

# Exercises on statistical inference using belief functions

Thierry Denoeux

2023-09-27

## Exercise 1

### Question 1

For a Gaussian contour function with parameters  $m$  and  $h$ , the set  $\Gamma(s)$  is the closed interval whose bounds are the solutions of the quadratic equation

$$-\frac{h}{2}(x - m)^2 = \log(s).$$

We thus have  $\Gamma(s) = m \pm \sqrt{-2\frac{\log(s)}{h}}$ . Now, the intersection of two closed intervals  $[a, b]$  and  $[c, d]$  is

$$[a, b] \cap [c, d] = \begin{cases} \emptyset & \text{if } \max(a, c) > \min(b, d) \\ [\max(a, c), \min(b, d)] & \text{otherwise.} \end{cases}$$

Finally, the degree of conflict between two random sets  $\Gamma_1$  and  $\Gamma_2$  is

$$\kappa = P(\{(s_1, s_2) \in S^2 : \Gamma_1(s_1) \cap \Gamma_2(s_2) = \emptyset\}).$$

We can now write the following function, which returns  $N$  focal sets of the combined belief function:

```
comb_Gaussian <- function(m1,h1,m2,h2,N=10^4){
  X <- matrix(0,N,2)
  n <- 0
  nc <- 0
  while(n<N){
    s1 <- runif(1)
    s2 <- runif(1)
    X1 <- c(m1-sqrt(-2*log(s1)/h1),m1+sqrt(-2*log(s1)/h1)) # Gamma(s1)
    X2 <- c(m2-sqrt(-2*log(s2)/h2),m2+sqrt(-2*log(s2)/h2)) # Gamma(s2)
    if(max(X1[1],X2[1])<= min(X1[2],X2[2])){ # The intersection is nonempty
      n <- n+1
      X[n,] <- c(max(X1[1],X2[1]), min(X1[2],X2[2])) # The intersection is a focal set
    } else nc <- nc+1
  }
  return(list(X=X,conf=nc/(n+nc)))
}
```

### Question 2

Let us compute  $10^4$  focal sets:

```

m1<-0; h1<-0.3
m2<-1; h2<-2
comb<-comb_Gaussian(m1,h1,m2,h2,N=10^4)
X<-comb$X
conf<-comb$conf

```

We compute the estimated contour function at equally spaced values in a vector  $x$ :

```

x<-seq(min(m1-2/sqrt(h1),m2-2/sqrt(h2)),max(m1+2/sqrt(h1),m2+2/sqrt(h2)),0.01)
nx<-length(x)
pl<-rep(0,nx)
for(i in 1:nx){
  pl[i]<-mean((X[,1]<=x[i])&(X[,2]<=x[i]))
}

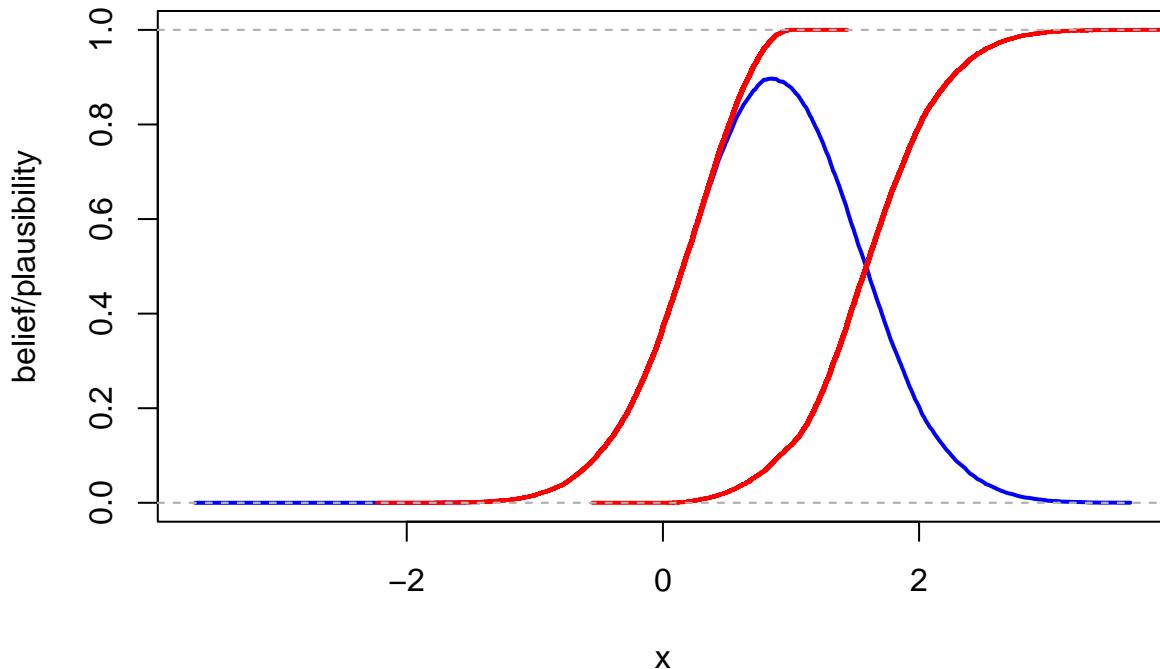
```

