Exercises on prediction using belief functions

Thierry Denoeux

09/09/2022

Exercise 1

Question 1

To use the probability integral transform, we need to compute the inverse of the cdf:

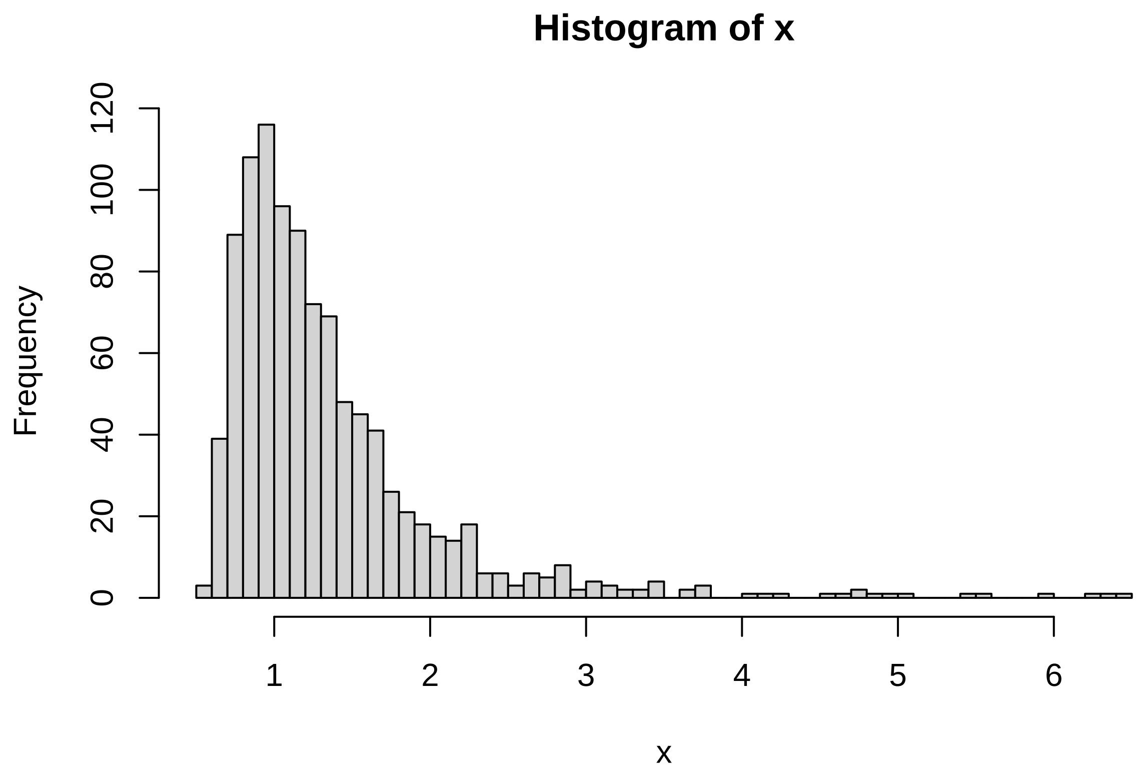
$$\exp(-X^{-\alpha}) = U \Leftrightarrow X = (-\ln U)^{-1/\alpha}.$$

We can then write the following function:

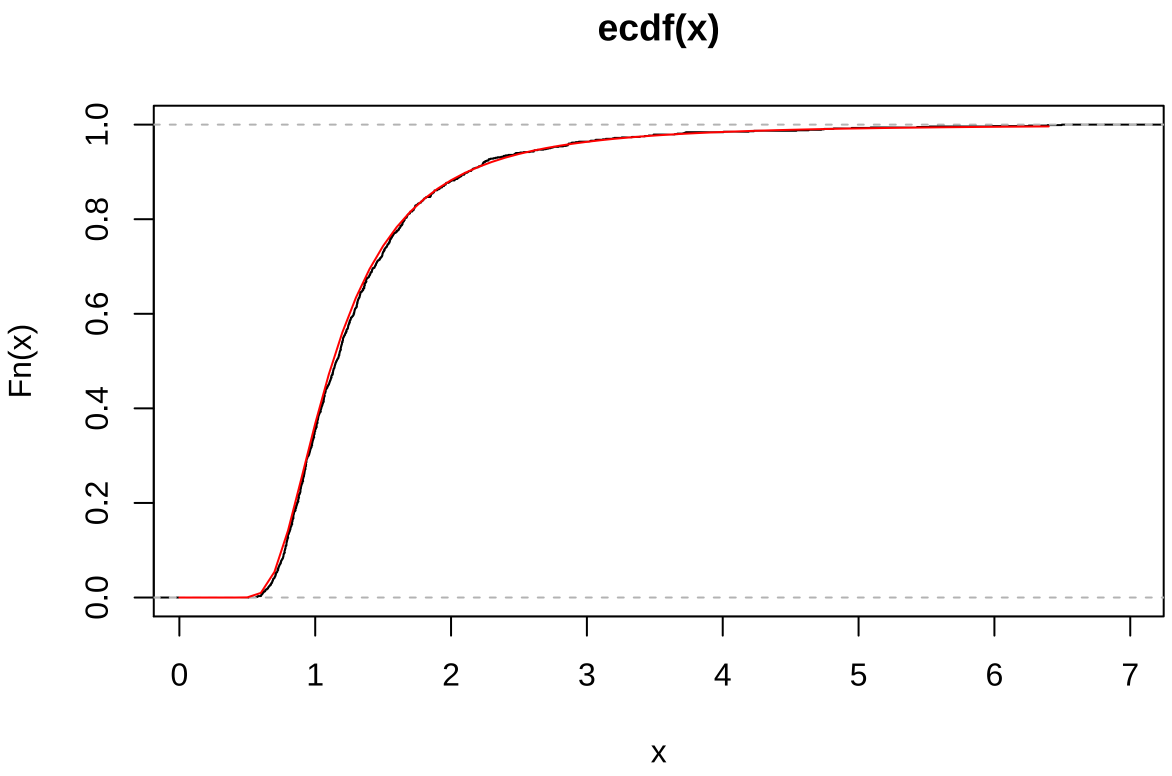
```r
rfrechet1 <- function(n,alpha) (-log(runif(n)))^(-1/alpha)
```

