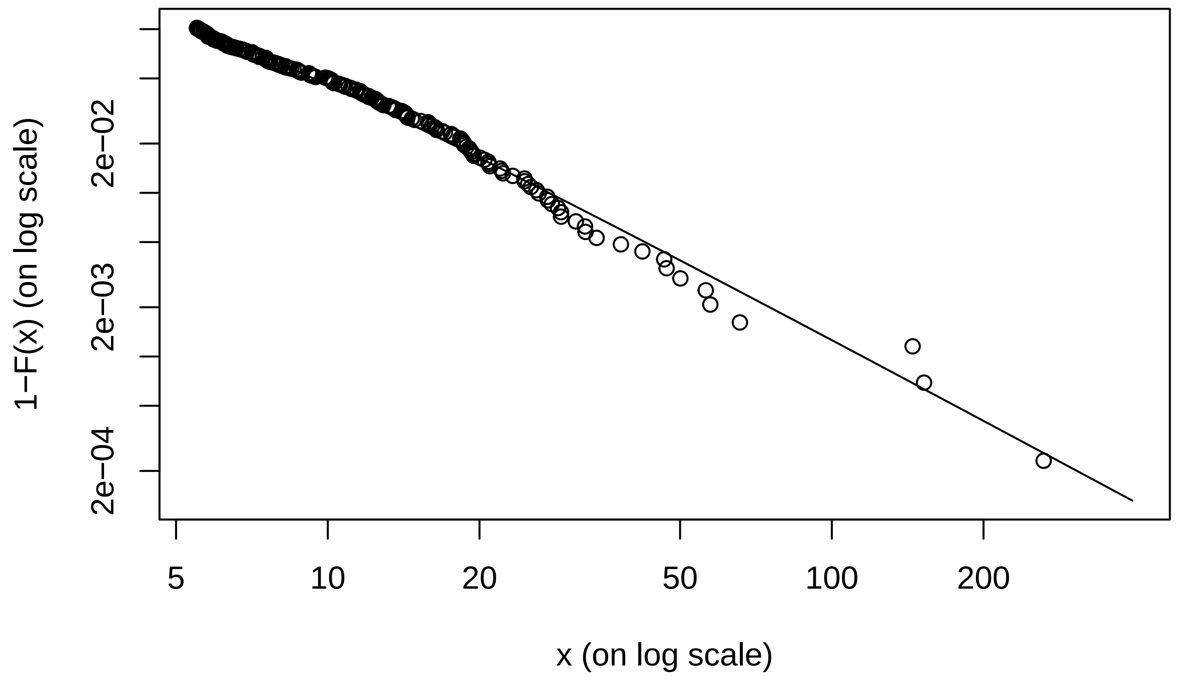
```
## $par.ests
##        xi      beta
## 0.602773 4.334780
##
## $par.ses
##         xi       beta
## 0.1191536 0.5664205
##
## $varcov
##              [,1]        [,2]
## [1,]  0.01419759 -0.04259004
## [2,] -0.04259004  0.32083219
##
```

```
## $information
## [1] "observed"
##
## $converged
## [1] 0
##
## $nllh.final
## [1] 678.3826
```

We get a xi=0.6 and significant (the s.e. is 0.12). For beta (or sigma in other parametrizations) we have 4.33, also significant (why?).

Let us verify the fitting of the tail. We can just write the following. Please remove the commenting #. Notice that an interactive menu opens with `plot(fit)` (in the pdf it will not show properly, please run the code).

```
#plot(fit)
tailplot(fit)
```



The fitting is quite satisfactory.

Would a higher threshold change the value of xi?

```
gpd(danish,20)$par.ests[1]
```

```
##        xi
## 0.6840479
```

```r
gpd(danish,20)$par.ses[1]
```

```
##        xi
## 0.2749542
```

Qualitatively we would say no: a higher threshold preserves the xi.

