TP 4 Statistiques

Adib Habbou - Adel Kebli - Clark Ji

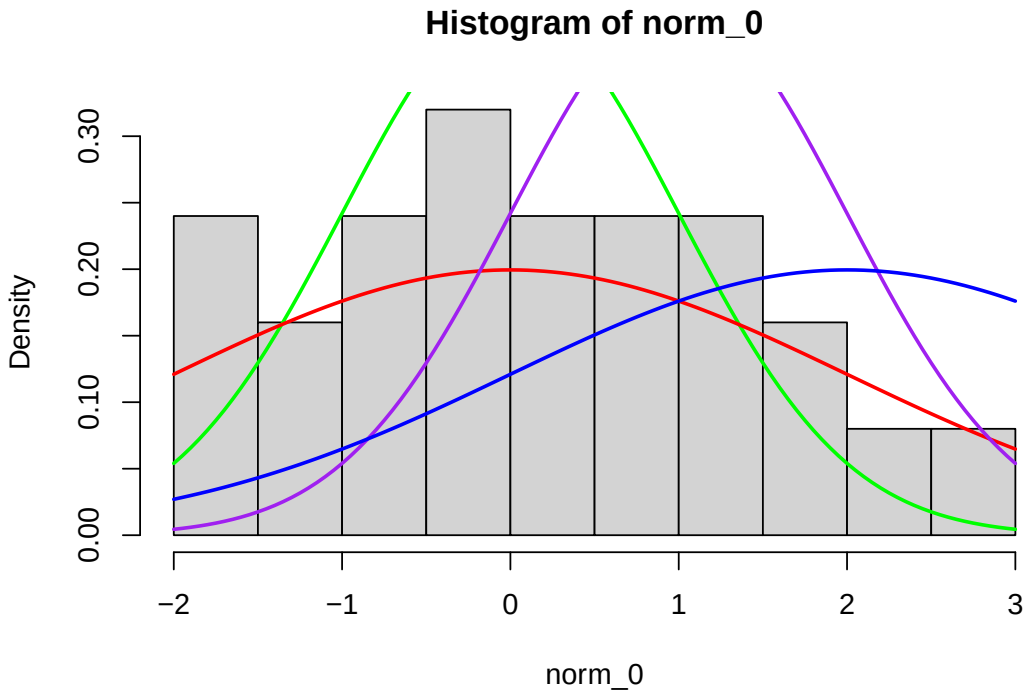
08/04/2022

Vraisemblance pour plusieurs paramètres

Loi normale

Question 1 :

```
norm_0 = rnorm(25, 0, 1)

hist(norm_0, breaks = 10, prob = TRUE)

curve(dnorm(x, 0, 1), add=TRUE, col="green", lwd=2)
curve(dnorm(x, 0, 2), add=TRUE, col="red", lwd=2)
curve(dnorm(x, 1, 1), add=TRUE, col="purple", lwd=2)
curve(dnorm(x, 2, 2), add=TRUE, col="blue", lwd=2)
```



On remarque que la densité verte correspond bien à une loi normale centrée réduite : $\mathcal{N}(0,1)$

Tandis que la courbe violette est un peu plus décalée vers la droite : $\mathcal{N}(1,1)$

La courbe rouge quant à elle est beaucoup plus aplatie que la verte : $\mathcal{N}(0,2)$

Pour finir, la courbe bleu est à la fois aplatie et décalée vers la droite : $\mathcal{N}(0,2)$

Question 2 :

La densité d'une loi normale est :

$$f(x,\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

Pour n échantillons indépendants et identiquements distribués, on a alors la vraisemblance qui est égale à :

$$L(x_1,...,x_n,\mu,\sigma) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}(\frac{x_i-\mu}{\sigma})^2}$$

$$L(x_1,...,x_n,\mu,\sigma) = (\frac{1}{\sqrt{2\pi\sigma^2}})^n \exp(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\mu)^2)$$

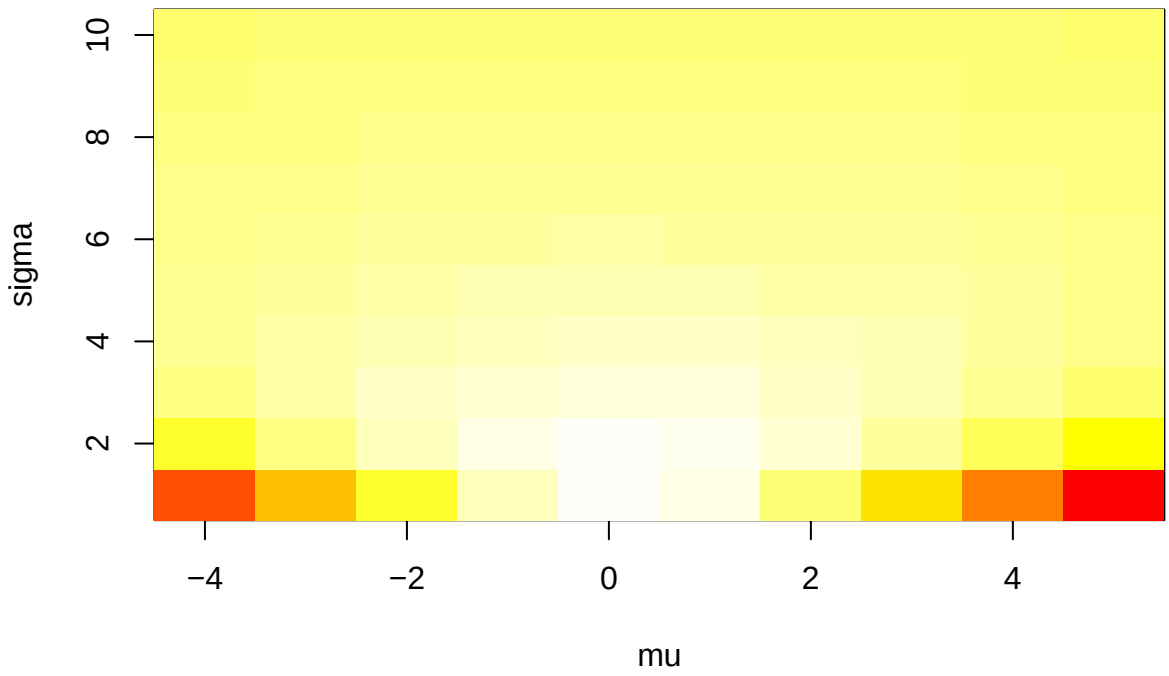
En passant à la log-vraisemblance, on a finalement :

$$\mathcal{L}(x_1,...,x_n,\mu,\sigma) = -\frac{n}{2}\ln(2\pi) - n\ln(\sigma) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\mu)^2$$

```
log_vraisemblance_norm <- function (mu, sigma, x) {
  n = length(x)
  somme = 0
  for (i in 1:n) {
    somme = somme + (x[i] - mu) ^ 2
  }
  return ((-n) * log(sigma) - (n/2) * log(2 * pi) - somme / (2 * (sigma ^ 2)))
}
```

```
mu = seq(from = -4, to = 5, by = 1)
sigma = seq(from = 1, to = 10, by = 1)
lvn0 = matrix(data = 1, nrow = 10, ncol = 10)
for (i in 1:10) {
  for (j in 1:10) {
    lvn0[i, j] = log_vraisemblance_norm(mu[i], sigma[j], norm_0)
  }
}
```

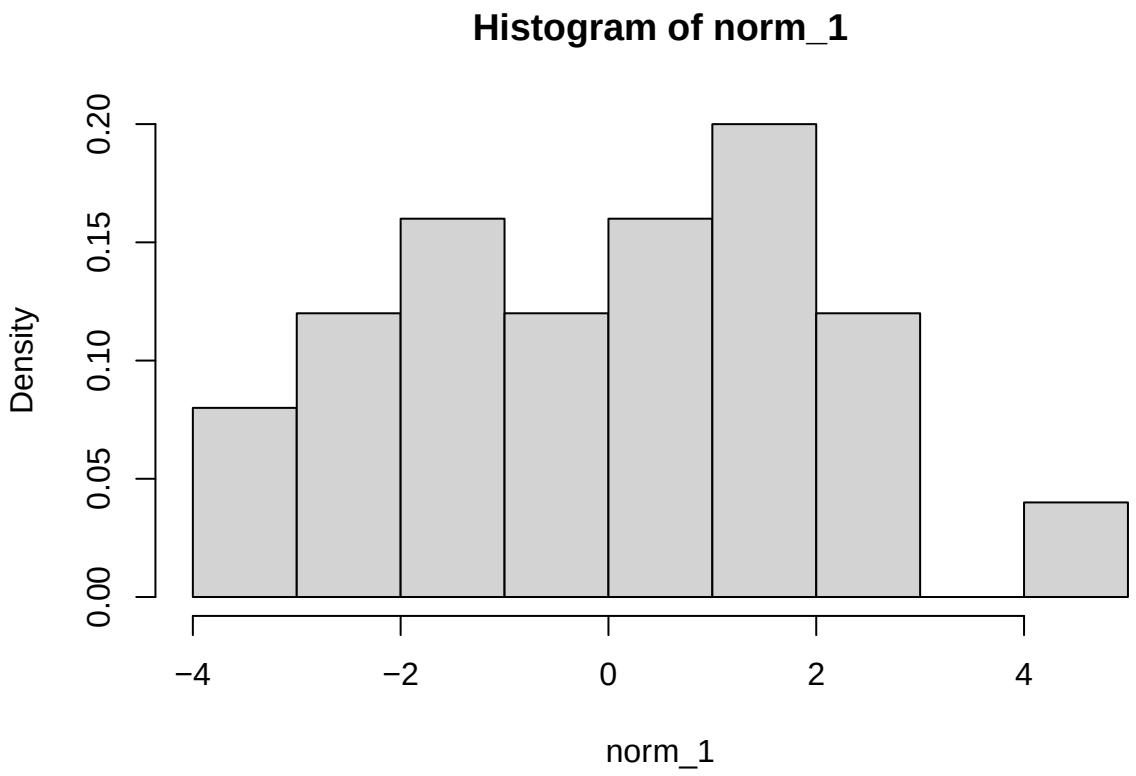
```
image(x = mu, y = sigma, z = lvn0, col = heat.colors(100))
```



On remarque d'après les variations de couleurs, que le carré le plus clair est le carré correspondant au point $\theta_0 = (0, 1)$. C'est donc le maximum de vraisemblance pour notre échantillon.