Let us generate a sample of size $n = 1000$ and draw the histogram as well as the empirical cdf together with the theoretical cdf:

```r
alpha<-3
x<-rfrechet1(1000,alpha)
hist(x,breaks=50)
```



```
plot(ecdf(x))
u<-seq(0,max(x),0.1)
lines(u,exp(-u^(-alpha)),col="red")
```



Question 2

The pdf is

$$f(x) = \alpha x^{-1-\alpha} \exp(-x^{-\alpha}) 1_{(0,+\infty)}(x).$$

It implementation in R is

```
dfrechet1 <- function(x,alpha) alpha*x^(-1-alpha)*exp(-x^(-alpha))
```

The likelihood function is

$$L(\alpha) = \alpha^n \left(\prod_{i=1}^{n} x_i^{-1-\alpha} \right) \exp\left(-\sum_{i=1}^{n} x_i^{-\alpha} \right)$$

and the log-liklihood is

$$\ell(\alpha) = n \ln \alpha - (1+\alpha) \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} x_i^{-\alpha}.$$

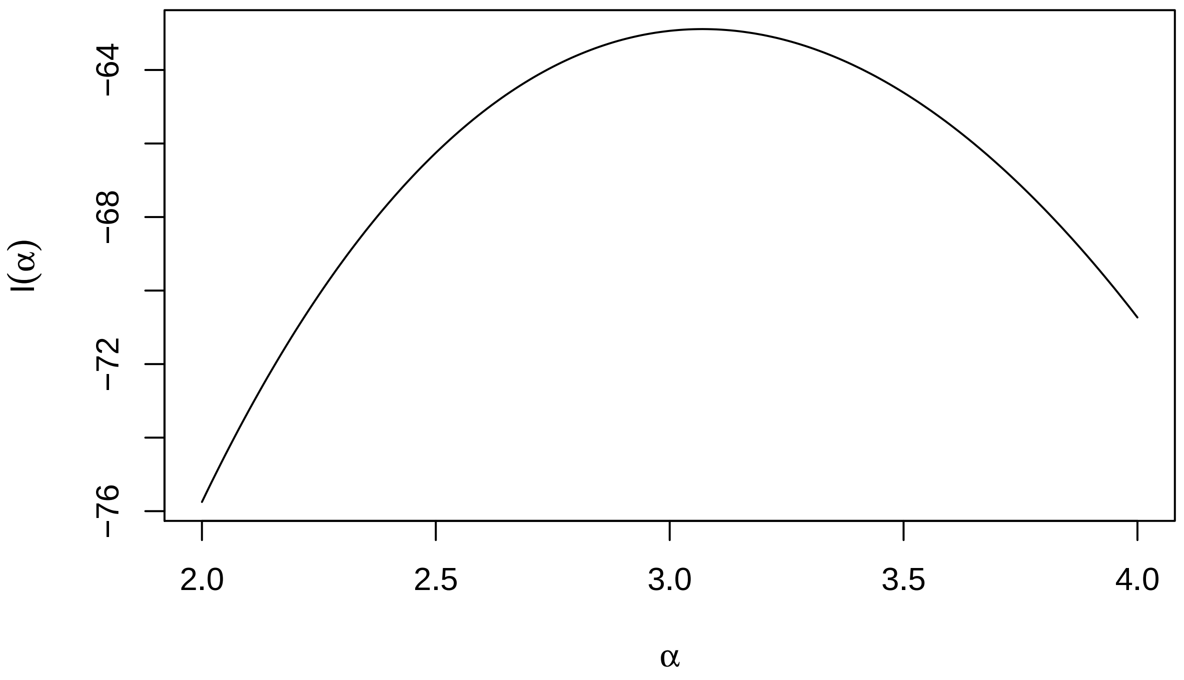
Implementation in R:

```
loglik<-function(alpha,x) sum(log(dfrechet1(x,alpha)))
```