A value of 0.6 or so seems plausible and in line with our previous findings. Notice that it is confirmed that the second moment is not finite!

Given the GPD fit, we could be interested in estimating a very high quantile (VaR) and the corresponding ES. This approach is much more reliable than using empirical estimates, especially under fat tails.

We can rely on a useful function in the `evir` package. The function requires a GPD fit in its arguments.

Let us start from a 99% confidence level.

```r
riskmeasures(fit,0.99)
```

```
##         p quantile    sfall
## [1,] 0.99 27.46474 71.70779
```

The empirical counterparts are

```r
quantile(danish,0.99) #99% Var
```

```
##      99%
## 26.04253
```

```r
mean(danish[danish>=quantile(danish,0.99)]) #99% ES
```

```
## [1] 58.58575
```

While the VaR is comparable, the empirical ES seems to underestimate the tail risk.

Let us consider the so-called worst-case scenario, i.e. quantities at the 99.9% confidence level.

```r
riskmeasures(fit,0.999)
```

```
##          p quantile    sfall
## [1,] 0.999 115.1248 292.3878
```

Notice that the empirical quantities, ignoring EVT, would make us underestimate the tail risk even more.

```r
quantile(danish,0.999) #99.9% Var
```

```
##    99.9%
## 131.5519
```

```r
mean(danish[danish>=quantile(danish,0.999)]) #99.9% ES
```

```
## [1] 186.7737
```

Notice that in this case also the empirical VaR is less reliable.

And for 99.99? We are really zooming into the tail here. Empirically it is like we are considering less than 1 observation in the sample!

```r
riskmeasures(fit,0.9999)
```

```
##           p quantile    sfall
## [1,] 0.9999 466.3411 1176.558
```

```r
quantile(danish,0.9999) #99.99% Var
```

```
##  99.99%
## 239.243
```

```r
mean(danish[danish>=quantile(danish,0.9999)]) #99.99% ES
```

