Extreme Value Theory - Practical session 1 : Heavy-tailed models

José G. Gómez-García

2023-09-24

Table des matières

1 Description of the practical session

Supporting data for this practical session are:

- Rainfall data from Orlando, Florida, USA, see Florida_rainfall.RData,
- Financial data on the CAC40 index, see CAC40.csv,
- The Society of Actuaries (SOA) Group Medical Insurance Large Claims Database from the R package `ReIns`,
- 2010 US Census data ranking US cities by population, see census_USA_2010.xlsx.

You may not have the time to work on all these real data examples. The idea is to allow you to work on a data set that suits your interests. 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 $1 \leq k \leq$, implement the Hill and Weissman estimators. The result should be a pair of vectors of size $n-1$. As one of the samples of data is fairly large, it will be useful to speed the code up as much as you can.
2. Implement diagnostic tools (QQ-plots…) to judge whether the heavy-tailed assumption makes sense.
3. Download a few packages. Good choices in this practical session are `evir` and `evt0`. These will help to see if your results make sense (and you might find that one of the packages does something wrong!)

```
library(ggplot2)
```

2 Cases studies

2.1 Rainfall data

The variable of interest is `sum_rain_2m_inches`, representing daily total rainfall at a weather station near Orlando, Florida, USA, during the American storm season, which lasts from August to October. Accompanying this variable is the day of recording, from 1st August 1998 to 31st October 2020.

```
load("~/Documents/GIT/teaching/extreme value theory/data sets for practical sessions/Florida_rainfall.RD
rainfall.data <- dataset[,-1]
```

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?

```
#plot.ts(rainfall.data[,2])
```

```
#hist(rainfall.data$sum_rain_2m_inches)
```

```
#rainfall.data.plus <- rainfall.data$sum_rain_2m_inches[rainfall.data$sum_rain_2m_inches>0]
#hist(rainfall.data.plus, breaks = 50, probability = TRUE)
```

2. Can you find a reasonable statistical model for the whole of the data?
3. Check that there is evidence of a heavy right tail in this data set.
4. Calculate an estimate for the extreme value index of the data. Justify your choice of tuning parameters.
5. Calculate an estimate for the extreme quantiles at level 0.99, 0.995 and 0.999. Do these make sense?
6. 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?

2.2 CAC40 data

Financial data about stock market indices and equity prices are typically not stationary. When using this data, isolate first the `Close` variable, representing daily closing prices. Then, construct the daily log-return variable X_t defined as follows: if S_t is the closing price on day t, the daily log-return on day t is $\log(S_t/S_{t-1})$. This variable X_t will be the variable of interest, rather than the nonstationary price S_t. Accompanying this variable is the date of recording, from 1st March 1990 to 8th July 2021.

```
cac40.data <- read.csv("~/Documents/GIT/teaching/extreme value theory/data sets for practical sessions/C
cac40.data$Close <- as.numeric(cac40.data$Close)
log.close <- data.frame('Date'=cac40.data$Date, 'lclose'=log(cac40.data$Close))
log.close <-na.omit(log.close)  #na.omit to get rid of NAs
```

It's also possible to remplace the NAs with the media in a window around the missing value if you to get fancy.

```
#some checks
any(log.close$lclose ==0)
```

```
## [1] FALSE
```

```
all(log.close$lclose > 0)
```

```
## [1] TRUE
```

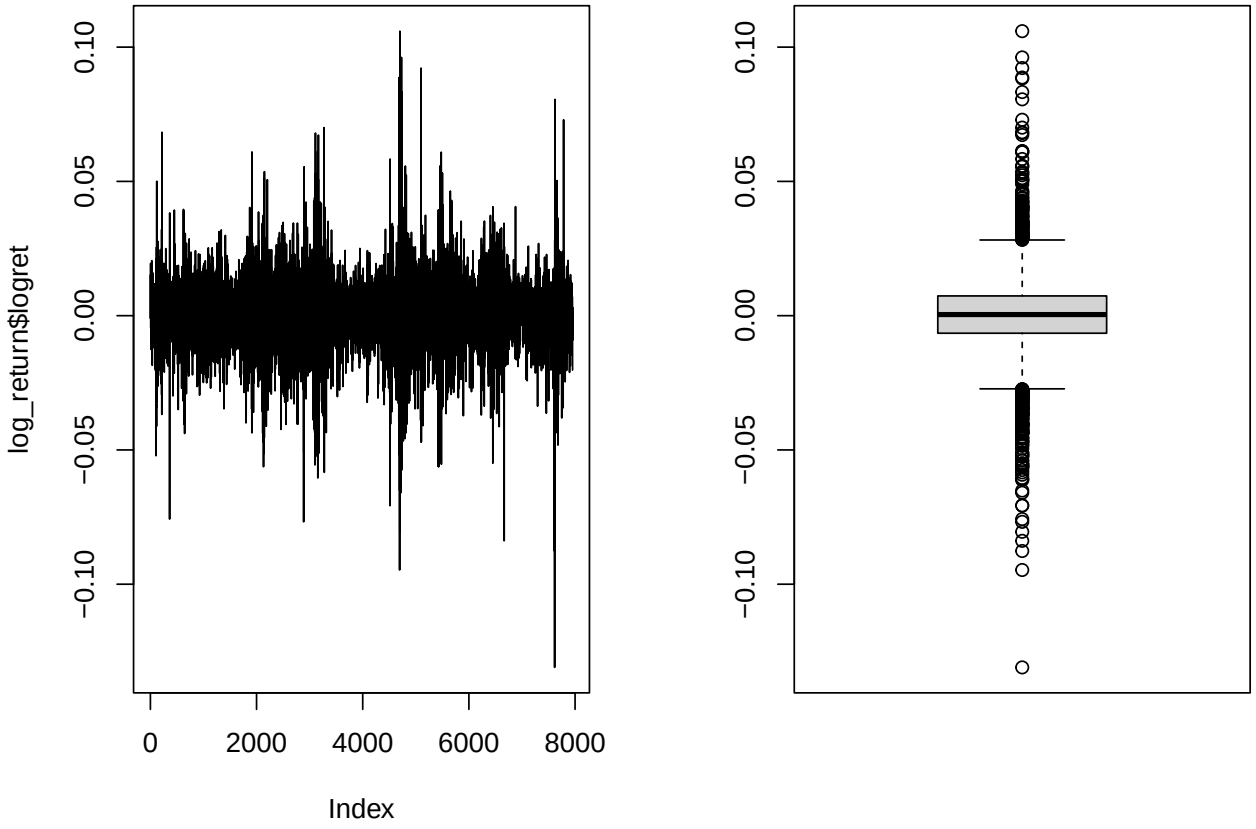
log returns can be computed by differencing the logarithms of the prices

```
log_return <- data.frame('Date'=log.close[-1,1], 'logret'=diff(log.close[,2]))
# or you can just use
log_return2 <- data.frame('Date'=log.close$Date[-1], 'logret'= log.close[2:nrow(log.close),2]-log.close
```

2.2.1 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?

- Let's look at some graphs:

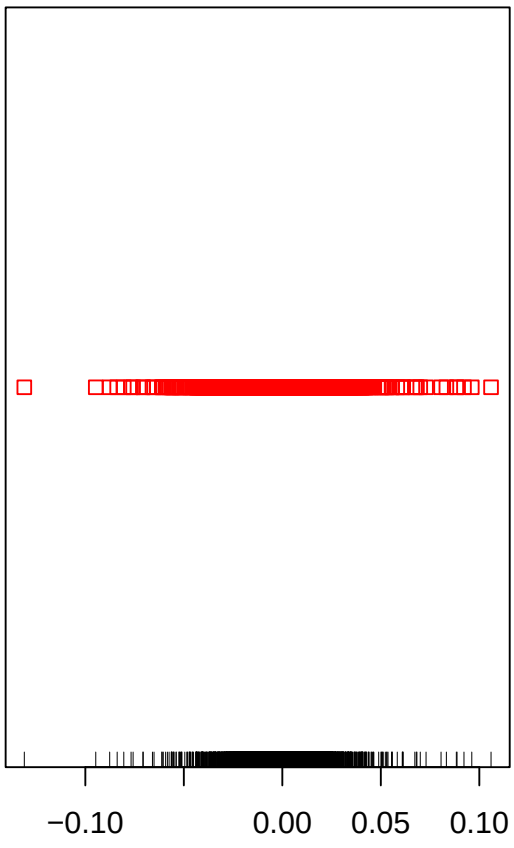
```
par(mfrow = c(1,2))
plot(log_return$logret, type = 'l')
boxplot(log_return$logret)
```

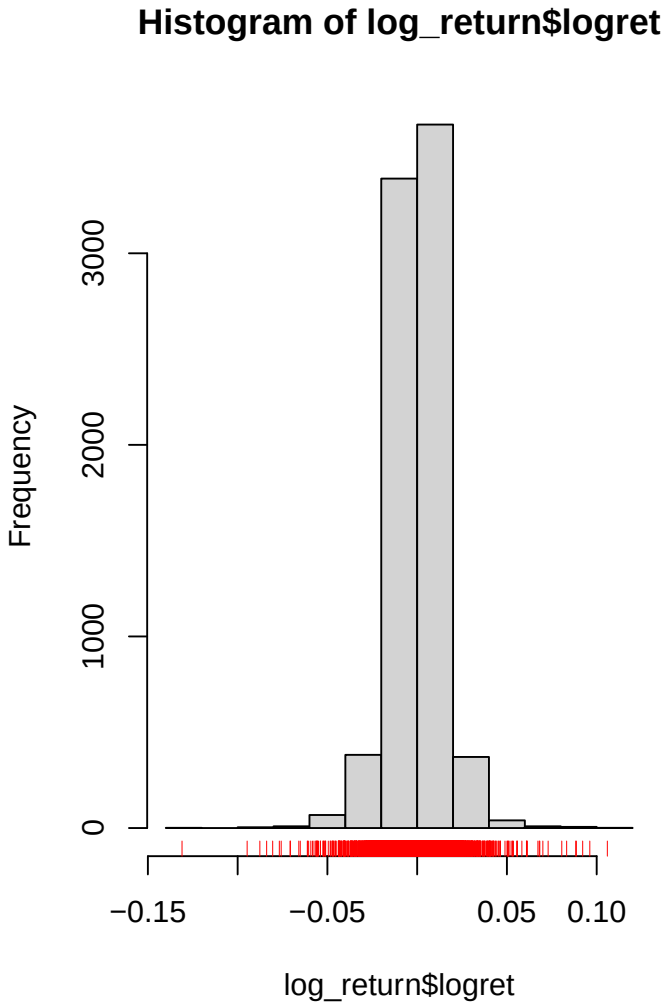


The boxplot shows very heavy tails (first sight).

```
par(mfrow=c(1,2))
stripchart(log_return$logret, col='red') #don't bother
rug(log_return$logret)

hist(log_return$logret)
rug(log_return$logret, col='red')
```

