Homework 3

Due on Feb 24, 2025 (Mon)

Instructions:

- Install `pdflatex`, `R`, and `RStudio` on your computer.
- Please edit the `HW3_First_Last.Rnw` file in `Rstudio` and compile with `knitr` instead of `Sweave`. Go to the menu `RStudio|Preferences...|Sweave` choose the `knitr` option, i.e., `Weave Rnw files using knitr?` You may have to install `knitr` and other necessary packages.
- Replace "First" and "Last" in the file-name with your first and last names, respectively. Complete your assignment by modifying and/or adding necessary R-code in the text below.
- You should submit both a **PDF file** and a **ZIP file** contining the data and the *HW3_First_Last.Rnw* file required to produce the PDF. The file should be named "HW3_First_Last.zip" and it should contain a single folder named "First_Last" with all the necessary data files, the `HW3_First_Last.Rnw` and `HW3_First_Last.pdf` file.

  This file must be obtained from compiling `HW3_First_Last.Rnw` with `knitr` and LaTeX. Again please replace "First_Last" with your name.
- The GSI grader will annotate the PDF file you submit. However, they will also check if the code you provide in the ZIP file compiles correctly. If the file fails to compile due to errors other than missing packages, there will be an automatic 10% deduction to your score.

Problems: For your convenience a number of R-functions are included in the .Rnw file and not shown in the compiled PDF.
Note: You may have to modify the provided functions if you think they have errors or they are unstable. It is your responsibility to provide correct and functioning code.

1. **The goal of this exercise is to practice the peaks-over-threshold methodology over simulated data.**

   (a) Simulate a sample of $n = 10\,000$ t-distributed random variables with degrees of freedom $\nu = 1.1,\ 2, 3, 4$. For each sample, produce *mean-excess plots* to determine visually a "reasonable" threshold u_0 above which the distribution can be modeled with a Generalized Pareto Distribution (GPD). Examine a range of thresholds obtained from the empirical quantiles of the data, e.g., $u = \texttt{quantile}(x, \texttt{seq}(\texttt{from} = 0.5, \texttt{to} = 0.999, \texttt{length.out} = 100))$ and provide plots of the empirical mean of the excesses as a function of u.

   Discuss: What is the effect of ν on the choice of u_0. Identify the quantile levels for the "best" choices of u_0.

   My choice on u_0 is (42, 8.5, 5, 4.5) for $\nu = 1.1, 2, 3, 4$, We can see that when nu is too small, the distribution will have heavy tail, making more extreme leading to a higher threshold u_0 for the Generalized Pareto Distribution model to be appropriate.

(b) For each ν from part **(a)** let now u_0 be chosen as the empirical quantile of the data. For each of the 4 degrees of freedom, fit the GPD using numerical maximization of the log-likelihood for the samples of excesses over the extreme thresholds u_0. Produce asymptotic 95% confidence intervals for the parameter ξ based on the Hessian.

(c) The goal of this part is to study the coverage probabilities of the confidence intervals for ξ obtained in part **(b)**.

Step 1. Fix ν and simulate a sample from the t-distribution of length $n = 10,\ 000$. Let the thresholds u be computed as the empirical quantiles of the data of levels $p = \texttt{seq}(from = 0.8, to = 0.99, length.out = 10)$. For each such threshold compute the 95% confidence intervals for ξ and note whether they cover the true $\xi = 1/\nu$.