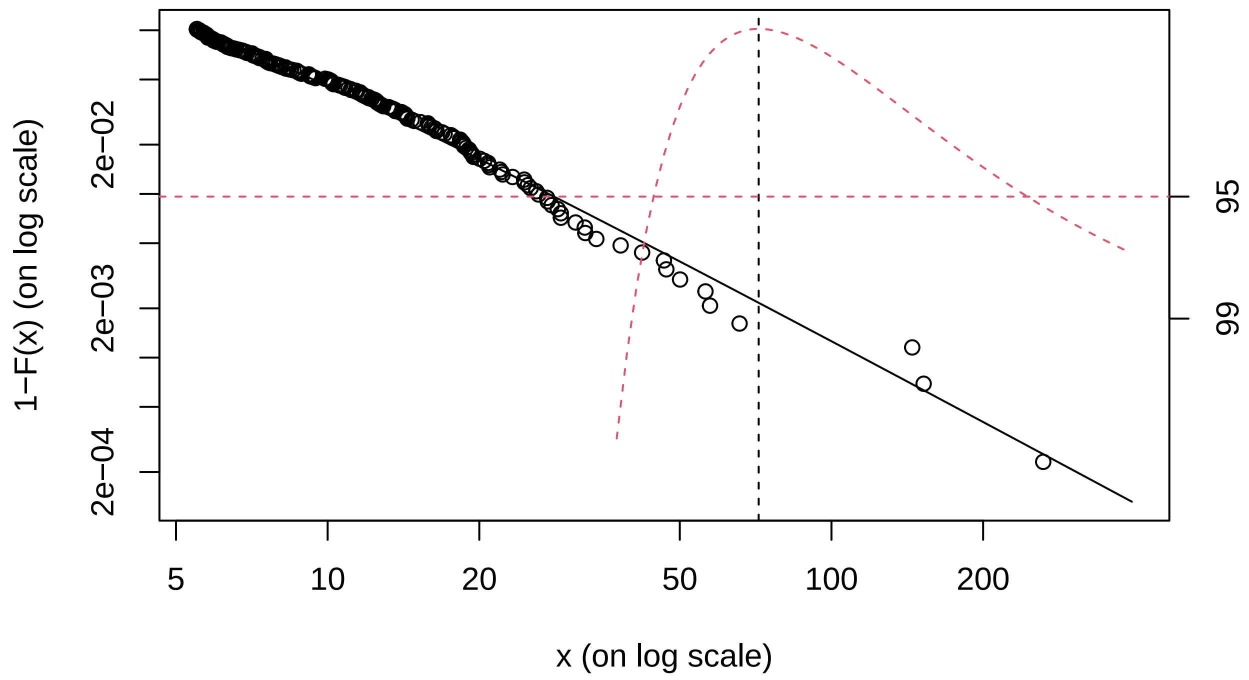
```
## [1] 263.2504
```

Oops! EVT definitely wins.

Two useful functions I also suggest to check are `gpd.q` and `gpd.sfall`, which also provide confidence intervals. They need to be combined with `tailplot`.

For example

```r
gpd.sfall(tailplot(fit), 0.99, ci.p = 0.95)
```



```
##  Lower CI  Estimate  Upper CI
##  44.65648  71.70779 244.01195
```

Exercise 4 U

A lot of people speak about cryptocurrencies and their extremely fat tails.$ Is this correct? In the following exercise, you will find out.

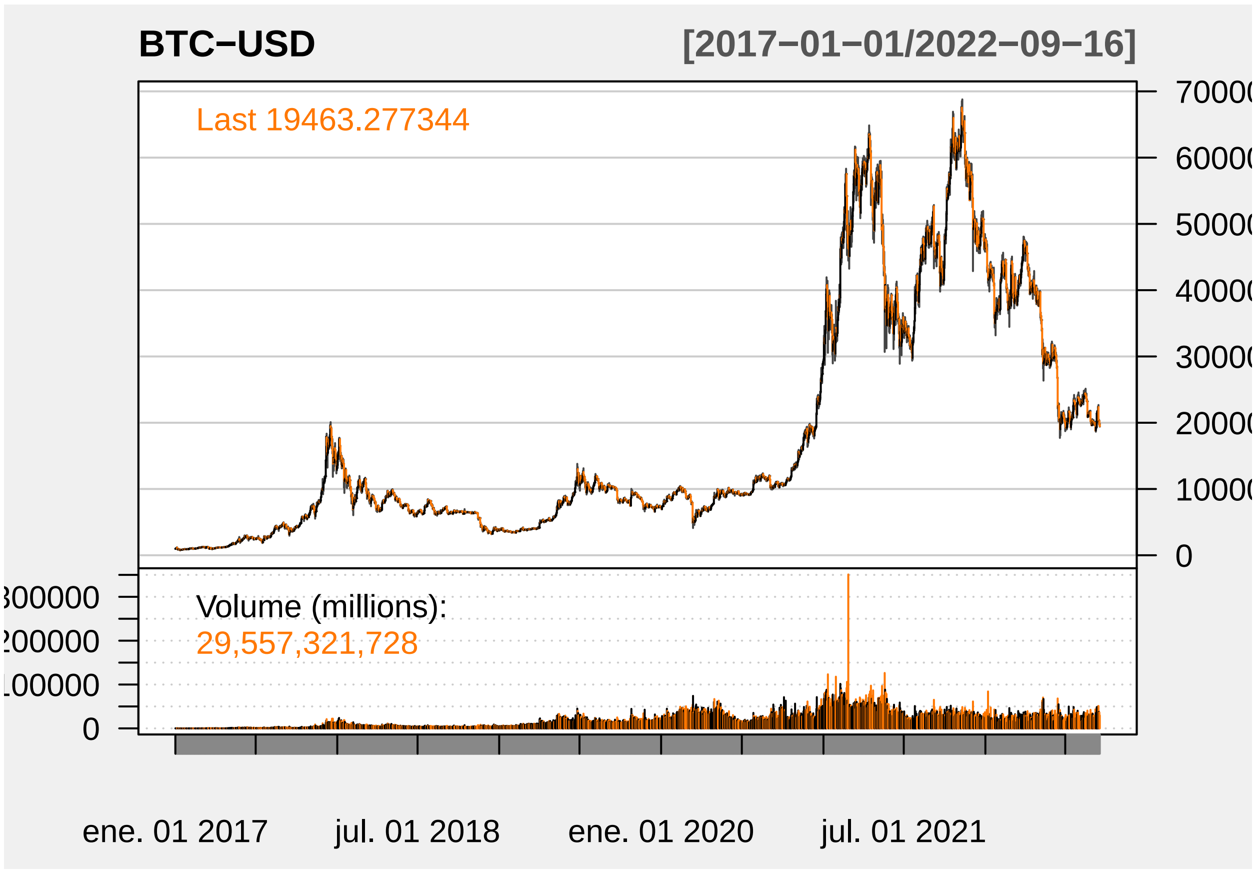
We will discuss the results together in a live class on Zoom, so please try to solve the different steps.