Example:

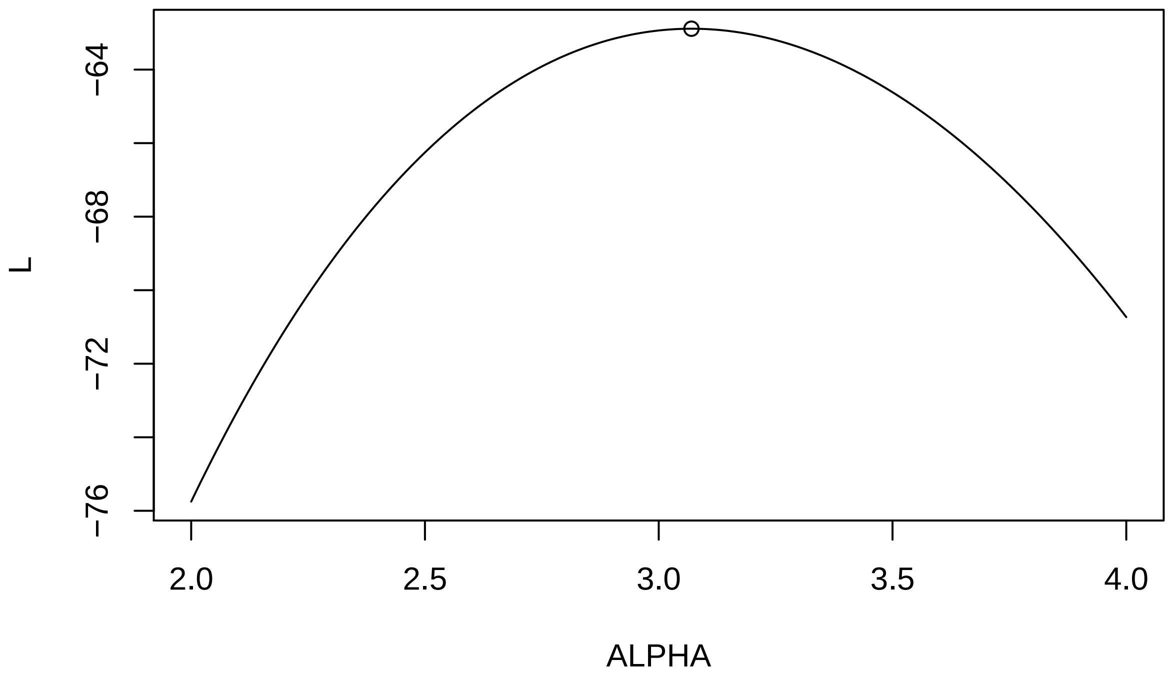
```
x<-rfrechet1(100,alpha)
ALPHA<-seq(2,4,0.01)
N<-length(ALPHA)
L<-rep(0,N)
```

```r
for(i in 1:N) L[i]<-loglik(ALPHA[i],x)
plot(ALPHA,L,type="l",xlab=expression(alpha),ylab=expression(l(alpha)))
```



To find the maximum likelihood estimate (MLE) $\widehat{\alpha}$, we need to use a numerical optimization procedure, such as function `optimize`:

```r
opt<-optimize(loglik,c(1,5),x=x,maximum = TRUE)
plot(ALPHA,L,type="l")
points(opt$maximum,opt$objective,xlab=expression(alpha),ylab=expression(l(alpha)))
```