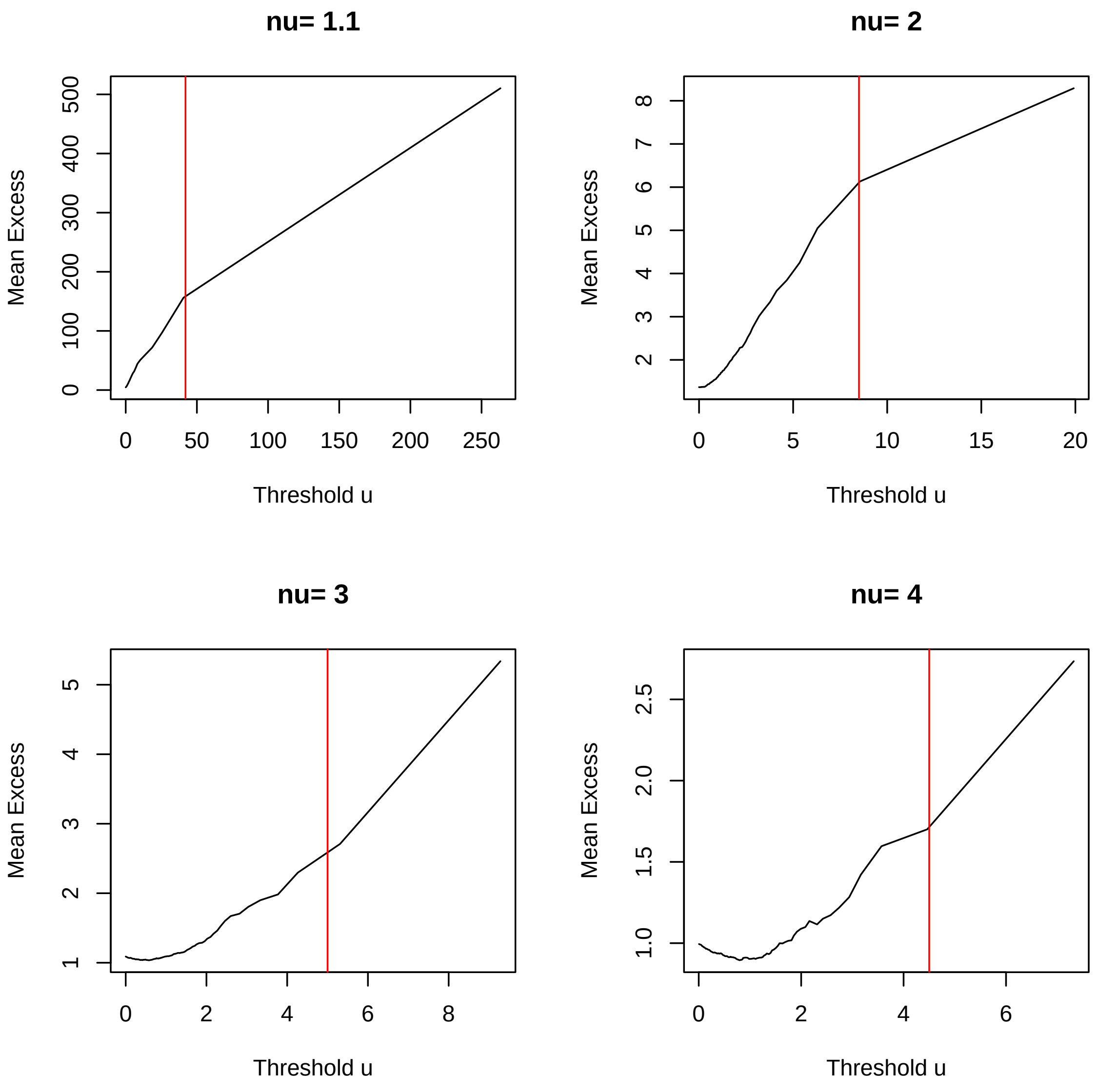
Step 2. Replicate Step 1 independently 500 times for $\nu = 1.1,\ 2,\ 3,\ 4$. Report the empirical coverages of the resulting confidence intervals for ξ in a 4×10 table, where the columns correspond to the quantile levels p of the thresholds u and the rows to the different values of the parameter ν of the t-distribution.

Discuss the results from this experiment.

The experiment demonstrates that selecting an appropriate threshold $\mu$is critical for obtaining reliable confidence intervals for ξ. When ν is small (heavy tails), coverage probabilities are generally high across a range of thresholds. However, for larger ν (lighter tails), only very extreme thresholds $p > 0.9$ lead to accurate inference.

Hint for part (a)

```r
nu = c(1.1,2,3,4);
pu = seq(from=0.5,to=0.999,length.out=100);

set.seed(0220)
par(mfrow=c(2,2))
u0 = c(42,8.5,5,4.5) # Result after observation
for (i in c(1:4)){
   x =rt(n=1e4,df=nu[i]);
   u=quantile(x,pu)
   me = mean.excess(x,u);
   plot(u,me,type="l",main=paste("nu=",nu[i]),xlab="Threshold u",ylab="Mean Excess");
   abline(v=u0[i], col='red')
}
```



(b)

```
library(knitr)
u0 = c(42,8.5,5,4.5);
ci =cbind(1/nu,0,0)
for (i in c(1:4)){
  x = rt(n=1e5,df=nu[i]);
  res=fit.gpd.Newton_Raphson(x,threshold=u0[i]);
  ci[i,2]=res$xi-1.96*res$se.xi;
  ci[i,3]=res$xi+1.96*res$se.xi;
}
```

```
  colnames(ci)=c("xi","Lower","Upper")
  kable(ci,digits=4)
```

xi	Lower	Upper
0.9091	0.7593	1.0739
0.5000	0.3678	0.5834
0.3333	0.1867	0.3574
0.2500	0.1412	0.3662

```
pu = round(seq(from = 0.8, to = 0.99,length.out = 10), 2);
coverages = matrix(0,4,length(pu));
iter = 500;
for (i in c(1:4)){
    for (k in c(1:iter)){
      x = rt(n=1e4,df=nu[i])
      res = fit.gpd.Newton_Raphson(x,pu=pu)
      coverages[i,] = coverages[i,]+
        as.numeric((res$xi+1.96*res$se.xi>=(1/nu[i]))
                   &(res$xi-1.96*res$se.xi<=(1/nu[i])));
  }
}
  coverages = coverages/iter;
  coverages = cbind(nu,coverages)
  colnames(coverages)=c("nu",as.character(pu))
  kable(coverages,digits=4)
```

nu	0.8	0.82	0.84	0.86	0.88	0.91	0.93	0.95	0.97	0.99
1.1	0.750	0.824	0.898	0.906	0.934	0.946	0.950	0.954	0.944	0.920
2.0	0.124	0.242	0.392	0.580	0.686	0.806	0.874	0.888	0.920	0.918
3.0	0.000	0.018	0.080	0.184	0.334	0.594	0.756	0.842	0.876	0.888
4.0	0.000	0.004	0.012	0.054	0.132	0.372	0.552	0.740	0.854	0.880