Consider Botcoin and Ethereum.

```
portfolio = c("BTC-USD","ETH-USD")
getSymbols(portfolio, src="yahoo", from="2017-01-01")
```

```
## [1] "BTC-USD" "ETH-USD"
```

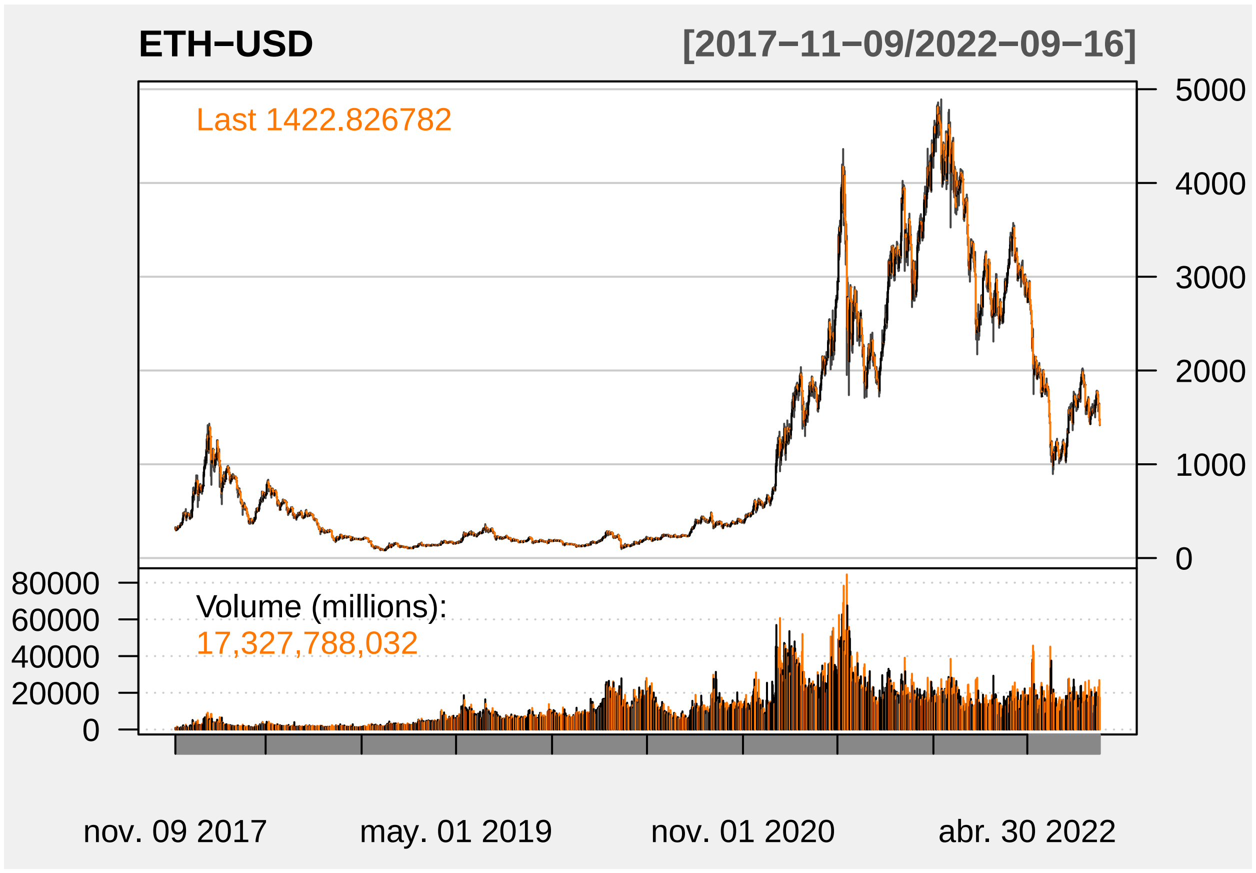
```
chartSeries(`BTC-USD`,up.col="black",theme="white")
```



```r
getSymbols(portfolio, src="yahoo", from="2017-01-01")
```

```
## [1] "BTC-USD" "ETH-USD"
```