The estimated lower and upper cdfs can be computed as the empirical cdfs of, respectively, the upper and lower bounds of the  $N$  focal sets:

```

plot(x,pl,col="blue",type="l",lwd=2,xlab="x",ylab="belief/plausibility",ylim=c(0,1))
lines(ecdf(X[,1]),do.points=FALSE, verticals=TRUE,col="red",lwd=2)
lines(ecdf(X[,2]),do.points=FALSE, verticals=TRUE,col="red",lwd=2)

```



### Question 3

We know that the combined contour function is equal to the product of the contour functions, devided by one minus the degree of conflict:

$$pl(x) = \frac{pl_1(x)pl_2(x)}{1 - \kappa}$$

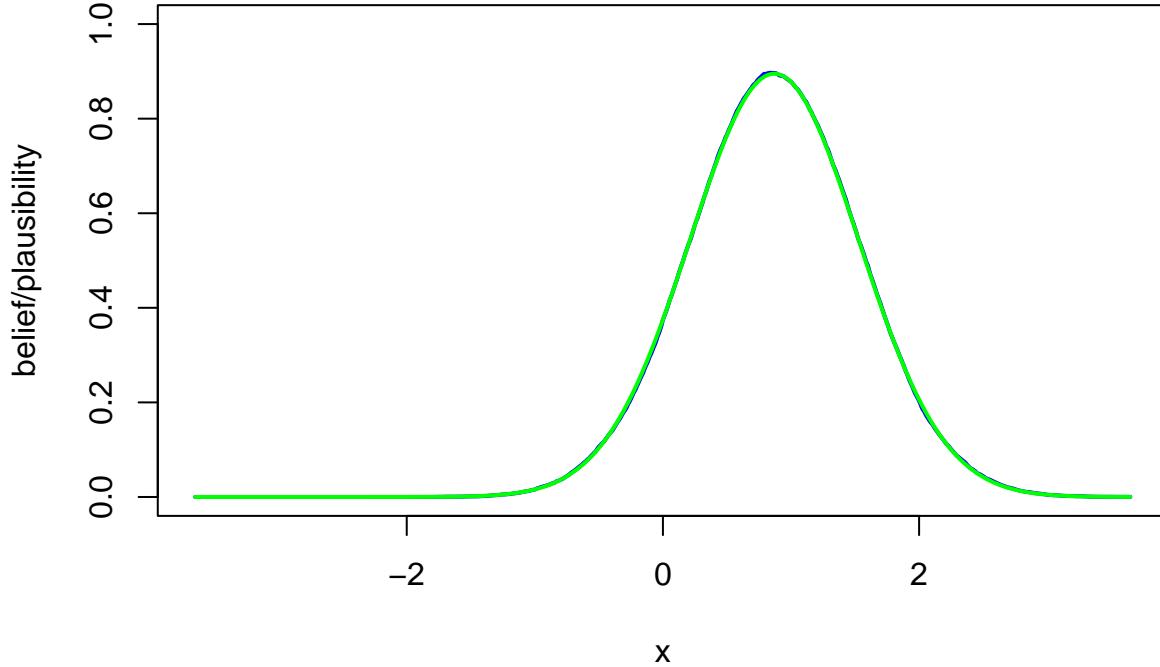
Let us verify this equality:

```

pl1<-exp(-0.5*h1*(x-m1)^2)
pl2<-exp(-0.5*h2*(x-m2)^2)

```

```
plot(x,pl,col="blue",type="l",lwd=2,xlab="x",ylab="belief/plausibility",ylim=c(0,1))
lines(x,pl1*pl2/(1-conf),lwd=2,col="green")
```