For a fixed ν, a higher quantile generally results in more experiments successfully covering the true ξ. Conversely, at the same quantile level, an increase in ν tends to decrease the coverage probability.

2. **The goal of this problem is to implement the Peaks-over-Threshold methodology for the predition of Value-at-Risk and Expected Shortfall.**

(a) Write an R-function which estimates Value-at-Risk and Expected Shortfall at level α for extremely small α by using the Peaks-over-Threshold methodology.

The function inputs are:

- `data:` a vector containing the data
- α: a vector of levels for VaR_α and ES_α
- p: a scalar $0 < p < 1 - \alpha$ – quantile level for the threshold, i.e., $u = \texttt{quantile}(\texttt{data}, p)$.

The function output: A list of two arrays containing the VaR_α and ES_α estimates.

The function should fit the GPD model to the data exceeding threshold u and use the tail of the estimated GPD to extrapolate the values of VaR_α and ES_α. Recall that the estimate of VaR_α is:

$$\widehat{\text{VaR}}_\alpha(u) := \left(\left[\frac{n\alpha}{N_u} \right]^{-\widehat{\xi}} - 1 \right) \times \frac{\widehat{\sigma}(u)}{\widehat{\xi}} + u,$$

where in this case $N_u/n \approx p$ will be the proportion of the sample exeeding u.

Derive an analytic expression. for $ES_\alpha = \mathbb{E}[X|X > VaR_\alpha]$ in terms of ξ and σ by assuming that $X - u|X > u$ is $GPD(\xi, \sigma)$ and $\xi < 1$. Use this expression in computing point-estimates and parametric-bootstrap confidence intervals for ES_α.

Hint to Part (a):

```r
x = rt(n=1e4,df=3)
get_VaR <- function(data, p=0.99, alpha){
  u = quantile(data,p);
  names(u)="";
  fit <- fit.gpd.Newton_Raphson(data,threshold = u);
  xi.hat <- fit$xi
  sig.hat <- fit$sig
  Cov = fit$Cov[1,,]
  VaR <-  ((alpha/mean(data>=u))^(-xi.hat) -1)*sig.hat/xi.hat+u
  return(list("VaR"=VaR,"alpha"=alpha,
              "Cov"=Cov,"xi"=xi.hat,
              "sig"=sig.hat))
}
#
# Add a function computing Expected Shortfall with similar inputs and outputs.
#
get_ES <- function(data, p=0.99, alpha){
  u = quantile(data,p);
  names(u)="";
  fit <- fit.gpd.Newton_Raphson(data,threshold = u);
  xi.hat <- fit$xi
  sig.hat <- fit$sig
  Cov = fit$Cov[1,,]
  VaR <-  ((alpha/mean(data>=u))^(-xi.hat) -1)*sig.hat/xi.hat+u
  ES = sig.hat/(1-xi.hat) + u + (VaR-u)/(1-xi.hat)

  return(list("ES"=ES,"alpha"=alpha,
              "Cov"=Cov,"xi"=xi.hat,
              "sig"=sig.hat))
}
```

(b) We want to obtain confidence intervals for VaR_α and ES_α. Here, we will implement the so-called **parametric bootstrap**.

Step 1. Use the function from part **(a)** to fit a GPD model to the excesses of the data over the p-th quantile.

Step 2. Pretending that the MLE is precisely asymptotically normal, we will assume that $(\hat{\xi}, \hat{\sigma}) \sim \mathcal{N}((\xi, \sigma), C)$, where C is the inverse of the Hessian.

Now, we simulate $N = 2\,000$ independent bivariate Normal random vectors

$$(\hat{\xi}_i^*, \hat{\sigma}_i^*) \sim \mathcal{N}((\hat{\xi}, \hat{\sigma}), C),\ i = 1, \cdots, N.$$

Step 3. Using the formulae obtained in part **(a)** for VaR_α and ES_α, by plugging in $(\hat{\xi}_i^*, \hat{\sigma}_i^*)$ for $(\hat{\xi}, \hat{\sigma})$, generate $N = 2\,000$ samples of $VaR_{\alpha,i}$ and $ES_{\alpha,i}$, $i = 1, \cdots, N$. Compute probability-symmetric empirical 95% confidence intervals for VaR_α and ES_α from these samples.