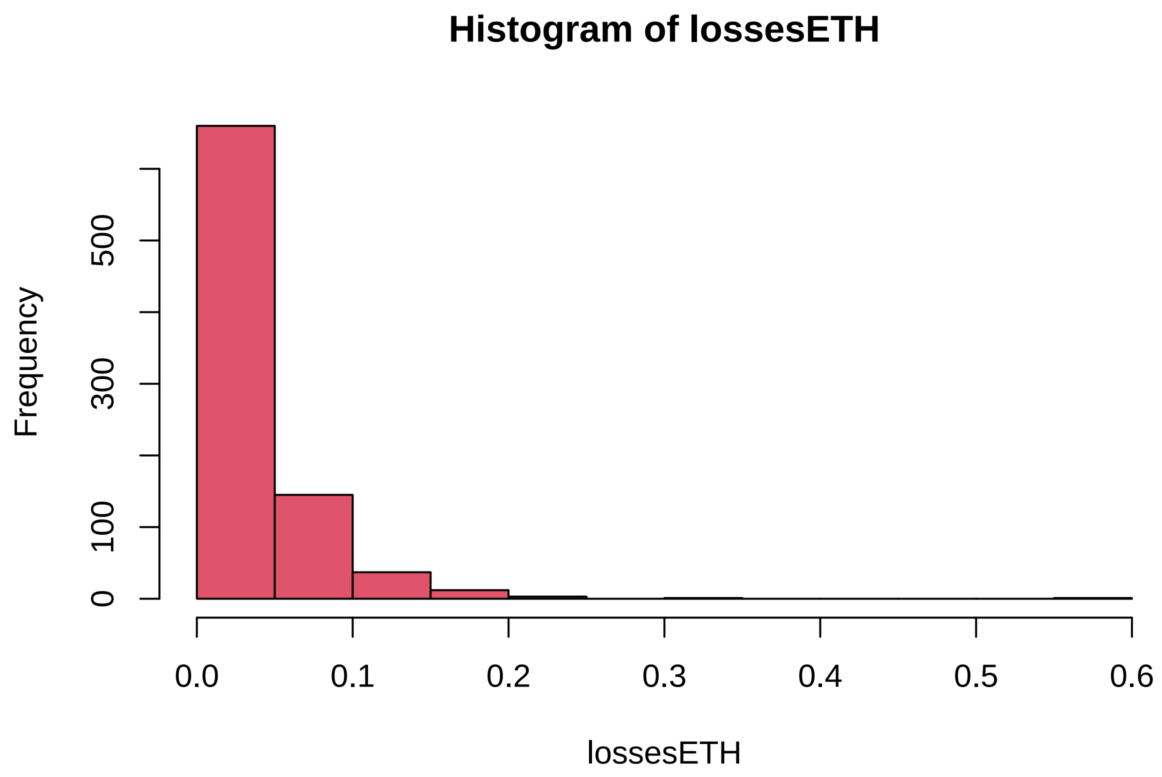
```r
chartSeries(`ETH-USD`,up.col="black",theme="white")
```



