```
##########################################################################

# TD - Exercises in R on uncertainty propagation


# Bertrand Iooss - Fabrice Gamboa - Barranquilla - 2015


# S O L U T I O N S


##########################################################################


rm(list=ls())

graphics.off()


library(triangle)

library(mistral)


#########################

#2.1

library(triangle)


# 2 solutions for visualizing the probability densities


# with plot and density

N <- 1e6

x11()

par(mfcol=c(2,2))

F1 <- rnorm(N,3e4,9e3)
```

```
plot(density(F1))

E1 <- rtriangle(N,2.8e7,4.8e7,3e7)

plot(density(E1))

L1 <- runif(N,250,260)

plot(density(L1))

I1 <- rtriangle(N,310,450,400)

plot(density(I1))


# or with curve

par(mfcol=c(2,2))

curve(dnorm(x,3e4,9e3),from = -10000 , to = 72000, col='blue',xlab='F',ylab='pdf',lwd=2)

curve(dtriangle(x,2.8e7,4.8e7,3e7),from = 2.7e7 , to = 4.9e7,col='red',xlab='E',ylab='pdf',lwd=2)

curve(dunif(x,250,260),from = 248 , to = 262,col=rgb(0.3,0.7,0.3),xlab='L',ylab='pdf',lwd=2)

curve(dtriangle(x,310,450,400),from = 305 , to = 455,col='orange',xlab='I',ylab='pdf',lwd=2)


#2.2


beam <- function(x){
  # parameters in the following order: F, E, L, I
  # this function takes as inputs a n x d matrix and returns a n-size vector as output
  if (is.vector(x)){x <- matrix(x,nrow=1)}  # turning a vector to a matrix
  result <- (x[,1]*x[,3]^3)/(3*x[,2]*x[,4])
  return(result)
}


#2.3

N <- 1000
```

```
F2 <- rnorm(N,3e4,9e3)

E2 <- rtriangle(N,2.8e7,4.8e7,3e7)

L2 <- runif(N,250,260)

I2 <- rtriangle(N,310,450,400)

data = cbind(F2,E2,L2,I2)

z = beam(data)

mean(z) ; var(z)


summary(z)


x11() ; hist(z)

?hist

hist(z,probability=TRUE,col='lightblue',main='histogram of the beam')

lines(density(z),col='blue',lwd=2)


# convergence :


m_hat_cv <- c()          # initialisation

var_hat_cv <- c()


seqn = c(seq(10,5000,by=50))# sample size in the loop


for (k in seqn){   # loop

  Fk <- rnorm(k,3e4,9e3)

  Ek <- rtriangle(k,2.8e7,4.8e7,3e7)

  Lk <- runif(k,250,260)

  Ik <- rtriangle(k,310,450,400)
```

```
  datak = cbind(Fk,Ek,Lk,Ik)

  Yk = beam(datak)

  m_hat_cv <- c(m_hat_cv,mean(Yk))

  var_hat_cv <- c(var_hat_cv,var(Yk))

}


# graph

x11()

par(mfrow = c(2, 1))

plot(seqn, m_hat_cv,col='blue',type='l',xlab='n',ylab='mean estimation',main='mean convergence')

plot(seqn , var_hat_cv,col='red',type='l',xlab='n',ylab='variance estimation',main='variance
convergence')



#2.4

F <- 3e4 ; E <- 3e7 ; L <- 250 ; I <- 400 # nominal point

Moycum <- beam(cbind(F,E,L,I))

print(Moycum)


help(deriv) # Formal differentiation

DE <- deriv(y ~ ( F * L^3 ) / (3 * E * I),c("F","E","L","I"))

print(DE)

#print(eval(DE))

grad <- eval(DE)

grad <- attributes(grad)$gradient[1,]

print("Dérivées exactes :")

print(grad)

#Quadratic summation
```

```
Varcum <- grad[1]^2*var(F1)+grad[2]^2*var(E1)+grad[3]^2*var(L1)+grad[4]^2*var(I1)
print(Varcum)

#2.5
quantile(z,0.95) # empirical quantile

##########################
#3.1

failure <- function(x){
  # there is a failure when this function is negative
  # parameters are in the following order: F, E, L, I
  # this function takes as inputs a n x d matrix and returns a n-size vector as output

  if (is.vector(x)){x <- matrix(x,nrow=1)}
  result <- 30 - (x[,1]*x[,3]^3)/(3*x[,2]*x[,4])
  return(result)
}

#3.2
n <- 10000

# radom sampling of inputs
F1 <- rnorm(n,3e4,9e3)
E1 <- rtriangle(n,2.8e7,4.8e7,3e7)
L1 <- runif(n,250,260)
I1 <- rtriangle(n,310,450,400)
```

```
data <- cbind(F1,E1,L1,I1)

fail_Y <- failure(data)


# use function sum

pf =  sum(fail_Y < 0) / n


cv <- sqrt((1-pf)/(n*pf))


#3.3


cvlim <- 0.015

Nlim <- 1e9


N = 1e6

Nrun = N

cv = 1

z = NULL

res = NULL


while ((Nrun <= Nlim) & (cv > cvlim)){

  F2 <- rnorm(N,3e4,9e3)

  E2 <- rtriangle(N,2.8e7,4.8e7,3e7)

  L2 <- runif(N,250,260)

  I2 <- rtriangle(N,310,450,400)

  data = cbind(F2,E2,L2,I2)

  z = c(z,failure(data))

  pf <- sum(z < 0) / Nrun
```

```
  cv <- sqrt((1-pf)/(Nrun*pf))
  Nrun <- length(z)
  res <- rbind(res,c(Nrun,pf,cv))
  print(c(Nrun,pf,cv))
}
x11()
plot(res[,1],res[,2])

#3.4

eps = 1e-7
Method = "AR"
IS = FALSE
N.calls = 2000
q = 1
u.dep = c(0,0,0,0)

choice.law = list()
choice.law[[1]] = list("norm",c(3e4,9e3))
choice.law[[2]] = list("triangle",c(2.8e7,4.8e7,3e7))
choice.law[[3]] = list("unif",c(250,260))
choice.law[[4]] = list("triangle",c(310,450,400))

res=FORM(f=failure,u.dep=u.dep,choice.law=choice.law,
        N.calls=N.calls,eps=eps,Method=Method,IS=IS,q=q)
print(res)
```

```
#3.5


eps = 1e-7

Method = "AR"

IS = TRUE

N.calls = 2000

q = 0.1

u.dep = c(0.1,0,0,0)


choice.law = list()

choice.law[[1]] = list("norm",c(3e4,9e3))

choice.law[[2]] = list("triangle",c(2.8e7,4.8e7,3e7))

choice.law[[3]] = list("unif",c(250,260))

choice.law[[4]] = list("triangle",c(310,450,400))


res=FORM(f=failure,u.dep=u.dep,choice.law=choice.law,

      N.calls=N.calls,eps=eps,Method=Method,IS=IS,q=q)

print(res)
```