Note: If $\hat{\sigma}_i^*$ is simulated as negative, you will have to drop this sample.

Hint to part (b):

```
get_par_boostrap_ci_VaR <- function(x,MC.iter=2000,p=0.99,alpha,p.lower=0.025,
                                     p.upper=0.975,qfactor=1.2){
  res <- get_VaR(data=x, p=p, alpha)
  u = quantile(x,p); names(u)="";
  pu = mean(x>=u);
  theta.star =
    matrix(c(res$xi,res$sig),2,1)%*%matrix(1,1,MC.iter) +
    qfactor*mat.power(res$Cov,1/2)%*% matrix(rnorm(2*MC.iter),2,MC.iter);
  xi.star = theta.star[1,]
  sig.star = theta.star[2,]
  #dropping negative sigma.star variates's
  xi.star=xi.star[sig.star>0]
  sig.star=sig.star[sig.star>0]
  ci = c();
  for (ia in c(1:length(alpha))) {
    a = alpha[ia]
    VaR.a = ((a/pu)^(-xi.star) -1)*sig.star/xi.star+u
     ci = cbind(ci, quantile(VaR.a, c(p.lower,p.upper)));
  }

  rownames(ci)<- c("Lower","Upper");
  colnames(ci) <- alpha;
  return(ci)
}

get_par_boostrap_ci_ES <- function(x,MC.iter=2000,p=0.99,alpha,p.lower=0.025,
                                    p.upper=0.975,qfactor=1.2){
  res <- get_VaR(data=x, p=p, alpha)
  u = quantile(x,p); names(u)="";
  pu = mean(x>=u);
  theta.star =
    matrix(c(res$xi,res$sig),2,1)%*%matrix(1,1,MC.iter) +
    qfactor*mat.power(res$Cov,1/2)%*% matrix(rnorm(2*MC.iter),2,MC.iter);
  xi.star = theta.star[1,]
  sig.star = theta.star[2,]
  #dropping negative sigma.star variates's
  xi.star=xi.star[sig.star>0]
  sig.star=sig.star[sig.star>0]
  ci = c();
  for (ia in c(1:length(alpha))) {
    a = alpha[ia]
    VaR.a = ((a/pu)^(-xi.star) -1)*sig.star/xi.star+u
    ES.a = sig.star/(1-xi.star) + u + (VaR.a-u)/(1-xi.star)
     ci = cbind(ci, quantile(ES.a, c(p.lower,p.upper)));
  }

  rownames(ci)<- c("Lower","Upper");
  colnames(ci) <- alpha;
  return(ci)
}
```

(c) The goal of this part is to check the coverage of the parametric-bootstrap based confidence intervals obtained in part **(b)**.

Step 1. Consider t-distribution model with $\nu = 3$ degrees of freedom. Using the Monte Carlo method, simulate a very large sample from the t-model and compute the true values of VaR_α and ES_α using empirical quantiles and averages, respectively for $\alpha = 10^{-4}, 10^{-5}, 10^{-6}$.

Step 2. Now, simulate $n = 10^4$ points from this model. Using $p = 0.95$, obtain the GPD fit and the resulting parametric-bootstrap 95% confidence intervals obtained in **(b)** for VaR_α and ES_α, $\alpha = 10^{-4}, 10^{-5}, 10^{-6}$.

Step 3. Replicate **Step 2.** independently, say 500 times and obtain the empirical coverages of the "true" values of VaR_α and ES_α from **Step 1.** Report these coverages in a table and comment.

Hint to part (c):

```
require(knitr)
library(statmod)

## Warning:  package 'statmod' was built under R version 4.3.3

alpha = c(1e-4,1e-5,1e-6)
VaR.true = - qt(alpha,df=3)
coverages = 0;
N.ci.replicates = 500
for (i in c(1:N.ci.replicates)){
  x = rt(n=1e4,df=3);
  ci = get_par_boostrap_ci_VaR(x,p=0.95,alpha=alpha,MC.iter = 2000);
  coverages = coverages +
    ( VaR.true<= ci[2,])*(ci[1,]<= VaR.true);
}

coverages=rbind(alpha,coverages/N.ci.replicates)
rownames(coverages)<-c("alpha","emp coverage")
kable(coverages)
```

	1e-04	1e-05	1e-06
alpha	0.0001	0.00001	0.000001
emp coverage	0.9600	0.95000	0.934000

```
#
# Do something similar with Expected Shortfall, where the "true" values can
# be computed using Monte Carlo.
#
coverages = 0

true_ES_integral = function(x, alpha, df) {
  sapply(alpha, function(a){
      VaR_true = - qt(a,df=df)
      integrate_res = integrate(function(x) x*dt(x, df=df),
                               lower = VaR_true,
                               upper = Inf)$value
      return(integrate_res/a)
})}

for (i in c(1:N.ci.replicates)){
  x = rt(n=1e4,df=3);
  ES.true = true_ES_integral(x, alpha, df=3)
  ci = get_par_boostrap_ci_ES(x,p=0.95,alpha=alpha,MC.iter = 2000);
```

```r
  coverages = coverages +
    ( ES.true<= ci[2,])*(ci[1,]<= ES.true);
}

coverages=rbind(alpha,coverages/N.ci.replicates)
rownames(coverages)<-c("alpha","emp coverage")
kable(coverages)
```