## Exercise 2

### Question 1

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

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

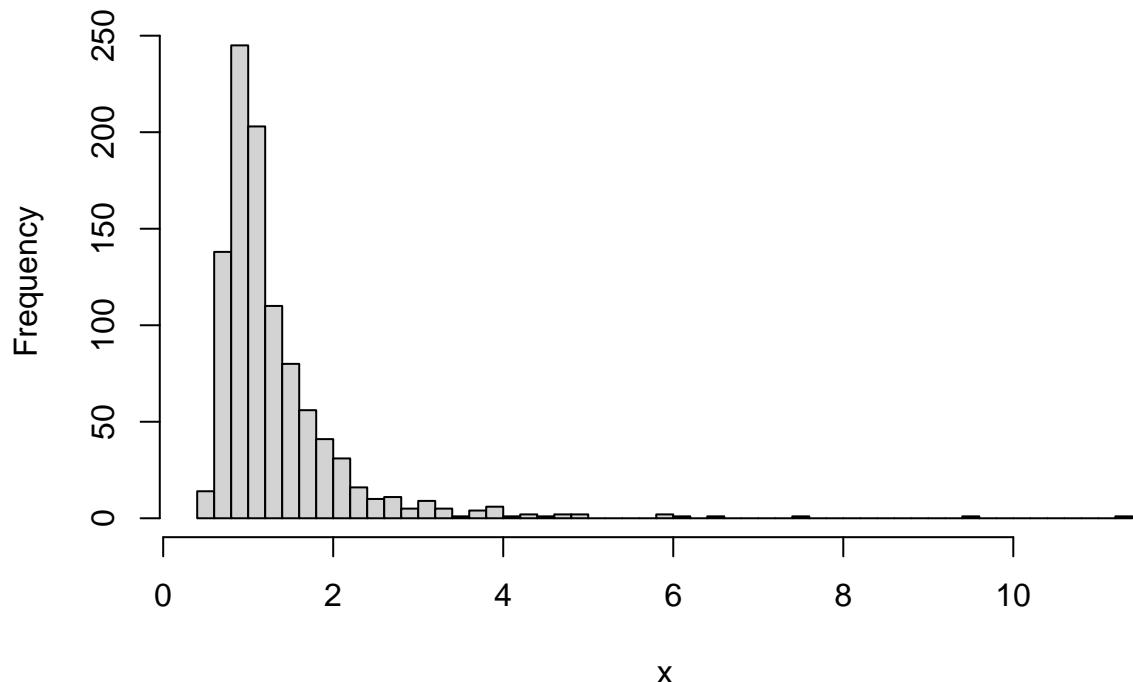
We can then write the following function:

```
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:

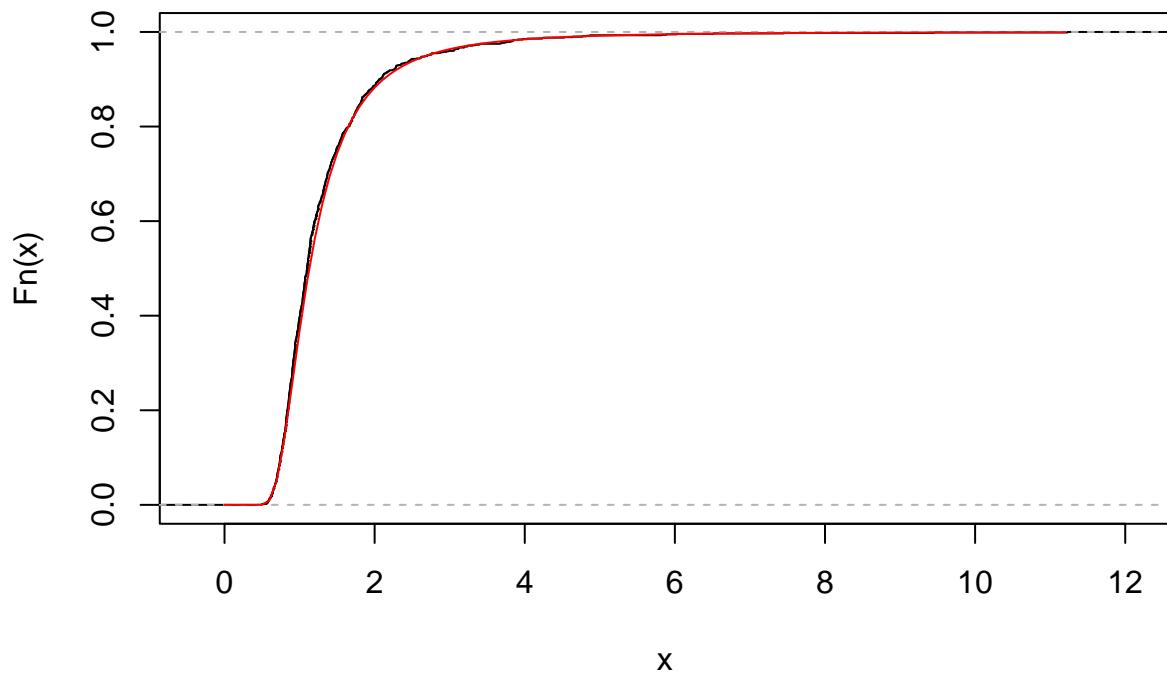
```
alpha<-3
x<-rfrechet1(1000,alpha)
hist(x,breaks=50)
```

### Histogram of x



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

