EVT in Action

A first simple data set

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)
```

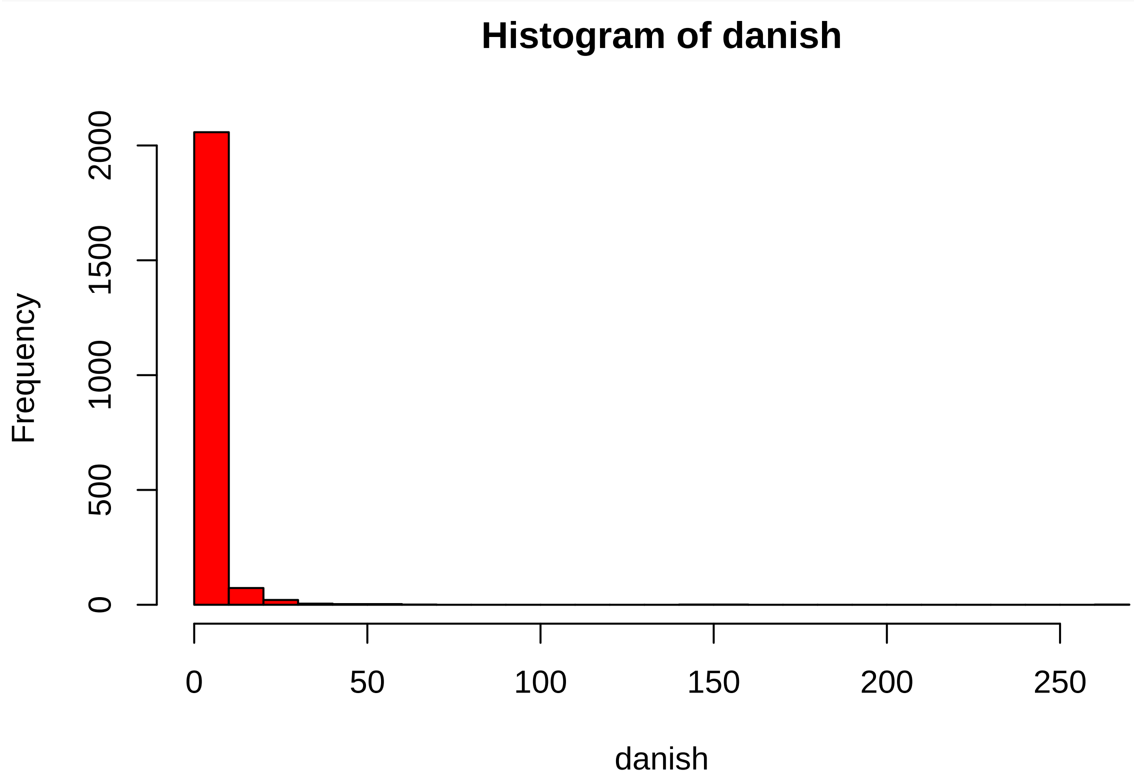
Let us consider the `danish` dataset, containing large fire insurance claims (expressed in 1000) in Denmark from Thursday 3rd January 1980 until Monday 31st December 1990. The data are available in the `evir` package.

```
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
```

```r
sd(danish)
```

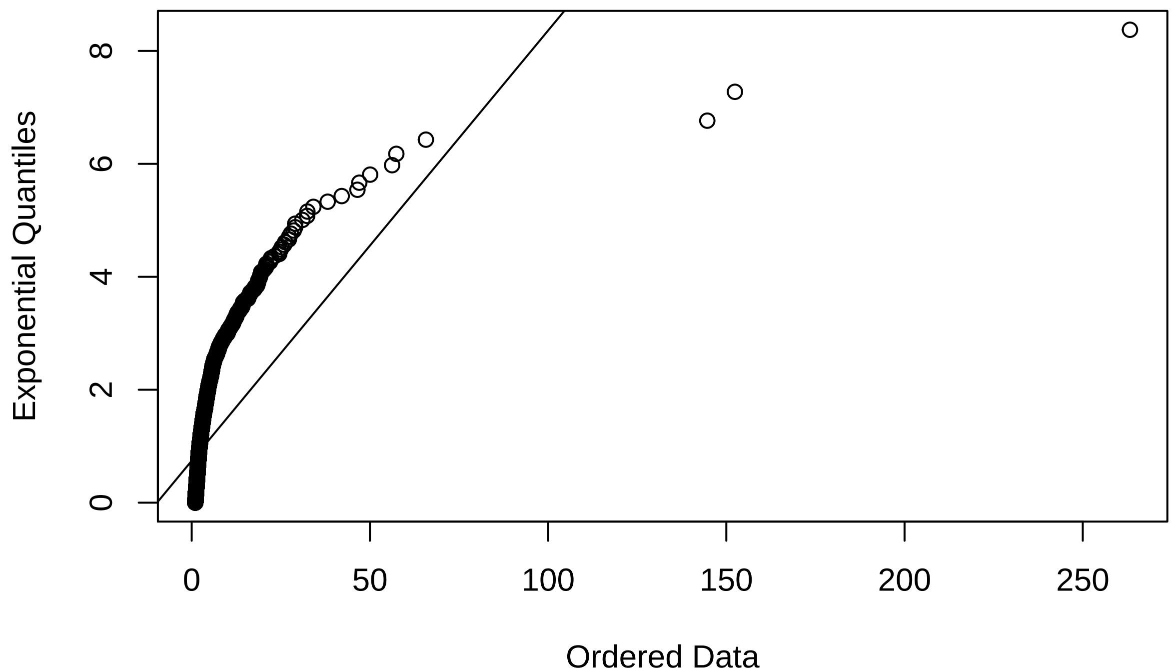
```
## [1] 8.507452
```

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 ξ to 0.

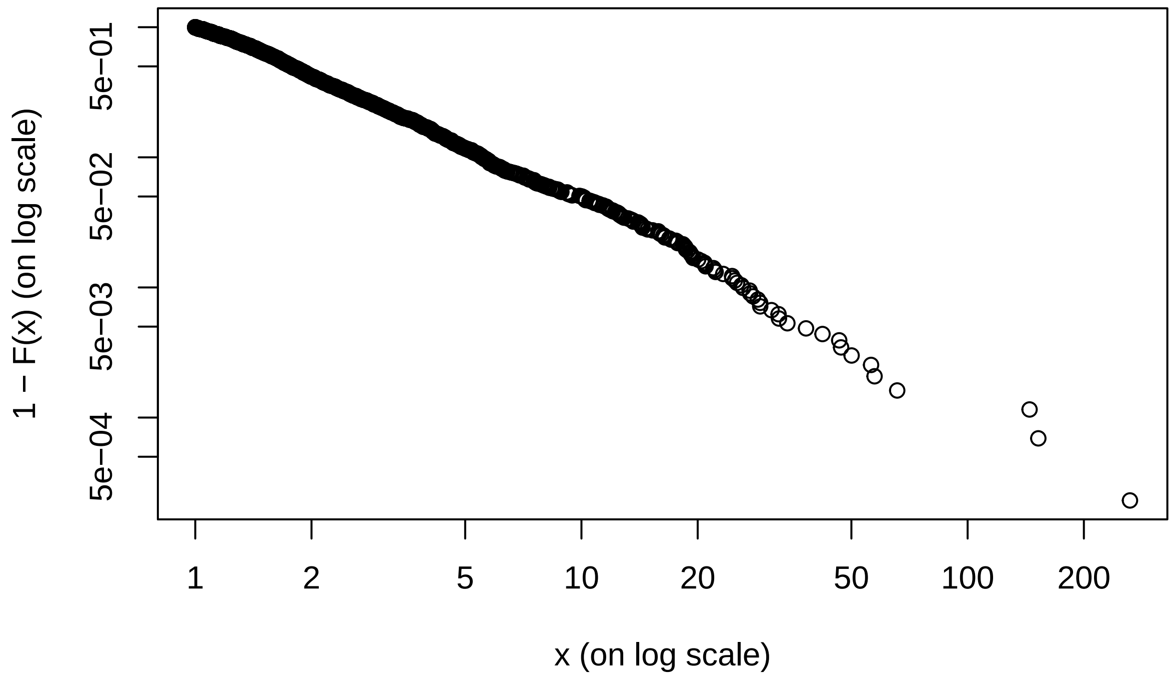
```r
qplot(danish,xi=0)
```



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 specifify the option *xy* to have a log-log representation.