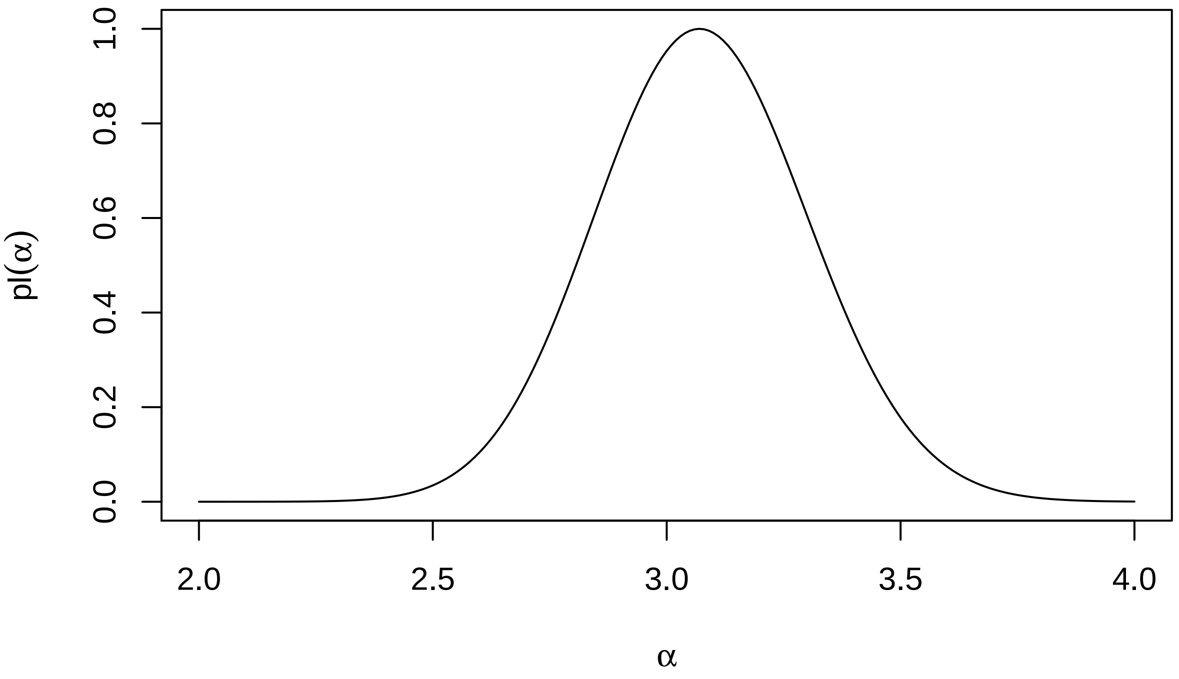
We can now write a function that computes the relative likelihood

$$pl(\alpha) = \frac{L(\alpha)}{L(\widehat{\alpha})}$$

```
rel_lik<-function(alpha,x,MLE) exp(loglik(alpha,x)-loglik(MLE,x))
```

and plot this function:

```
L<-rep(0,N)
for(i in 1:N) L[i]<-rel_lik(ALPHA[i],x,opt$maximum)
plot(ALPHA,L,type="l",xlab=expression(alpha),ylab=expression(pl(alpha)))
```



Question 3

To make predictions we will consider again the φ-equation derived in our answer to Question 1:

$$X_{n+1} = (-\ln U)^{-1/\alpha} = \varphi(\alpha, U).$$

Here, $\varphi(\alpha, U)$ is an increasing function of α if $-\ln U \geq 1$, and a decreasing function of α otherwise. Let $\Gamma(S) = [\underline{\alpha}(S), \overline{\alpha}(S)]$ be interval of values of α such that $pl(\alpha) \geq S$. For α in that interval, the range of X_{n+1} is, thus,

$$\varphi(\Gamma(S), U) = \begin{cases} [(-\ln U)^{-1/\underline{\alpha}(S)}, (-\ln U)^{-1/\overline{\alpha}(S)}] & \text{if } -\ln U \geq 1, \\ [(-\ln U)^{-1/\overline{\alpha}(S)}, (-\ln U)^{-1/\underline{\alpha}(S)}] & \text{otherwise.} \end{cases}$$

When S and U are drawn independently from standard uniform distributions, the above equation defines a predictive random set for X_{n+1}.

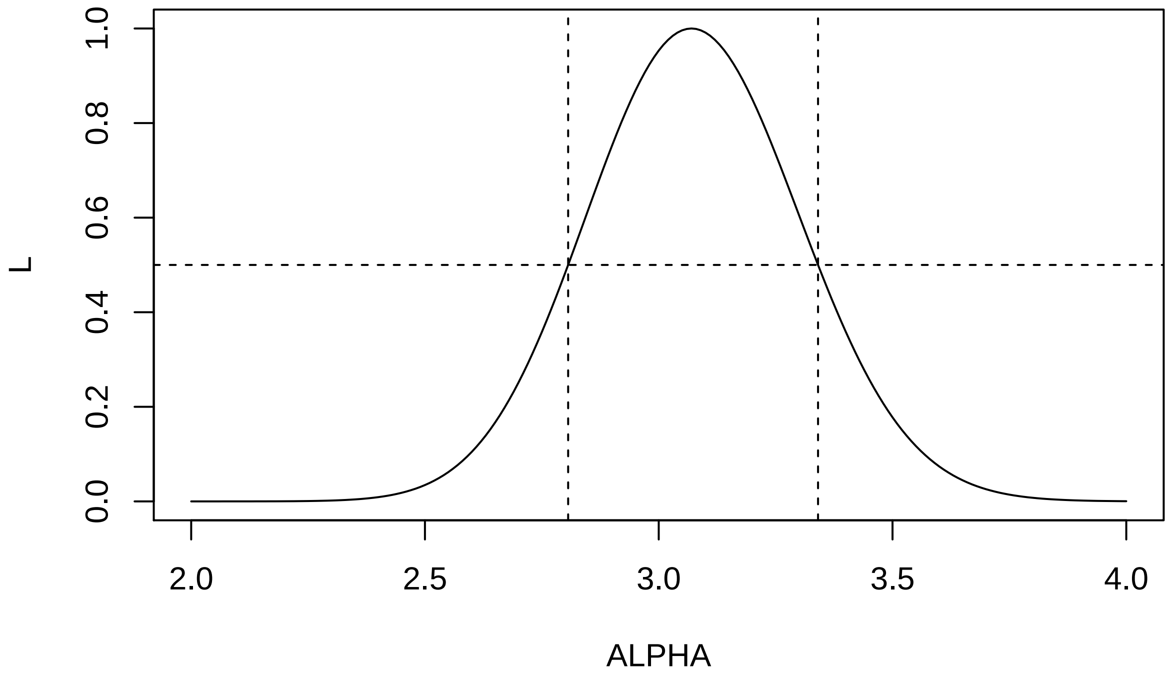
To compute $\underline{\alpha}(S)$ and $\overline{\alpha}(S)$, we write a function `bounds_alpha` that uses bthe built-in function `uniroot`:

```
bounds_alpha<-function(s,x,MLE){
  upper<-1.5*MLE
  while(rel_lik(upper,x,MLE) >s) upper<-2*upper
  lower<-0.75*MLE
```

```r
  while(rel_lik(lower,x,MLE) >s) lower<-0.5*lower
  f<-function(alpha,s,x,MLE) rel_lik(alpha,x,MLE)-s
  sol1<-uniroot(f,c(lower,MLE),s=s,x=x,MLE=MLE)
  sol2<-uniroot(f,c(MLE,upper),s=s,x=x,MLE=MLE)
  return(list(min=sol1$root,max=sol2$root))
}
```