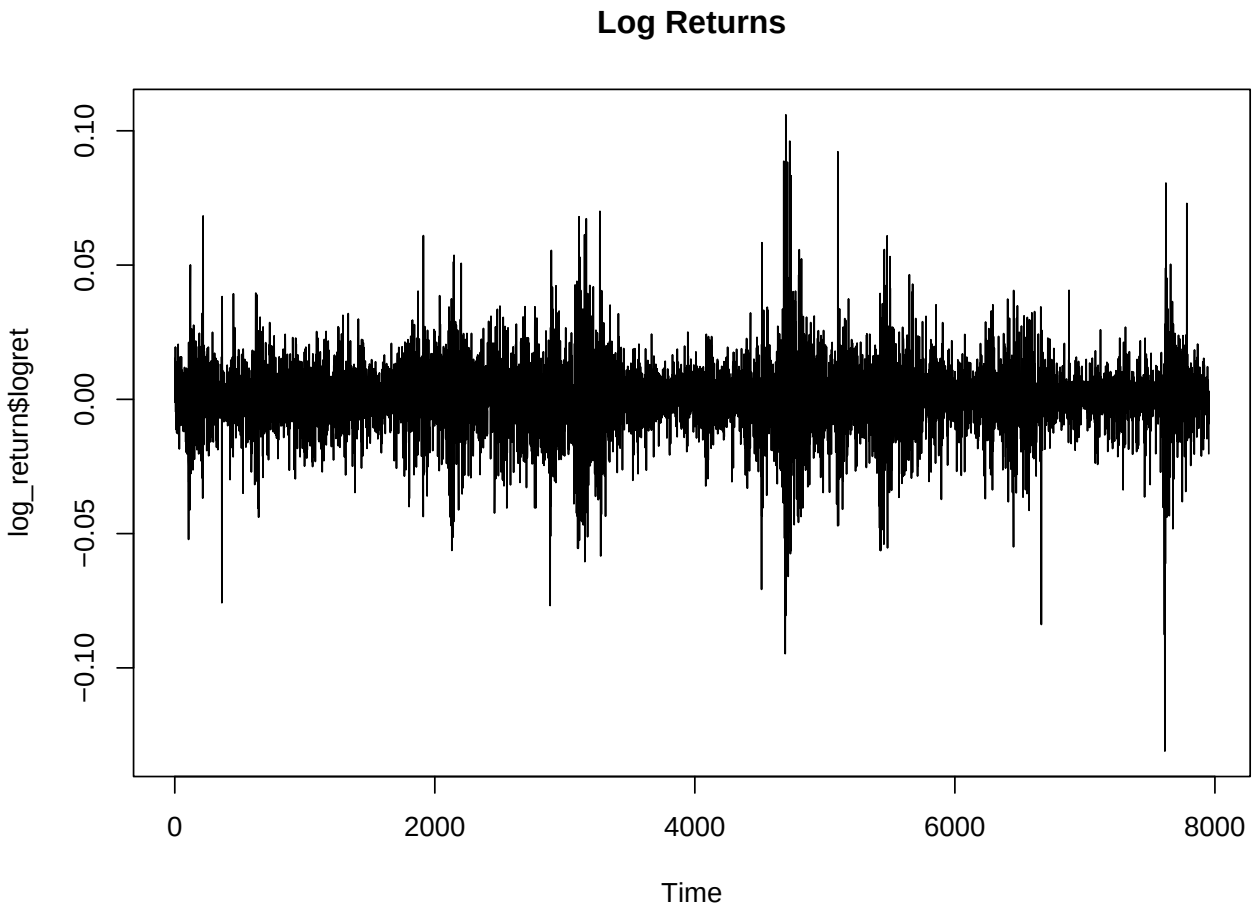




we once again see a rather heavy tail.

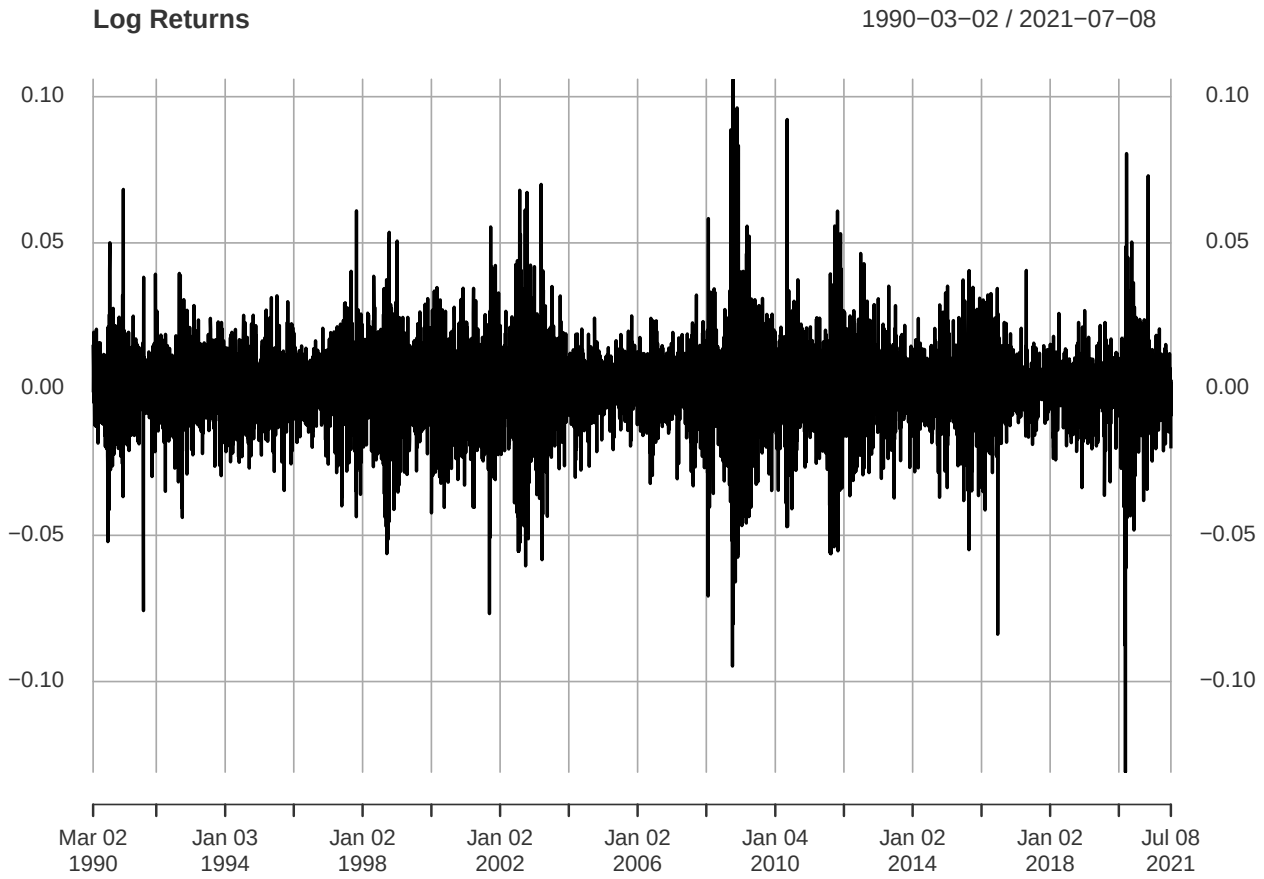
We now show the time series:

```
plot.ts(log_return$logret, main="Log Returns")
```



- We can use `xts` (eXtensible Time Series) package to maximize the preservation of information in native format.

```
library(xts)
log_return.ts <- as.xts(log_return$logret, as.Date(log_return$Date))
plot(log_return.ts, main="Log Returns")
```



- Stationary test

We are dealing with log returns, so stationarity is not an issue. However, we cans till test for it using the ADF and/or KPSS tests.

```
library(aTSA)
stationary.test(log_return$logret) # same as adf.test
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##       lag   ADF p.value
##  [1,]   0 -89.7    0.01
##  [2,]   1 -64.3    0.01
##  [3,]   2 -54.2    0.01
##  [4,]   3 -45.4    0.01
##  [5,]   4 -42.3    0.01
##  [6,]   5 -39.1    0.01
##  [7,]   6 -35.3    0.01
##  [8,]   7 -32.9    0.01
##  [9,]   8 -31.3    0.01
## [10,]   9 -29.7    0.01
## [11,]  10 -28.1    0.01
## Type 2: with drift no trend
##       lag   ADF p.value
##  [1,]   0 -89.7    0.01
```

```
##  [2,]   1 -64.3    0.01
##  [3,]   2 -54.2    0.01
##  [4,]   3 -45.4    0.01
##  [5,]   4 -42.3    0.01
##  [6,]   5 -39.1    0.01
##  [7,]   6 -35.3    0.01
##  [8,]   7 -32.9    0.01
##  [9,]   8 -31.3    0.01
## [10,]   9 -29.7    0.01
## [11,]  10 -28.1    0.01
## Type 3: with drift and trend
##       lag   ADF p.value
##  [1,]   0 -89.7    0.01
##  [2,]   1 -64.3    0.01
##  [3,]   2 -54.2    0.01
##  [4,]   3 -45.4    0.01
##  [5,]   4 -42.3    0.01
##  [6,]   5 -39.1    0.01
##  [7,]   6 -35.3    0.01
##  [8,]   7 -32.9    0.01
##  [9,]   8 -31.3    0.01
## [10,]   9 -29.7    0.01
## [11,]  10 -28.1    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```
library(urca)
summary(ur.df(log_return$logret, lags=trunc(sqrt(length(log_return$logret))), selectlags = "AIC"))
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.134613 -0.006484  0.000506  0.007406  0.098843
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## z.lag.1     -1.11500    0.02865 -38.916  < 2e-16 ***
## z.diff.lag1  0.10798    0.02590   4.169 3.10e-05 ***
## z.diff.lag2  0.08989    0.02308   3.895 9.91e-05 ***
## z.diff.lag3  0.04942    0.01973   2.505   0.0123 *
## z.diff.lag4  0.06775    0.01599   4.237 2.29e-05 ***
## z.diff.lag5  0.02338    0.01128   2.073   0.0382 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

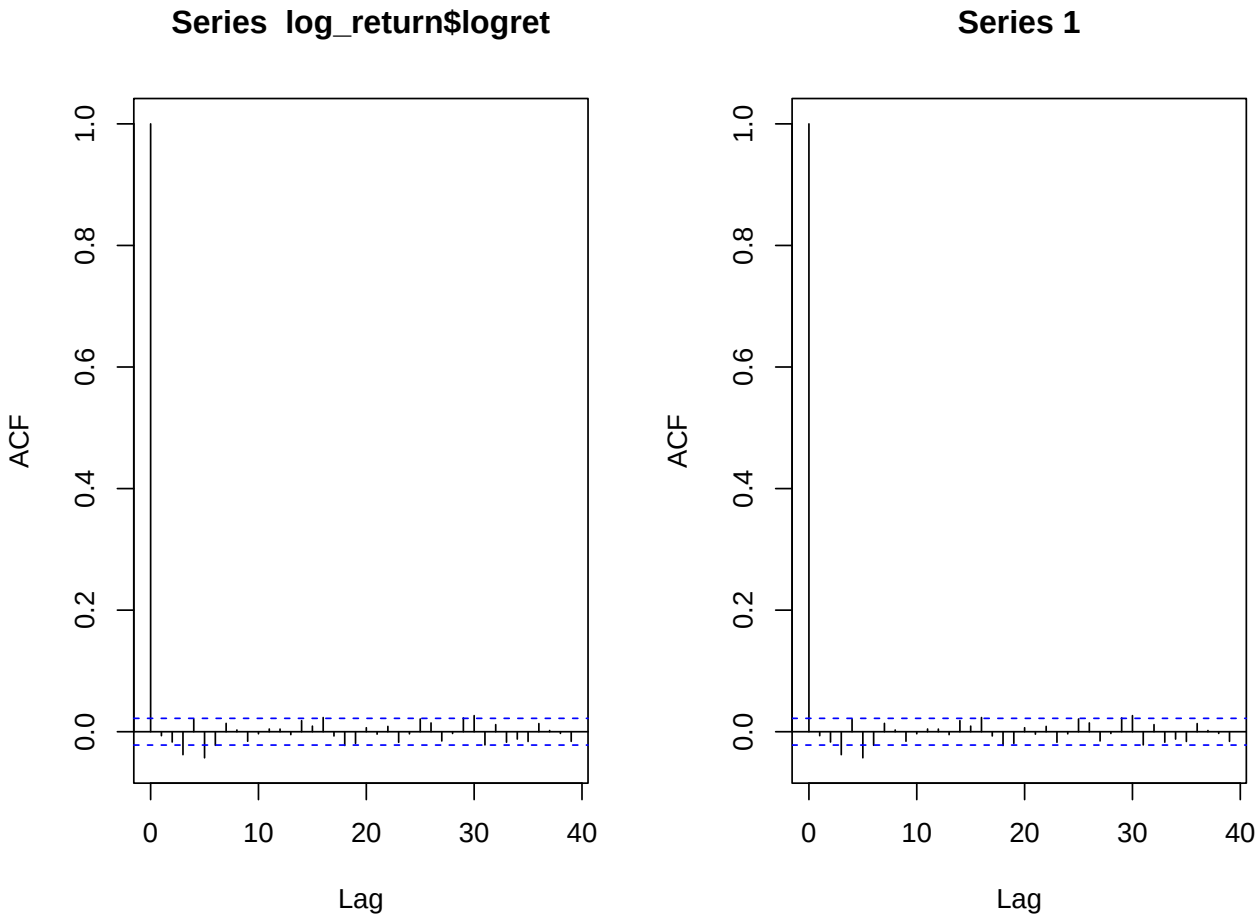
```
## Residual standard error: 0.01375 on 7857 degrees of freedom
## Multiple R-squared:  0.5057, Adjusted R-squared:  0.5053
## F-statistic:  1340 on 6 and 7857 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -38.9162
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.58 -1.95 -1.62
```