```r
emplot(danish,'xy') # Zipf plot
```

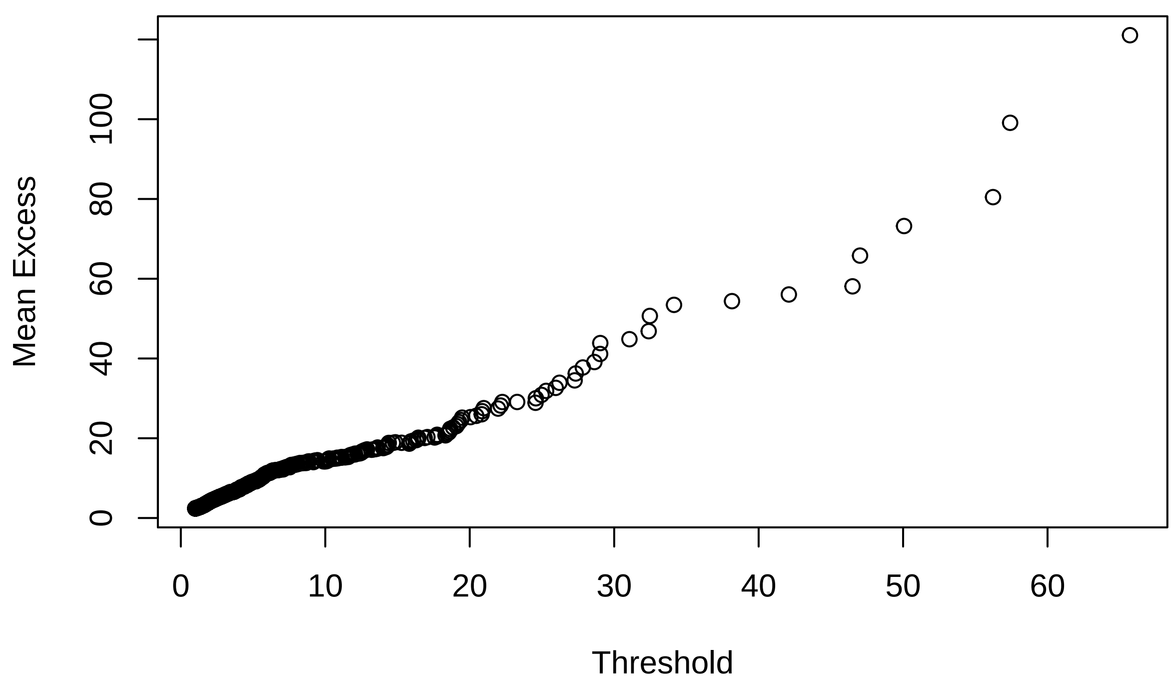


A clear negative linear slope is present. This is a first signal of the fat tailed nature of the data. But remember: a Zipf plot verifies a necessary yet not sufficient condition! Looking at the range of the plot, the credibility of the plot seems ok. The mean and the variance are below 5, while we observe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

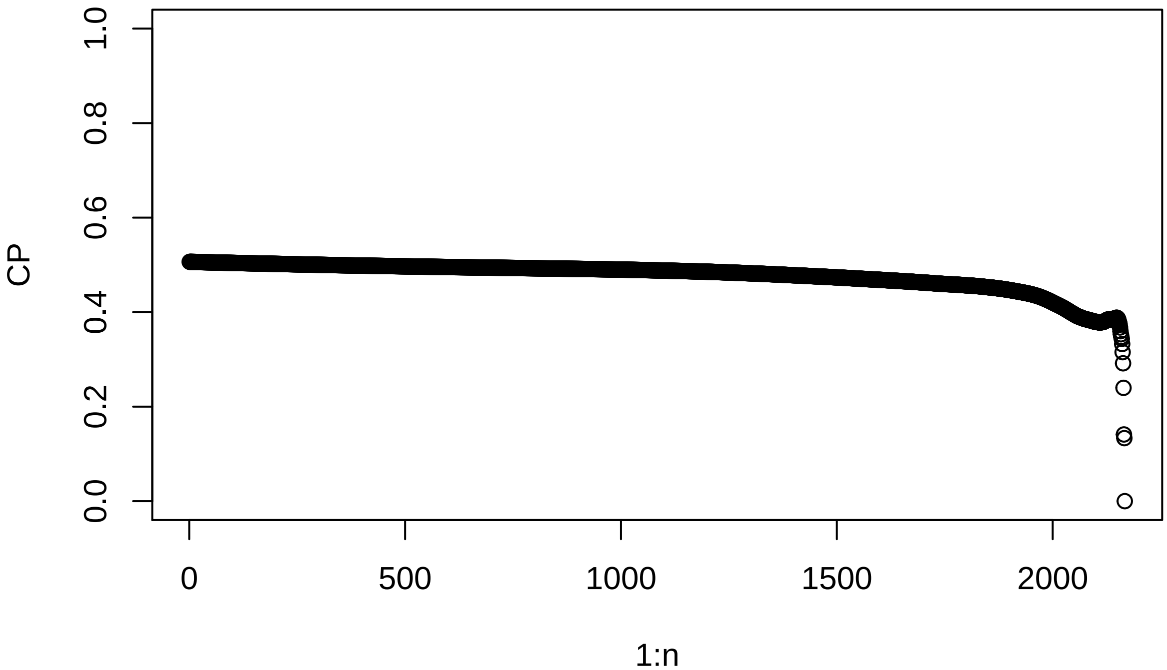
```
meplot(danish)
```



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.

```
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))
```



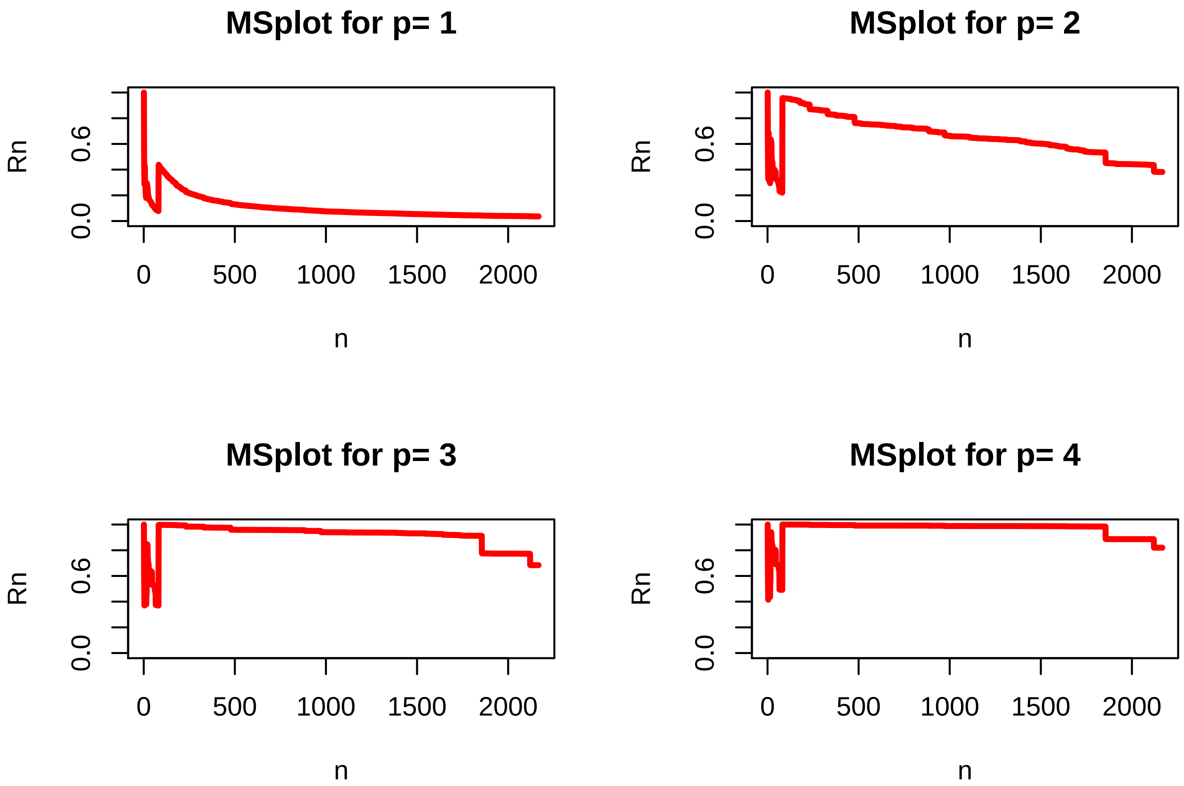
The nearly horizontal behavior we observe (the last part of the plot is to be ignored for the limited amount of points considered) can be seen as a further signal of Paretianity.

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.

```r
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 for inference. In estimating ξ, we would expect a value between 0.5 and 1.

```r
fit=gpd(danish,5)
fit
```

```
## $n
## [1] 2167
##
## $data
##    [1]   8.725274   7.898975   7.320644  11.374817  26.214641  14.122076
##    [7]   5.424253  11.713031  12.465593   5.875902   5.417277  17.569546
##   [13]   7.320644   7.320644  13.620791  21.961933   5.563852 263.250366
##   [19]   9.882870   6.319914   5.417277   7.613470   6.674817  19.070278
##   [25]   6.325037   5.402635   5.314788   5.856515  19.472914   6.915630
##   [31]   8.256881  34.141547   6.888453   9.174312  20.969856   5.242464
##   [37]   8.551769  12.895151   8.777628  56.225426   7.863696   7.558322
##   [43]   5.111402   5.242464  10.222805  14.678899   5.937759   7.109870
##   [49]   7.693316   8.453735   6.422018  50.065531   5.698583  10.178024
##   [55]  10.820452   5.469679  24.970273   5.231867  11.890606   5.695600
##   [61]   7.074911  20.049941   5.507729   5.001735  65.707491  27.262595
##   [67]  15.926278   5.469679  22.258226   5.588585   6.234705   5.561735
##   [73]  10.011123  10.072303   7.992070   5.925473  12.631813   6.916554
##   [79]   5.300504   5.672970  13.348165  11.431591  11.123471   9.314136
##   [85]   5.026178  11.623037  14.293194  13.623037   5.445026  18.646484
##   [91]  15.811518  19.162304  18.848168   5.305578   7.235602   7.643979
##   [97]   7.539267   5.026178  22.137567  16.300000   6.143355  46.500000
```

```
## [103]   6.200000   7.085000   6.306654   6.563000  10.500000   5.500000
## [109]   9.200000  12.225000  14.239000   5.200000  57.410636   5.850000
## [115]  14.300000   6.167000  13.500000   6.700000  10.700000  19.400000
## [121]   7.230000   8.710000   5.600000   5.207329   6.798457  12.054002
## [127]   5.207329  16.441659  29.026037  12.536162   5.785921   5.400193
## [133]   5.785921   6.075217  18.322083   5.785921   5.593057   8.678881
## [139]   8.100289  10.270010  17.068467  17.743491   5.496625  23.283859
## [145]   6.586271   7.142857  11.131725  12.523191   5.751391   5.565863
## [151]   5.751391   8.812616  32.467532  29.037106  18.552876  16.883117
## [157]   7.792208  27.829314   5.899814  12.059369   6.307978   9.461967
## [163]   7.606679  11.595547   5.194805   5.102041   7.101113  16.415262
## [169]   5.094055   6.015972  18.424135   7.985803   6.511091  38.154392
## [175]   5.210293   5.767524   6.832298   7.542147   5.681455   9.228039
## [181]  27.338066   7.542147   6.140195  11.801242  25.288376  10.204082
## [187]   5.989352  20.452529   5.323869   7.098492  47.019521  24.578527
## [193]  15.882875   7.542147   5.501331  25.953860   8.873114   6.211180
## [199]  10.825200   5.767524  31.055901   5.376799  24.555461  42.091448
## [205]  14.394581  20.863675   9.229467   5.927180   5.664691  14.394581
## [211]  10.137172   6.773920   5.715495   6.435224   5.503810   5.080440
## [217]   5.080440  12.801863  16.088061 152.413209   6.287045  14.013548
## [223]   8.367485   9.398815   5.770533   5.249788  12.701101  11.685013
## [229]   6.011854  32.387807  10.584251  18.628281   7.425743  10.998350
## [235]   8.250825  12.376238   5.198020   5.612211  15.284653  10.184818
## [241]   6.372937   8.085809   6.200495  13.201320  20.826733   5.940594
## [247]   8.250825  14.851485 144.657591  28.630363  19.265677   5.528053
## [253]   5.775578  17.739274
##
## $threshold
## [1] 5
##
## $p.less.thresh
## [1] 0.8827873
##
## $n.exceed
## [1] 254
##
## $method
## [1] "ml"
##
## $par.ests
##        xi      beta
## 0.6320499 3.8074817
##
## $par.ses
##        xi      beta
## 0.1117143 0.4637269
##
## $varcov
##             [,1]        [,2]
## [1,]  0.01248007 -0.03203283
## [2,] -0.03203283  0.21504268
##
## $information
## [1] "observed"
```

```
##
## $converged
## [1] 0
##
## $nllh.final
## [1] 754.1115
##
## attr(,"class")
## [1] "gpd"
```

```
# plot(fit) # Commented to produce the pdf.
```