Let us check this function with an example:

```r
plot(ALPHA,L,type="l")
sol<-bounds_alpha(s=0.5,x,MLE=opt$maximum)
abline(h=0.5,lty=2)
abline(v=sol$min,lty=2)
abline(v=sol$max,lty=2)
```

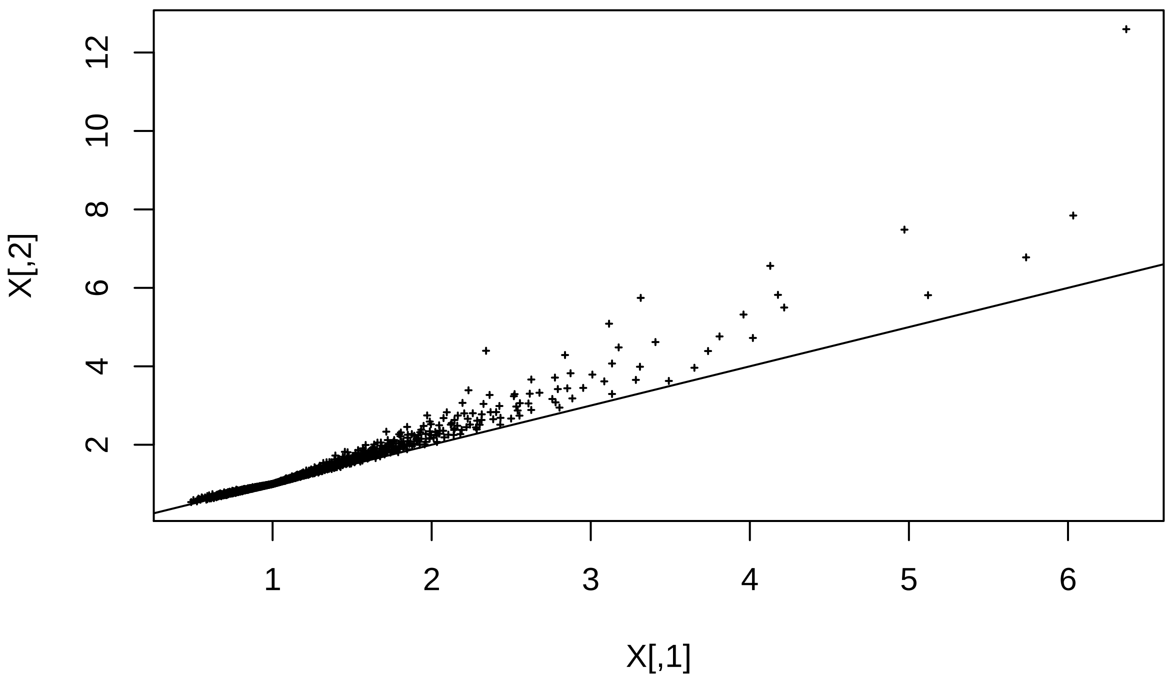


We can now generate $N = 1000$ draws from the predictive random set on X_{n+1}. We will use a Halton sequence to simulate random draws from U and S:

```r
library('randtoolbox')
N<-1000
SU<-halton(N,2)
X<-matrix(0,N,2)
for(i in 1:N){
  sol<-bounds_alpha(SU[i,1],x,MLE=opt$maximum)
  z<--log(SU[i,2])
  if(z>=1) X[i,]<-z^c(-1/sol$min,-1/sol$max) else
    X[i,]<-z^c(-1/sol$max,-1/sol$min)
}
```