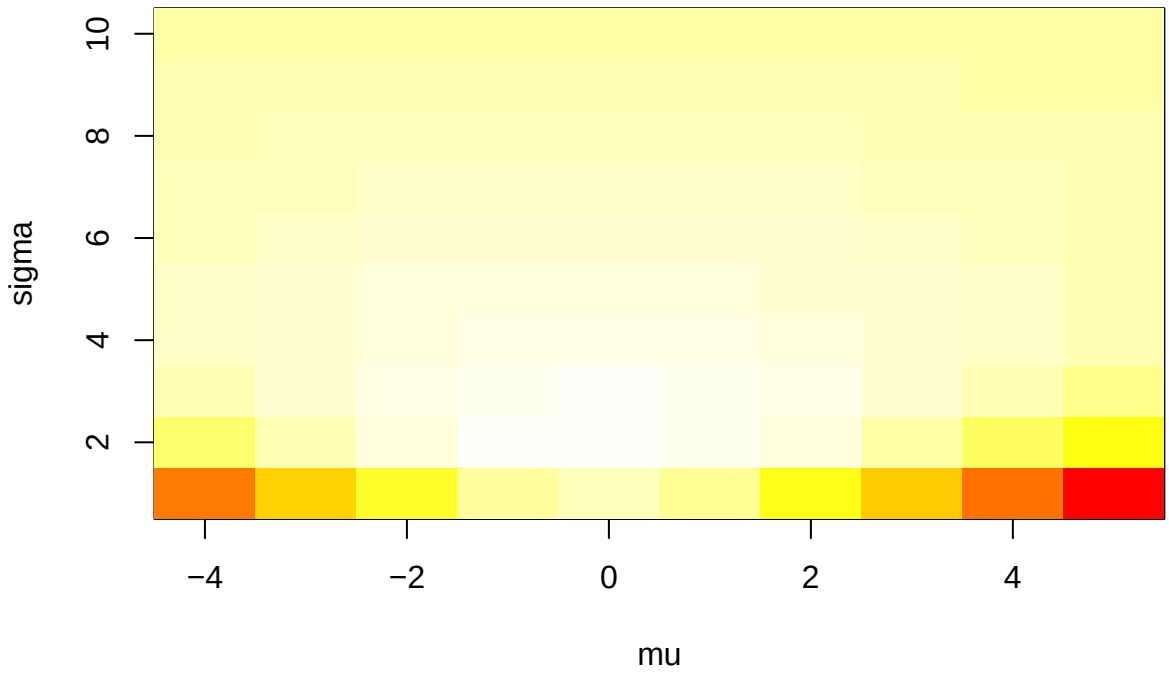
Question 3 :

```
norm_1 = rnorm(25, 0, 2)

hist(norm_1, breaks = 10, prob = TRUE)
```



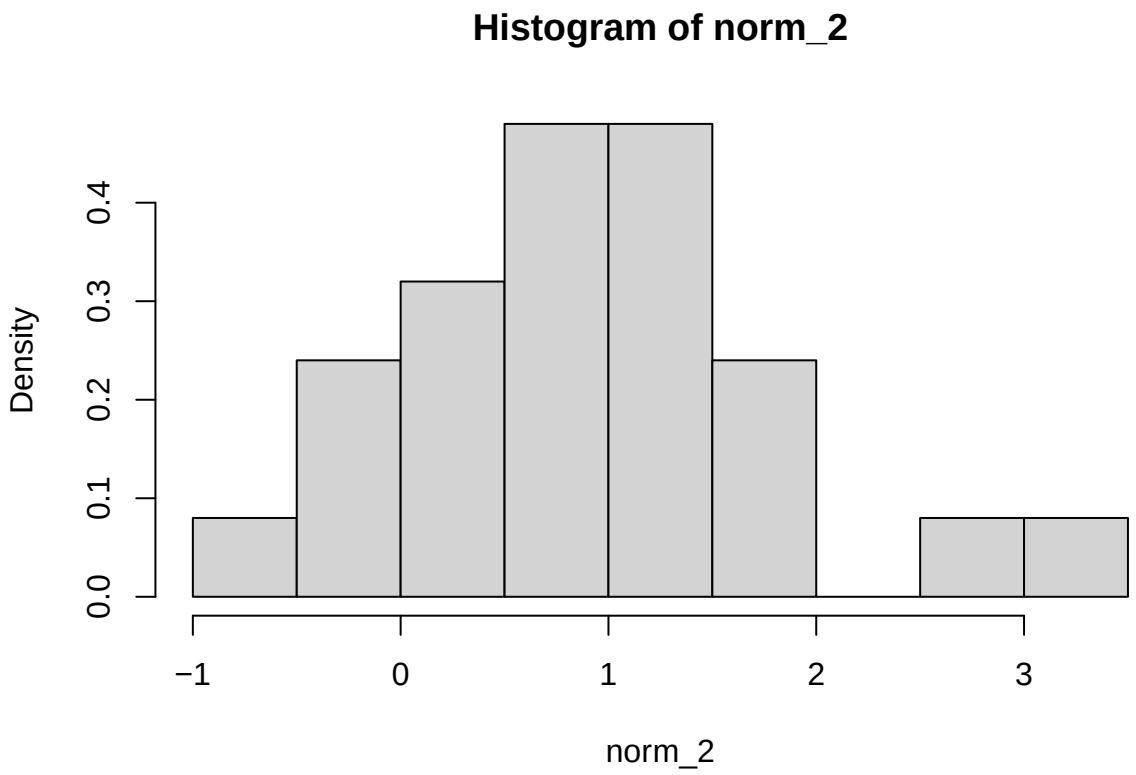

```
lvn1 = matrix(data = 1, nrow = 10, ncol = 10)
for (i in 1:10) {
  for (j in 1:10) {
    lvn1[i, j] = log_vraisemblance_norm(mu[i], sigma[j], norm_1)
  }
}
```

```r
image(x = mu, y = sigma, z = lvn1, col = heat.colors(100))
```



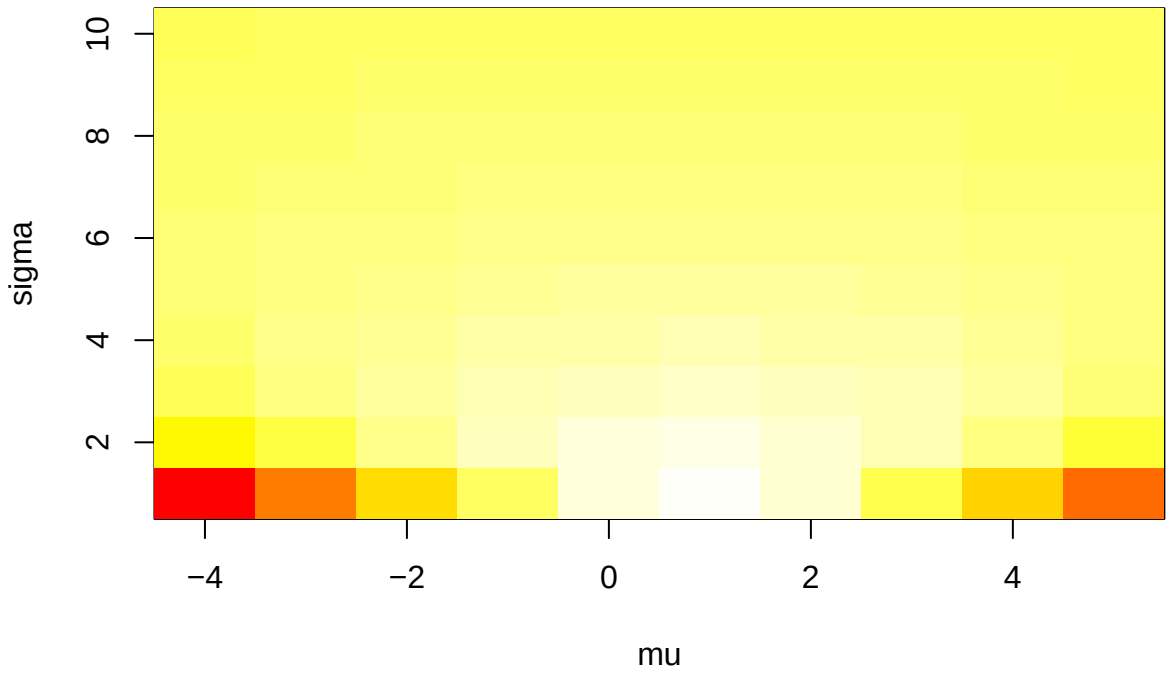
On remarque d'après les variations de couleurs, que le carré le plus clair est le carré correspondant au point $\theta_1 = (0, 2)$. C'est donc le maximum de vraisemblance pour notre échantillon.

```r
norm_2 = rnorm(25, 1, 1)

hist(norm_2, breaks = 10, prob = TRUE)
```



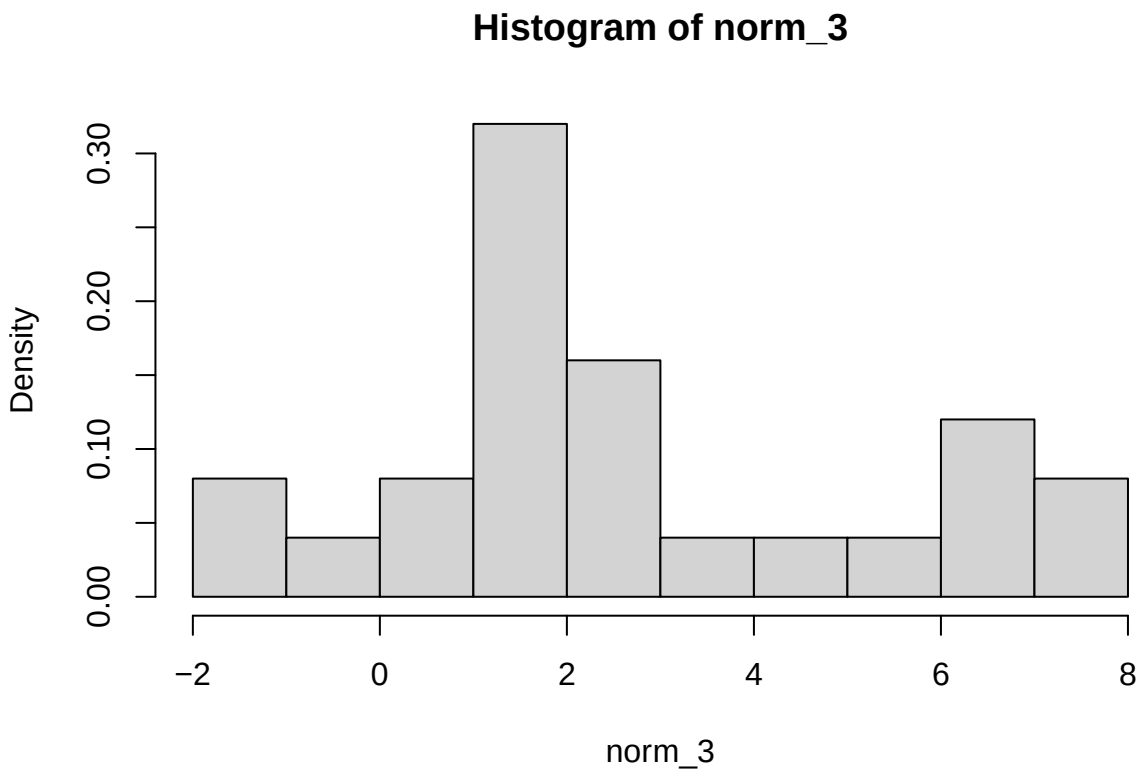

```r
lvn2 = matrix(data = 1, nrow = 10, ncol = 10)
for (i in 1:10) {
  for (j in 1:10) {
    lvn2[i, j] = log_vraisemblance_norm(mu[i], sigma[j], norm_2)
  }
}
```

```r
image(x = mu, y = sigma, z = lvn2, col = heat.colors(100))
```