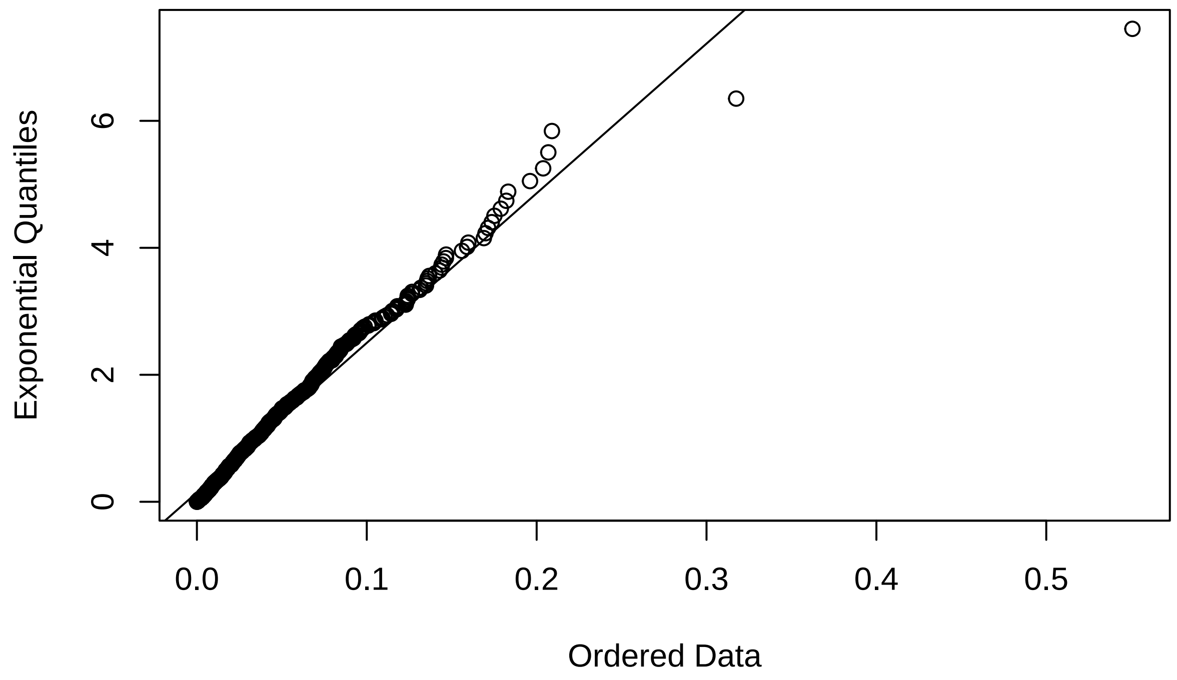
Now, for both Bitcoin and Etheruom, consider losses and returns. For example, for Ethereum:

```r
ethereum=Ad(`ETH-USD`) # Cl(`ETH-USD`)
lossreturnsETH=diff(log(ethereum))
lossesETH=-lossreturnsETH[lossreturnsETH<0] # Change the sign!
returnsETH=lossreturnsETH[lossreturnsETH>0]
```

```r
hist(lossesETH,col=2)
```



```
qplot(lossesETH,0)
```



And now continue with the analysis of tails. . .