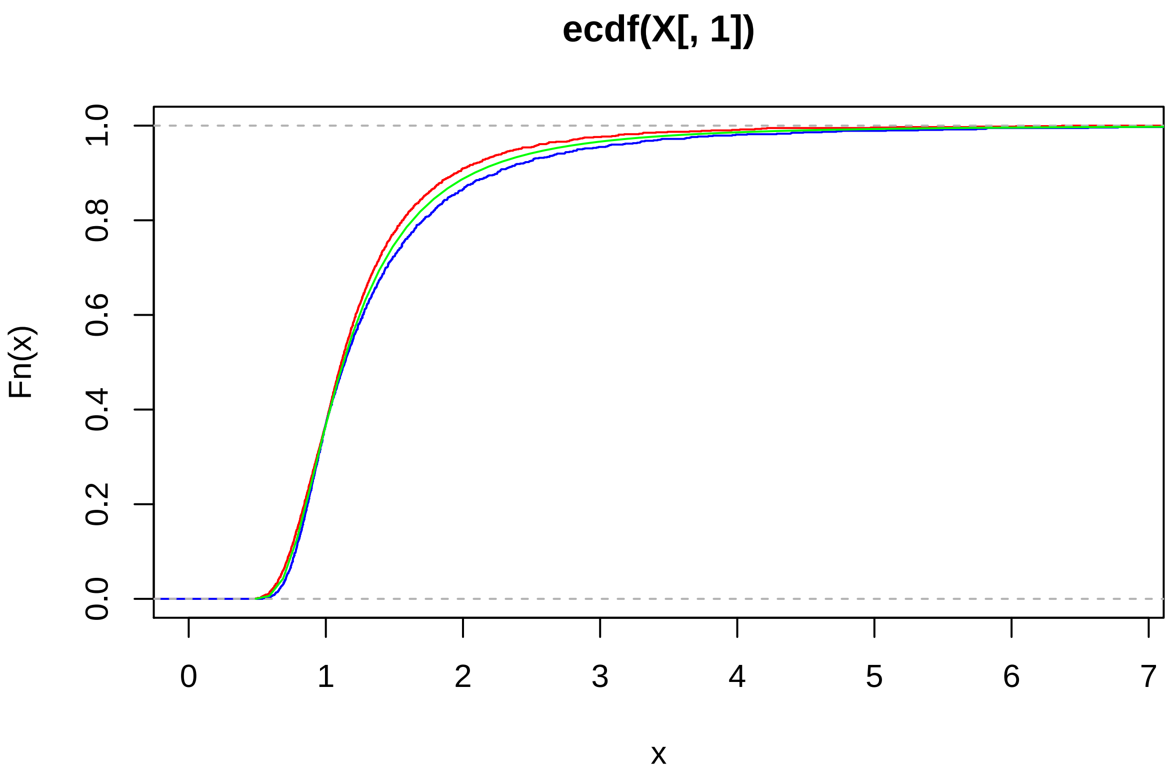
Let us draw the intervals:

```r
plot(X,pch=3,cex=0.3)
abline(0,1)
```



The lower and upper cdfs of the predictive belief function on X can be estimated, respectively, by the empirical cdfs of the upper and lower bounds of the predictive random interval. Let us plot these two functions, together with the plug-in cdf:

```
plot(ecdf(X[,1]),verticals = FALSE,col="red")
plot(ecdf(X[,2]),verticals = FALSE,add=TRUE,col="blue")
u<-seq(min(X[,1]),max(X[,2]),0.1)
lines(u,exp(-u^(-opt$maximum)),col="green")
```



We can now write the function that will compute, more generally, the belief and plausibility of any interval $[a, b]$:

```
belpl<-function(a,b,X){
  bel<-mean((X[,1]>= a) & (X[,2]<= b))
  pl<-mean((X[,2]>= a) & (X[,1]<= b))
  return(list(bel=bel,pl=pl))
}
```

For instance, we have

```
print(belpl(1,2,X))
```

```
## $bel
## [1] 0.496
##
## $pl
## [1] 0.54
```

```
print(belpl(1,3,X))
```

```
## $bel
## [1] 0.585
##
## $pl
## [1] 0.606
```

Exercise 2

We will now use package `VGAM`, which contains functions related to the Fréchet distribution. To guarantee the conditions $\alpha > 0$ and $\sigma > 0$ when maximizing the log-likelihood, we will use the following alternative parameterization: $\alpha = \beta^2$, $\sigma = \xi^2$, and $\theta = (\beta, \xi)$. The vector of original parameters will be denoted by $\omega = (\alpha, \sigma)$. The log-likelihood is then easily computed as

```
library(VGAM)
loglik2<-function(theta,x){
  sum(log(dfrechet(x,shape=theta[1]^2,scale=theta[2]^2)))
}
```

Let us generate a sample of size $n = 100$ from the Fréchet distribution with parameters $\alpha = 3$ and $\sigma = 2$ using the built-in function `rfrechet`:

```
n<-100
x<-rfrechet(n,scale=2,shape=3)
```

