TP 5 : Révisions : simulation et estimation

Exercice 1 : Simulation et estimation pour la Loi Gamma.

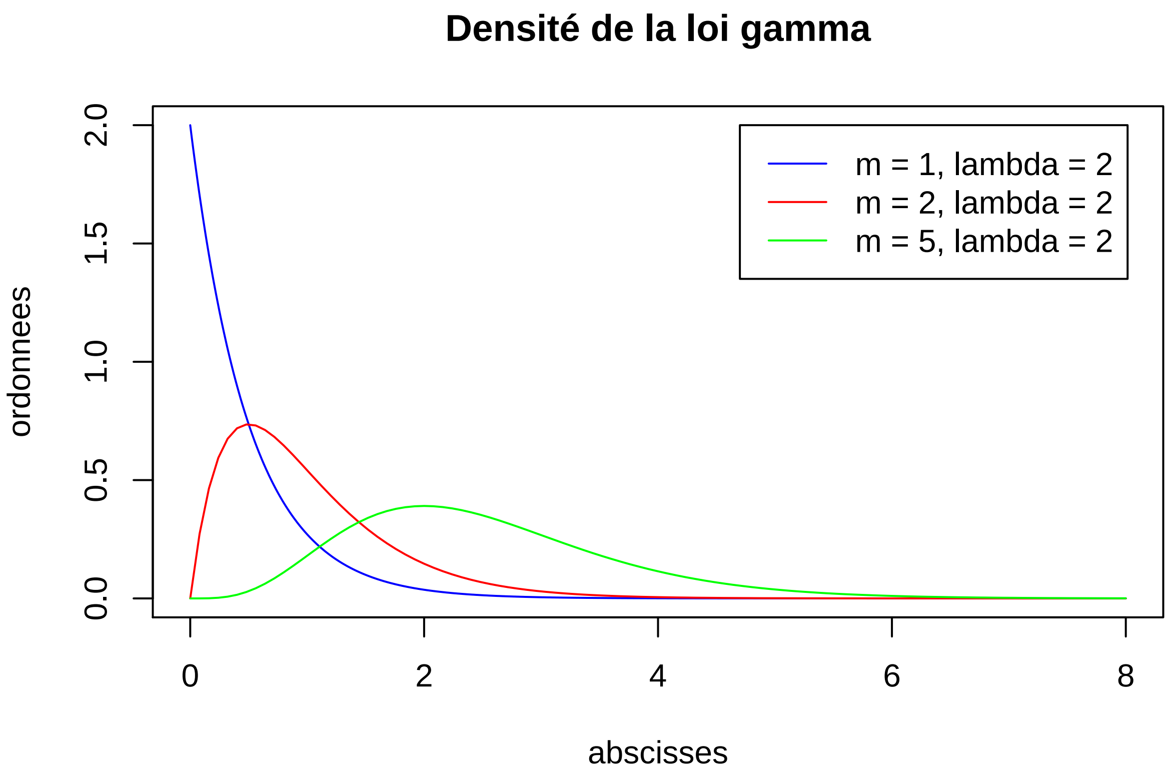
1. Réponse : On remarque facilement qu'une variable de loi $\Gamma(m, \lambda)$ quand m vaut 1 est une variable de loi exponentielle (cf la densité), de paramètre λ.

2. Voici le code pour la fonction `f`:

```
f = function(m,lambda,x){
  x^(m-1) * lambda^m * exp(-lambda*x)/factorial(m-1)
}
```

Et voici le code pour la représentation graphique demandée :