### ecdf(x)



## Question 2

The pdf is

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

Its 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-likelihood is

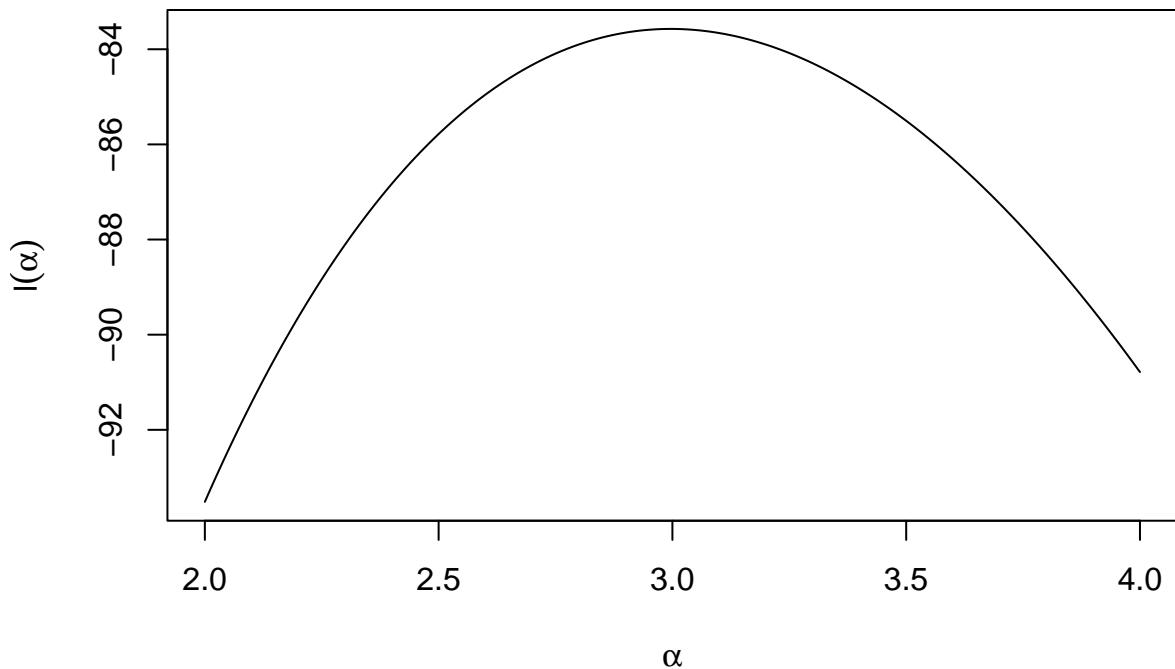
$$\ell(\alpha) = n \ln \alpha - (1 + \alpha) \sum_{i=1}^n x_i - \sum_{i=1}^n x_i^{-\alpha}.$$

Implementation in R:

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

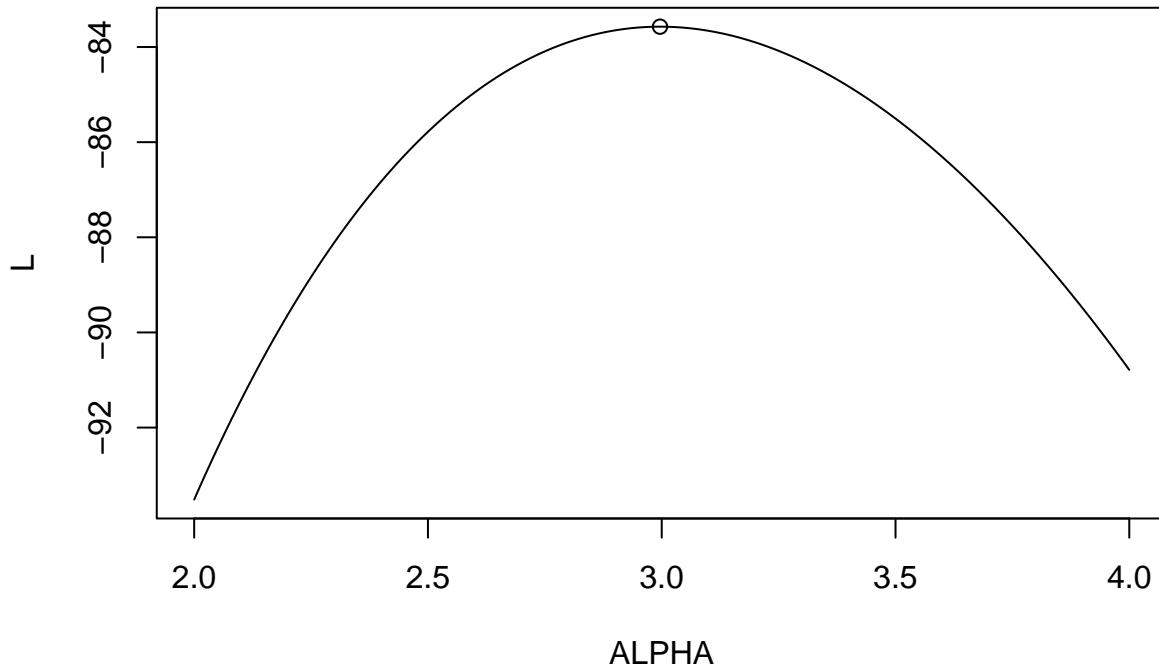
Example:

```
x<-rfrechet1(100,alpha)
ALPHA<-seq(2,4,0.01)
N<-length(ALPHA)
L<-rep(0,N)
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)  $\hat{\alpha}$ , we need to use a numerical optimization procedure, such as function `optimize`:

```
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)))
```



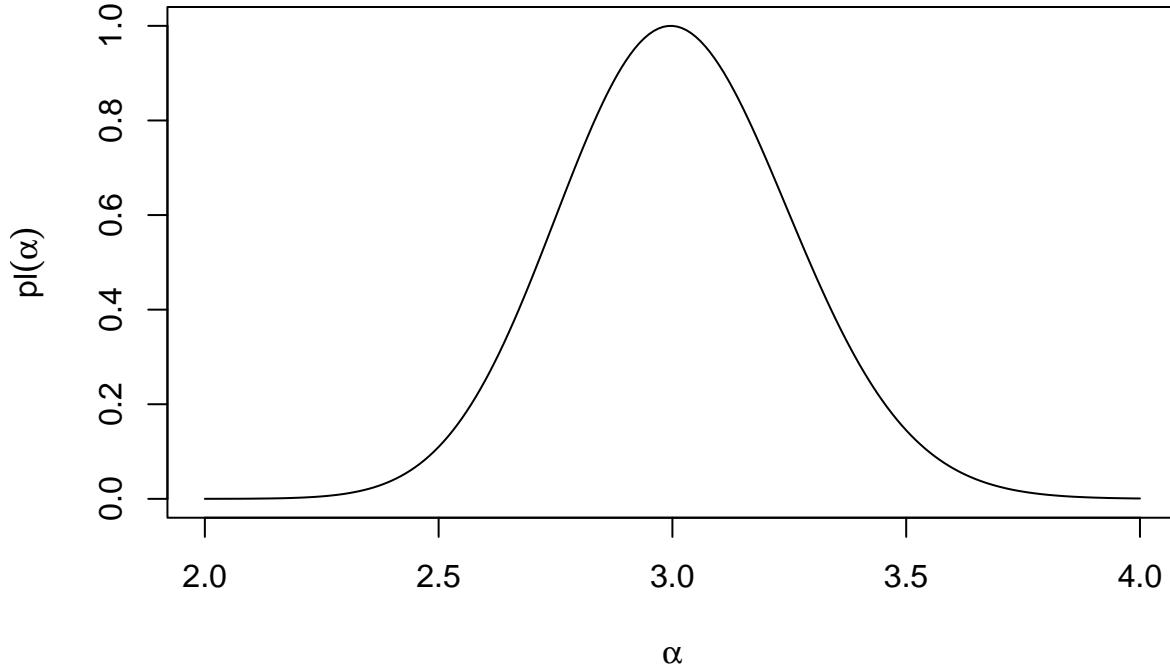
We can now write a function that computes the relative likelihood

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

```
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

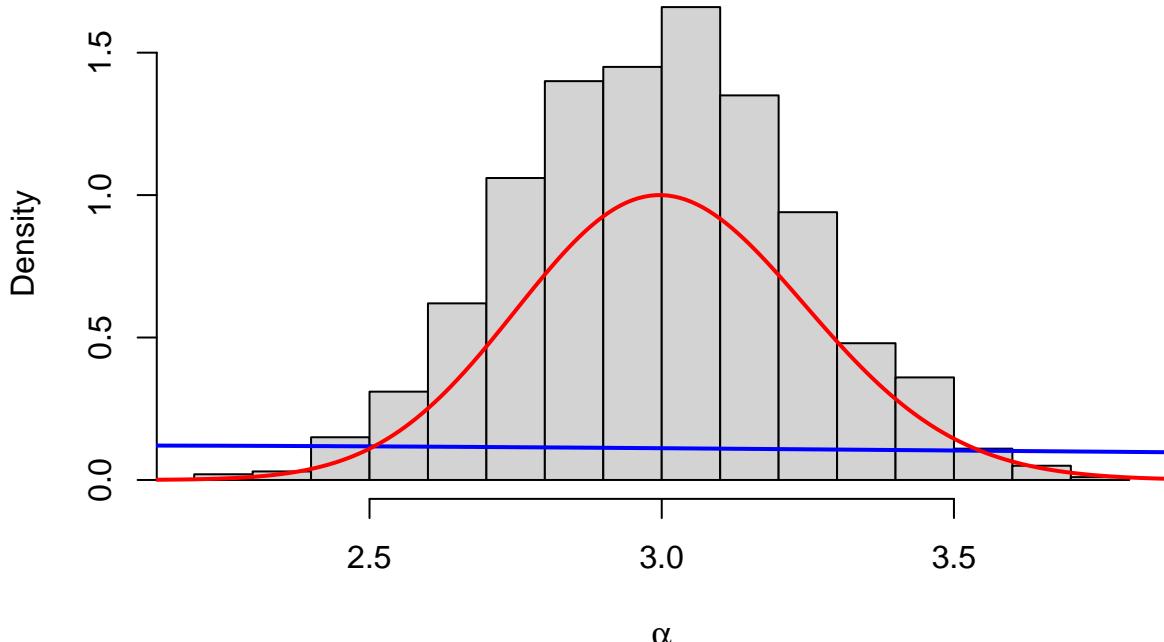
Let us write a function that generates a random sample of size  $N$  from the distribution obtained by combining the likelihood belief function with a lognormal prior  $LN(\mu, \sigma^2)$ :

```
combln <- function(mu,sig,N,x,MLE){
  A<-rep(0,N)
  i<-0
  while(i<N){
    s<-runif(1)
    alpha<-rlnorm(1,meanlog=mu,sdlog=sig)
    if(rel_lik(alpha,x,MLE) > s){
      i<-i+1
      A[i]<-alpha
    }
  }
  return(A)
}
```

Let us assume a weakly informative prior with parameters  $\sigma = 1$  and  $\mu = \log(2) + \sigma^2$  (the mode of the distribution is then  $\exp(\mu - \sigma^2) = 2$ ):