```
summary(ur.kpss(log_return$logret, type = 'mu'))
```

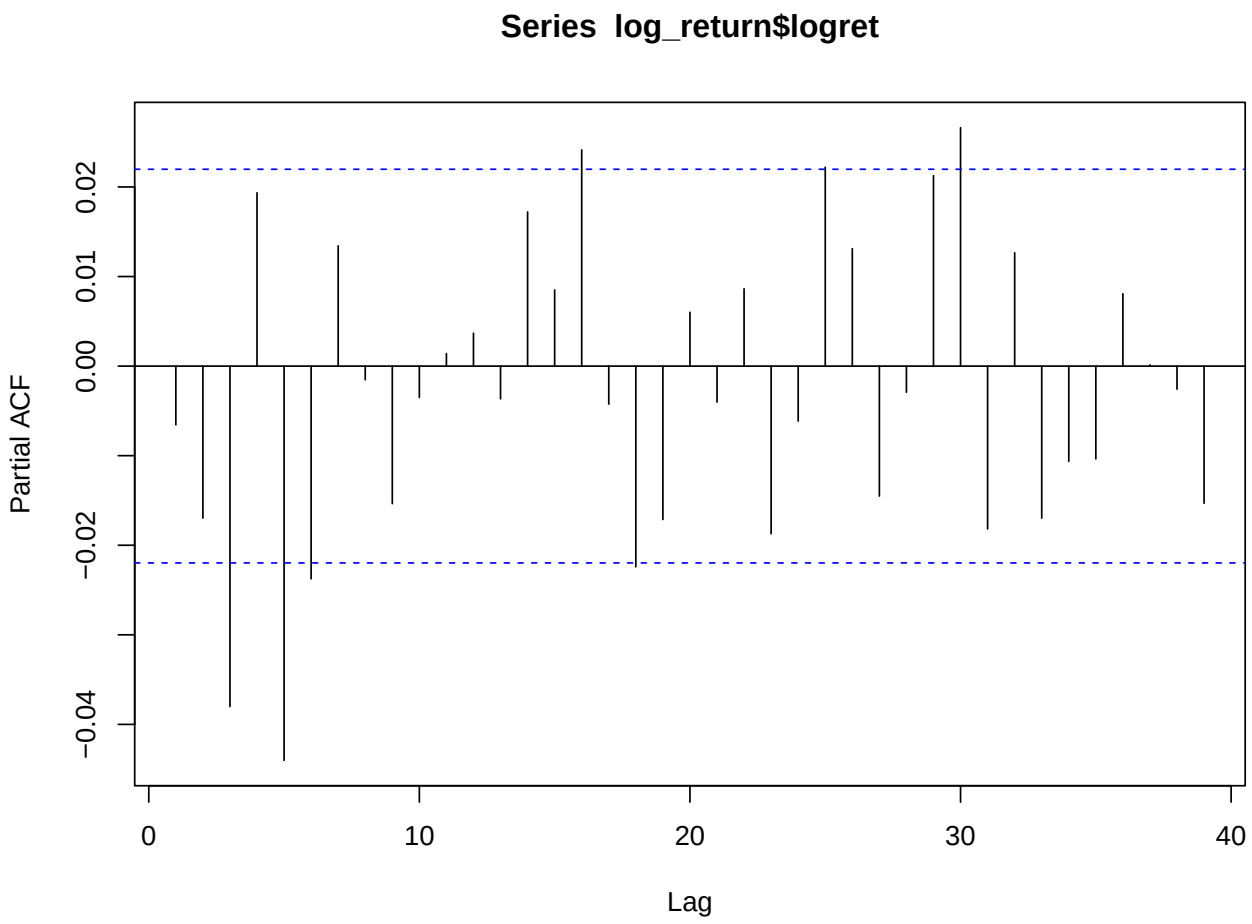
```
##
## ########################
## # KPSS Unit Root Test #
## ########################
##
## Test is of type: mu with 11 lags.
##
## Value of test-statistic is: 0.0607
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```

- Checking autocorrelations

```
par(mfrow = c(1,2))
acf(log_return$logret)
acf(coredata(log_return.ts))
```

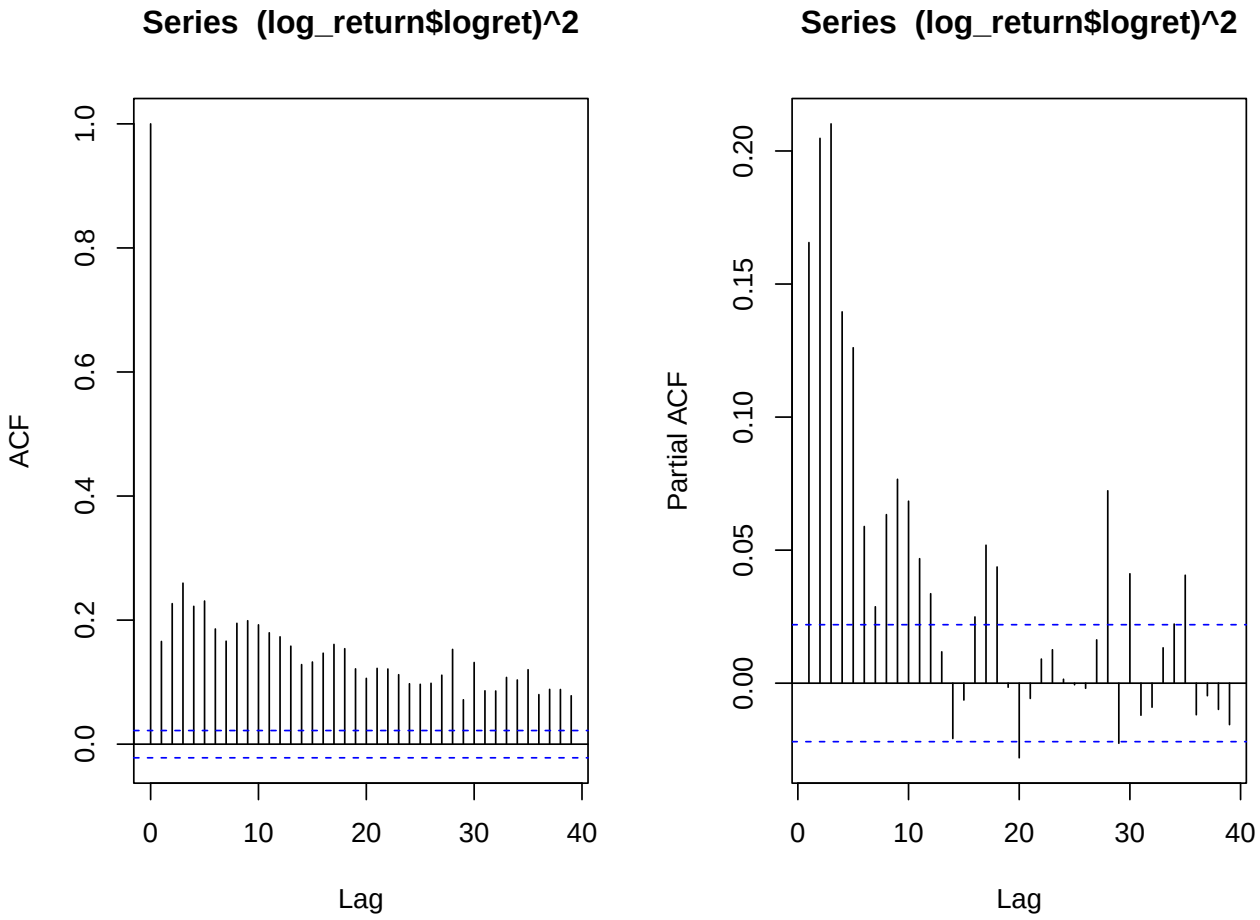



```
pacf(log_return$logret)
```



The results seem unconvincing. Let's check the squared returns instead:

```
par(mfrow=c(1,2))
acf((log_return$logret)^2)
pacf((log_return$logret)^2)
```

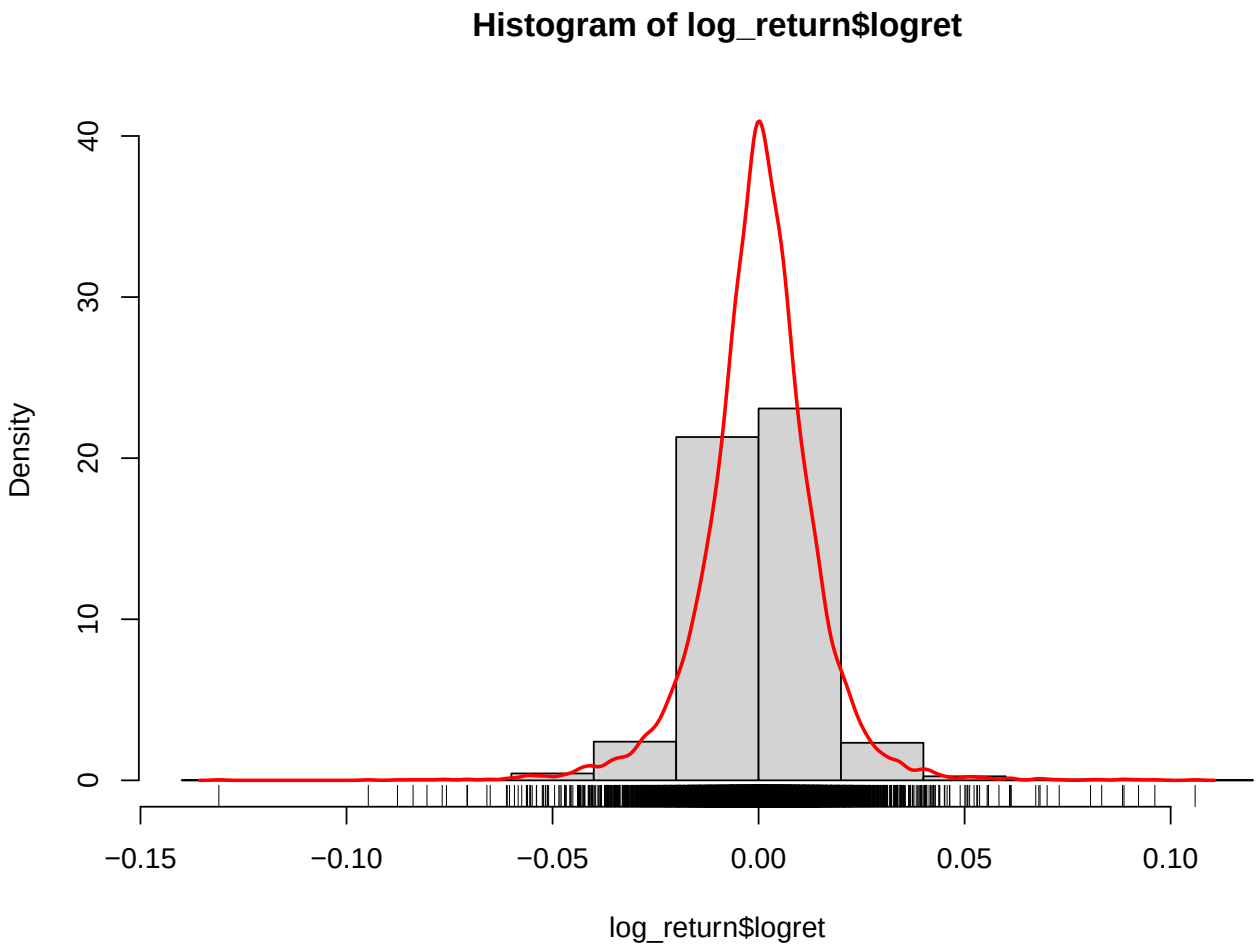


There it is! - severe autocorrelation.