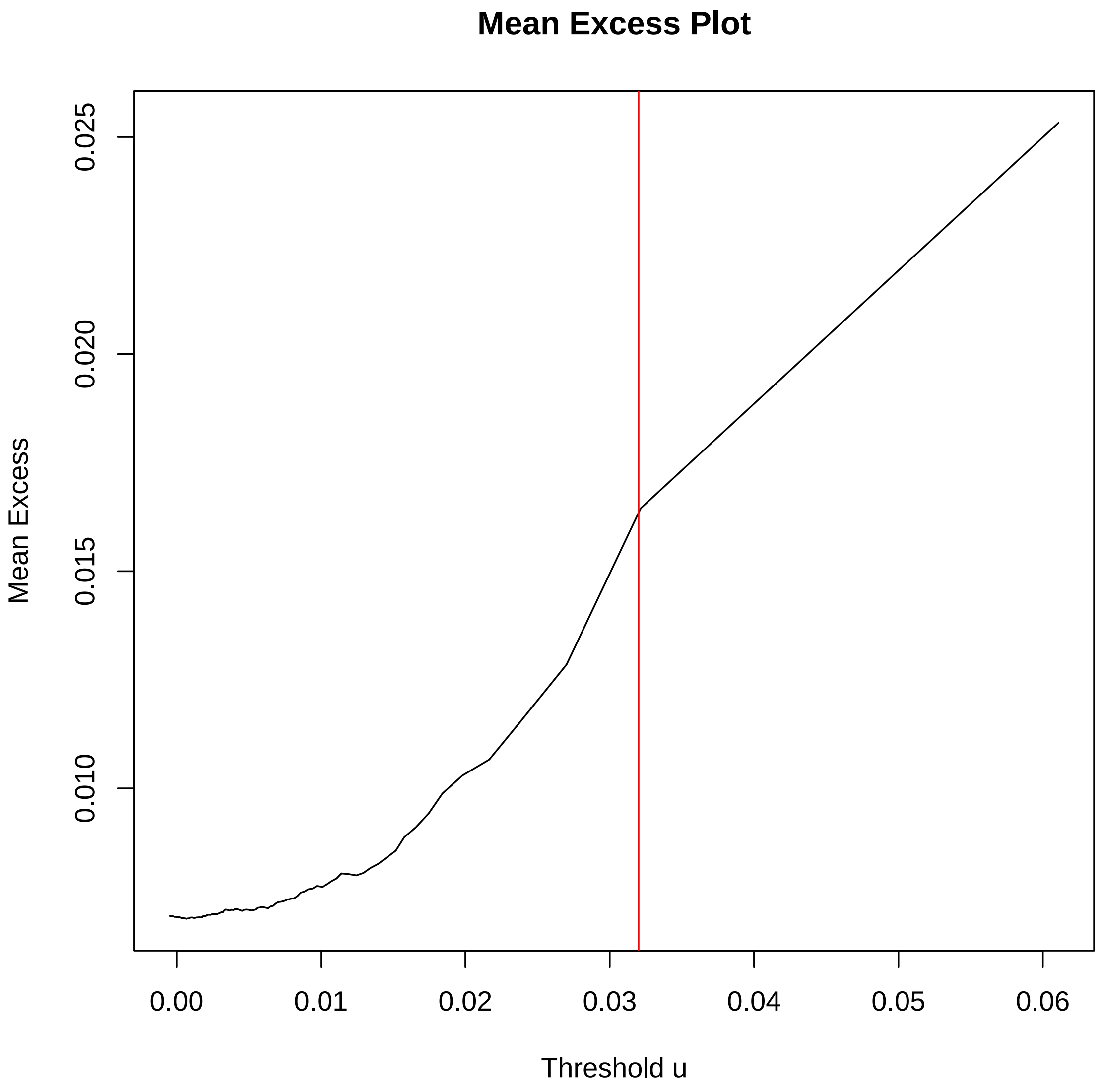
	1e-04	1e-05	1e-06
alpha	0.0001	0.00001	0.000001
emp coverage	0.9540	0.94000	0.932000

As α decreases (i.e., as we estimate more extreme quantiles), the empirical coverage drops slightly from 95.4% at $\alpha = 10^{-4}$ to 93.2% at $\alpha = 10^{-6}$. This suggests that for extremely rare events, the bootstrap method may slightly underestimate uncertainty, potentially due to finite sample effects or limitations in GPD approximation.