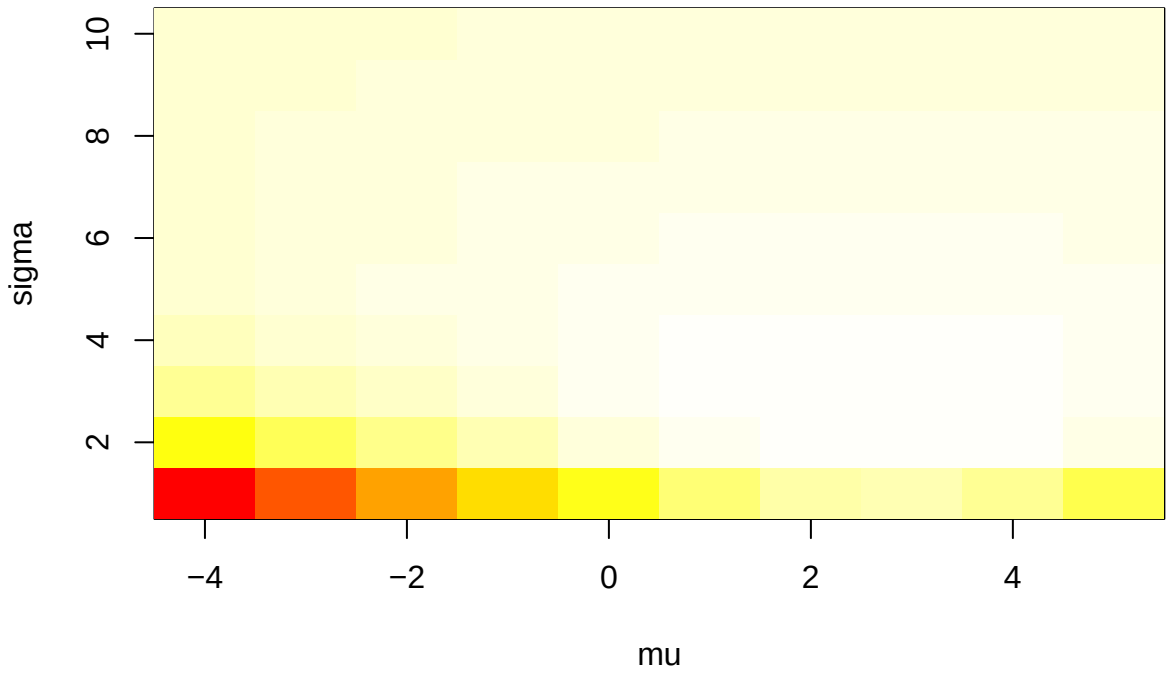
On remarque d'après les variations de couleurs, que le carré le plus clair est le carré correspondant au point $\theta_2 = (1, 1)$. C'est donc le maximum de vraisemblance pour notre échantillon.

```r
norm_3 = rnorm(25, 2, 2)

hist(norm_3, breaks = 10, prob = TRUE)
```




```r
lvn3 = matrix(data = 1, nrow = 10, ncol = 10)
for (i in 1:10) {
  for (j in 1:10) {
    lvn3[i, j] = log_vraisemblance_norm(mu[i], sigma[j], norm_3)
  }
}
```

```
image(x = mu, y = sigma, z = lvn3, col = heat.colors(100))
```

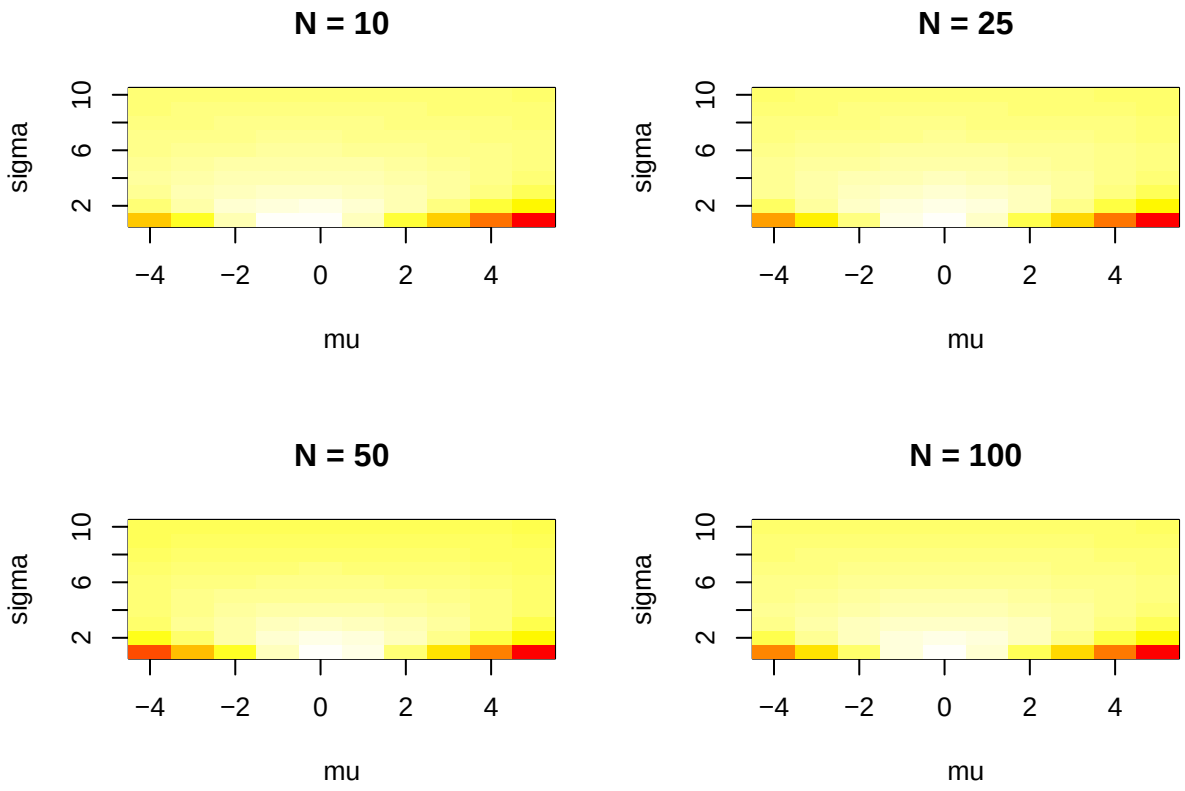


On remarque d'après les variations de couleurs, que le carré le plus clair est le carré correspondant au point $\theta_0 = (2, 2)$. C'est donc le maximum de vraisemblance pour notre échantillon.

Question 4 :

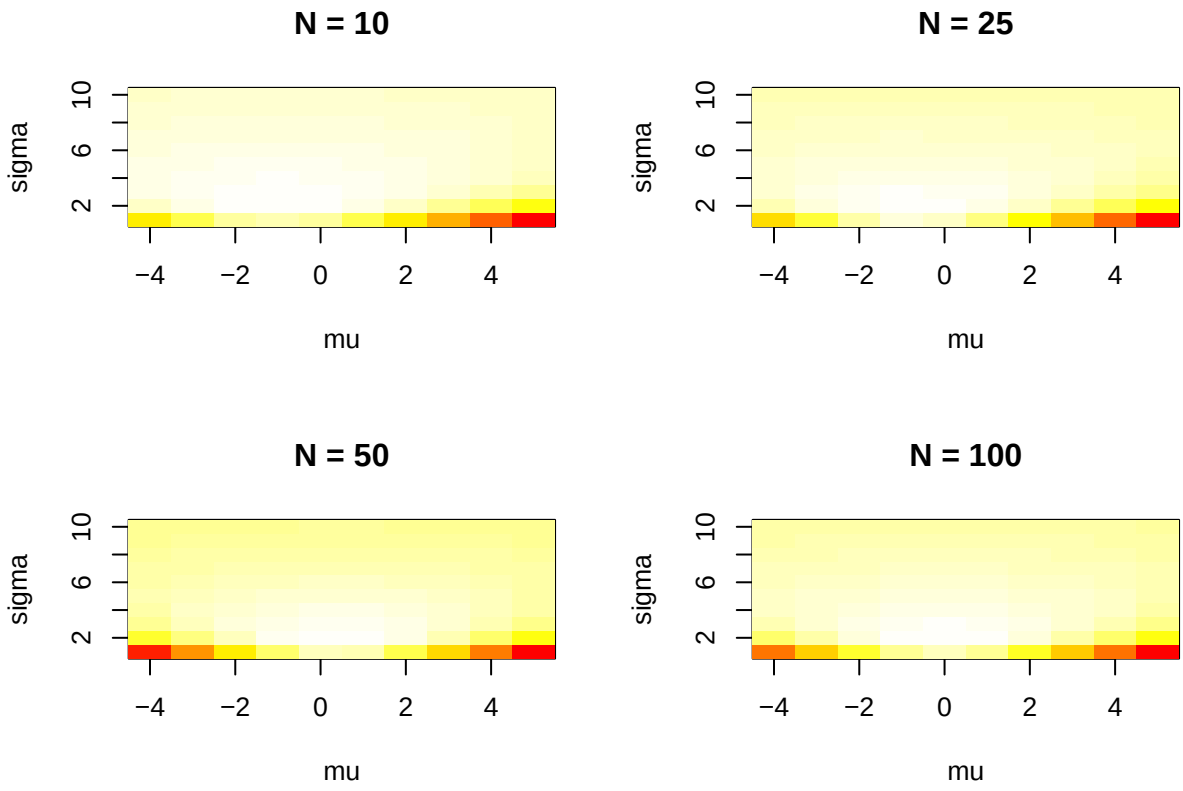
```
norm_0_10 = rnorm(10, 0, 1)
norm_0_25 = rnorm(25, 0, 1)
norm_0_50 = rnorm(50, 0, 1)
norm_0_100 = rnorm(100, 0, 1)

lvn0_10 = matrix(data = 1, nrow = 10, ncol = 10)
lvn0_25 = matrix(data = 1, nrow = 10, ncol = 10)
lvn0_50 = matrix(data = 1, nrow = 10, ncol = 10)
lvn0_100 = matrix(data = 1, nrow = 10, ncol = 10)

for (i in 1:10) {
  for (j in 1:10) {
    lvn0_10[i, j] = log_vraisemblance_norm(mu[i], sigma[j], norm_0_10)
  }
}
for (i in 1:10) {
  for (j in 1:10) {
    lvn0_25[i, j] = log_vraisemblance_norm(mu[i], sigma[j], norm_0_25)
  }
}
for (i in 1:10) {
  for (j in 1:10) {
    lvn0_50[i, j] = log_vraisemblance_norm(mu[i], sigma[j], norm_0_50)
  }
}
for (i in 1:10) {
  for (j in 1:10) {
    lvn0_100[i, j] = log_vraisemblance_norm(mu[i], sigma[j], norm_0_100)
  }
}
```

```r
par(mfrow = c(2, 2))

image(x = mu, y = sigma, z = lvn0_10, col = heat.colors(100), main = "N = 10")
image(x = mu, y = sigma, z = lvn0_25, col = heat.colors(100), main = "N = 25")
image(x = mu, y = sigma, z = lvn0_50, col = heat.colors(100), main = "N = 50")
image(x = mu, y = sigma, z = lvn0_100, col = heat.colors(100), main = "N = 100")
```