Remark: this dependence will cause issues with variance/SE estimation.

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

```
hist(log_return$logret, probability = TRUE, ylim=c(0,max(density(log_return$logret)$y)))
lines(density(log_return$logret), lwd=2, col="red")
rug(log_return$logret)
```

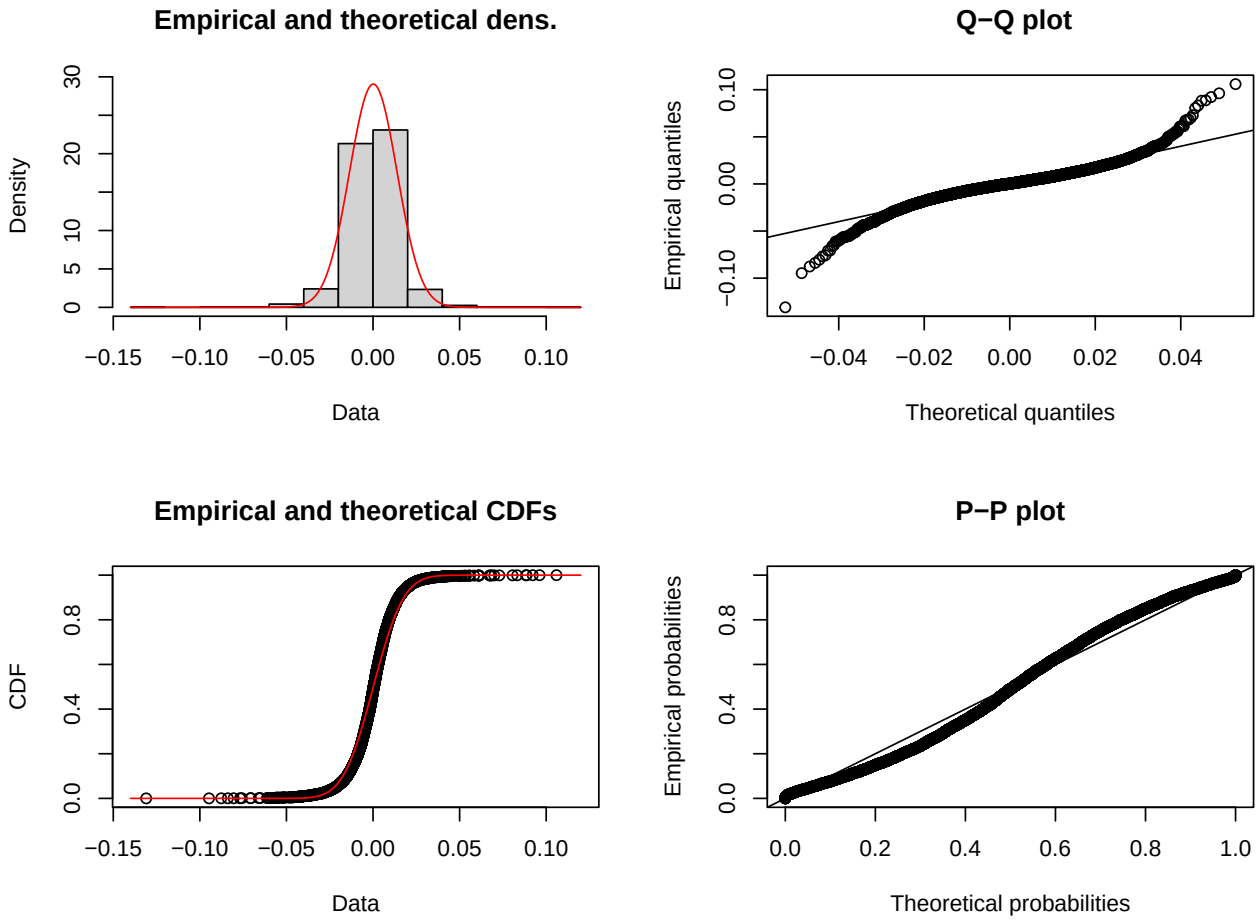



```
summary(log_return$logret)
```

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -0.1309835 -0.0065411  0.0004147  0.0001572  0.0073485  0.1059459
```

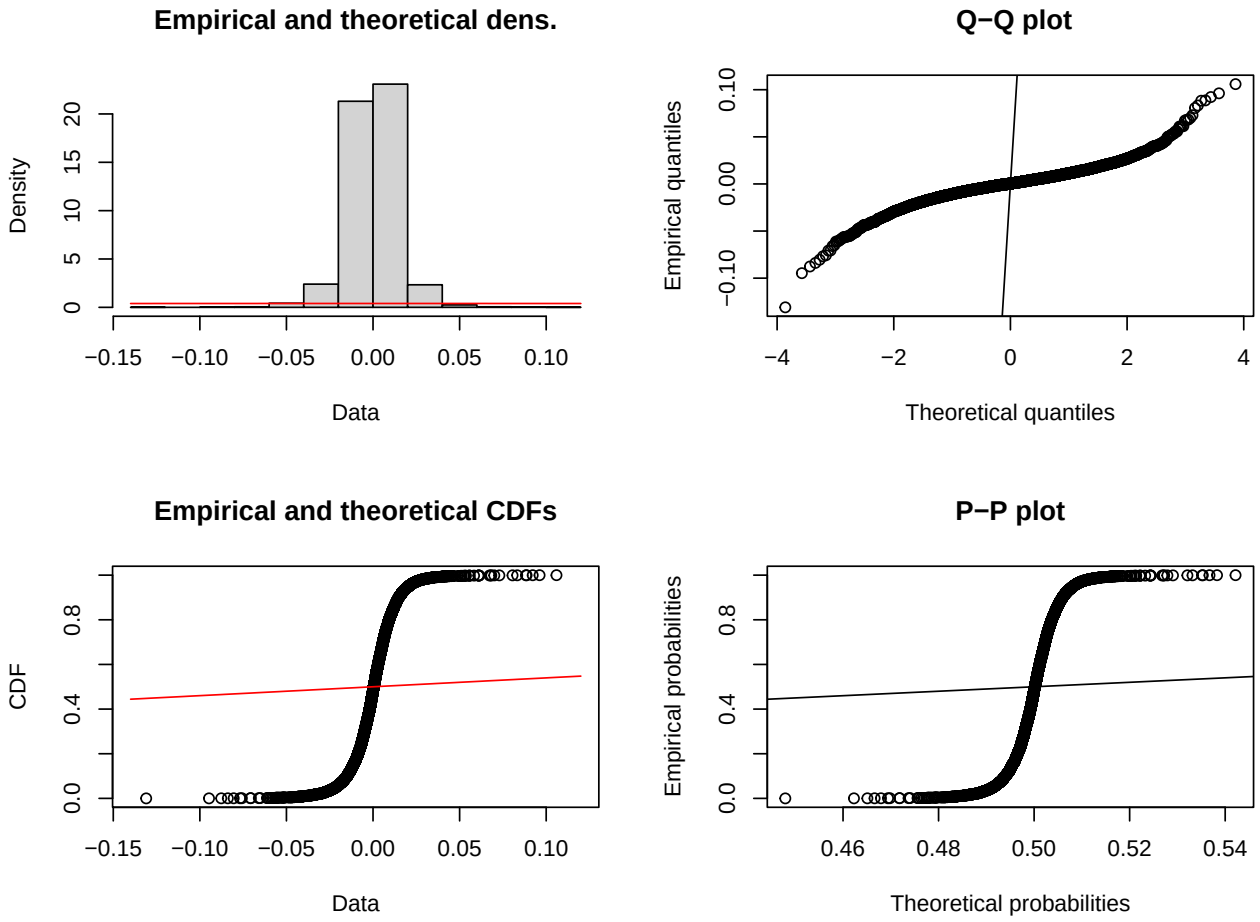
It appears bell-shaped, but the tails are too heavy to be Gaussian. We'll check it anyway.

```
library(fitdistrplus)
norm.fit <- fitdistrplus::fitdist(log_return$logret, "norm", method = 'mle')
plot(norm.fit)
```



Problem in the tail of the distribution. We will try with a student distribution.

```
t.fit <- fitdistrplus::fitdist(log_return$logret, "t", start = list(df=3))
plot(t.fit)
```



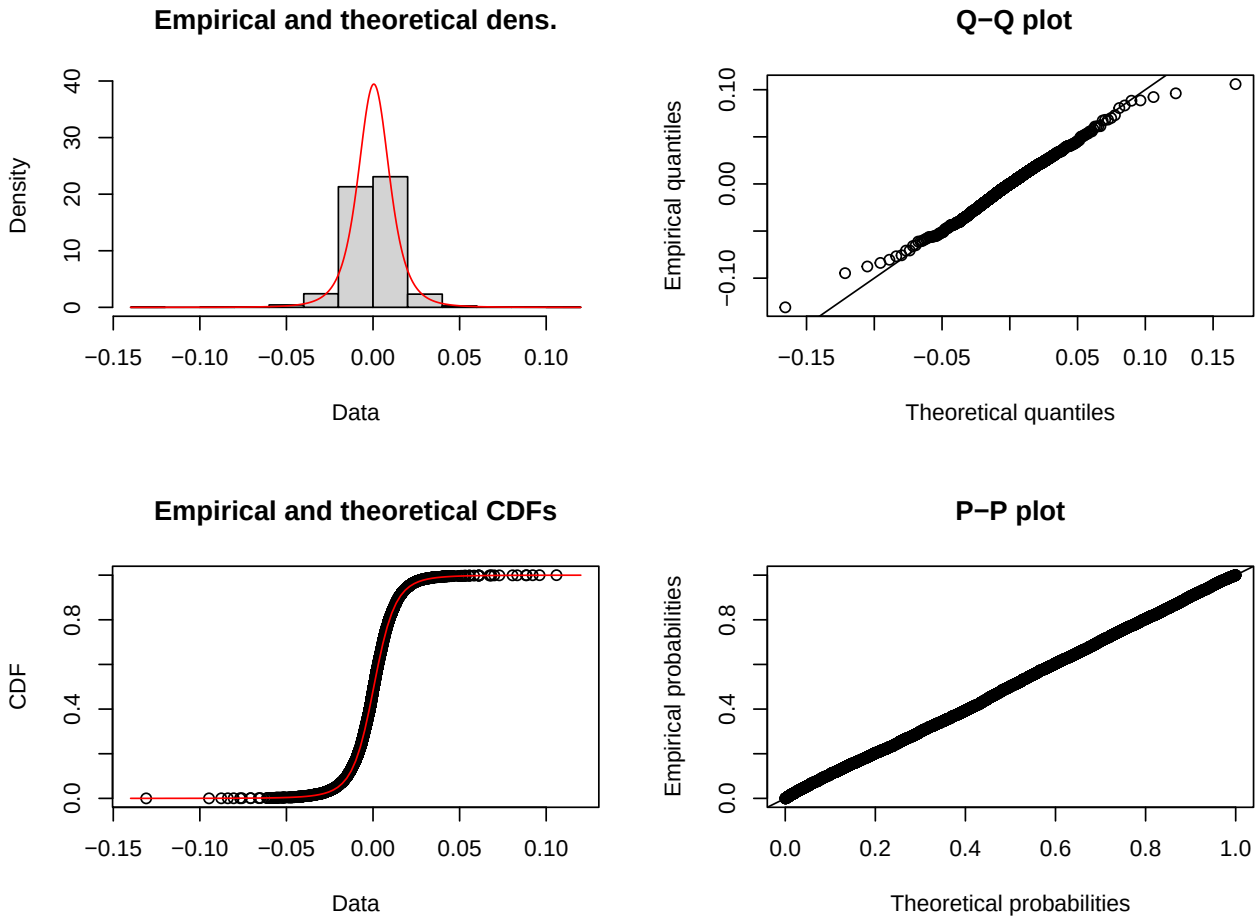
The result is horrible. However, it must be the wrong location-scale.

- Check a location scale shift student distribution:

```
#manual creation of LS student. Exercice
dt_ls <- function(x, df=1, mu=0, sigma=1) 1/sigma * dt((x - mu)/sigma, df)
pt_ls <- function(q, df=1, mu=0, sigma=1) pt((q - mu)/sigma, df)
qt_ls <- function(p, df=1, mu=0, sigma=1) qt(p, df)*sigma + mu
rt_ls <- function(n, df=1, mu=0, sigma=1) rt(n,df)*sigma + mu
t.fit2<- fitdistrplus::fitdist(log_return$logret, 't_ls', start =list(df=1,mu=mean(log_return$logret),s
t.fit2
```

```
## Fitting of the distribution ' t_ls ' by maximum likelihood
## Parameters:
##           estimate   Std. Error
## df    3.6066698101 0.1588165266
## mu    0.0004560405 0.0001268895
## sigma 0.0094429860 0.0001335163
```

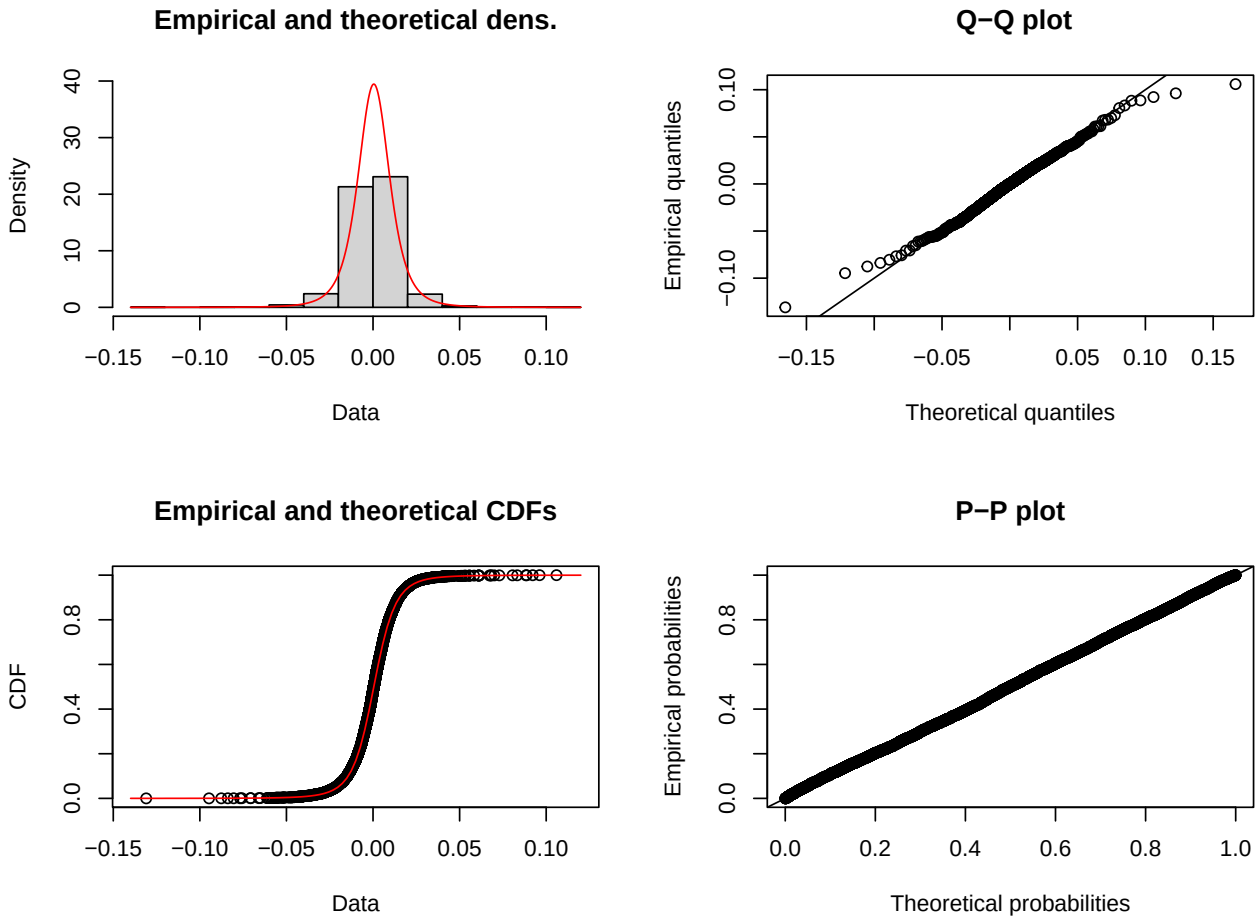
```
plot(t.fit2)
```



The result is much more satisfactory.

There is obviously an automatic version:

```
library(metRology)
t.fit3 <- fitdistrplus::fitdist(log_return$logret, "t.scaled", start=list(df=3,mean=mean(log_return$logr
plot(t.fit3) #identical results to t.fit2
```




```
summary(t.fit3)
```

```
## Fitting of the distribution ' t.scaled ' by maximum likelihood
## Parameters :
##          estimate   Std. Error
## df   3.6071226283 0.1589293320
## mean 0.0004540197 0.0001268762
## sd   0.0094414481 0.0001335199
## Loglikelihood:  23469.55   AIC:  -46933.1   BIC:  -46912.16
## Correlation matrix:
##                df        mean          sd
## df    1.00000000 -0.04056638  0.67096304
## mean -0.04056638  1.00000000 -0.03741193
## sd    0.67096304 -0.03741193  1.00000000
```

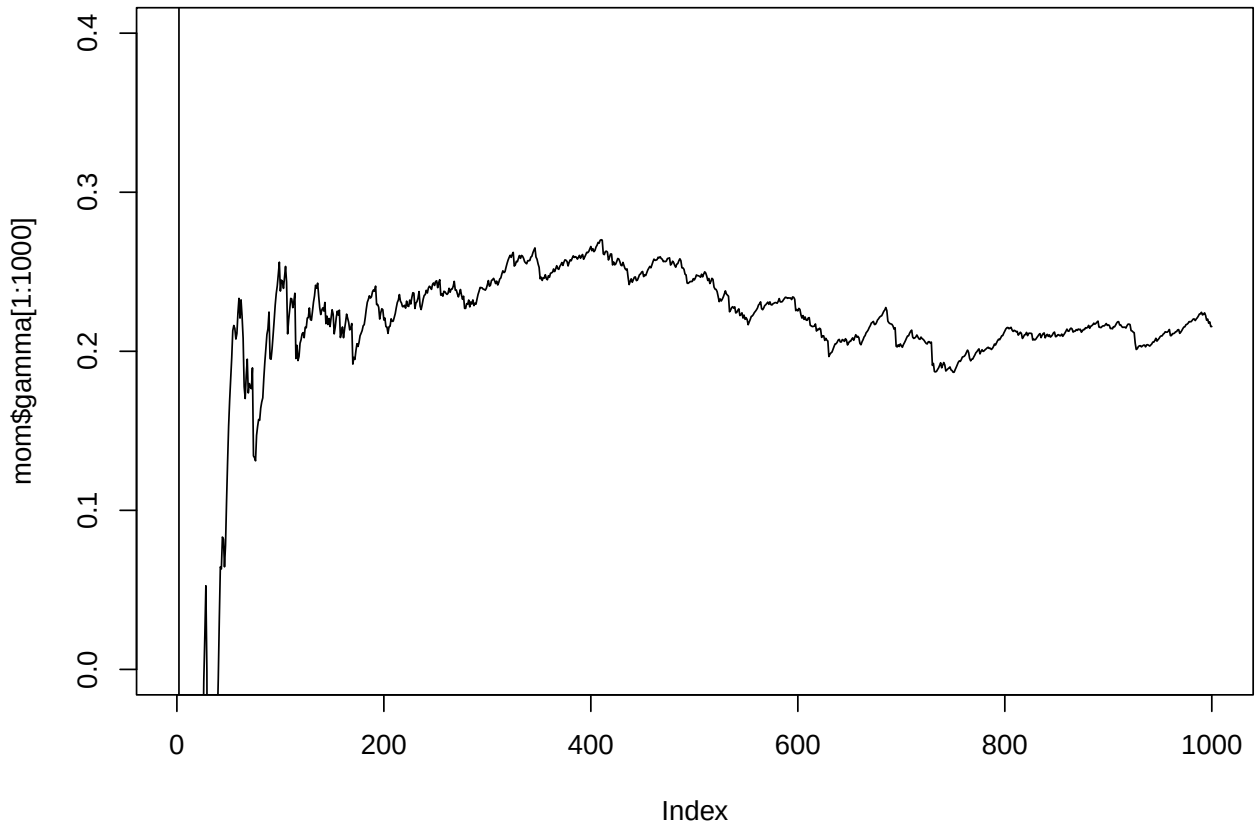
```
detach('package:metRology') #detach clashing packages
```

2.2.3 Check that there is evidence of a heavy right tail in this data set.

The visual inspection done earlier clearly implies that the data is heavy tailed. If you want to confirm that, use the moment/generalized Hill/MLE estimators of the extreme value index, and check if you end up with positive values.

- The moment estimator