3. **The goal of this exercise is to apply the methodology in the previous two exercises to financial data.** Use the SP500 time series for the period 1957/01/02 through 2023/12/29 (just drop the missing values)

   (a) Load the sp500 time series and focus on the **losses** (negative daily returns). Examine the mean-excess plot to determine a suitable threshold u_0, above which the excesses are likely to follow a GPD model. Proved the plots and comment.

```r
data = read.csv("sp500_full_1957_2023.csv", header=TRUE)
data$caldt <- as.Date(as.character(data$caldt), format = "%Y%m%d")
data <- subset(data,
caldt >= as.Date("1957-01-02") & caldt <= as.Date("2023-12-29"))
data <- na.omit(data)

losses = -data$sprtrn

pu = seq(from=0.5,to=0.999,length.out=100);

u = quantile(losses, pu)
me = mean.excess(losses, u);
plot(u, me, type="l",main="Mean Excess Plot",
     xlab="Threshold u",ylab="Mean Excess");
abline(v=0.032, col="red")
```



(b) Fit the GPD model and produce plots of the point estimates and 95% confidence intervals for ξ over a range of thresholds.

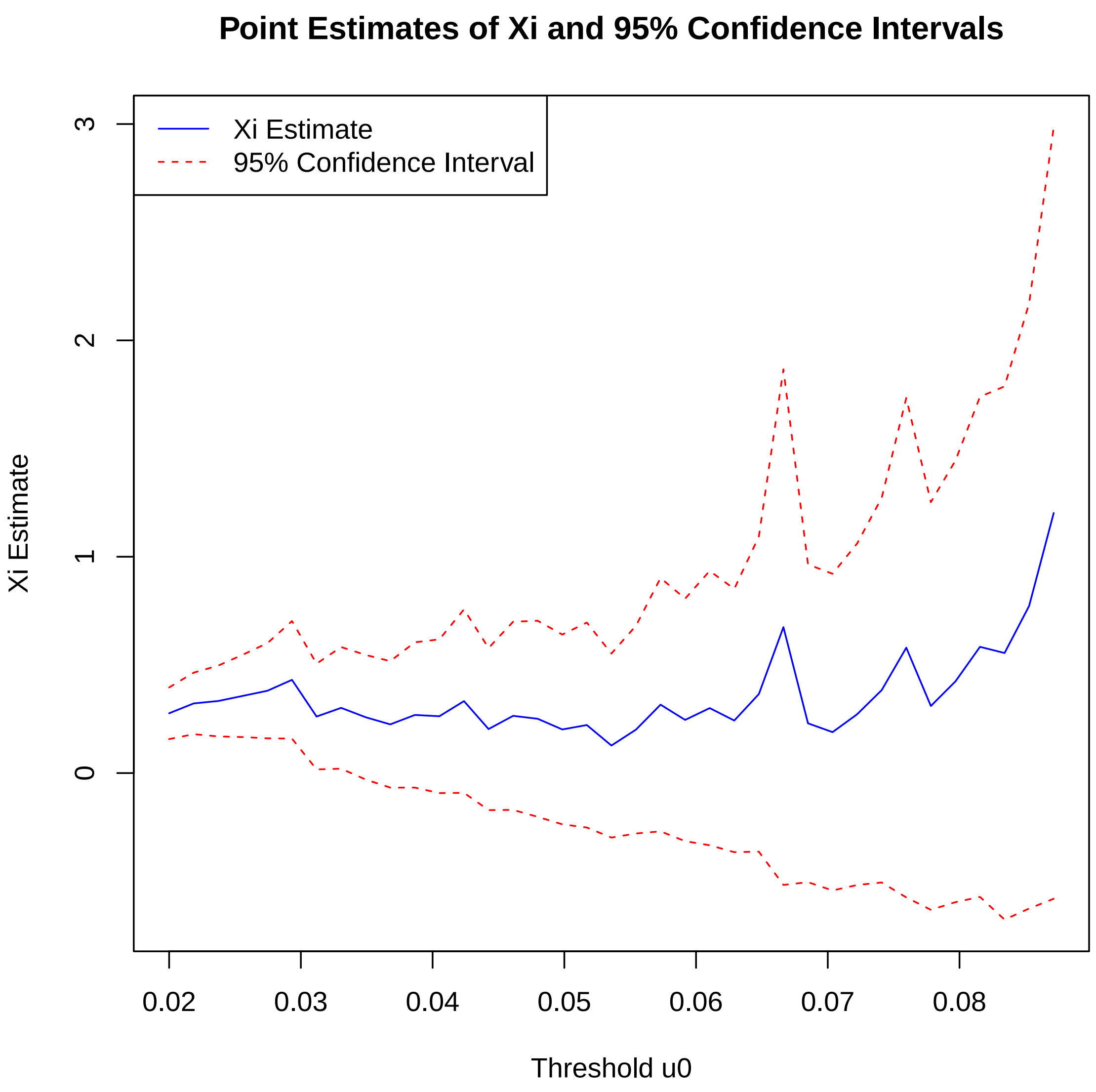
```
 library(knitr)


# setting range of thresholds
u0_values <- seq(0.02, max(losses), length.out = 100)

xi_estimates <- c()
lower_bounds <- c()
upper_bounds <- c()
```

```
mu = c()

for (u0 in u0_values) {
 res = fit.gpd.Newton_Raphson(losses, threshold = u0)
 if (!is.na(res$se.xi)){
   mu <- c(mu, u0)
   xi_estimates <- c(xi_estimates, res$xi)
   lower_bounds <- c(lower_bounds, res$xi - 1.96 * res$se.xi)
   upper_bounds <- c(upper_bounds, res$xi + 1.96 * res$se.xi)
 }
}

## Error in solve.default(get_gpd_Hessian(xi = theta[1], sig = theta[2], :  system
is computationally singular:  reciprocal condition number = 3.83026e-21

results <- data.frame(Threshold = mu,
                      Xi = xi_estimates,
                      Lower = lower_bounds,
                      Upper = upper_bounds)

kable(results, digits = 4)
```