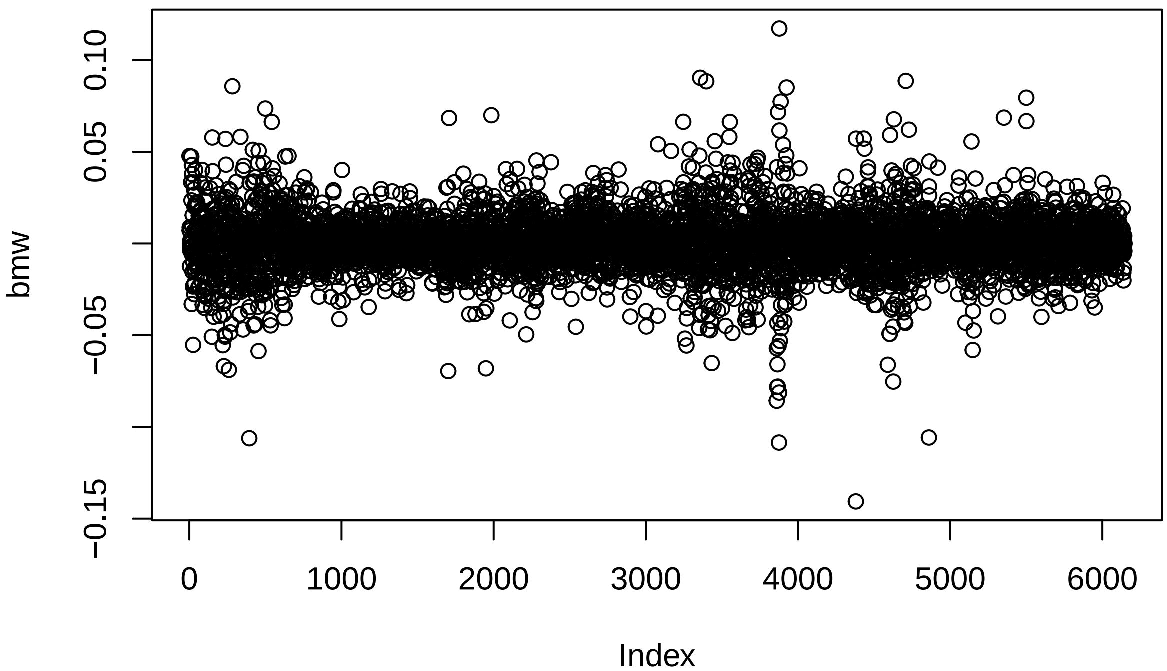
A sligthly more complicated case.

Let us now consider the BMW data set in the `evir` package.

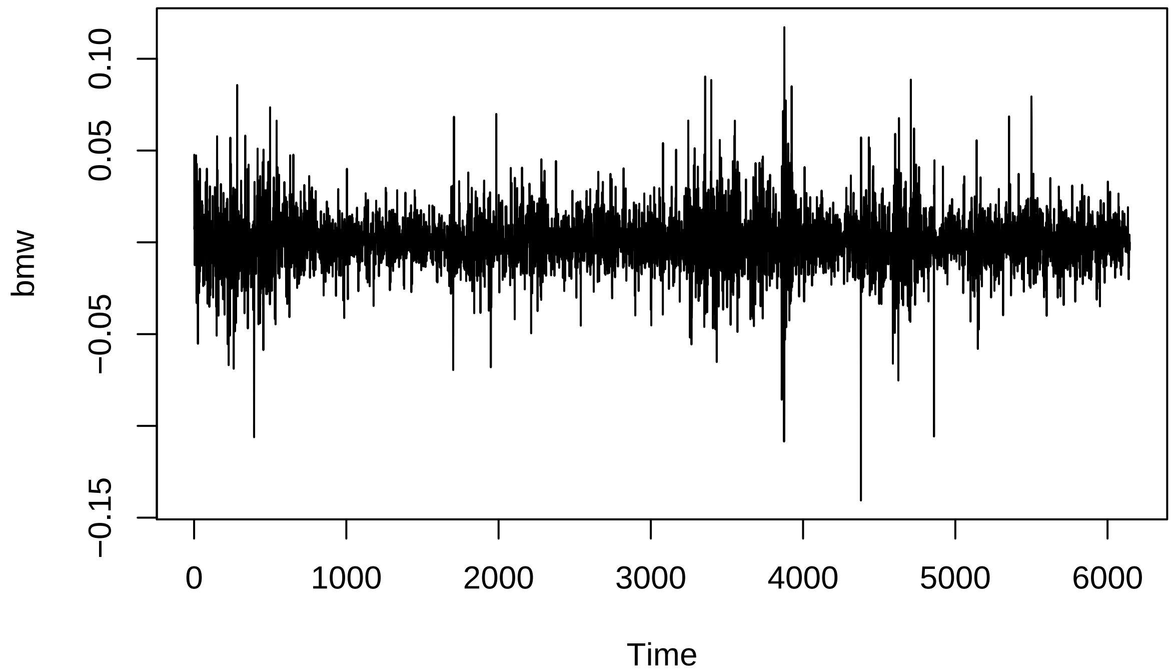
```
data(bmw)
? bmw
head(bmw)
```

```
## [1]  0.047704097  0.007127223  0.008883307 -0.012440569 -0.003569961
## [6]  0.000000000
```

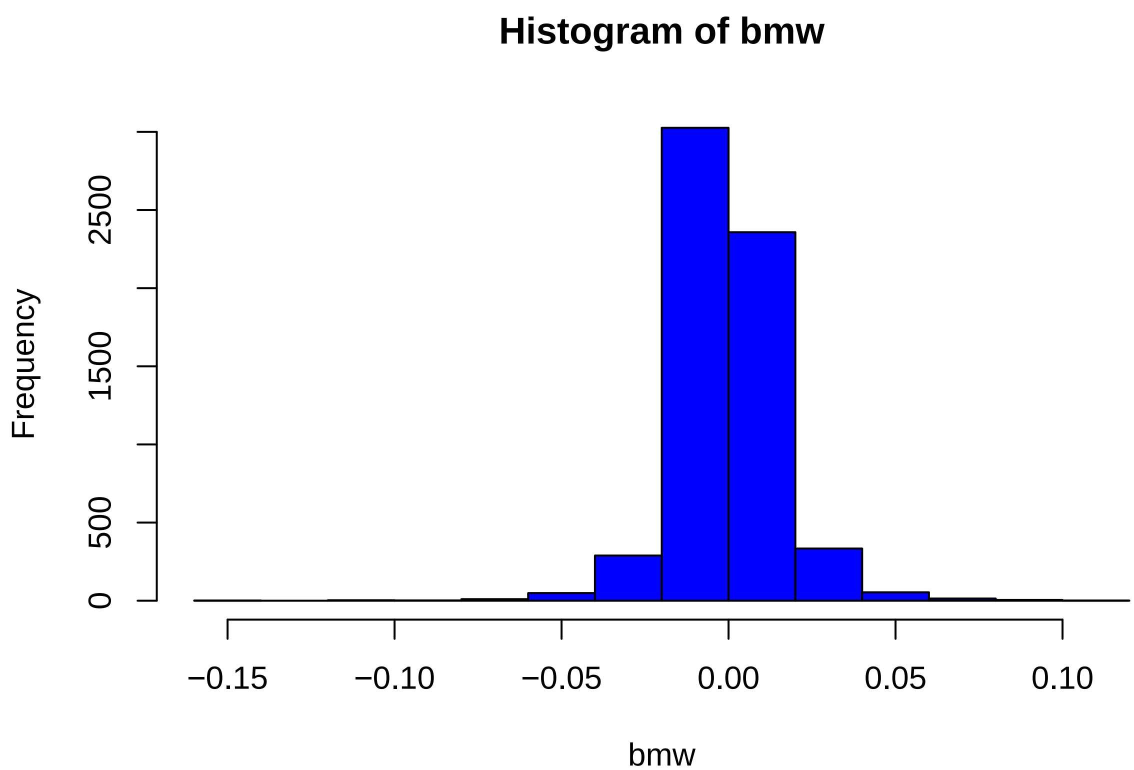
```
plot(bmw)
```




```
plot.ts(bmw)
```