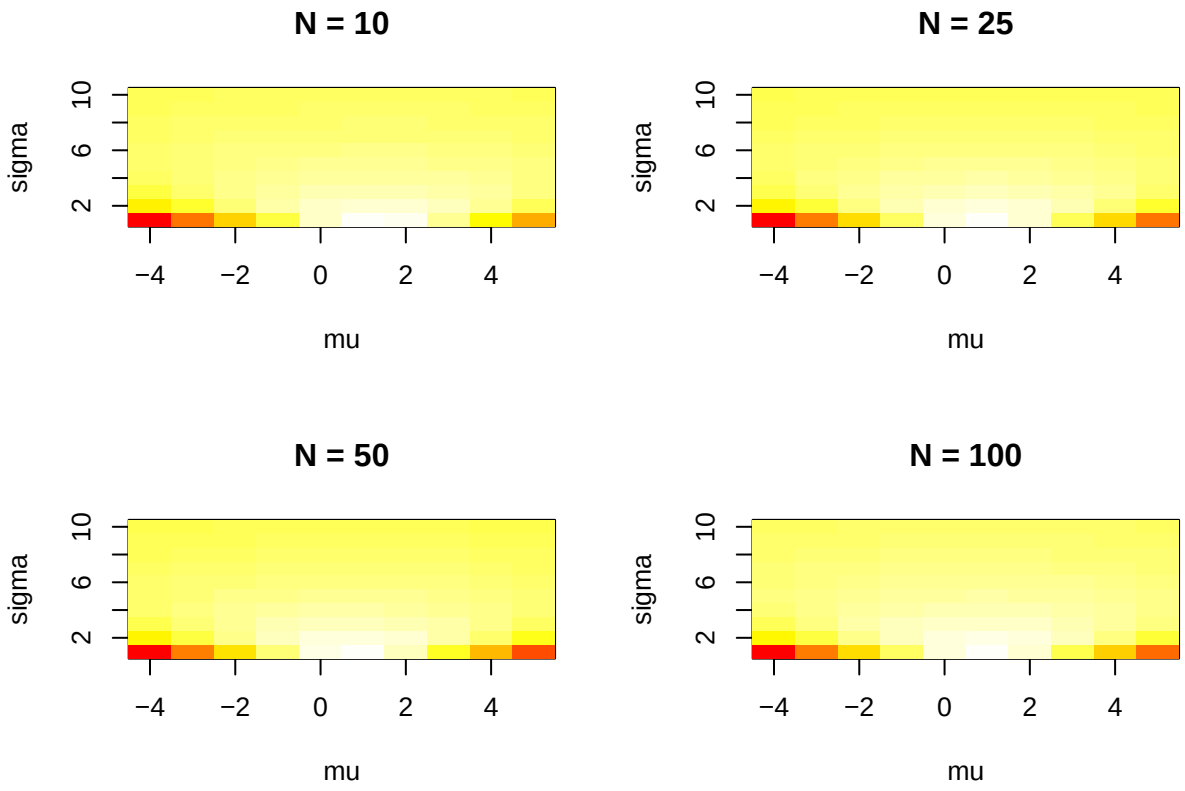
```r
norm_1_10 = rnorm(10, 0, 2)
norm_1_25 = rnorm(25, 0, 2)
norm_1_50 = rnorm(50, 0, 2)
norm_1_100 = rnorm(100, 0, 2)

lvn1_10 = matrix(data = 1, nrow = 10, ncol = 10)
lvn1_25 = matrix(data = 1, nrow = 10, ncol = 10)
lvn1_50 = matrix(data = 1, nrow = 10, ncol = 10)
lvn1_100 = matrix(data = 1, nrow = 10, ncol = 10)

for (i in 1:10) {
  for (j in 1:10) {
    lvn1_10[i, j] = log_vraisemblance_norm(mu[i], sigma[j], norm_1_10)
  }
}
for (i in 1:10) {
  for (j in 1:10) {
    lvn1_25[i, j] = log_vraisemblance_norm(mu[i], sigma[j], norm_1_25)
  }
}
for (i in 1:10) {
  for (j in 1:10) {
    lvn1_50[i, j] = log_vraisemblance_norm(mu[i], sigma[j], norm_1_50)
  }
}
for (i in 1:10) {
  for (j in 1:10) {
    lvn1_100[i, j] = log_vraisemblance_norm(mu[i], sigma[j], norm_1_100)
  }
}
```

```
par(mfrow = c(2, 2))

image(x = mu, y = sigma, z = lvn1_10, col = heat.colors(100), main = "N = 10")
image(x = mu, y = sigma, z = lvn1_25, col = heat.colors(100), main = "N = 25")
image(x = mu, y = sigma, z = lvn1_50, col = heat.colors(100), main = "N = 50")
image(x = mu, y = sigma, z = lvn1_100, col = heat.colors(100), main = "N = 100")
```



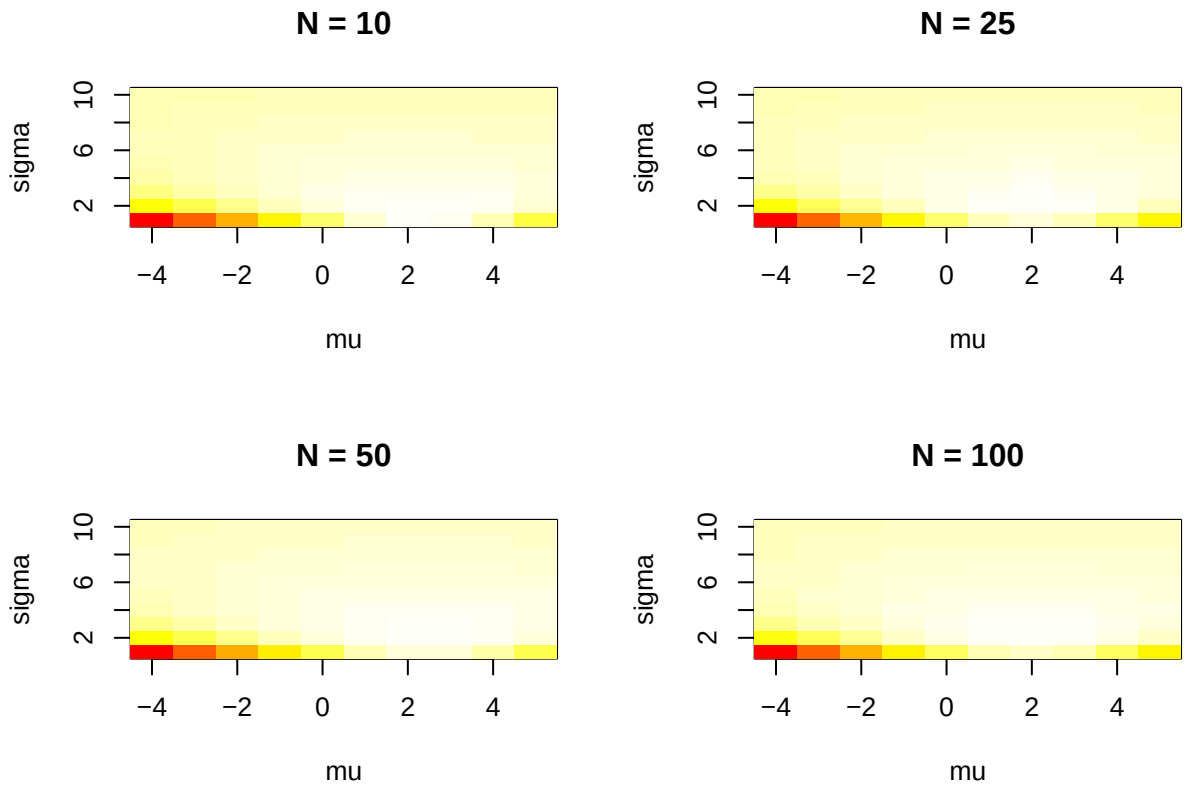
```
norm_2_10 = rnorm(10, 1, 1)
norm_2_25 = rnorm(25, 1, 1)
norm_2_50 = rnorm(50, 1, 1)
norm_2_100 = rnorm(100, 1, 1)

lvn2_10 = matrix(data = 1, nrow = 10, ncol = 10)
lvn2_25 = matrix(data = 1, nrow = 10, ncol = 10)
lvn2_50 = matrix(data = 1, nrow = 10, ncol = 10)
lvn2_100 = matrix(data = 1, nrow = 10, ncol = 10)

for (i in 1:10) {
  for (j in 1:10) {
    lvn2_10[i, j] = log_vraisemblance_norm(mu[i], sigma[j], norm_2_10)
  }
}
for (i in 1:10) {
  for (j in 1:10) {
    lvn2_25[i, j] = log_vraisemblance_norm(mu[i], sigma[j], norm_2_25)
  }
}
for (i in 1:10) {
  for (j in 1:10) {
    lvn2_50[i, j] = log_vraisemblance_norm(mu[i], sigma[j], norm_2_50)
  }
}
for (i in 1:10) {
  for (j in 1:10) {
    lvn2_100[i, j] = log_vraisemblance_norm(mu[i], sigma[j], norm_2_100)
  }
}
```

```r
par(mfrow = c(2, 2))

image(x = mu, y = sigma, z = lvn2_10, col = heat.colors(100), main = "N = 10")
image(x = mu, y = sigma, z = lvn2_25, col = heat.colors(100), main = "N = 25")
image(x = mu, y = sigma, z = lvn2_50, col = heat.colors(100), main = "N = 50")
image(x = mu, y = sigma, z = lvn2_100, col = heat.colors(100), main = "N = 100")
```



```r
norm_3_10 = rnorm(10, 2, 2)
norm_3_25 = rnorm(25, 2, 2)
norm_3_50 = rnorm(50, 2, 2)
norm_3_100 = rnorm(100, 2, 2)

lvn3_10 = matrix(data = 1, nrow = 10, ncol = 10)
lvn3_25 = matrix(data = 1, nrow = 10, ncol = 10)
lvn3_50 = matrix(data = 1, nrow = 10, ncol = 10)
lvn3_100 = matrix(data = 1, nrow = 10, ncol = 10)

for (i in 1:10) {
  for (j in 1:10) {
    lvn3_10[i, j] = log_vraisemblance_norm(mu[i], sigma[j], norm_3_10)
  }
}
for (i in 1:10) {
  for (j in 1:10) {
    lvn3_25[i, j] = log_vraisemblance_norm(mu[i], sigma[j], norm_3_25)
  }
}
for (i in 1:10) {
  for (j in 1:10) {
    lvn3_50[i, j] = log_vraisemblance_norm(mu[i], sigma[j], norm_3_50)
  }
}
for (i in 1:10) {
  for (j in 1:10) {
    lvn3_100[i, j] = log_vraisemblance_norm(mu[i], sigma[j], norm_3_100)
  }
}
```

```r
par(mfrow = c(2, 2))

image(x = mu, y = sigma, z = lvn3_10, col = heat.colors(100), main = "N = 10")
image(x = mu, y = sigma, z = lvn3_25, col = heat.colors(100), main = "N = 25")
image(x = mu, y = sigma, z = lvn3_50, col = heat.colors(100), main = "N = 50")
image(x = mu, y = sigma, z = lvn3_100, col = heat.colors(100), main = "N = 100")
```



On peut dire que plus la taille de l'échantillon est grande, plus le calcul de la log-vraisemblance est précis. Par conséquent on arrive à voir beaucoup plus nettement le carré le plus clair sur la représentation graphique de la surface de la log-vraisemblance. Cela nous permet de déterminer de manière plus précise le maximum de vraisemblance.