```
library(ReIns)
mom<-Moment(log_return$logret[log_return$logret>0])
plot(mom$gamma[1:1000], type='l', lwd=1, ylim=c(0,0.4))
```



The extreme value index is clearly positive, which confirms that the data is heavy-tailed.

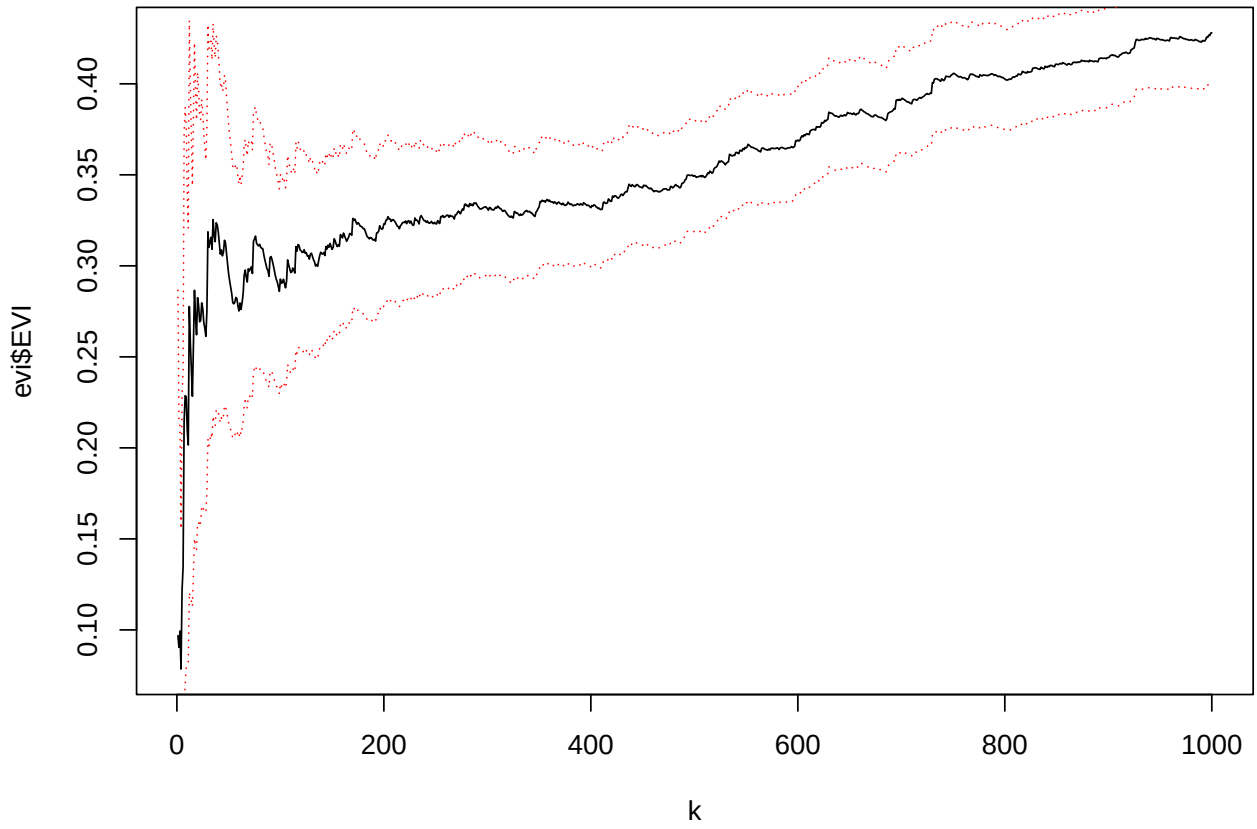
2.2.4 Calculate an estimate for the extreme value index of the data. Justify your choice of tuning parameters.

- Tail index estimation using the Hill estimator.

Since the data is heavy-tailed, we use the hill estimator to find reasonable values of the tail index.

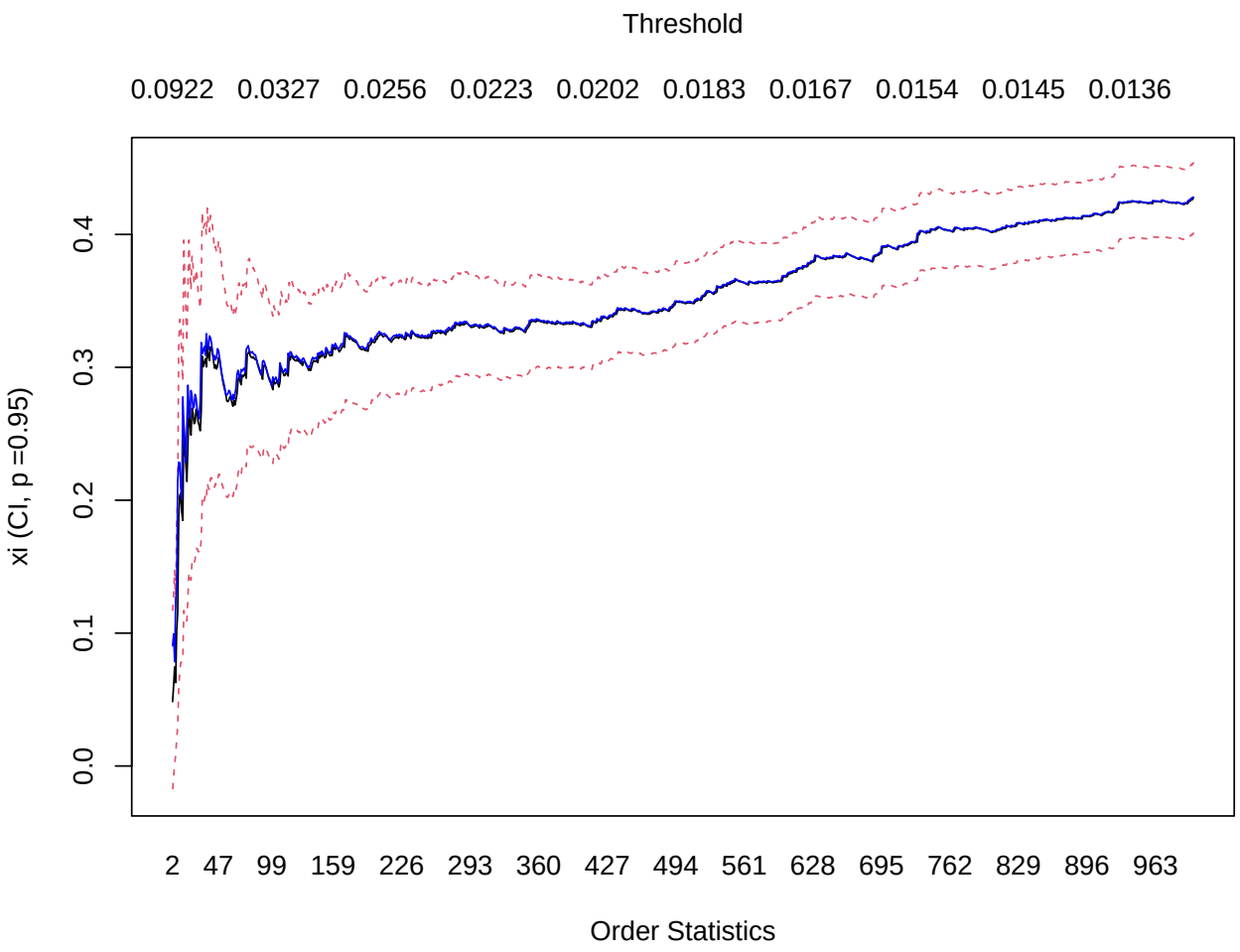
```
k=1:1000
alpha<-0.05

library(evt0) #trusted package (you can use ReIns as well)
evi <-mop(log_return$logret, k=k, p=0) #p=0 for Hill estimator (see Mean of order p (mop) statistic for
plot(k,evi$EVI, type='l', lty=1)
#confidence intervals (these are meaningless in the context of this data)
qcrit<-qnorm(1-alpha/2)
u<-evi$EVI+qcrit*evi$EVI/sqrt(k) #confidence intervals
l<-evi$EVI-qcrit*evi$EVI/sqrt(k) #however, they're meaningless because of dependence
lines(k,u,lty=3,col='red')
lines(k,l,lty=3,col='red')
```



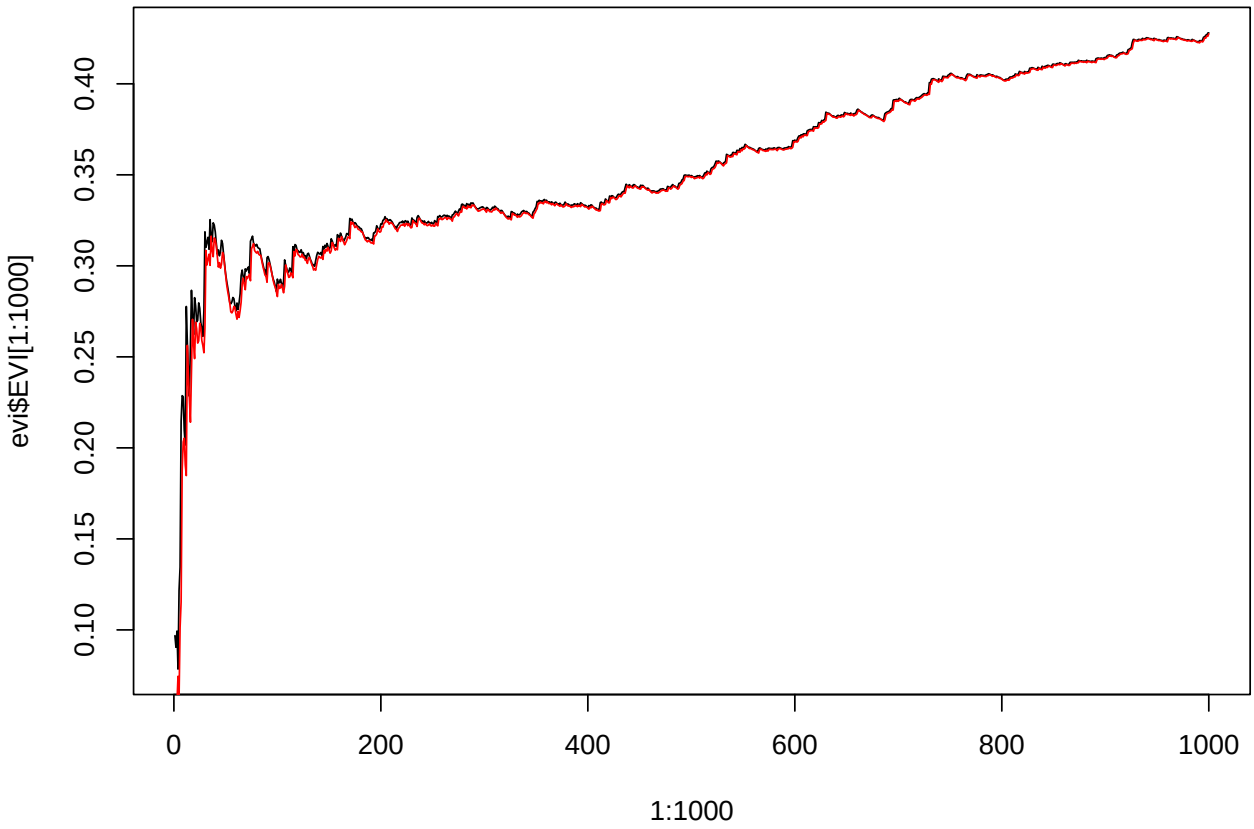
- Trying a different package

```
library(evir)
evi.evir <- hill(log_return$logret, start=2, end=1000, option = "xi")
lines(2:1000, evi$EVI[2:1000], col="blue")
```



Trouble signs, `evir` has issues with its hill code, so don't use it !