```
sig<-1
mu<- log(2)+sig^2
A <- combln(mu,sig,N=1000,x,opt$maximum)
hist(A,freq=FALSE,xlab=expression(alpha))
lines(ALPHA,dlnorm(ALPHA,mu,sig),col="blue",lwd=2)
lines(ALPHA,L,col="red",lwd=2)
```

**Histogram of A**



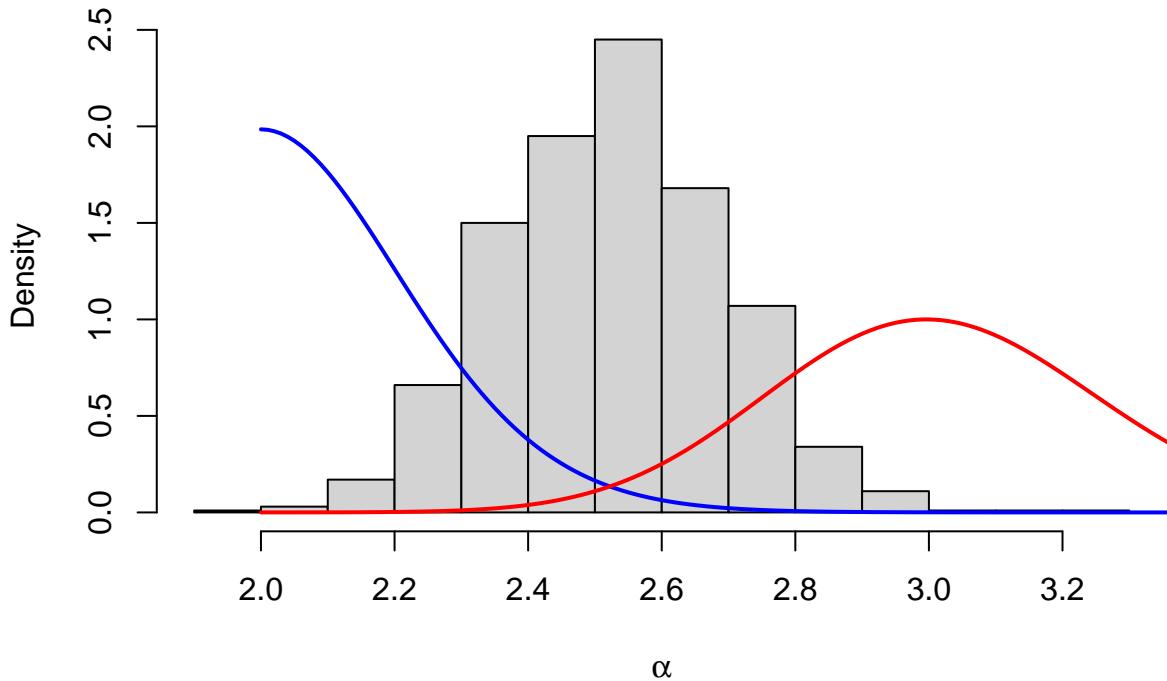
Let us now consider a more informative prior with  $\sigma = 0.1$ :

```

sig<-0.1
mu<- log(2)+sig^2
A <- combln(mu,sig,N=1000,x,opt$maximum)
hist(A,freq=FALSE,xlab=expression(alpha))
lines(ALPHA,dlnorm(ALPHA,mu,sig),col="blue",lwd=2)
lines(ALPHA,L,col="red",lwd=2)

```

Histogram of A



#### Question 4

To make predictions we will consider again the  $\varphi$ -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  $\alpha$  if  $-\ln U \geq 1$ , and a decreasing function of  $\alpha$  otherwise. Let  $\Gamma(S) = [\underline{\alpha}(S), \bar{\alpha}(S)]$  be interval of values of  $\alpha$  such that  $pl(\alpha) \geq S$ . For  $\alpha$  in that interval, the range of  $X_{n+1}$  is, thus,

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

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  $\bar{\alpha}(S)$ , we write a function `bounds_alpha` that uses the built-in function `uniroot`:

```

bounds_alpha<-function(s,x,MLE){
  upper<-1.5*MLE
  while(rel_llik(upper,x,MLE) >s) upper<-2*upper
  lower<-0.75*MLE
  while(rel_llik(lower,x,MLE) >s) lower<-0.5*lower
  f<-function(alpha,s,x,MLE) rel_llik(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))
}