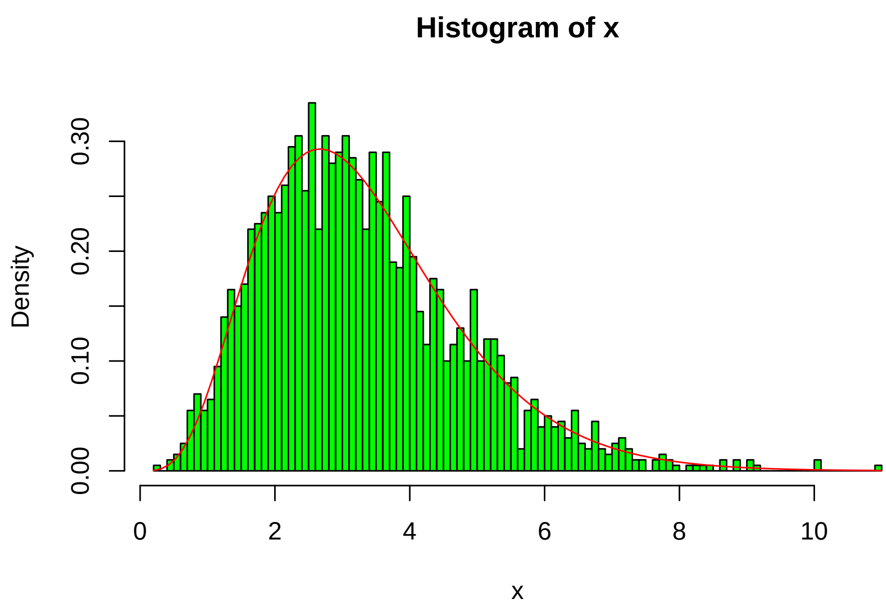
```
abscisses = seq(0,8,by=0.01) #abscisses pour le premier plot
ordonnees = f(m=1,lambda = 2,abscisses) #ordonnées pour le premier plot
plot(abscisses,ordonnees,type="l",col="blue",main = "Densité de la loi gamma")
curve(f(m=2,lambda=2,x),add=TRUE,col="red")
curve(f(m=5,lambda=2,x),add=TRUE,col="green")
legend(4.7,2,legend=c("m = 1, lambda = 2","m = 2, lambda = 2","m = 5, lambda = 2"),
       col=c("blue", "red","green"),lty=1)
```



On pourra prêter attention à l'utilisation de la fonction `legend`.

3. Voici le code pour la simulation et l'affichage de l'histogramme.

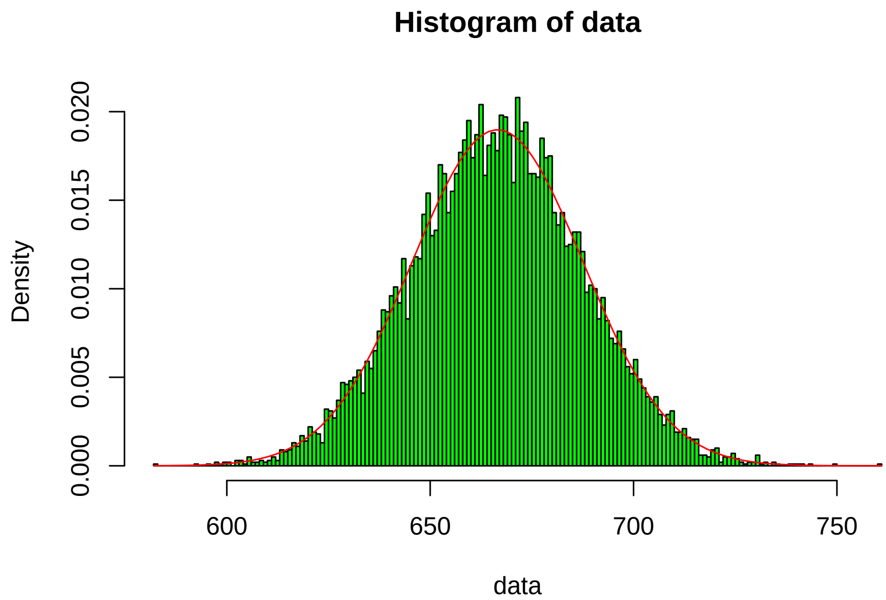
```
n = 2000
m = 5
lambda = 1.5
x = rgamma(n,shape=m,rate=lambda)

hist(x,freq=FALSE,breaks=80,col="green")
curve(f(m,lambda,x),add=TRUE,col="red")
```



L'histogramme est proche de la densité théorique, comme attendu.

4. Réponse : on peut faire une approximation gaussienne. Vérifions-le en superposant à l'histogramme la courbe de densité de la loi Gamma en bleu, et celle de la loi normale (de bonne moyenne et variance) en rouge.