Maximum de vraisemblance pour plusieurs paramètres

Question 5 :

```
optim(par = c(10, 10), fn = function (theta)
  {log_vraisemblance_norm(theta[1], theta[2], norm_0)},
  method = "L-BFGS-B", lower = c(-Inf, 0.1), control = list(fnscale = -1))$par
```

```
[1] 0.2020378 1.2329968
```

En reprenant la log-vraisemblance obtenue précédemment :

$$\mathcal{L}(x_1, ..., x_n, \mu, \sigma) = -\frac{n}{2}\ln(2\pi) - n\ln(\sigma) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i - \mu)^2$$

Pour calculer le maximum de vraisemblance, on cherche respectivement les valeurs de μ et σ telles que :

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial \mu} = 0 \\ \\ \frac{\partial \mathcal{L}}{\partial \sigma} = 0 \end{cases}$$

On a :

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial \mu} = \frac{1}{\sigma^2}\sum_{i=1}^{n}(x_i - \mu) = -\frac{n\mu}{\sigma^2} + \frac{1}{\sigma^2}\sum_{i=1}^{n} x_i = 0 \\ \\ \frac{\partial \mathcal{L}}{\partial \sigma} = -\frac{n}{\sigma} + \frac{1}{\sigma^3}\sum_{i=1}^{n}(x_i - \mu)^2 = 0 \end{cases}$$

$$\iff \begin{cases} \widehat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i \\ \\ \widehat{\sigma} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2} \end{cases}$$

On a donc finalement :

$$\widehat{\theta}^{MV} = \left(\frac{1}{n}\sum_{i=1}^{n} x_i, \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}\right)$$

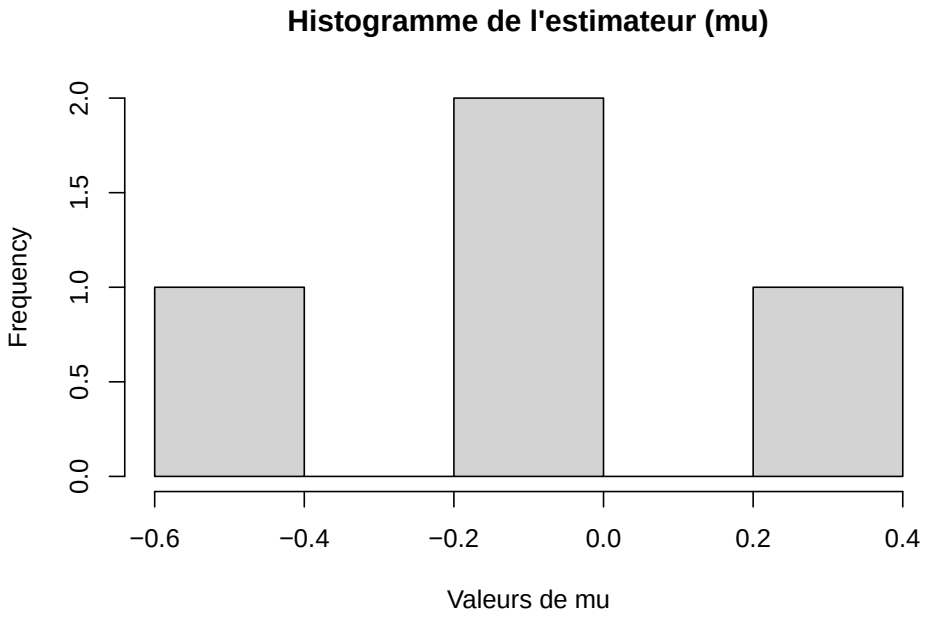
```
n = length(norm_0)
somme_x = 0
somme_x_u = 0
for (i in 1:n) {
  somme_x = somme_x + norm_0[i]
}
e_mu = somme_x / n
for (i in 1:n) {
  somme_x_u = somme_x_u + (norm_0[i] - e_mu) ^ 2
}
e_sigma = sqrt(somme_x_u / n)
```

Avec la méthode numérique on trouve comme valeur : $\widehat{\mu}_{num} = 0.02401617$ et $\widehat{\sigma}_{num} = 1.02854968$. Avec la méthode analytique on trouve comme valeur : $\widehat{\mu}_{ana} = 0.02401632$ et $\widehat{\sigma}_{ana} = 1.028549$

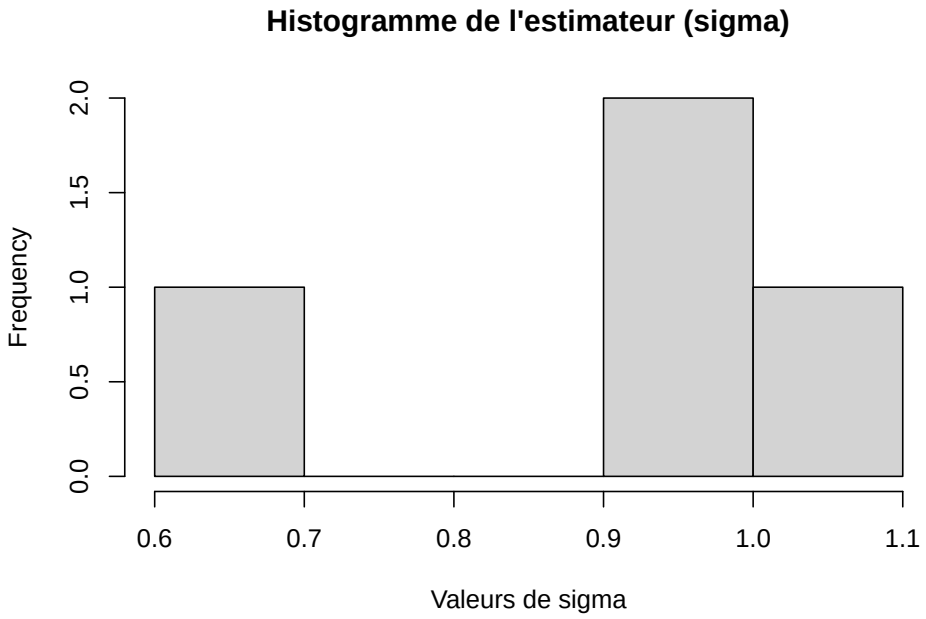
Question 6 :

```
emvn_0_10 = optim(par = c(10, 10), fn = function (theta)
  {log_vraisemblance_norm(theta[1], theta[2], norm_0_10)},
  method = "L-BFGS-B", lower = c(-Inf, 0.1), control = list(fnscale = -1))$par

emvn_0_25 = optim(par = c(10, 10), fn = function (theta)
  {log_vraisemblance_norm(theta[1], theta[2], norm_0_25)},
  method = "L-BFGS-B", lower = c(-Inf, 0.1), control = list(fnscale = -1))$par

emvn_0_50 = optim(par = c(10, 10), fn = function (theta)
  {log_vraisemblance_norm(theta[1], theta[2], norm_0_50)},
  method = "L-BFGS-B", lower = c(-Inf, 0.1), control = list(fnscale = -1))$par

emvn_0_100 = optim(par = c(10, 10), fn = function (theta)
  {log_vraisemblance_norm(theta[1], theta[2], norm_0_100)},
  method = "L-BFGS-B", lower = c(-Inf, 0.1), control = list(fnscale = -1))$par
```

```
hist(c(emvn_0_10[1], emvn_0_25[1], emvn_0_50[1], emvn_0_100[1]), prob = FALSE, breaks = 4,
     xlab = "Valeurs de mu", main = "Histogramme de l'estimateur (mu)")
```

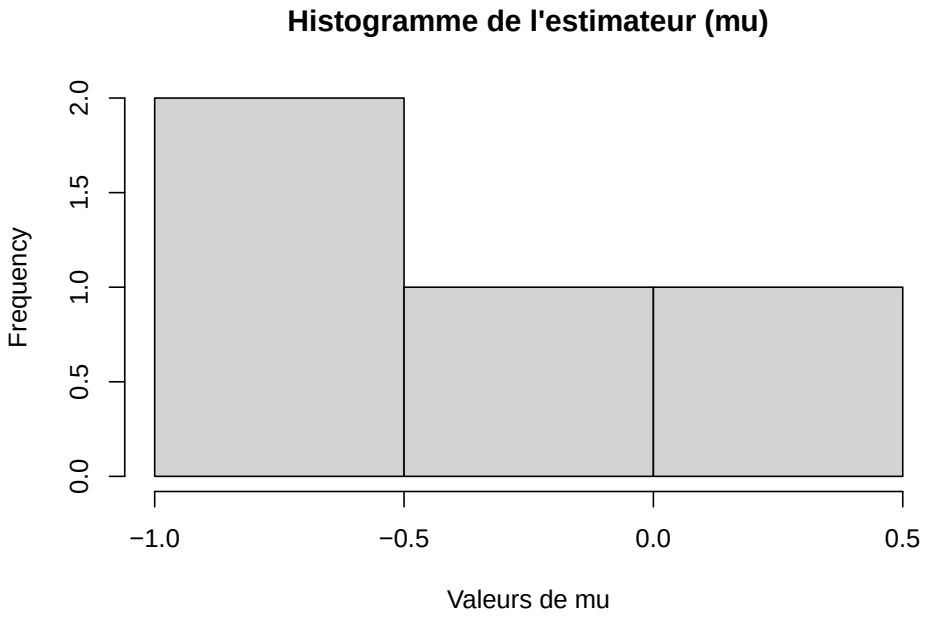


```r
hist(c(emvn_0_10[2], emvn_0_25[2], emvn_0_50[2], emvn_0_100[2]), prob = FALSE, breaks = 4,
     xlab = "Valeurs de sigma", main = "Histogramme de l'estimateur (sigma)")
```