```
plot(1:1000,evi$EVI[1:1000], type='l')
lines(2:1000, evi.evir$y[length(evi.evir$y):1], col='red')
```

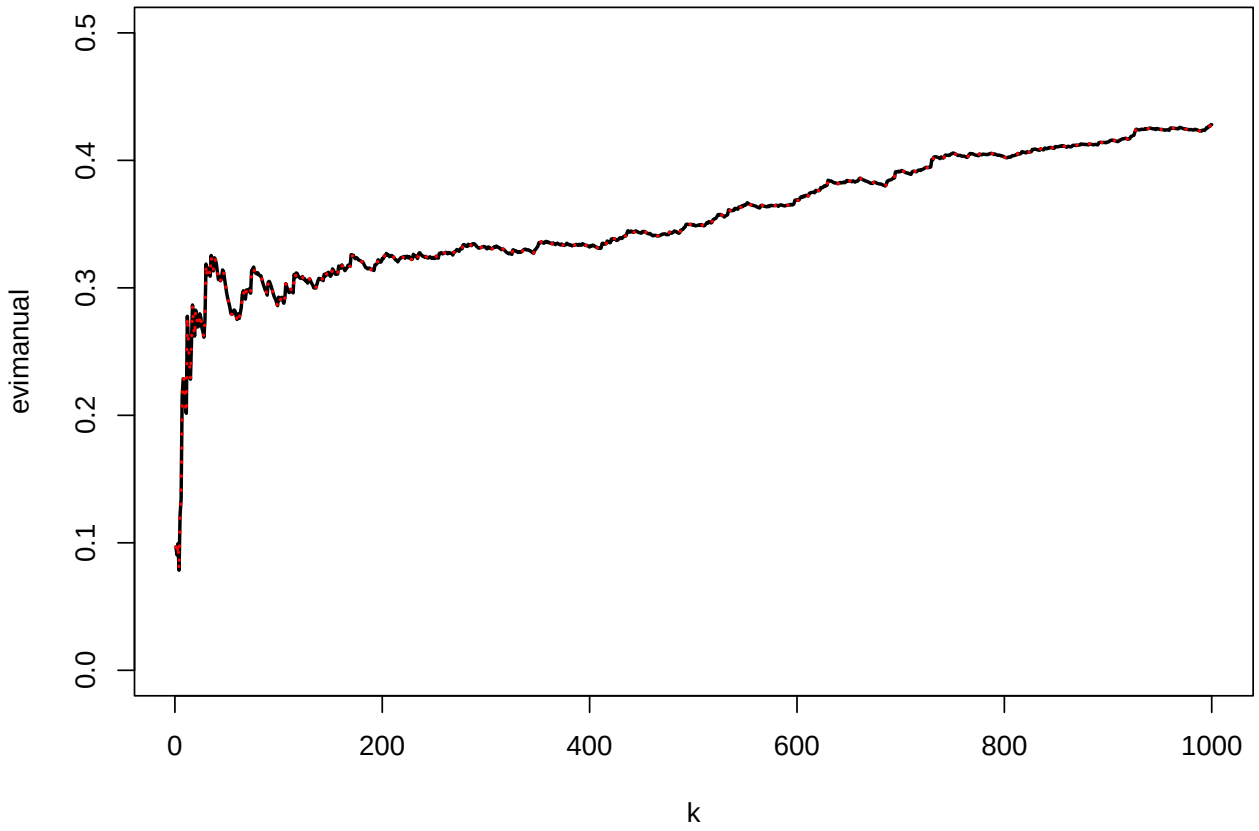


Stick to mop from `evt0`, or Hill from `ReIns` or write it yourself:

- Manual tail index estimator (with loop)

```
hillmanual  <- function(y,k){
  n=length(y)
  ysort=sort(y)
  som1=log(ysort[n])
  evi<-som1-log(ysort[n-1])
  for (i in 1:k){
    som1 = som1 + (log(ysort[n-i]))
    ev = som1/(i+1)-log(ysort[n-i-1])
    evi<-c(evi,ev)
  }
  return(evi)
}
```

```
evimanual<-hillmanual(log_return$logret, 999)
plot(k, evimanual, ylim=c(0,0.5), type='l', lwd=2)
lines(1:1000,evi$EVI[1:1000], col='red', lty=3, lwd=2)
```

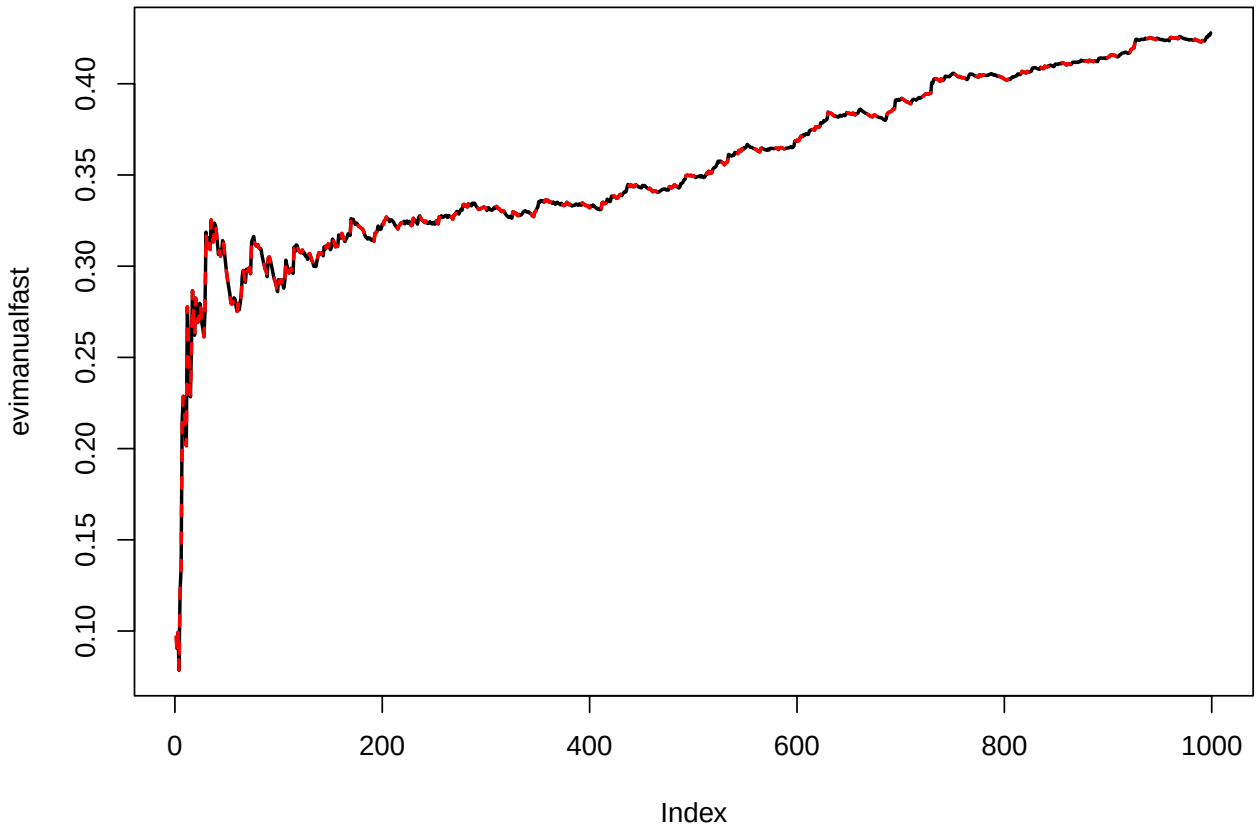


Perfect match with mop !

- Manual tail index estimator (without loop)

```
hillmanualfast<-function(y,k){
  ysort=sort(y, decreasing = T)
  ysort<-ysort[ysort>0]
  logy<-log(ysort)[1:k]
  cumlogy<-cumsum(logy)/(1:k)
  xi<-c(NA,(cumlogy[-length(cumlogy)]-logy[-1])[2:k])
  return(xi)
}
```

```
evimanualfast<-hillmanualfast(log_return$logret, k=1000)
plot(evimanualfast, type='l', lwd=2)
lines(1:1000,evi$EVI[1:1000], col='red', lty=2, lwd=2) #
```



Perfect match with mop !

- Let's look at the speed of each code:

```
x <- log_return$logret
system.time(hillmanualfast(x, k=1000))
```

```
##    user  system elapsed
##   0.001   0.000   0.001
```

```
system.time(hillmanual(x, k=1000))
```

```
##    user  system elapsed
##   0.001   0.001   0.002
```

```
system.time(mop(x, k=k, p=0)$EVI)
```

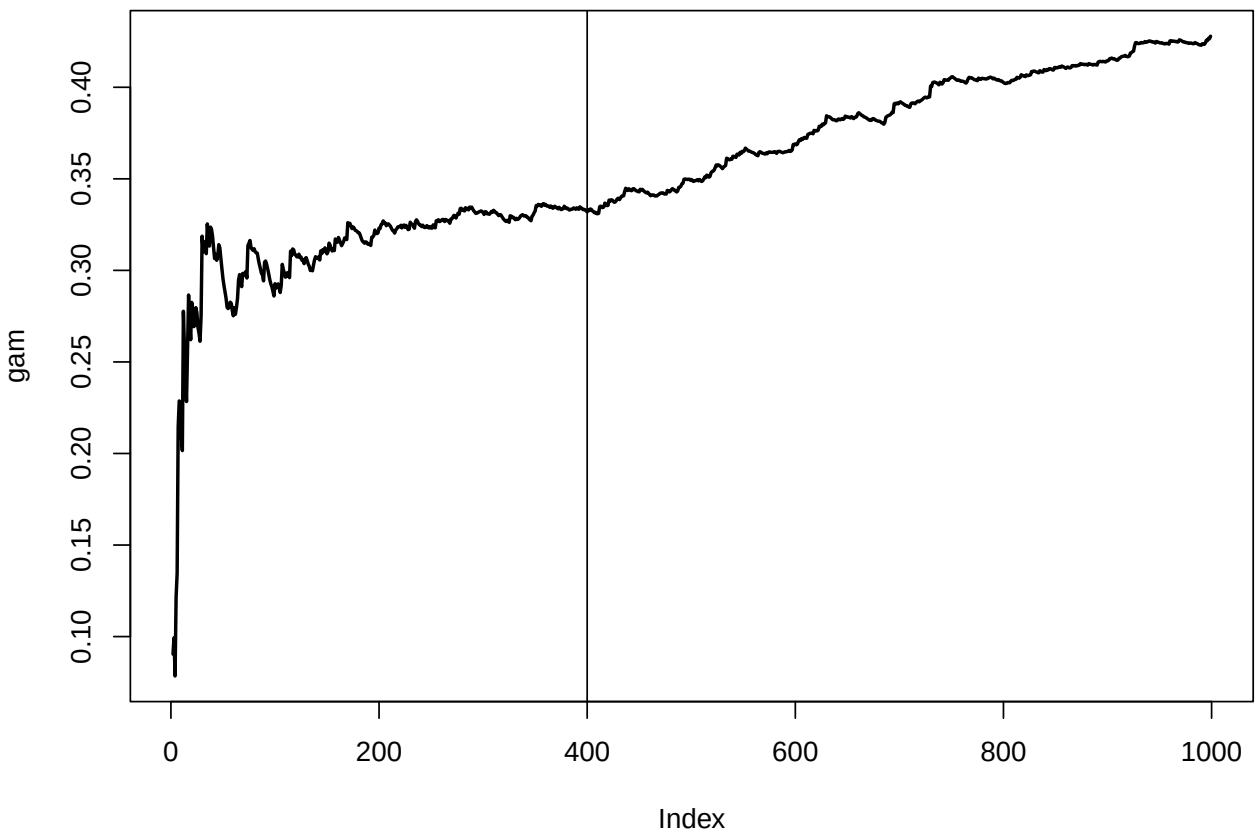
```
##    user  system elapsed
##       0       0       0
```

Quite similar in all three cases, but the code with loops is a bit slower (results may vary depending on pc specs).

- Extrapolation

To extrapolate, we need to choose a specific value of the Hill estimator, which corresponds to the choosing k