Threshold	Xi	Lower	Upper
0.0200	0.2764	0.1570	0.3957
0.0219	0.3219	0.1797	0.4642
0.0237	0.3331	0.1695	0.4968
0.0256	0.3568	0.1665	0.5471
0.0275	0.3805	0.1605	0.6006
0.0293	0.4309	0.1592	0.7026
0.0312	0.2613	0.0170	0.5056
0.0331	0.3014	0.0203	0.5824
0.0349	0.2581	-0.0298	0.5460
0.0368	0.2251	-0.0672	0.5174
0.0387	0.2687	-0.0671	0.6044
0.0405	0.2628	-0.0926	0.6182
0.0424	0.3326	-0.0914	0.7565
0.0442	0.2035	-0.1711	0.5781
0.0461	0.2646	-0.1703	0.6995
0.0480	0.2509	-0.2023	0.7041
0.0498	0.2017	-0.2368	0.6402
0.0517	0.2220	-0.2517	0.6956
0.0536	0.1272	-0.2981	0.5525
0.0554	0.2009	-0.2793	0.6810
0.0573	0.3162	-0.2693	0.9018
0.0592	0.2457	-0.3149	0.8063
0.0610	0.3004	-0.3337	0.9344
0.0629	0.2429	-0.3655	0.8513
0.0648	0.3651	-0.3633	1.0935
0.0666	0.6745	-0.5171	1.8662
0.0685	0.2300	-0.5044	0.9645
0.0704	0.1892	-0.5426	0.9211
0.0722	0.2719	-0.5175	1.0612
0.0741	0.3834	-0.5057	1.2725
0.0760	0.5795	-0.5747	1.7336
0.0778	0.3102	-0.6320	1.2524
0.0797	0.4236	-0.5966	1.4439
0.0816	0.5835	-0.5725	1.7395
0.0834	0.5549	-0.6774	1.7873
0.0853	0.7726	-0.6262	2.1714
0.0872	1.2020	-0.5811	2.9852

```r
plot(mu, xi_estimates, type = "l", col = "blue",
    xlab = "Threshold u0", ylab = "Xi Estimate",
    ylim = c(min(lower_bounds), max(upper_bounds)),
    main = "Point Estimates of Xi and 95% Confidence Intervals")
lines(mu, lower_bounds, col = "red", lty = 2)
lines(mu, upper_bounds, col = "red", lty = 2)
```

```
legend("topleft", legend = c("Xi Estimate", "95% Confidence Interval"),
      col = c("blue", "red"), lty = c(1, 2))
```



(c) Produce parametric-bootstrap confidence intervals for VaR_α and ES_α for $\alpha = 1/252, 1/(5*252), 1/(10*252)$, which are levels of risk corresponding to return periods of $1-$, $5-$ and 10-years.

```
alpha = c(1/252, 1/(5*252), 1/(10*252))
CI_VaR_95 = get_par_boostrap_ci_VaR(losses, p=0.95, alpha=alpha, MC.iter = 2000);
CI_ES_95 = get_par_boostrap_ci_ES(losses, p=0.95, alpha=alpha, MC.iter = 2000);
colnames(CI_VaR_95)=c("1/252","1/(5*252)","1/(10*252)")
colnames(CI_ES_95)=c("1/252","1/(5*252)","1/(10*252)")
```

```r
kable(CI_VaR_95)
```

	1/252	1/(5*252)	1/(10*252)
Lower	0.0355188	0.0548944	0.0649026
Upper	0.0412169	0.0747627	0.0961130

```r
kable(CI_ES_95)
```