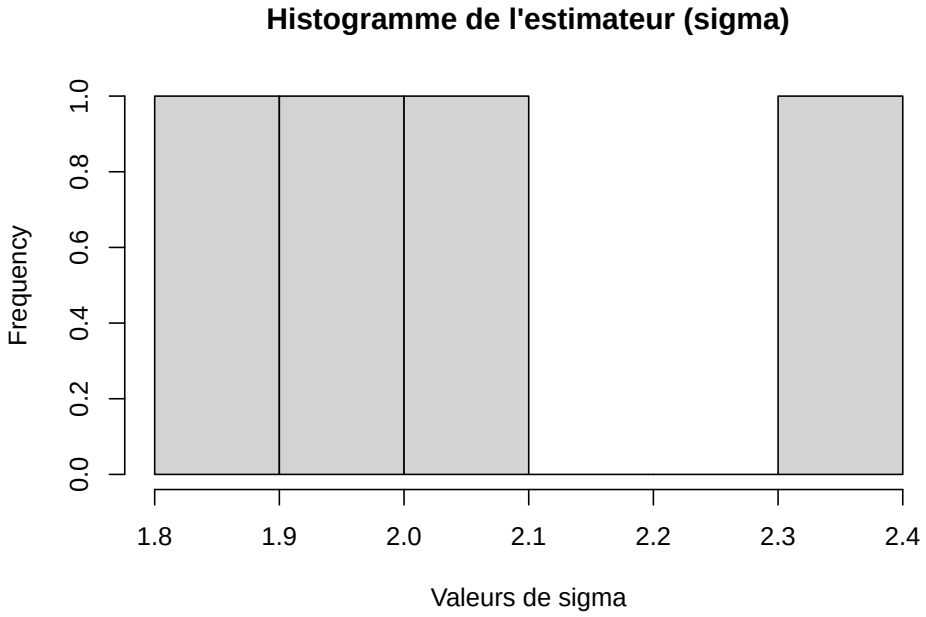



```r
emvn_1_10 = optim(par = c(10, 10), fn = function (theta)
  {log_vraisemblance_norm(theta[1], theta[2], norm_1_10)},
  method = "L-BFGS-B", lower = c(-Inf, 0.1), control = list(fnscale = -1))$par

emvn_1_25 = optim(par = c(10, 10), fn = function (theta)
  {log_vraisemblance_norm(theta[1], theta[2], norm_1_25)},
  method = "L-BFGS-B", lower = c(-Inf, 0.1), control = list(fnscale = -1))$par

emvn_1_50 = optim(par = c(10, 10), fn = function (theta)
  {log_vraisemblance_norm(theta[1], theta[2], norm_1_50)},
  method = "L-BFGS-B", lower = c(-Inf, 0.1), control = list(fnscale = -1))$par

emvn_1_100 = optim(par = c(10, 10), fn = function (theta)
  {log_vraisemblance_norm(theta[1], theta[2], norm_1_100)},
  method = "L-BFGS-B", lower = c(-Inf, 0.1), control = list(fnscale = -1))$par
```

```r
hist(c(emvn_1_10[1], emvn_1_25[1], emvn_1_50[1], emvn_1_100[1]), prob = FALSE, breaks = 4,
     xlab = "Valeurs de mu", main = "Histogramme de l'estimateur (mu)")
```

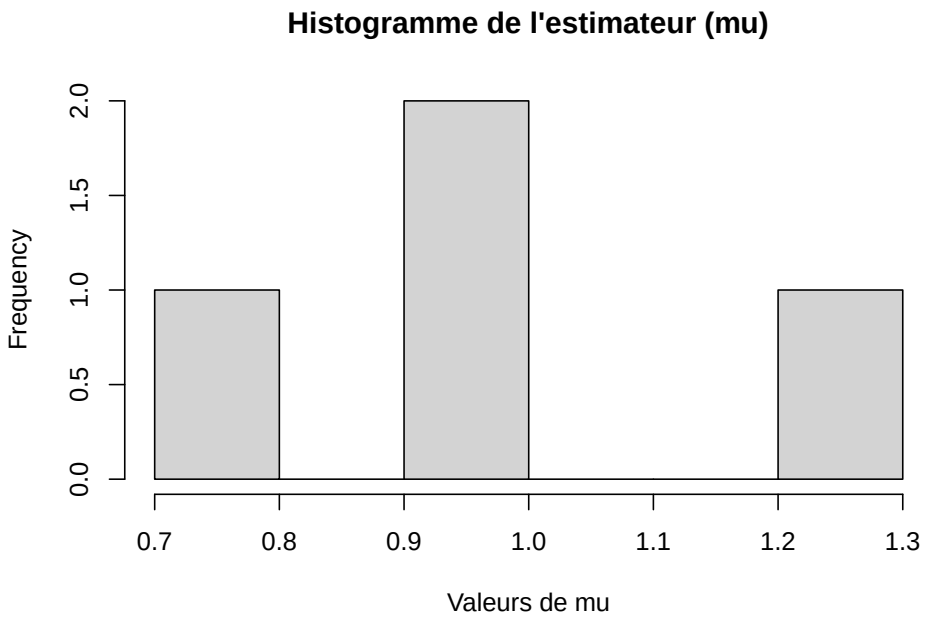



```r
hist(c(emvn_1_10[2], emvn_1_25[2], emvn_1_50[2], emvn_1_100[2]), prob = FALSE, breaks = 4,
     xlab = "Valeurs de sigma", main = "Histogramme de l'estimateur (sigma)")
```

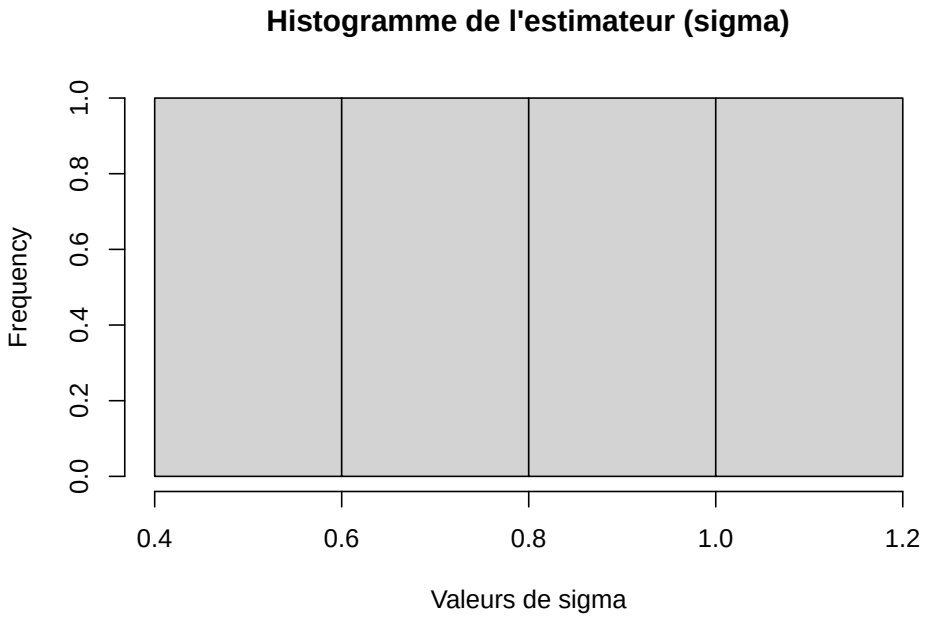


```r
emvn_2_10 = optim(par = c(10, 10), fn = function (theta)
  {log_vraisemblance_norm(theta[1], theta[2], norm_2_10)},
  method = "L-BFGS-B", lower = c(-Inf, 0.1), control = list(fnscale = -1))$par

emvn_2_25 = optim(par = c(10, 10), fn = function (theta)
  {log_vraisemblance_norm(theta[1], theta[2], norm_2_25)},
  method = "L-BFGS-B", lower = c(-Inf, 0.1), control = list(fnscale = -1))$par

emvn_2_50 = optim(par = c(10, 10), fn = function (theta)
  {log_vraisemblance_norm(theta[1], theta[2], norm_2_50)},
  method = "L-BFGS-B", lower = c(-Inf, 0.1), control = list(fnscale = -1))$par

emvn_2_100 = optim(par = c(10, 10), fn = function (theta)
  {log_vraisemblance_norm(theta[1], theta[2], norm_2_100)},
  method = "L-BFGS-B", lower = c(-Inf, 0.1), control = list(fnscale = -1))$par
```

```r
hist(c(emvn_2_10[1], emvn_2_25[1], emvn_2_50[1], emvn_2_100[1]), prob = FALSE, breaks = 4,
     xlab = "Valeurs de mu", main = "Histogramme de l'estimateur (mu)")
```

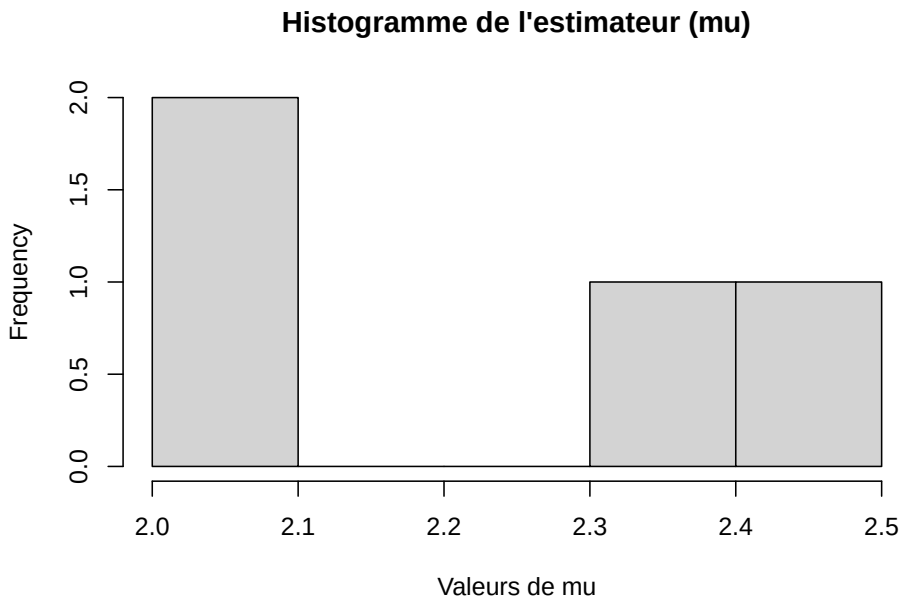


```r
hist(c(emvn_2_10[2], emvn_2_25[2], emvn_2_50[2], emvn_2_100[2]), prob = FALSE, breaks = 4,
     xlab = "Valeurs de sigma", main = "Histogramme de l'estimateur (sigma)")
```