To find the MLE $\widehat{\omega}$, we will use the BFGS algorithm implemented in function `optim`:

```
opt<-optim(c(1,1),loglik2,x=x,method="BFGS", control=list(fnscale=-1,trace=3))
```

```
## initial  value 231.540615
## iter  10 value 152.372962
## iter  10 value 152.372962
## iter  10 value 152.372962
## final  value 152.372962
## converged
```

```
omegah<-opt$par[1:2]^2
print(omegah)
```

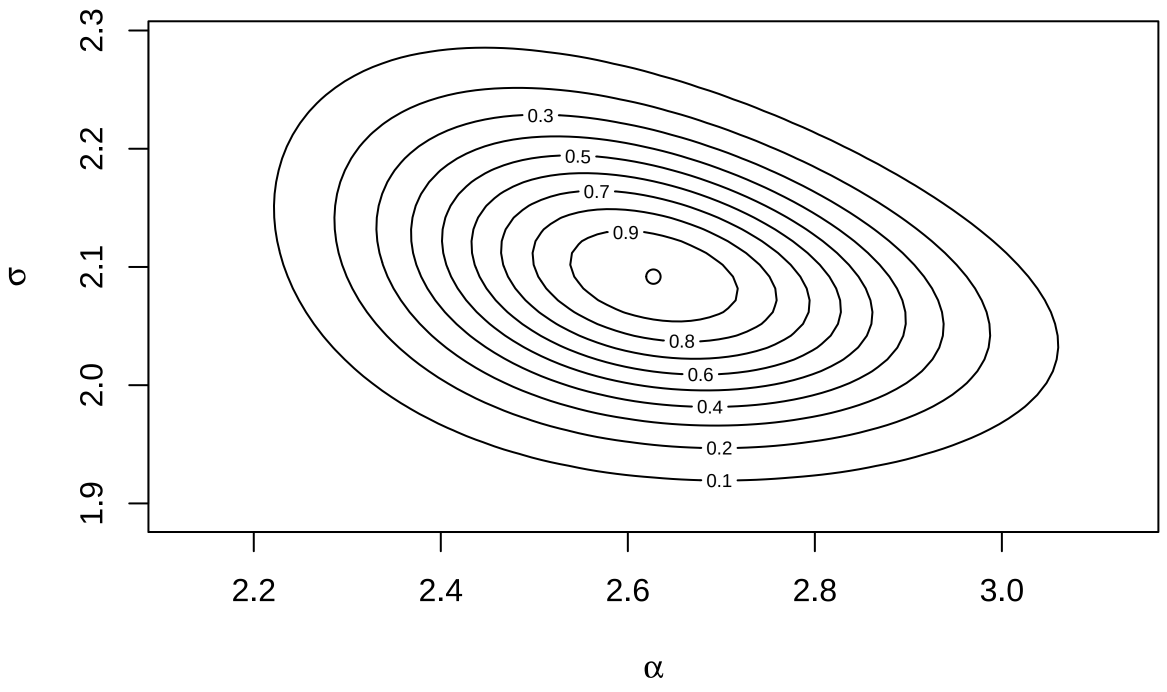
```
## [1] 2.627359 2.091808
```

We can now write a function for computing the relative likelihood:

```
rel_lik2<-function(omega,x,omegah){
  theta<-sqrt(abs(omega[1:2]))
  thetah<-sqrt((omegah[1:2]))
  return(exp(loglik2(theta,x)-loglik2(thetah,x)))
}
```

We can plot the contours of this relative log-likelihood function:

```
ALPHA<-seq(omegah[1]-0.5,omegah[1]+0.5,0.01)
SIG<-seq(omegah[2]-0.2,omegah[2]+0.2,0.01)
N1<-length(ALPHA)
N2<-length(SIG)
L<-matrix(0,N1,N2)
for(i in 1:N1){
  for(j in 1:N2){
    L[i,j]<-rel_lik2(c(ALPHA[i],SIG[j]),x,omegah)
  }
}
contour(ALPHA,SIG,L,level=seq(0.1,0.9,0.1),xlab=expression(alpha),ylab=expression(sigma))
points(omegah[1],omegah[2])
```



For prediction, we first write the φ-equation as

$$X_{n+1} = \varphi(\omega, U) = F_\omega^{-1}(U) = \sigma(-\ln U)^{-1/\alpha},$$

where U has a standard uniform distribution in $[0,1]$. To compute the intervals $\varphi(\Gamma(s), u)$, we solve the following optimization problems:

$$\min_\omega \varphi(\omega, u) \quad \text{and} \quad \max_\omega \varphi(\omega, u)$$

such that $pl(\omega;x) \geq s$, for each pair (s,u) drawn from the uniform distribution in $[0,1]^2$.