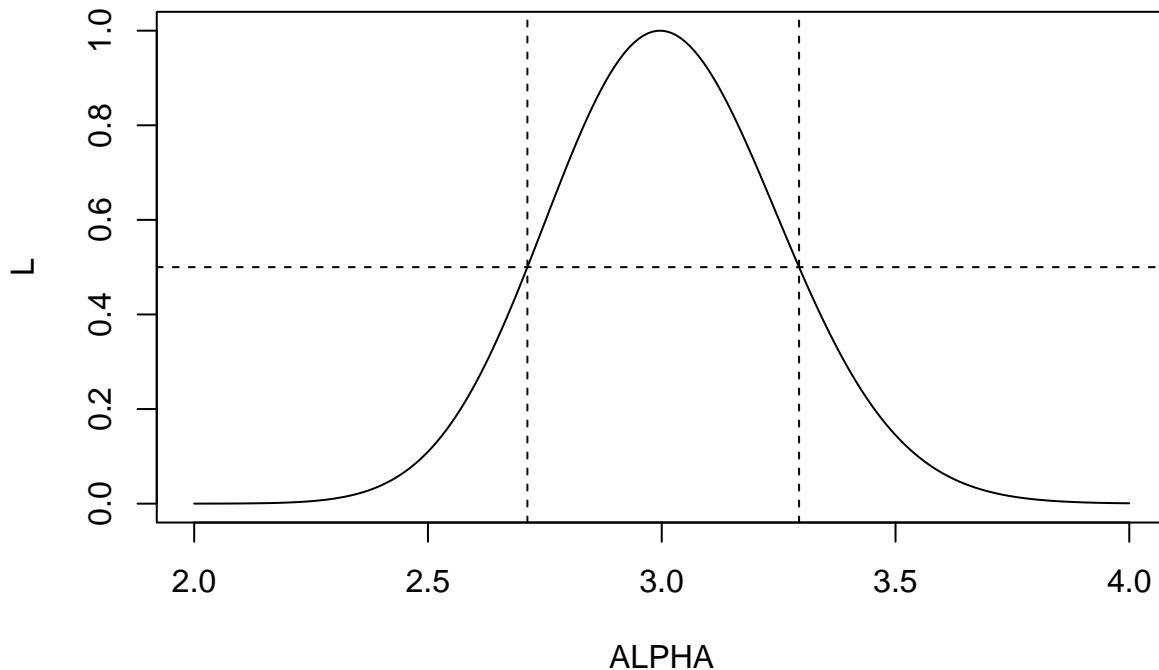
```

Let us check this function with an example:

```

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  $S$ :

```

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)
}

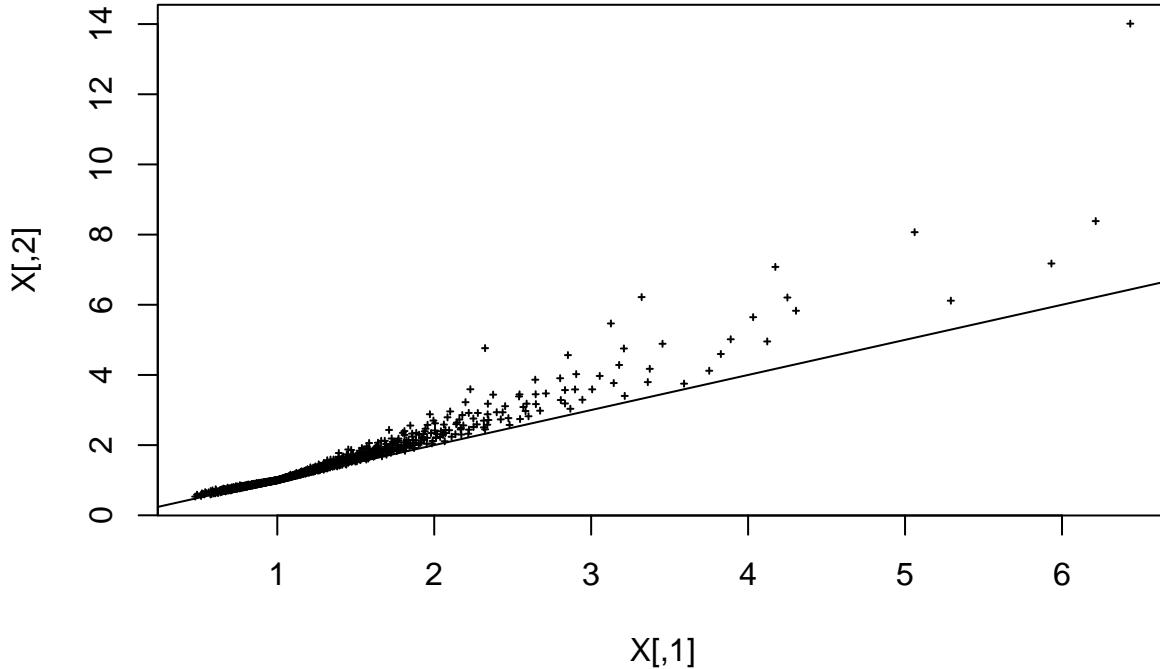
```