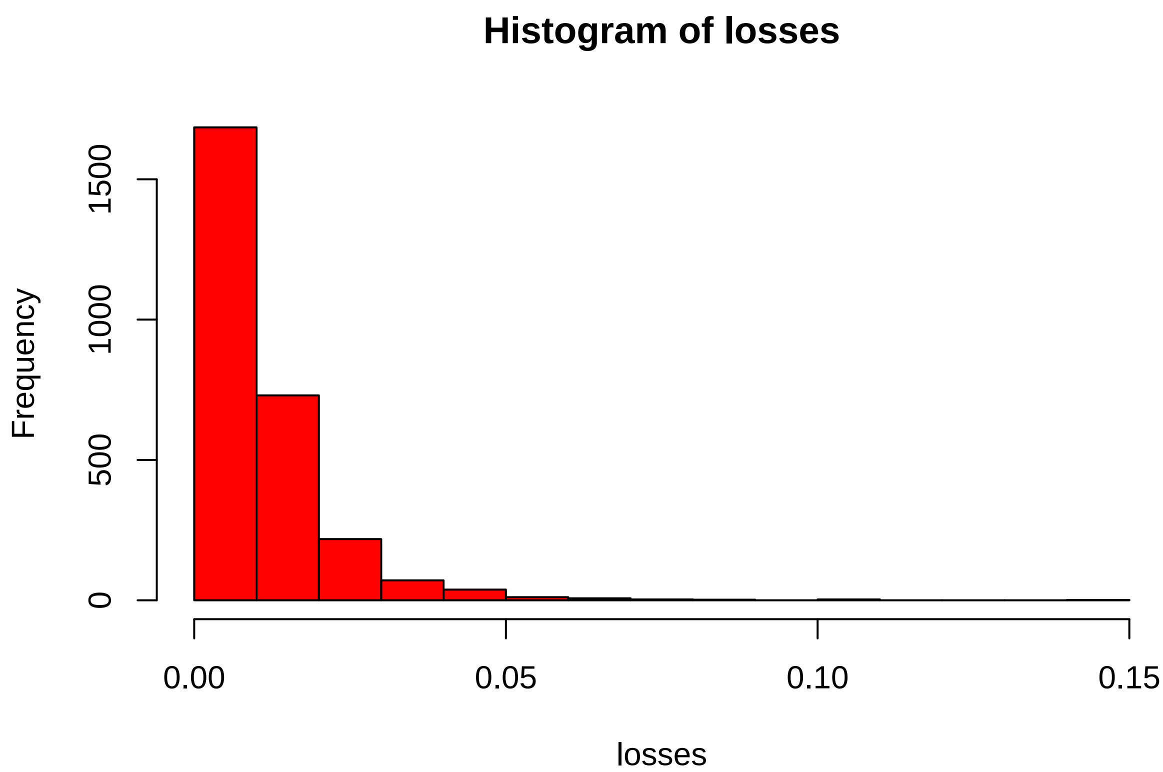



```
hist(bmw, col=4)
```

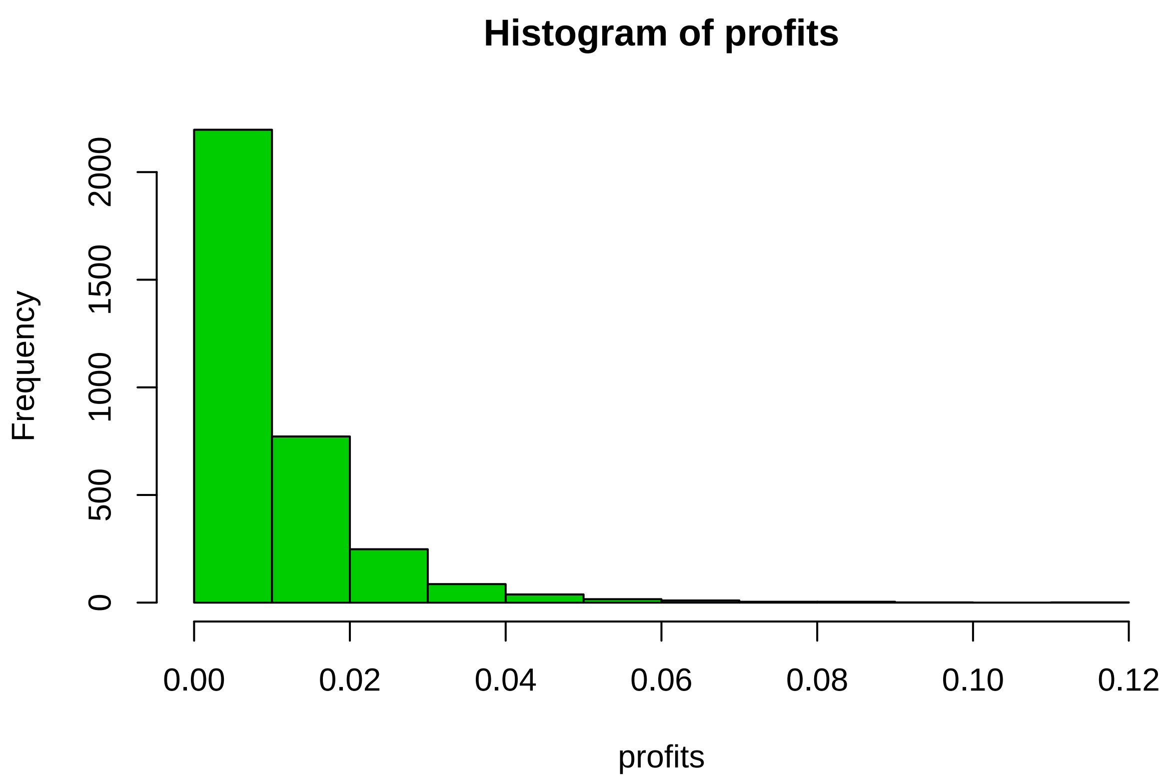



```
losses=-bmw[bmw<0]
profits=bmw[bmw>=0]
```

```
hist(losses,col=2)
```




```r
hist(profits,col=3)
```




```r
summary(losses); sd(losses)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0002287 0.0036486 0.0079052 0.0106849 0.0138228 0.1406157
```

```
## [1] 0.01079672
```

```r
summary(profits); sd(profits)
```

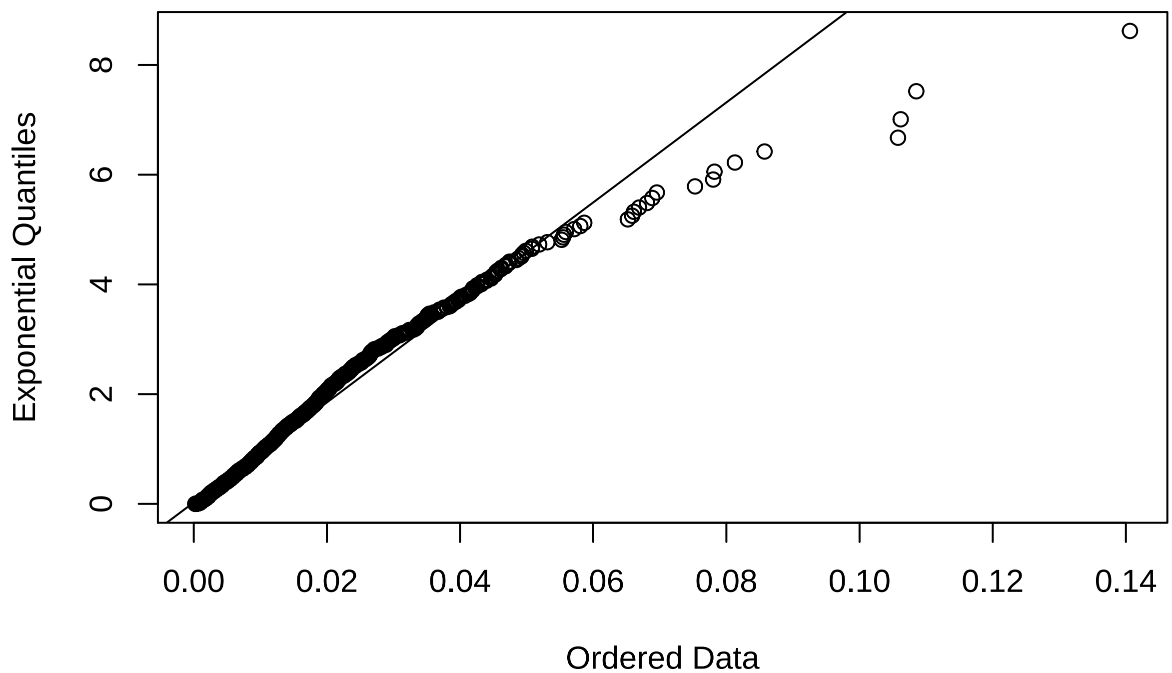
```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.000000 0.001909 0.006211 0.009381 0.013168 0.117192

## [1] 0.01092088
```