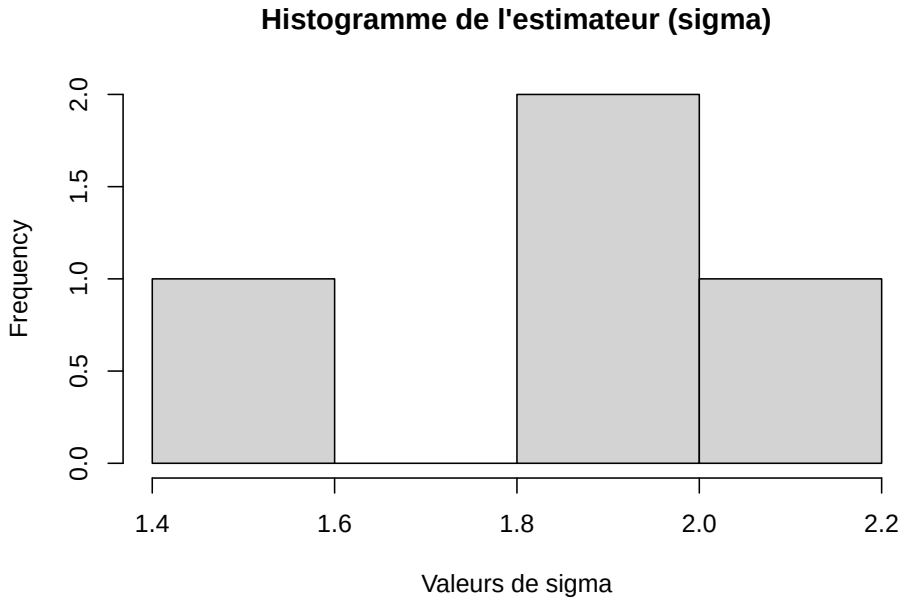



```r
emvn_3_10 = optim(par = c(10, 10), fn = function (theta)
  {log_vraisemblance_norm(theta[1], theta[2], norm_3_10)},
  method = "L-BFGS-B", lower = c(-Inf, 0.1), control = list(fnscale = -1))$par

emvn_3_25 = optim(par = c(10, 10), fn = function (theta)
  {log_vraisemblance_norm(theta[1], theta[2], norm_3_25)},
  method = "L-BFGS-B", lower = c(-Inf, 0.1), control = list(fnscale = -1))$par

emvn_3_50 = optim(par = c(10, 10), fn = function (theta)
  {log_vraisemblance_norm(theta[1], theta[2], norm_3_50)},
  method = "L-BFGS-B", lower = c(-Inf, 0.1), control = list(fnscale = -1))$par

emvn_3_100 = optim(par = c(10, 10), fn = function (theta)
  {log_vraisemblance_norm(theta[1], theta[2], norm_3_100)},
  method = "L-BFGS-B", lower = c(-Inf, 0.1), control = list(fnscale = -1))$par
```

```r
hist(c(emvn_3_10[1], emvn_3_25[1], emvn_3_50[1], emvn_3_100[1]), prob = FALSE, breaks = 4,
     xlab = "Valeurs de mu", main = "Histogramme de l'estimateur (mu)")
```




```r
hist(c(emvn_3_10[2], emvn_3_25[2], emvn_3_50[2], emvn_3_100[2]), prob = FALSE, breaks = 4,
     xlab = "Valeurs de sigma", main = "Histogramme de l'estimateur (sigma)")
```

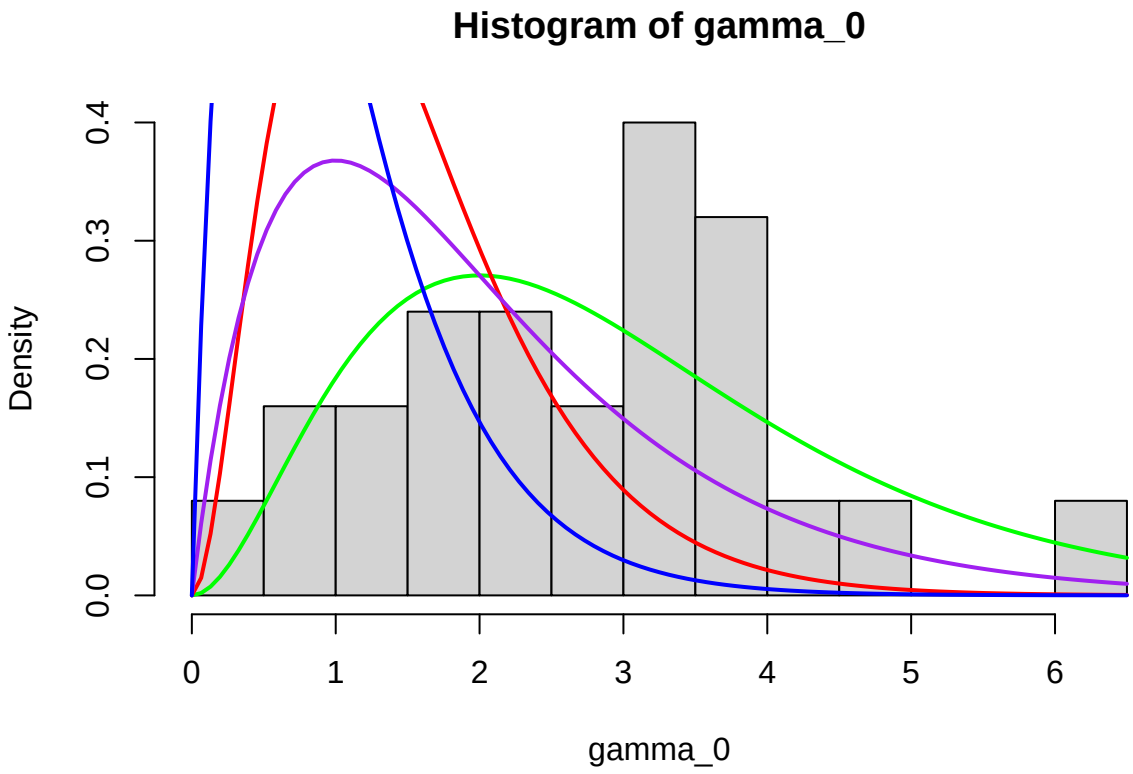


On remarque que plus la taille de l'échantillon est grande, plus la valeur obtenue pour l'estimateur est proche de la valeur théorique. Pour des échantillons de taille 10 ou 25 on a des valeurs trop éloignées, mais pour des échantillons de taille 50 ou 100, on arrive à des valeurs très proches de la valeur théorique.

La Loi Gamma

Question 7 :

```
gamma_0 = rgamma(25, 3, 1)

hist(gamma_0, breaks = 10, prob = TRUE)

curve(dgamma(x, 3, 1), add=TRUE, col="green", lwd=2)
curve(dgamma(x, 3, 2), add=TRUE, col="red", lwd=2)
curve(dgamma(x, 2, 1), add=TRUE, col="purple", lwd=2)
curve(dgamma(x, 2, 2), add=TRUE, col="blue", lwd=2)
```



On remarque que la densité verte correspond bien à une loi gamma désirée : $\Gamma(3,1)$

Tandis que la courbe violette est un peu plus décalée vers la gauche : $\Gamma(2,1)$

La courbe rouge quant à elle est beaucoup plus élargie que la verte : $\Gamma(3,2)$

Pour finir, la courbe bleue est à la fois élargie et décalée vers la gauche : $\Gamma(2,2)$

Question 8 :

La densité de la loi Gamma est :

$$f(x, \alpha, \beta) = \frac{1}{\Gamma(\alpha)} x^{\alpha-1} \beta^{\alpha} e^{-\beta x}$$

Pour n échantillons iid, on a alors la vraisemblance qui est égale à :

$$L(x_1, ..., x_n, \alpha, \beta) = \prod_{i=1}^{n} \frac{1}{\Gamma(\alpha)} x_i^{\alpha-1} \beta^{\alpha} e^{-\beta x_i}$$

$$L(x_1, ..., x_n, \alpha, \beta) = \left(\frac{\beta^{\alpha}}{\Gamma(\alpha)} \right)^{n} \prod_{i=1}^{n} e^{(\alpha-1)\ln(x_i) - \beta x_i}$$

$$L(x_1, ..., x_n, \alpha, \beta) = \left(\frac{\beta^{\alpha}}{\Gamma(\alpha)} \right)^{n} \exp\left(\sum_{i=1}^{n} (\alpha - 1)\ln(x_i) - \beta x_i \right)$$

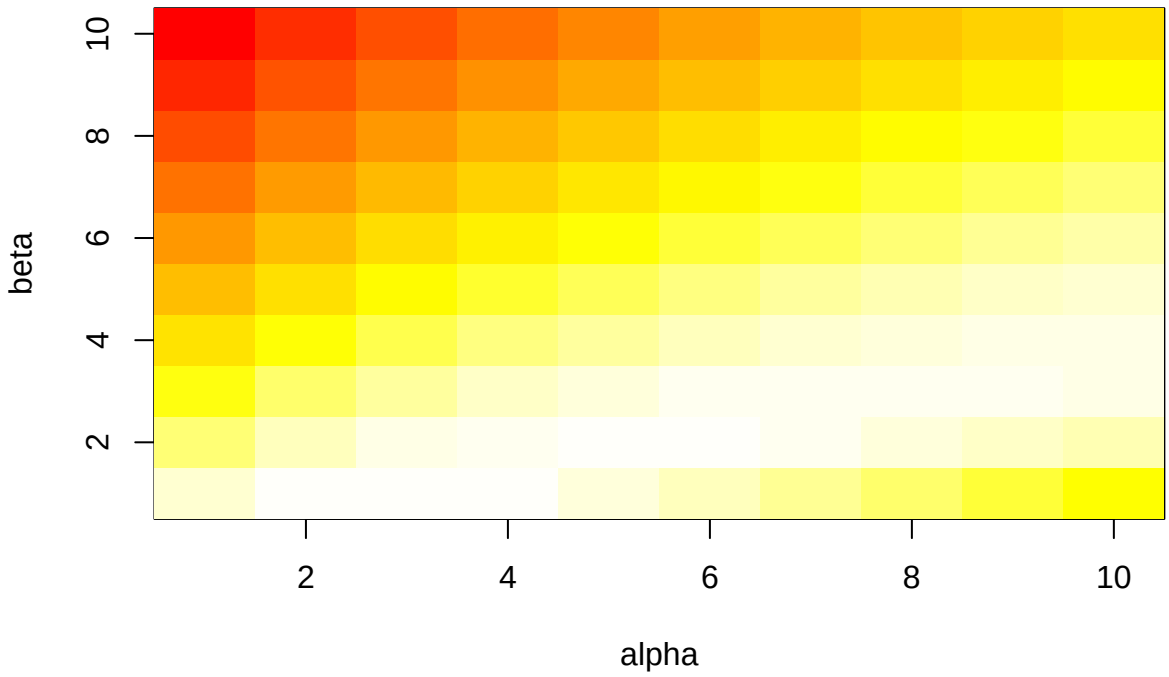
En passant à la log-vraisemblance, on a finalement :

$$\mathcal{L}(x_1, ..., x_n, \alpha, \beta) = n \ (\ \alpha \ln(\beta) - \ln(\Gamma(\alpha)) \ ) + (\alpha - 1) \sum_{i=1}^{n} \ln(x_i) - \beta \sum_{i=1}^{n} x_i$$

```
log_vraisemblance_gamma <- function (alpha, beta, x) {
  n = length(x)
  somme_log = 0
  somme_x = 0
  for (i in 1:n) {
    somme_log = somme_log + log(x[i])
    somme_x = somme_x + x[i]
  }
  return (n * alpha * log(beta) - n * log(gamma(alpha))
          + (alpha - 1) * somme_log - beta * somme_x)
}
```

```
alpha = seq(from = 1, to = 10, by = 1)
beta = seq(from = 1, to = 10, by = 1)
lvg0 = matrix(data = 1, nrow = 10, ncol = 10)
for (i in 1:10) {
  for (j in 1:10) {
    lvg0[i, j] = log_vraisemblance_gamma(alpha[i], beta[j], gamma_0)
  }
}
```

```r
image(x = alpha, y = beta, z = lvg0, col = heat.colors(100))
```