```
n<-length(x)
gam<-evimanualfast
plot(gam, type='l', lwd=2) #check for stable region, then pick the value of k at the end of it
#250-400 looks fairly stable, so we choose k=400
abline(v=400)
```



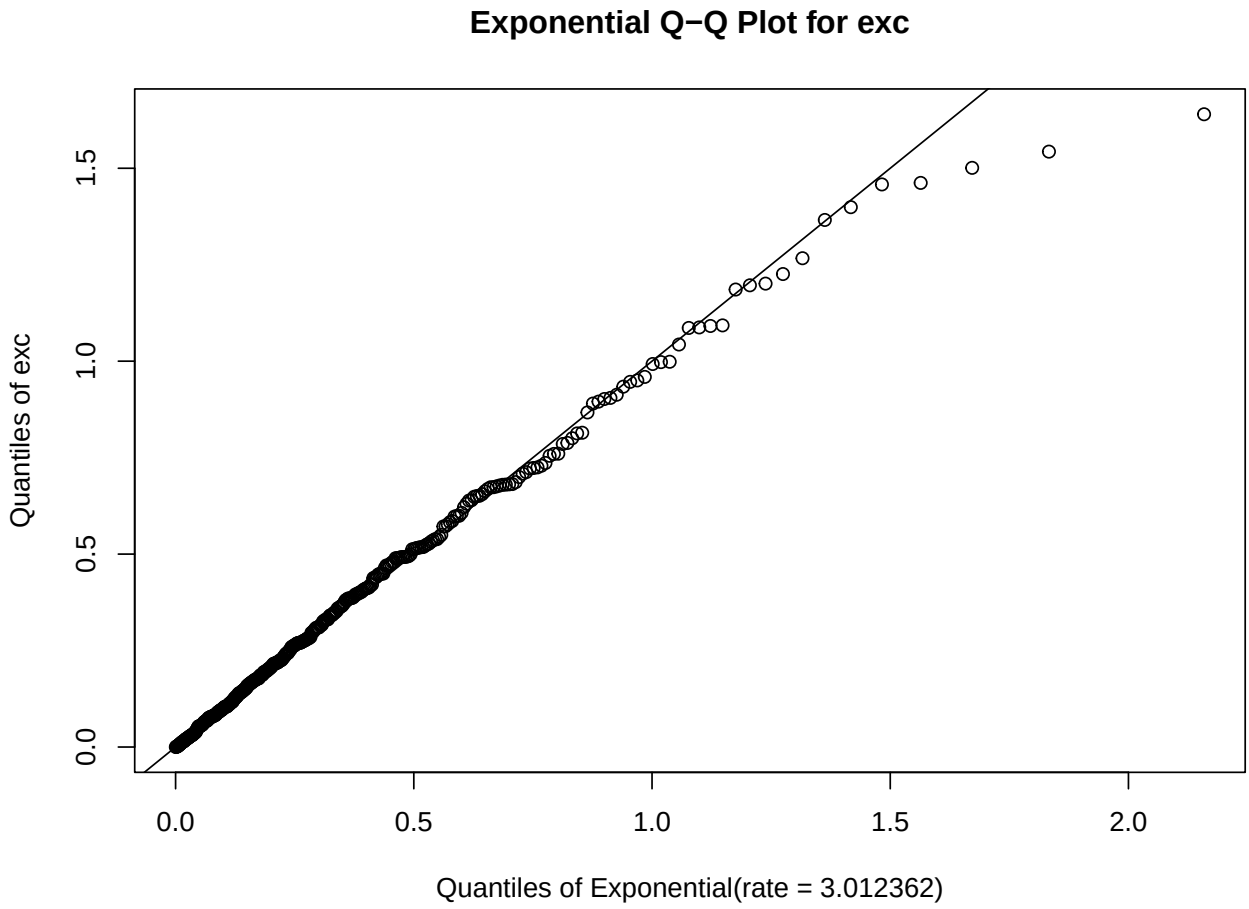

```
#check choice of k using exponential QQ
kn<-400

sorted<-sort(log(x), decreasing = T)
exc<-(sorted-sorted[kn])[1:kn] #these log exceedances should be exponentially with mean=tail index
any(is.na(exc)) #a quick way to check for NAs in your data
```

```
## [1] FALSE
```

- QQ plot vs exponential with parameter=1/gamma

```
library(EnvStats)
qqPlot(exc,y=NULL, "exp", param.list = list(rate=1/gam[kn]))
abline(0,1)
```



The result looks good !

2.2.5 Calculate an estimate for the extreme quantiles at level 0.995 and $1 - 1/n$, where n denotes the sample size. Do these make sense?

```
#we now have our tail index estimate
gamhat<-gam[kn]

#0.995 extrapolation (0.995 might seem extreme, but the sample size is quite large)
ext.quantile <-quantile(x,1-kn/n, type=3)*((1-0.995)/(kn/n))^{-gamhat} #type=3 is needed for order stat
empirical.quantile <- quantile(x,0.995)
c(ext.quantile, empirical.quantile)
```

```
##  94.97045%      99.5%
## 0.04421507 0.04247024
```

Extrapolated and empirical quantiles are very close.

```
#truly extreme level
taunp<-1-1/n
taunp
```

```
## [1] 0.9998743
```

```
ext.quantile.t<-quantile(x,1-kn/n, type=3)*((1-taunp)/(kn/n))^{-gamhat}
empirical.quantile.t <- quantile(x,taunp)
c(ext.quantile.t, empirical.quantile.t)
```

```
##  94.97045%  99.98743%
## 0.15015826 0.09617054
```

Extrapolation goes way past the empirical quantile, as expected.

2.2.6 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?

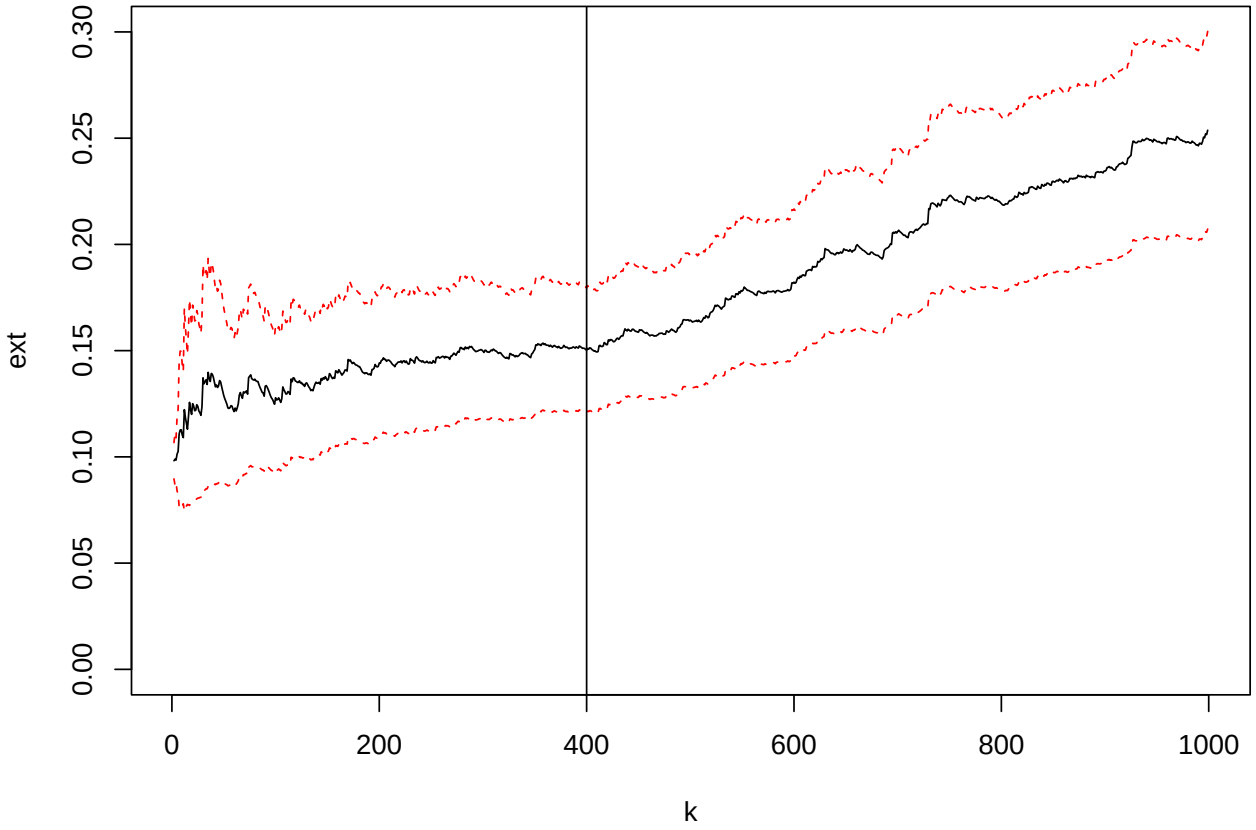
For confidence intervals, we already know they're meaningless because of dependence. However, here's a quick way to calculate these meaningless intervals

```
c1 <-ext.quantile.t * (1-qnorm(1-alpha/2)*log((kn/n)/(1-taunp))*sqrt(1/kn)*gamhat)
c2 <-ext.quantile.t * (1+qnorm(1-alpha/2)*log((kn/n)/(1-taunp))*sqrt(1/kn)*gamhat)
c(c1,ext.quantile.t,c2)
```

```
## 94.97045% 94.97045% 94.97045%
## 0.1208903 0.1501583 0.1794263
```

You can plot extrapolated quantiles and their CIs against k as well using

```
ext<-quantile(x,1-k/n, type=3)*((1-taunp)/(k/n))^{-gam} #make use of the vectorization, no loops
c1 <-ext * (1-qnorm(1-alpha/2)*log((k/n)/(1-taunp))*sqrt(1/k)*gam)
c2 <-ext * (1+qnorm(1-alpha/2)*log((k/n)/(1-taunp))*sqrt(1/k)*gam)
plot(k,ext, type = 'l', ylim=c(0,0.3))
lines(k,c1,col='red',lty=2)
lines(k,c2,col='red',lty=2)
abline(v=kn)
```



Remark To generate meaningful confidence intervals, we need to account for dependence. Prof. Stupfler's package `ExtremeRisks` has some nice tools for that

```
#library(ExtremeRisks)
#?HTailIndex
```

2.3 SOA data

The only variable provided in this example is the amount of money related to a medical claim exceeding \$25,000 in 1991 in the USA. This data set can be loaded by typing `data(soa)` after having loaded the `ReIns` package.

```
library(ReIns)
data("soa")
```

1. Represent the data. Can you find a reasonable statistical model for the whole of the data?
2. Check that there is evidence of a heavy right tail in this data set.
3. Calculate an estimate for the extreme value index of the data. Justify your choice of tuning parameters.
4. Calculate an estimate for the extreme quantiles at level 0.995 and $1-1/n$, where n denotes the sample size. Do these make sense? Do you think that the level 0.995 can be considered extreme here?
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?

2.4 US Census data

The variable of interest is the 2010 population size in US cities having more than 50,000 inhabitants. Other variables provided are population projections in subsequent years.

```
#library("readxl")
census <- read.csv("~/Documents/GIT/teaching/extreme value theory/data sets for practical sessions/censu
census$Census <- as.numeric(census$Census)
census.2010 <- census$Census
```

1. Represent the data. Can you find a reasonable statistical model for the whole of the data?
2. Check that there is evidence of a heavy right tail in this data set.
3. Calculate an estimate for the extreme value index of the data. Justify your choice of tuning parameters.
4. Calculate an estimate for the extreme quantiles at level 0.99, 0.995 and 0.999. Do these make sense? Can you provide an interpretation for such 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?