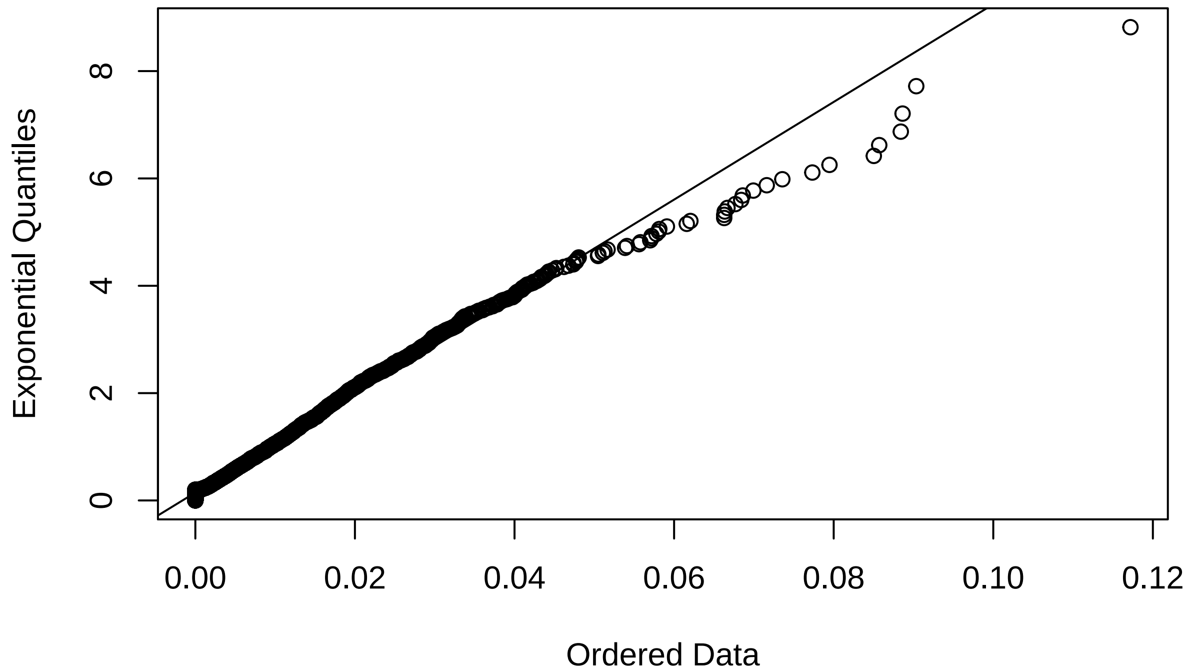
```r
summary(bmw)
```

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -0.1406157 -0.0066560  0.0000000  0.0003407  0.0071261  0.1171918
```

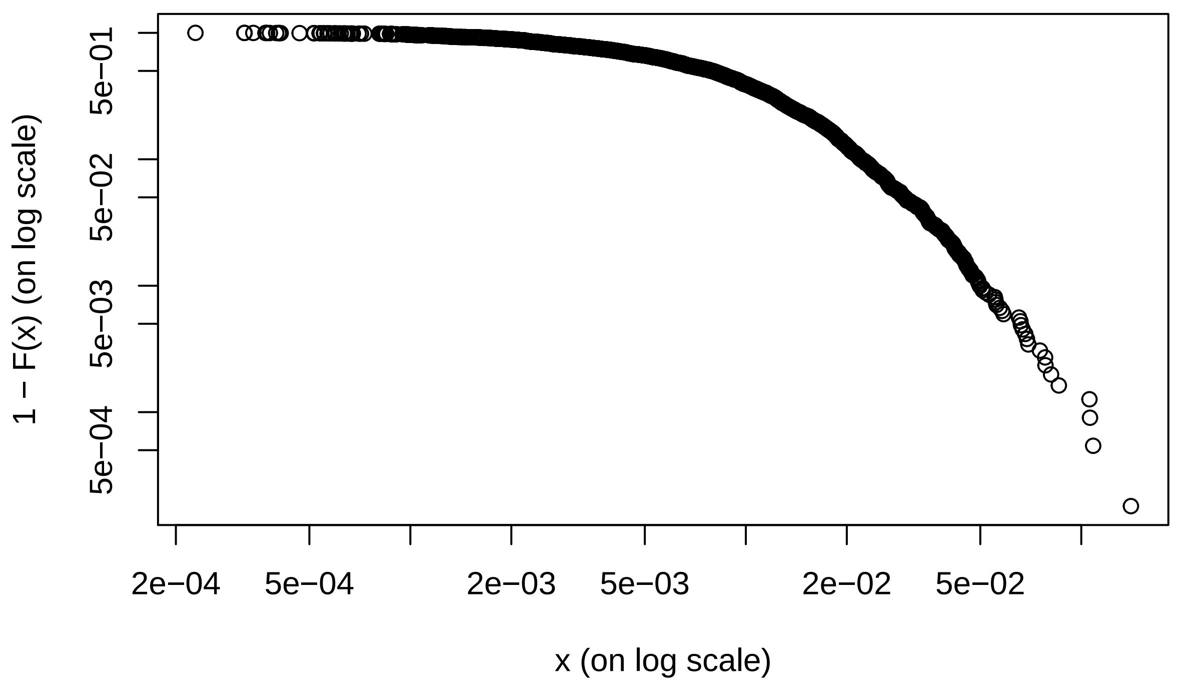
```r
qplot(losses,xi=0)
```



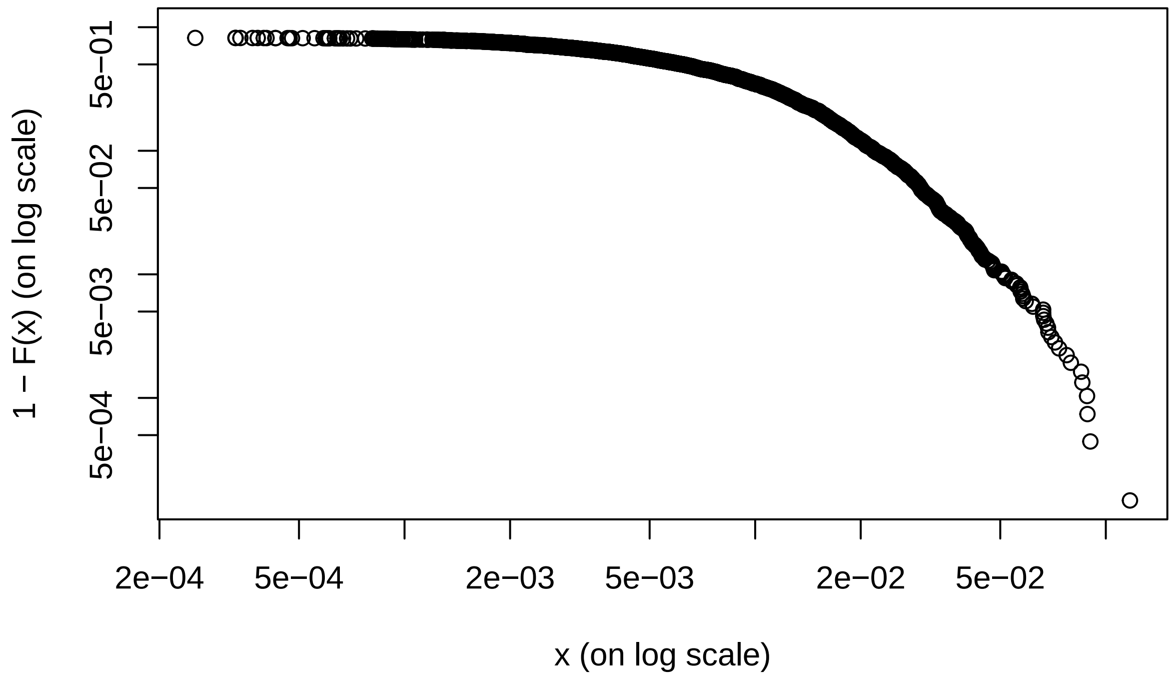

```r
qplot(profits,xi=0)
```




```r
emplot(losses,'xy')
```



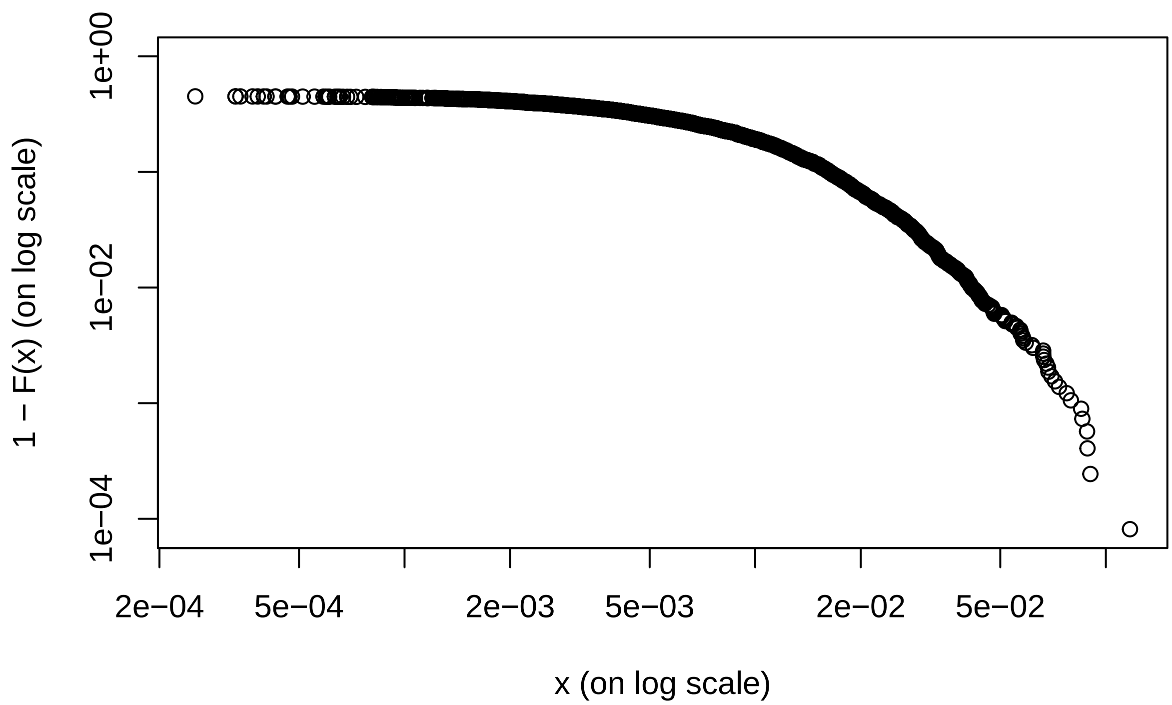

```r
emplot(profits,'xy')
```



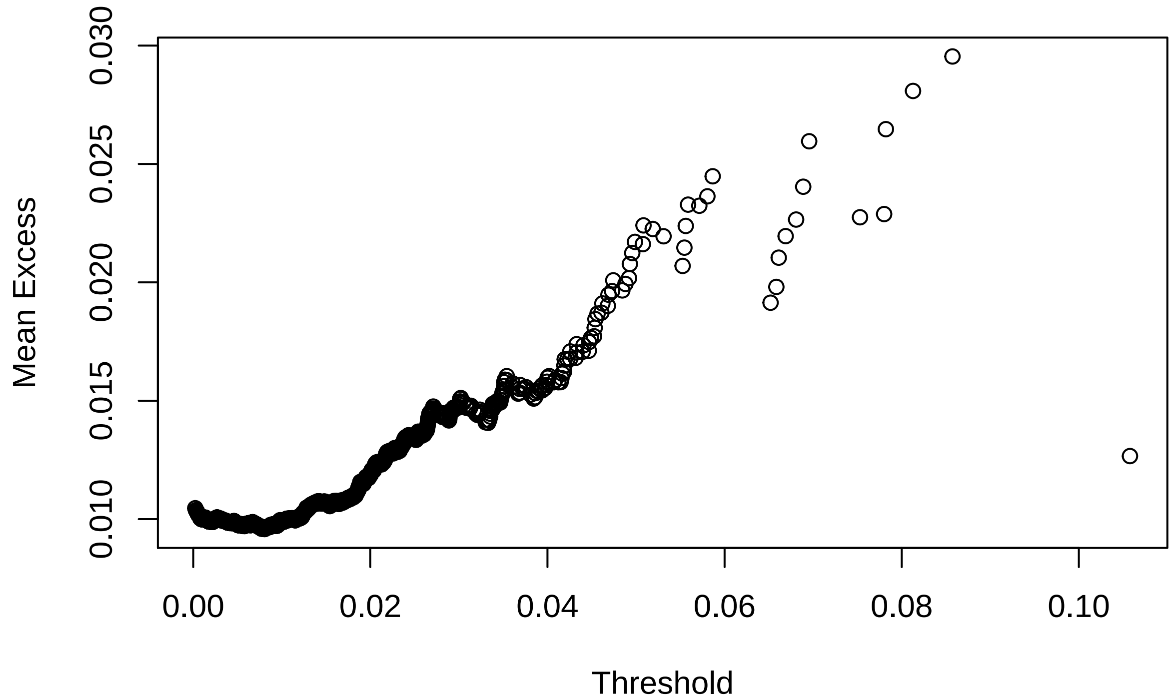
Please notice:

```r
emplot(bmw,'xy')
```

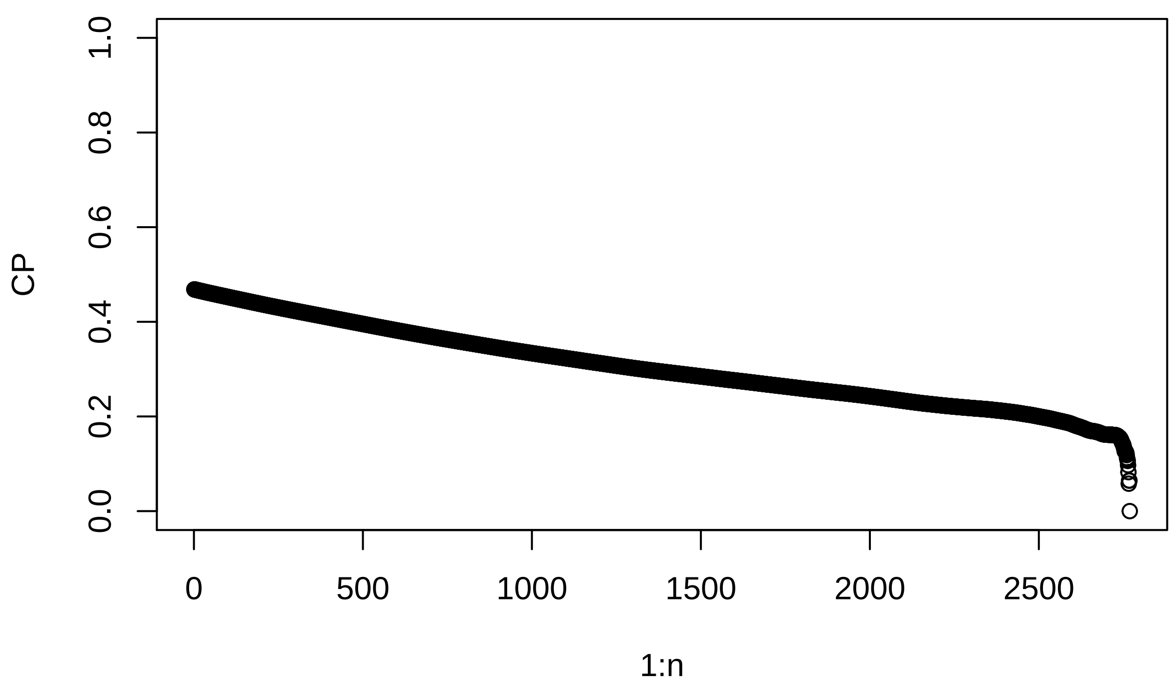
```
## Warning in xy.coords(x, y, xlabel, ylabel, log): 3380 x values <= 0 omitted from
## logarithmic plot
```