For that purpose, we use function `constrOptim.nl` in package `alabama`, which solves constrained nonlinear optimization problems. We function write R functions for the $\varphi(\omega,u)$ and the inequality constraints ($pl(\omega;x) \geq s$, $\alpha > 0$ and $\sigma > 0$):

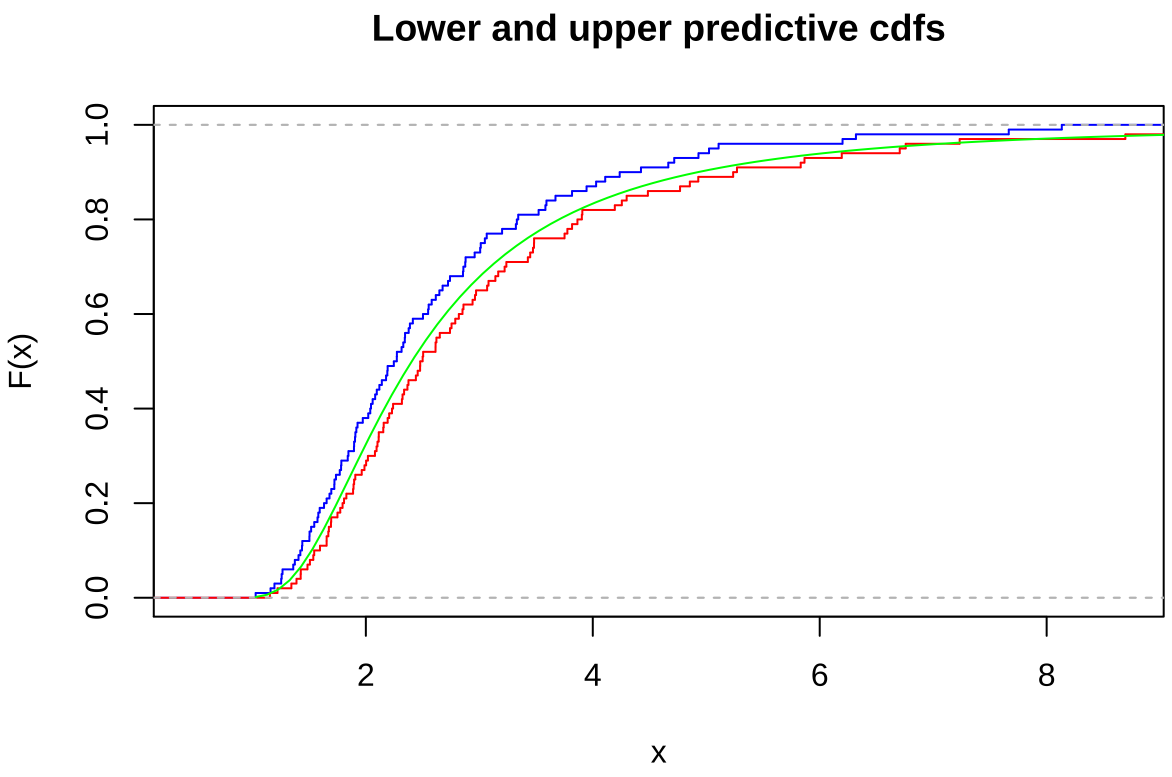
```
library(alabama)
fun_pred<- function(par,X,omegah,S,u){
  return(par[2]*(-log(u))^(-1/par[1]))
}
logpl_cstr <- function(par,X,omegah,S,u){
  pl<-rel_lik2(par,X,omegah)
  return(c(pl-S,par-1e-6))
}
```

We then generate $N = 100$ focal sets $\varphi(\Gamma(s),u)$. (It would actually be better to have N equal to a few thousands, but we set $N = 100$ here to limit the computing time).

```
N<-100
SU<-halton(N,2)
B<-matrix(0,N,2)
for(i in (1:N)){
  s<- SU[i,1]
  u<- SU[i,2]
  opt_min<- constrOptim.nl(par=omegah, fn=fun_pred, hin = logpl_cstr,
                           control.outer=list(trace=0),
                           control.optim=list(trace=0),
                           X=x,omegah=omegah,S=s,u=u)
  opt_max<- constrOptim.nl(par=omegah, fn=fun_pred, hin = logpl_cstr,
                           control.outer=list(trace=0),
                           control.optim=list(trace=0,fnscale=-1),
                           X=x,omegah=omegah,S=s,u=u)
  B[i,]<-c(opt_min$value,opt_max$value)
}
```

We can now plot the estimated lower and upper predictive cdfs, together with the plug-in predictive cdf $F_{\widehat{\omega}}(x)$:

```
plot(ecdf(B[,1]),col="blue",verticals=TRUE,pch="",main='Lower and upper predictive cdfs',ylab="F(x)")
plot(ecdf(B[,2]),col="red",verticals=TRUE,add=TRUE,pch="")
u<-seq(min(B[,1]),max(B[,2]),0.1)
lines(u,pfrechet(u,shape=omegah[1],scale=omegah[2]),col="green")
```



Lower and upper predictive cdfs
F(x)
0.0
0.2
0.4
0.6
0.8
1.0
2
4
6
8
x