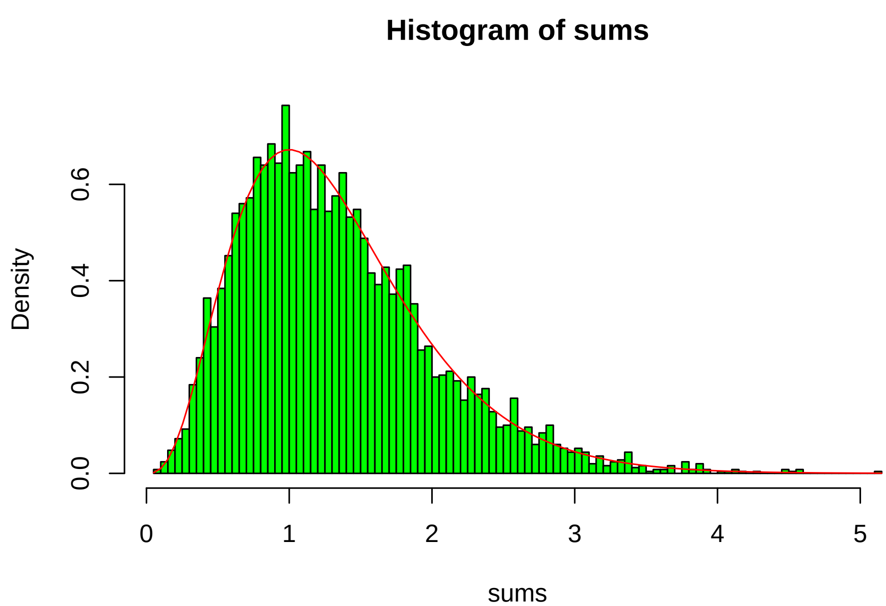
```
n = 10^4
m = 1000
lambda = 1.5
data = rgamma(n,shape=m,rate = lambda)

hist(data,freq=FALSE,breaks=150,col="green")
curve(f(m,lambda,x),add=TRUE,col="blue")
curve(dnorm(x,mean = mean(data), sd = sd(data)),add=TRUE,col="red") # on superpose aussi la densité gau
```



Les résultats sont satisfaisants, et ce d'autant plus que m est grand.

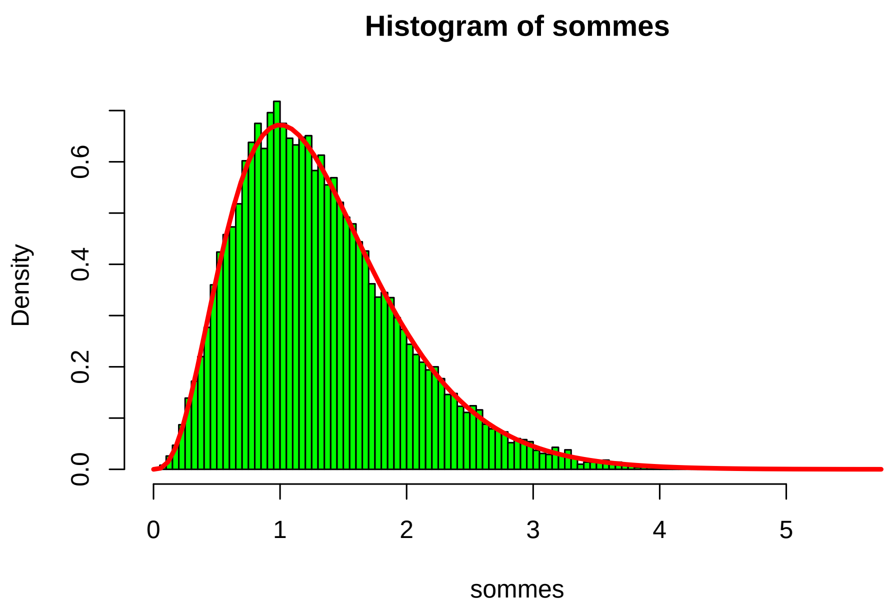
5. On simule $Y_1 + Y_2 + Y_3 + Y_4$ où les Y_i sont i.i.d. de loi exponentielle de paramètre $\lambda = 3$. On superpose à l'histogramme la densité de la loi Gamma correspondante.

```
n = 5000
m = 4
lambda = 3
sums = replicate(n,0)
for (i in 1:n){
  sums[i] = sum(rexp(m,rate= lambda))
}

hist(sums,freq=F,breaks=100,col="green")
curve(f(m,lambda,x),add=TRUE,col="red")
```



Voici un autre code, plus efficace.

```r
n = 20000
m = 4
lambda = 3
sommes = replicate(n,0)
for (j in 1:m){
  sommes = sommes + rexp(n,rate= lambda)
}

hist(sommes,freq=F,breaks=100,col="green")
curve(f(m,lambda,x),add=TRUE,col="red",lwd=3) #je rajoute lwd pour épaissir la ligne
```



L'histogramme et la densité sont proches : ces résultats illustrent bien cette égalité en loi.

6. Réponse : avec le résultat précédent, et la loi des grands nombres, on sait que $M_n \to m/\lambda$ p.s. et que $V_n \to m/\lambda^2$ p.s. Cela donne donc $\hat{\lambda} := M_n/V_n$ et $\hat{m} = M_n^2/V_n$.