```r
meplot(losses)
```



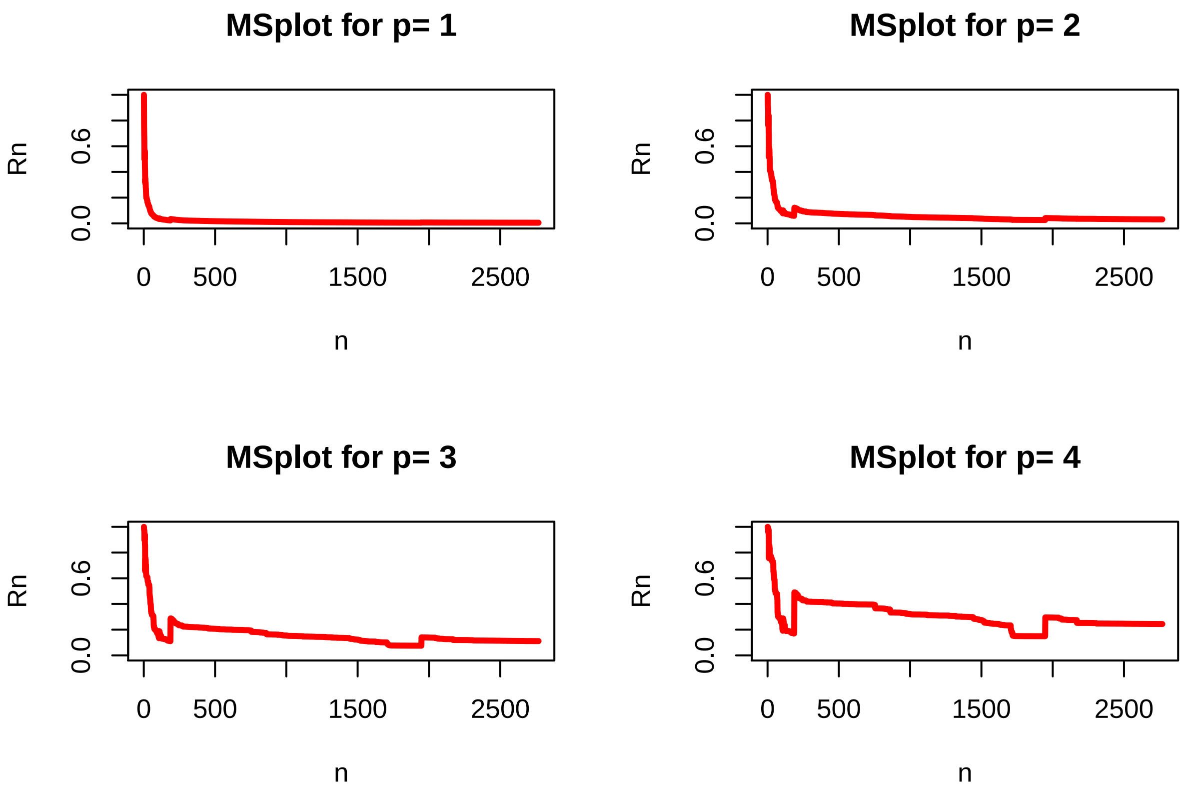

```
library(ineq)
sort_losses=sort(losses) # We sort the data
n=length(losses)
CP=c() #Empty array for storage
for (i in 1:n) {
  CP[i]=ineq(sort_losses[i:n],type="Gini") # Truncated Gini
}
plot(1:n,CP,ylim=c(0,1))
```



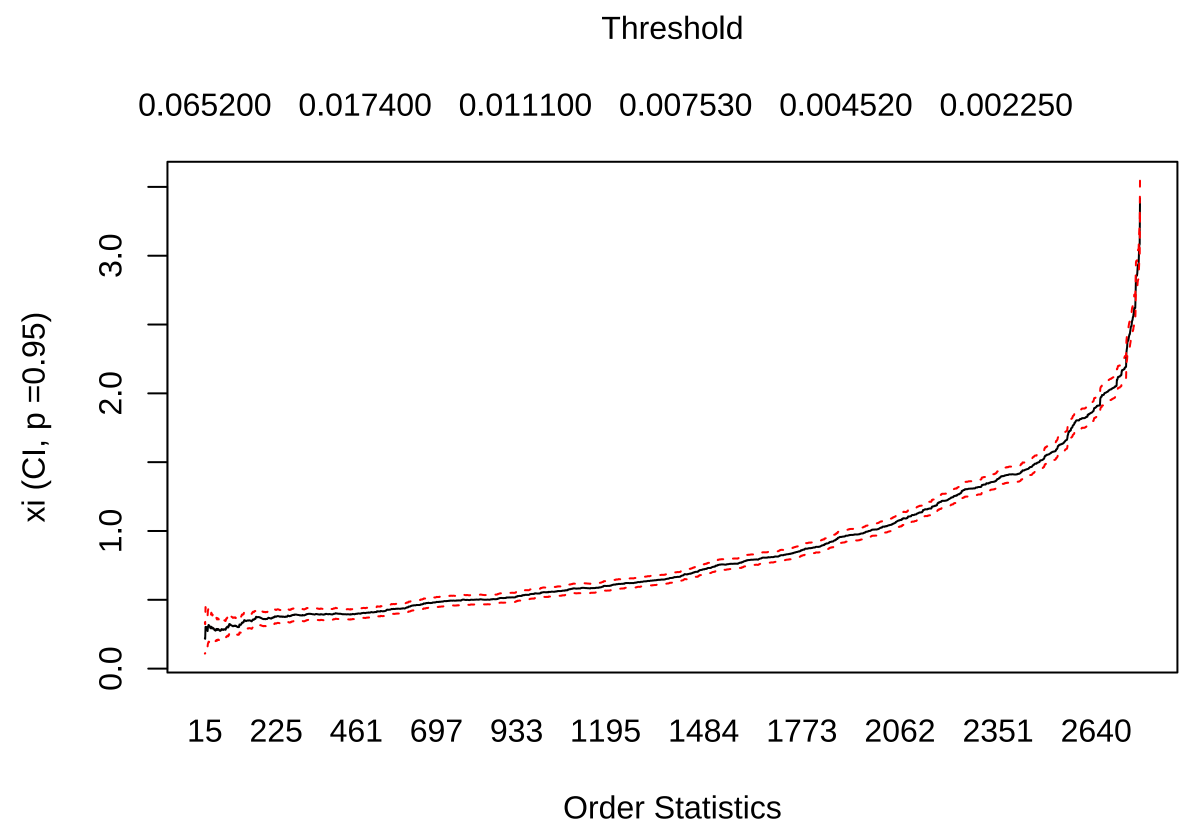
```r
MSplot <- function(data,p=4) {
  par(mfrow = c(2, 2))
  x=abs(data)
  for (i in 1:p) {
    y=x^i
    S=cumsum(y)
    M=cummax(y)
    R=M/S
    plot(1:length(x),R,type='l', col=2, lwd=3, ylim=c(0,1),xlab='n', ylab='Rn',
         main=paste("MSplot for p=",i))
  }
  par(mfrow = c(1, 1))
 # return(R)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

Let us run it.

```r
MSplot(losses)
```




```r
hill(losses,'xi')
```




```r
fit=gpd(losses,0.02)
tail(fit)
```

```
## $par.ests
##          xi        beta
## 0.223297186 0.009248302
##
## $par.ses
##           xi          beta
## 0.0683658225 0.0007703803
##
## $varcov
##                  [,1]          [,2]
## [1,]  4.673886e-03 -3.494755e-05
## [2,] -3.494755e-05  5.934858e-07
##
## $information
## [1] "observed"
##
## $converged
## [1] 0
##
## $nllh.final
## [1] -1224.767
```

```r
# plot(fit) # Commented to produce the pdf.
```

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

```
##          p  quantile     sfall
## [1,] 0.999 0.1009303 0.1361044
```