	1/252	1/(5*252)	1/(10*252)
Lower	0.0478196	0.0700832	0.0818719
Upper	0.0653292	0.1198766	0.1555031

```r
# Comparison for different threshold p0
CI_VaR_90 = get_par_boostrap_ci_VaR(losses, p=0.90, alpha=alpha, MC.iter = 2000);
CI_ES_90 = get_par_boostrap_ci_ES(losses, p=0.90, alpha=alpha, MC.iter = 2000);
colnames(CI_VaR_90)=c("1/252","1/(5*252)","1/(10*252)")
colnames(CI_ES_90)=c("1/252","1/(5*252)","1/(10*252)")
kable(CI_VaR_90)
```

	1/252	1/(5*252)	1/(10*252)
Lower	0.0359867	0.0538047	0.0625280
Upper	0.0411523	0.0683630	0.0839511

```r
kable(CI_ES_90)
```

	1/252	1/(5*252)	1/(10*252)
Lower	0.0471753	0.0671628	0.0769473
Upper	0.0593839	0.0958035	0.1163749

Discuss the effect of the threshold used in the GPD inference on the parametric-bootstrap intervals. Interpret these intervals, i.e., Is it reasonable to expect that over a period of 10 years the SP500 index will see a daily drop of around 7 percent?

Estimated VaR depends on p_0 which depend on $u_0 -> p_0$ increase then estimated VaR decrease overall. And based on the table we created, we can see that under the threshold $p_0 = 0.9$, the CI covers 0.07, which mean it is reasonable to expect over a period of 10 years will see a daily drop of around 7 percent

4. **The goal of this exercise is to understand the accuracy of the vanilla bootstrap for obtaining confidence intervals for VaR_α and ES_α for non-extreme levels of $\alpha \gg 1/n$, where n is the sample size.**

   Warning: This experiment may require some time to run since the vanilla bootstrap method is computationally intensive and we need to repeat the bootstrap calculation for multiple replications of the data.

   (a) Complete/modify the following code to obtain simple R-functions for computing empirical estimates of VaR$_\alpha$ and ES_α:

```
emp.VaR <- function(data,VaR.alpha=c(0.01),losses=TRUE){
  if (losses==FALSE){
    return(-quantile(data,1-VaR.alpha))
  } else {
    return(quantile(data,1-VaR.alpha))
  }
}

emp.ES <- function(data,ES.alpha=c(0.01),losses=TRUE){
  if (losses==FALSE){data=-data}
  VaR = emp.VaR(data,VaR.alpha=ES.alpha)
  return(mean(data[data>VaR]))
}
```

   (b) Using the function `bcanon` from the package`bootstrap`, produce probability-symmetric bootstrap confidence intervals for VaR_α and ES_α at level `alpha=0.01` for a random sample of $n = 10^4$ from the t-distribution with $\nu = 3$ degrees of freedom.

   Hint for part (b).

```
library(bootstrap)
x = rt(n=1e4,df=3);
out <- bcanon(x = x,nboot = 500,emp.VaR,VaR.alpha=0.01)
Lower = out$confpoints[1,2];
Upper = out$confpoints[8,2];
```

   (c) Continuing on the study from part **(b)**... Obtain the true values of VaR_α and ES_α for this model using a large Monte Carlo experiment or an exact analytical calculation. Then, simulate $N = 100$ independent samples of size $n = 10^4$ from the t-distribution with $\nu = 3$ degrees of freedom. For each of these samples, keep track of whether the bootstrap-based 95-percent confidence intervals cover the true values of VaR_α and ES_α (obtained from **(b)**). Produce a table with the empirical coverage proportions. **Discuss**.

   Hint for part (c).

```
library(bootstrap)
VaR.true = -qt(0.01,df=3)

MC.iterates = 100;
coverages_VaR = 0;
coverages_ES = 0;

for (i in c(1:MC.iterates)){
 set.seed(0220)
 x = rt(n=1e4,df=3);
 out_VaR <- bcanon(x = x, nboot = 500, emp.VaR, VaR.alpha=0.01)
 out_ES <- bcanon(x = x, nboot = 500, emp.ES, ES.alpha=0.01)

 ES.true = true_ES_integral(x, 0.01, df=3)

 Lower_VaR = out_VaR$confpoints[1,2];
 Upper_VaR = out_VaR$confpoints[8,2];

 Lower_ES = out_ES$confpoints[1,2];
 Upper_ES = out_ES$confpoints[8,2];

 coverages_VaR = coverages_VaR + (VaR.true<=Upper_VaR)*(VaR.true>=Lower_VaR);
 coverages_ES = coverages_ES + (ES.true<=Upper_ES)*(ES.true>=Lower_ES);

 cat('.')
}

## ..........................................................................

coverages_VaR/MC.iterates

## bca point
##         1

coverages_ES/MC.iterates

## bca point
##         1
```

The empirical coverage for both VaR and ES confidence intervals is 100%, meaning that in all 100 replications, the true values of VaR and ES were within their respective bootstrap-based confidence intervals. This suggests that the bootstrap method provides highly reliable confidence intervals for this specific tail risk estimation problem.