7. Voici le code attendu :

```r
N = 2000 #nombre d'echantillons
n = 5000 #taille de chaque echantillon
m = 5
lambda = 5

m.est = replicate(N,0)
lambda.est = replicate(N,0)

for (i in 1:N){# chaque échantillon
  y = rgamma(n,shape=m,rate = lambda) # on simule l'échantillon
  lambda.est[i] = mean(y)/var(y) # on calcule l'estimateur de lambda
  m.est[i] = mean(y)^2/var(y) # on calcule l'estimateur de m
}

biais.m = mean(m.est - m)
var.m = var(m.est)
risque.m = mean((m.est - m)^2)
```

```r
resultats.m = c(biais.m,var.m,risque.m)

biais.lambda = mean(lambda.est - lambda)
var.lambda = var(lambda.est)
risque.lambda = mean((lambda.est - lambda)^2)

resultats.lambda = c(biais.lambda,var.lambda,risque.lambda)

rbind(resultats.m,resultats.lambda)
```

```
##                         [,1]       [,2]       [,3]
## resultats.m      0.006934368 0.01204512 0.01208718
## resultats.lambda 0.007210613 0.01283590 0.01288148
```

Les risques sont faibles (par exemple 0.01 « 5^2, c'est bien), et ils dépendent principalement du terme de variance, le biais étant toujours très faible. Une question : qu'est ce que c'est qu'un risque faible ? C'est un risque quadratique petit devant le carré de la valeur typique du paramètre.

Pour ceux qui veulent aller plus loin : Une transition de phase pour l'estimation

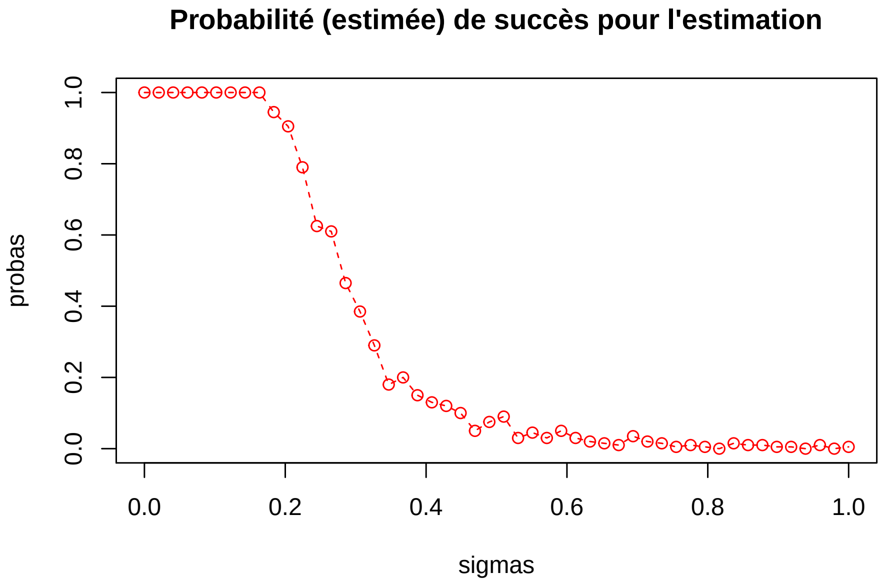
1. Voici le code pour la fonction `simu`.

```r
simu = function(n,sigma){
  i = sample.int(n, size = 1) #choix d'un entier uniforme entre 1 et n
  ei = replicate(n,0)
  ei[i] = 1 #définition de ei
  Z = rnorm(n)
  X = ei + sigma*Z
  c(X,i)
}
```

2. Voici le code attendu ici.

```r
n_sigmas = 50
sigmas = seq(0,1,length.out=n_sigmas)
n = 4000 # taille de l'échantillon
N = 200 # nombre de tests pour chaque valeur de sigma
probas = replicate(n_sigmas,0)

for (s in 1:n_sigmas){
  sigma = sigmas[s]
  score = 0
  for (j in 1:N){
    realisation = simu(n,sigma)
    X = realisation[1:n] #la X simulé
    i = realisation[n+1] #le vrai i
    hati = which(X==max(X))[1] #une façon de trouver l'argmax du vecteur X
    if (hati == i){
      score = score + 1
    }
  }
  probas[s] = score/N
}
```

```
plot(sigmas,probas,col="red",main="Probabilité (estimée) de succès pour l'estimation")
lines(sigmas,probas,col="red",lty=2)
```



Le code met un peu de temps à tourner, à vous d'adapter les paramètres n et N. Pour des valeurs faibles de σ, l'estimateur marche très bien, et il y a ensuite une barrière assez abrupte au-delà de laquelle l'estimateur de fonctionne plus pour retrouver i : c'est ce que l'on appelle une *transition de phase*. On pourra bien sûr essayer ce code avec d'autres paramètres, et vérifier que la transition a lieu pour $\sigma \sim \frac{1}{\sqrt{2\log n}}$ (pourquoi ?)

```
1/sqrt(2*log(n))
```

```
## [1] 0.2455284
```