The empirical counterparts are

```r
quantile(losses,0.9999) #99% Var
```

```
##   99.99%
## 0.131732
```

```r
mean(losses[losses>=quantile(losses,0.9999)]) #99% ES
```

```
## [1] 0.1406157
```

```r
fitBM=gev(-bmw,"month")
fitBM
```

```
## $n.all
## [1] 6146
##
## $n
## [1] 283
##
## $data
##   [1] 0.033019677 0.055259919 0.066891428 0.068877999 0.023835128 0.023092449
##   [7] 0.035115366 0.030063440 0.050842876 0.039941694 0.031013959 0.039396412
##  [13] 0.048453302 0.029417583 0.033783061 0.026425841 0.026598358 0.038631413
##  [19] 0.046887076 0.036826025 0.106175195 0.044687230 0.058654777 0.028176031
##  [25] 0.044666259 0.019492352 0.008697785 0.019301478 0.013453385 0.029693058
##  [31] 0.040710083 0.013700930 0.019466501 0.024850203 0.017798788 0.018079701
##  [37] 0.018323847 0.008458086 0.030940396 0.007513321 0.016826245 0.028967661
##  [43] 0.016654017 0.018659487 0.029137226 0.018070536 0.018235302 0.041225459
##  [49] 0.007976837 0.026597797 0.013237166 0.026068195 0.015605237 0.012444757
##  [55] 0.021876256 0.023924786 0.034657755 0.018418544 0.011214472 0.009001829
##  [61] 0.015595478 0.013506861 0.010298892 0.011157337 0.012895238 0.025355326
##  [67] 0.008765121 0.027107808 0.010206402 0.011275617 0.015205631 0.013417535
##  [73] 0.012360765 0.021851512 0.021111211 0.020871901 0.018310419 0.012981633
##  [79] 0.016119731 0.028002605 0.069554216 0.017148877 0.020496807 0.015062588
##  [85] 0.038596396 0.020995751 0.016289593 0.023379800 0.038451263 0.024264302
##  [91] 0.012085575 0.068078687 0.013428524 0.009963349 0.027324896 0.020475480
##  [97] 0.041947815 0.013293615 0.018199016 0.017977661 0.017773094 0.025637988
## [103] 0.017038250 0.049572259 0.027796630 0.037448413 0.031355418 0.021699626
## [109] 0.018211113 0.010931429 0.016123773 0.017726532 0.013882750 0.026525588
## [115] 0.021081640 0.018482081 0.019868453 0.030210991 0.045415661 0.010078086
## [121] 0.026954120 0.006155256 0.020924273 0.012877299 0.021631830 0.009804200
## [127] 0.017655988 0.017050711 0.030511670 0.019076682 0.013192899 0.013831479
## [133] 0.014225826 0.039844882 0.014622816 0.025401283 0.026486105 0.017457957
## [139] 0.014043695 0.013402024 0.045258579 0.019724648 0.017905046 0.039381058
## [145] 0.023808056 0.021783929 0.046091107 0.038087270 0.032365285 0.016007375
## [151] 0.020097807 0.051884835 0.055613170 0.030047360 0.031916728 0.022432575
## [157] 0.046816163 0.065191831 0.017443027 0.020191780 0.034982211 0.036689697
## [163] 0.020569730 0.035401927 0.044916984 0.021291135 0.048802440 0.018587223
## [169] 0.041909867 0.045651541 0.108521596 0.042561770 0.021587455 0.041483519
## [175] 0.017679856 0.026193527 0.023709698 0.016034448 0.025052880 0.085731355
## [181] 0.033531013 0.017630058 0.016876907 0.014403847 0.029542861 0.017305069
```

```
## [187] 0.032251678 0.017839652 0.023564053 0.017936849 0.012596981 0.013623010
## [193] 0.023088421 0.018867592 0.024643272 0.017334021 0.015556229 0.007666020
## [199] 0.022727182 0.021268204 0.013623010 0.020601472 0.019833375 0.140615651
## [205] 0.028951321 0.028348474 0.034301830 0.034748401 0.020399753 0.033601200
## [211] 0.024278141 0.012070965 0.020200330 0.066112864 0.075291383 0.030853653
## [217] 0.043286834 0.034055559 0.014359090 0.022956749 0.022105005 0.018651503
## [223] 0.026770798 0.032213305 0.015825121 0.105774626 0.014809025 0.013005720
## [229] 0.009730427 0.009160196 0.024005906 0.014909519 0.010516251 0.014676834
## [235] 0.027642028 0.020805380 0.043177266 0.029605139 0.058061830 0.047431870
## [241] 0.030041206 0.026409171 0.026883515 0.016880066 0.020540366 0.022792643
## [247] 0.039735321 0.008322718 0.028928575 0.019620427 0.015899478 0.011814395
## [253] 0.024768601 0.022733053 0.034124046 0.015018239 0.022108402 0.014911906
## [259] 0.029923051 0.040018459 0.010374733 0.018794127 0.030126705 0.029413885
## [265] 0.018667209 0.012987196 0.013509372 0.022035888 0.032328887 0.018827493
## [271] 0.022735633 0.018158735 0.019656653 0.020389956 0.031233020 0.034902872
## [277] 0.006851475 0.013087635 0.019297206 0.014371505 0.017753757 0.005153175
## [283] 0.020144566
##
## $block
## [1] "month"
##
## $par.ests
##          xi       sigma          mu
## 0.232488476 0.008931895 0.018679433
##
## $par.ses
##           xi        sigma           mu
## 0.0484898747 0.0004400282 0.0005968189
##
## $varcov
##               [,1]         [,2]          [,3]
## [1,]  2.351268e-03 2.758886e-07 -8.164903e-06
## [2,]  2.758886e-07 1.936248e-07  1.527495e-07
## [3,] -8.164903e-06 1.527495e-07  3.561929e-07
##
## $converged
## [1] 0
##
## $nllh.final
## [1] -850.8664
##
## attr(,"class")
## [1] "gev"
```

```
# plot(fitBM) # Commented to produce the pdf.
```

Another step:

Let's start by uploading the required packages.

```
library(quantmod)
```

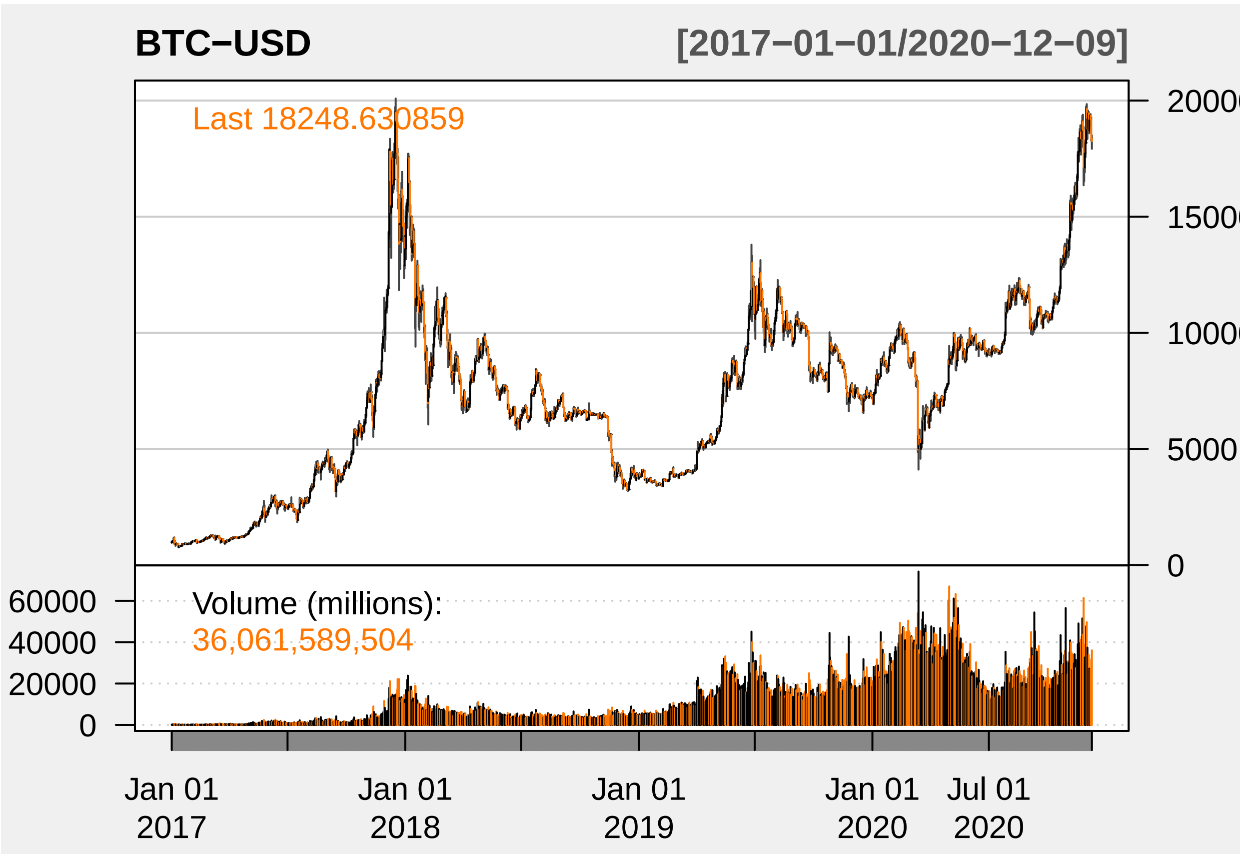
Data exploration

We will study the tails of Bitcoin.

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

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

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



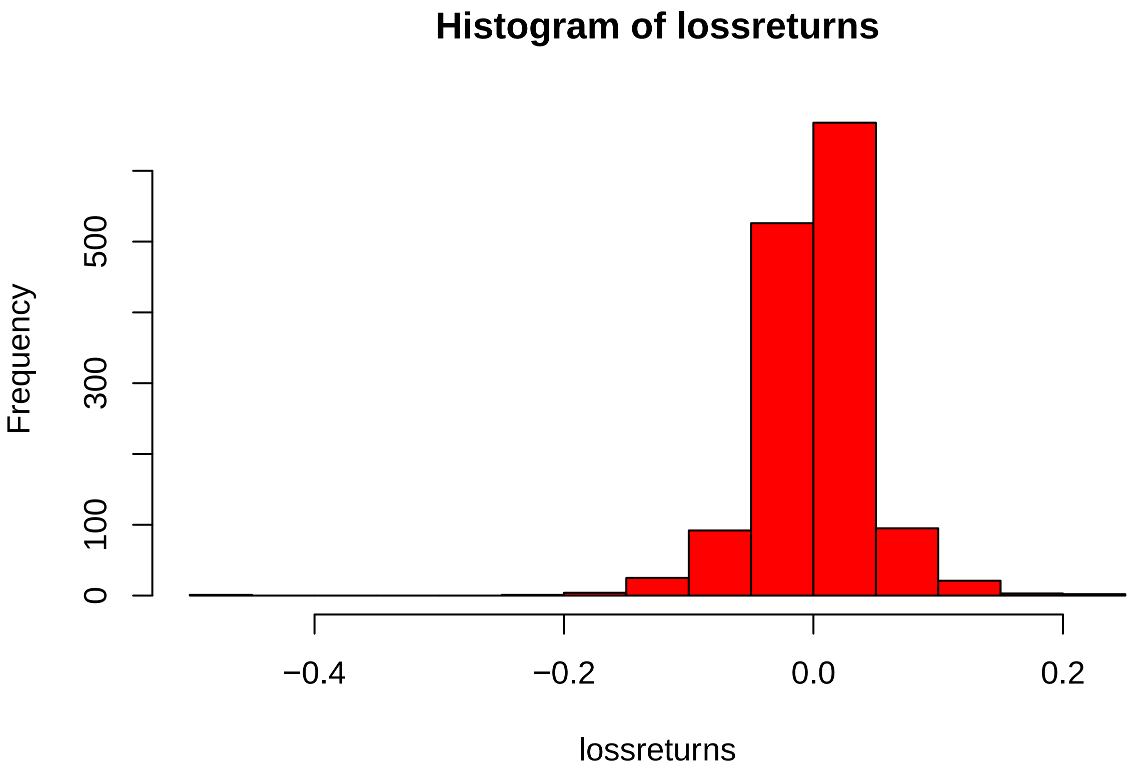
Let's consider losses and returns.

```
bitcoin=Ad(`BTC-USD`)
lossreturns=diff(log(bitcoin))
losses=-lossreturns[lossreturns<0]
returns=lossreturns[lossreturns>0]
summary(lossreturns)
```