Let us draw the intervals:

```

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

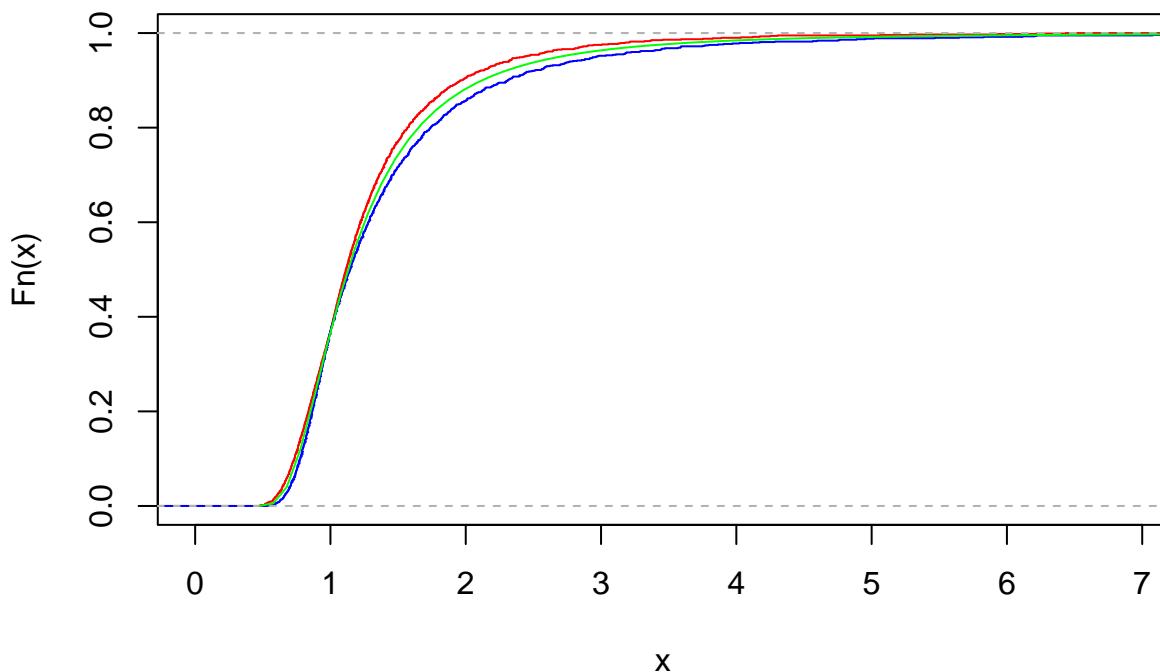
```



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")
```

### ecdf(X[, 1])



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] 0.487
##
## $pl
## [1] 0.535
print(belpl(1,3,X))

## $bel
## [1] 0.582
##
## $pl
## [1] 0.605
```

## Exercise 3

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  $\hat{\omega}$ , 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 232.942758
## iter 10 value 157.213906
## iter 10 value 157.213906
## iter 10 value 157.213906
## final value 157.213906
## converged
```

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

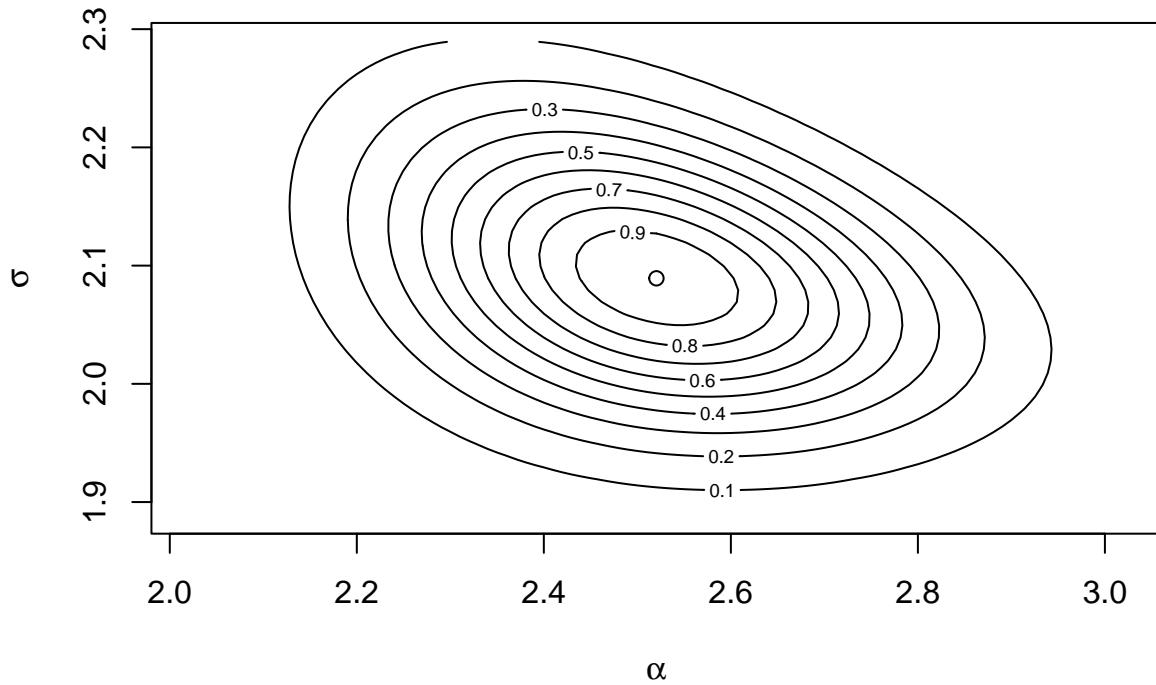
```
## [1] 2.520389 2.089253
```

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  $\varphi$ -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_liik2(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  $F_{\hat{\theta}}(x)$ :

```
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