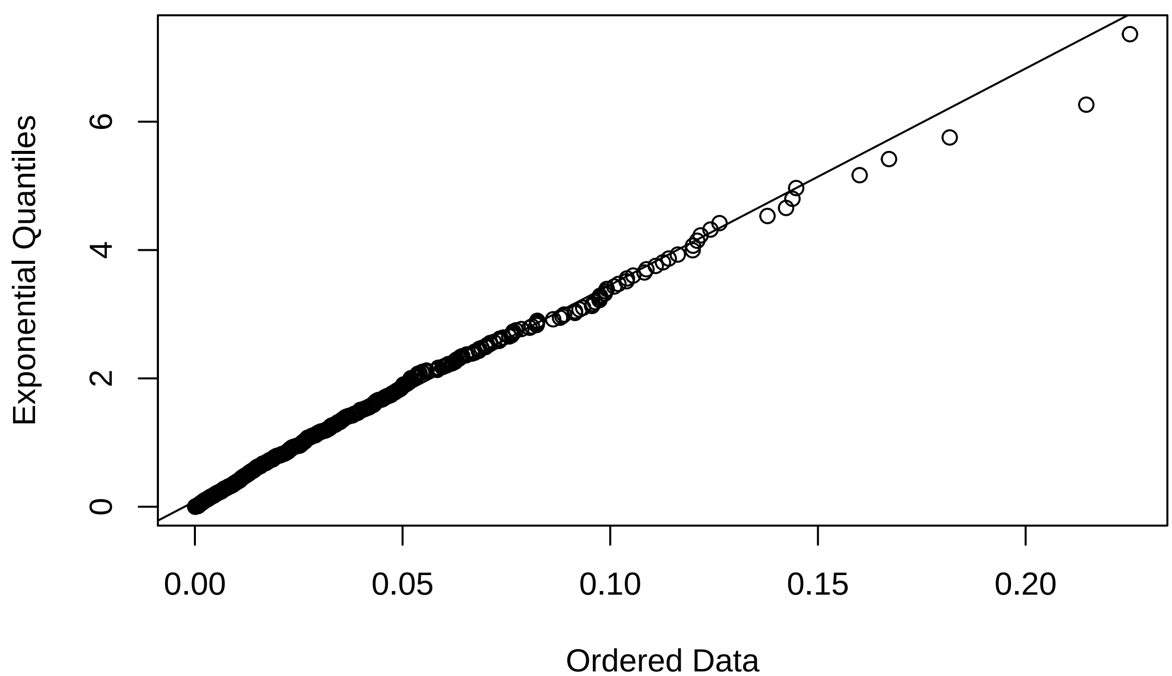
```
##      Index               BTC-USD.Adjusted
##  Min.   :2017-01-01   Min.   :-0.464730
##  1st Qu.:2017-12-26   1st Qu.:-0.014151
##  Median :2018-12-21   Median : 0.002251
##  Mean   :2018-12-21   Mean   : 0.002021
##  3rd Qu.:2019-12-15   3rd Qu.: 0.019602
##  Max.   :2020-12-09   Max.   : 0.225119
##                       NA's   :1
```

Some first necessary plots.

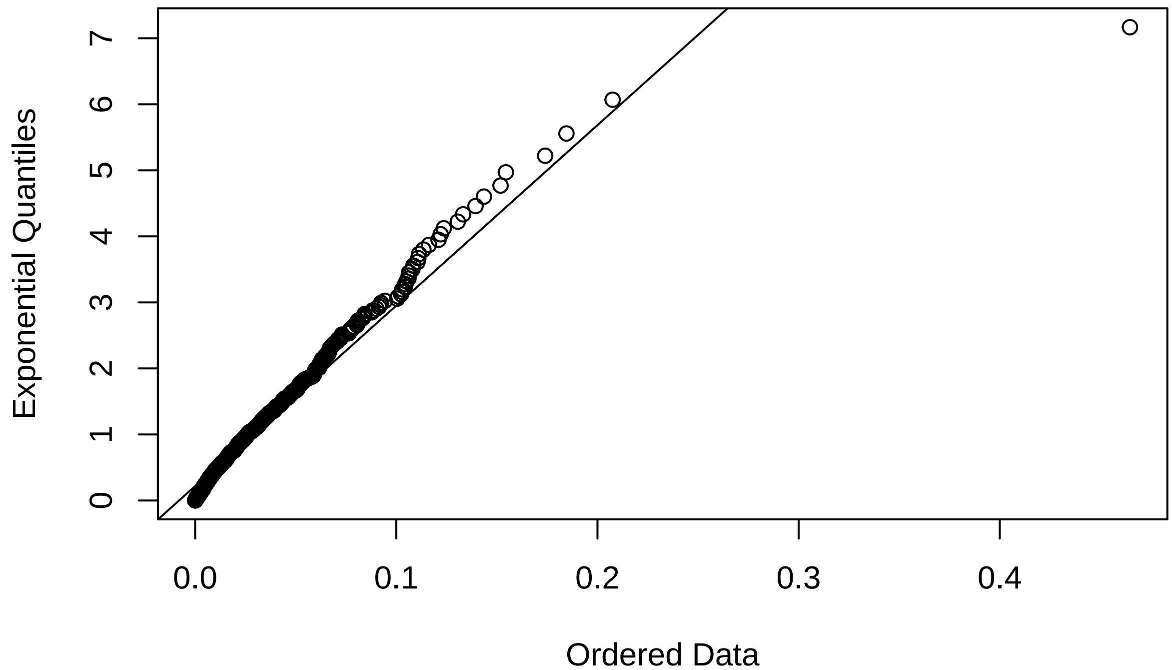
```r
hist(lossreturns,breaks=20,col=2)
```




```r
qplot(returns)
```

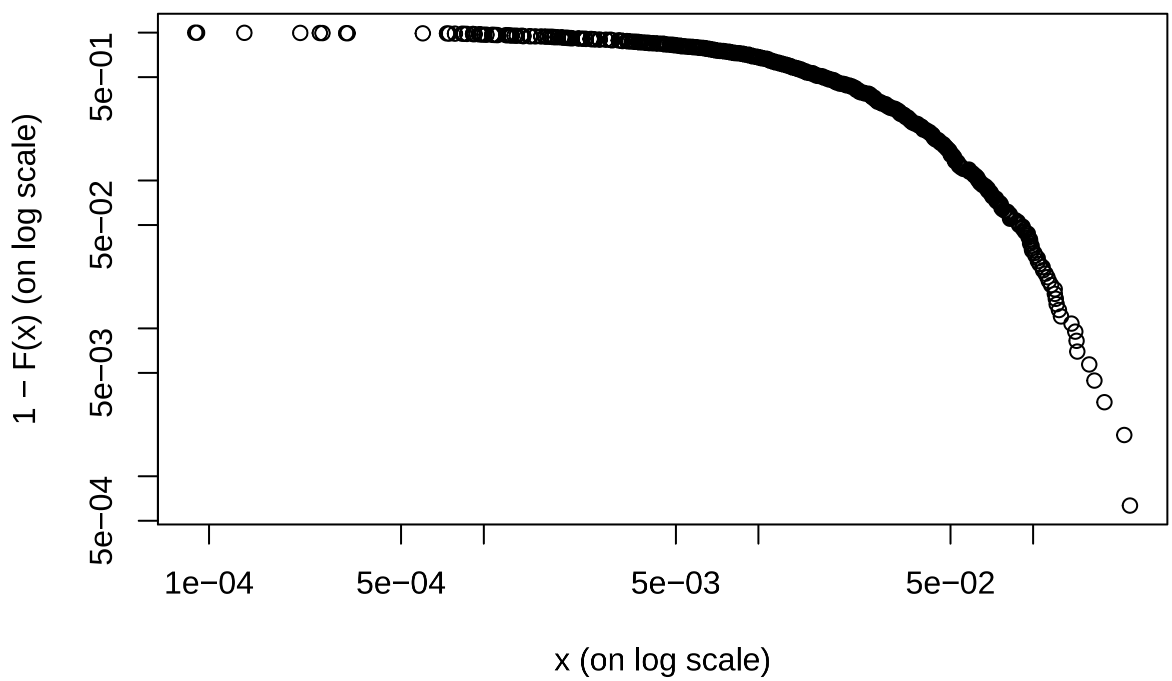



```r
qplot(losses)
```

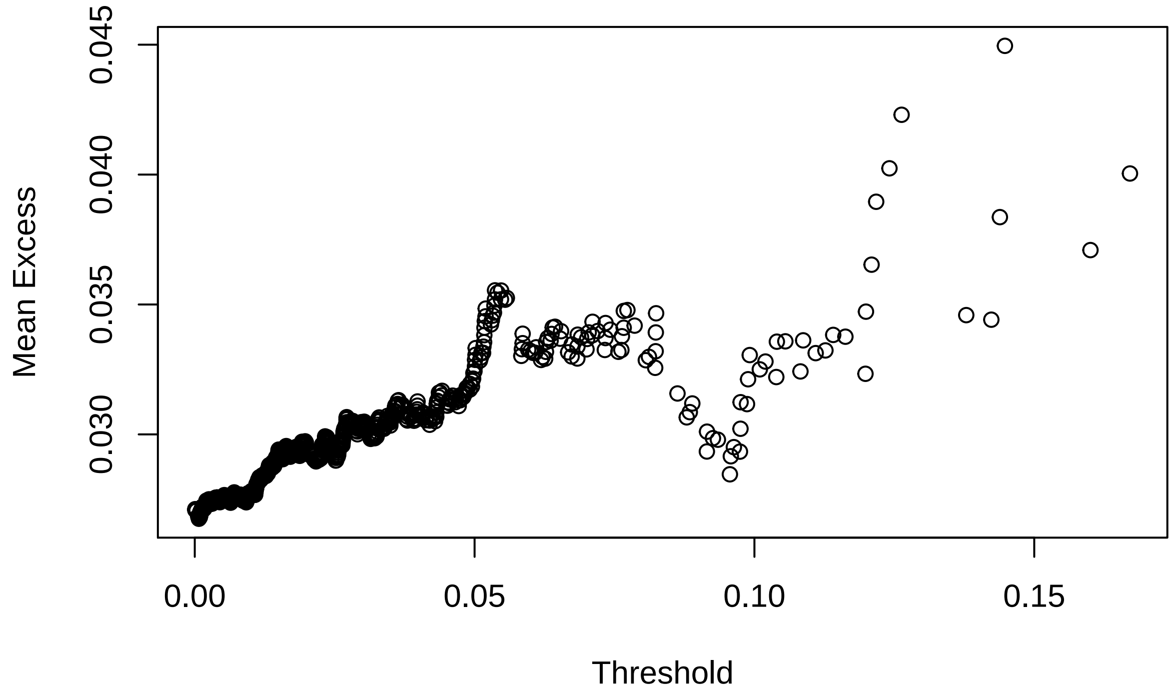


Zipf Plot and Meplot

```
emplot(returns,'xy')
```




```
meplot(returns)
```



Continue the analysis for both returns and losses. What do you find? You can share your results/ideas on the course forum. We will comment upon them in the next live class as well. ##################################