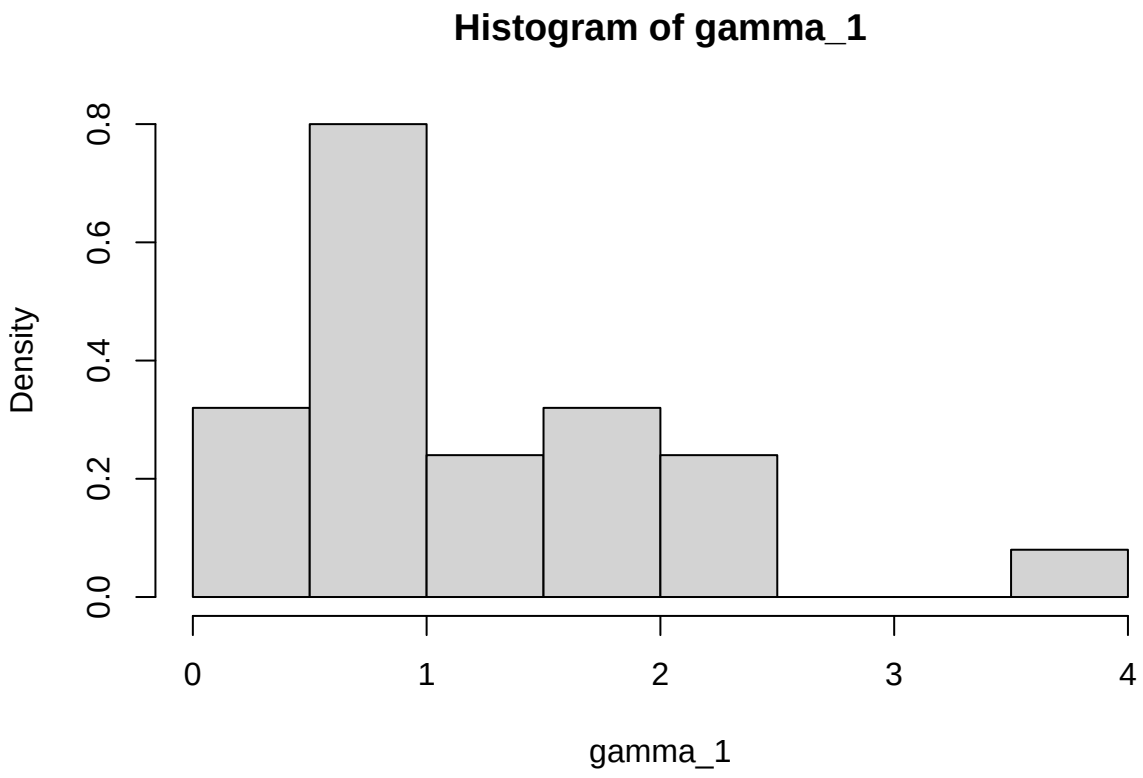
On remarque d'après les variations de couleurs, que le carré le plus clair est le carré correspondant au point $\theta_0 = (3, 1)$. C'est donc le maximum de vraisemblance pour notre échantillon.

Par rapport au cas de la loi normale, où le maximum est localisé en un point, pour la loi gamma le maximum obéit a une relation de linéarite entre les paramètres α et β. Chose qui fait que l'on observe une ligne de couleur très claire.

Pour le cas de la loi normale, le minimum apparaît aux deux extrémités lorsque μ et σ ont des valeurs assez faibles et également lorsque μ a une valeur assez grande tandis que σ a une valeur faible. Pour le cas de la loi gamma, on observe le minimum uniquement à une extrémité lorsque α a une valeur assez faible et que β a une valeur assez grande.

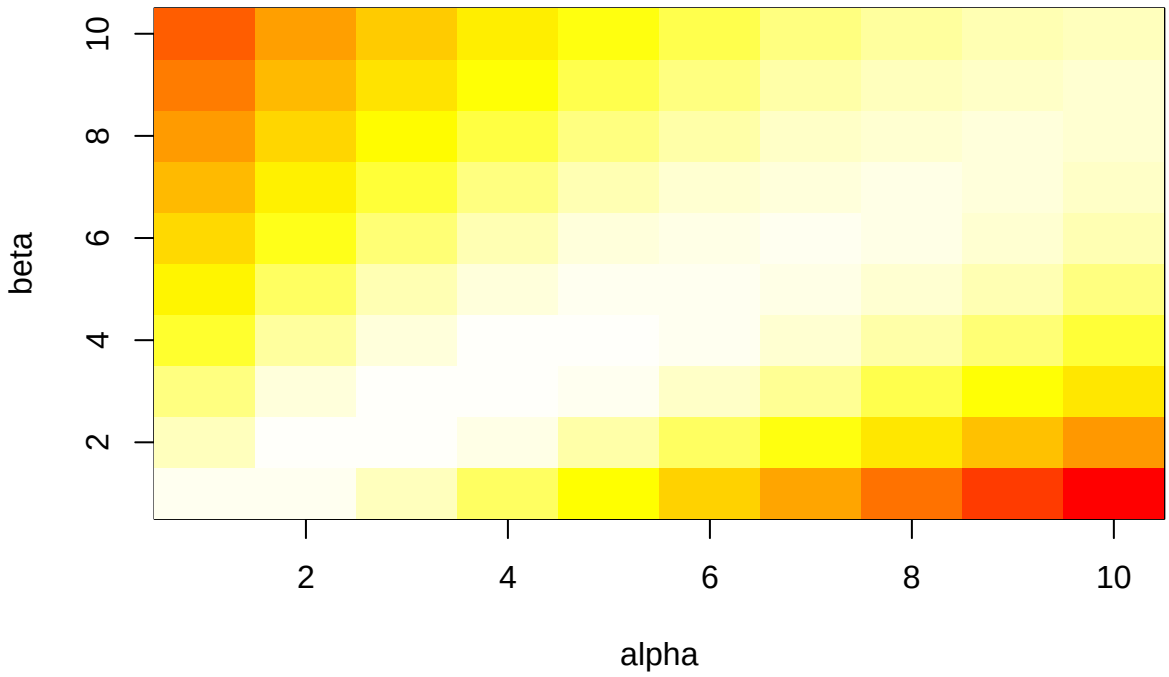
Question 9 :

```
gamma_1 = rgamma(25, 3, 2)

hist(gamma_1, breaks = 10, prob = TRUE)
```



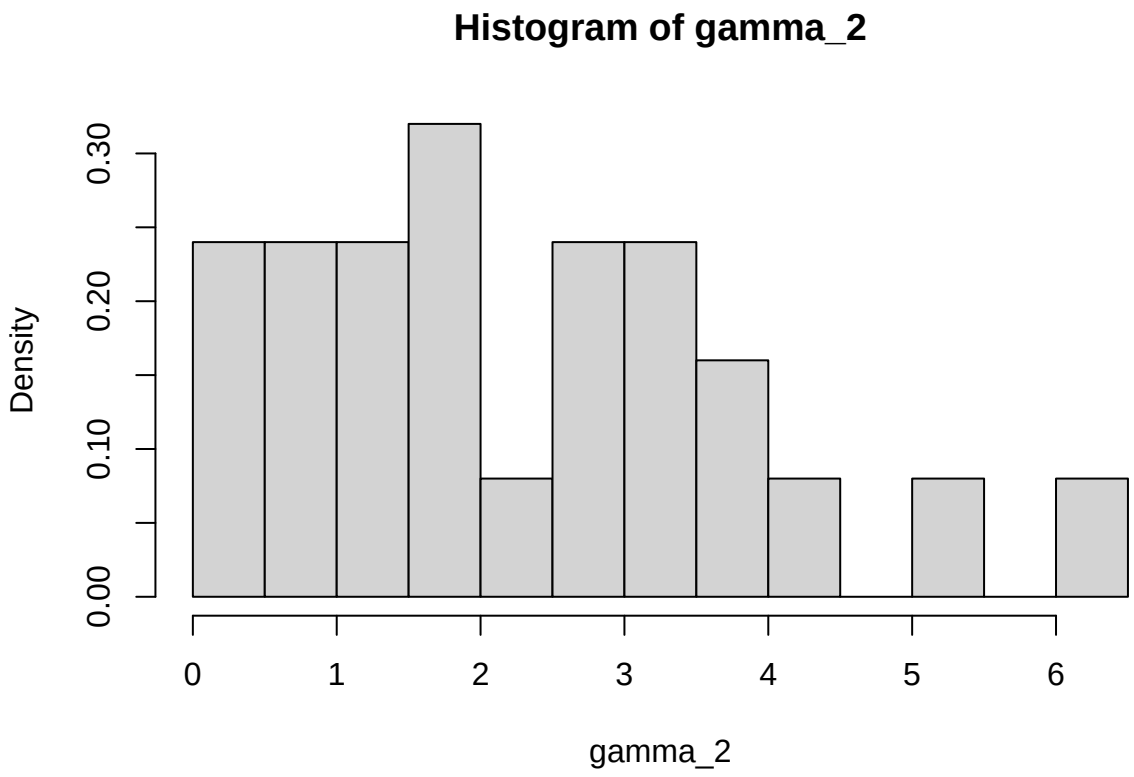

```
lvg1 = matrix(data = 1, nrow = 10, ncol = 10)
for (i in 1:10) {
  for (j in 1:10) {
    lvg1[i, j] = log_vraisemblance_gamma(alpha[i], beta[j], gamma_1)
  }
}
```

```
image(x = alpha, y = beta, z = lvg1, col = heat.colors(100))
```



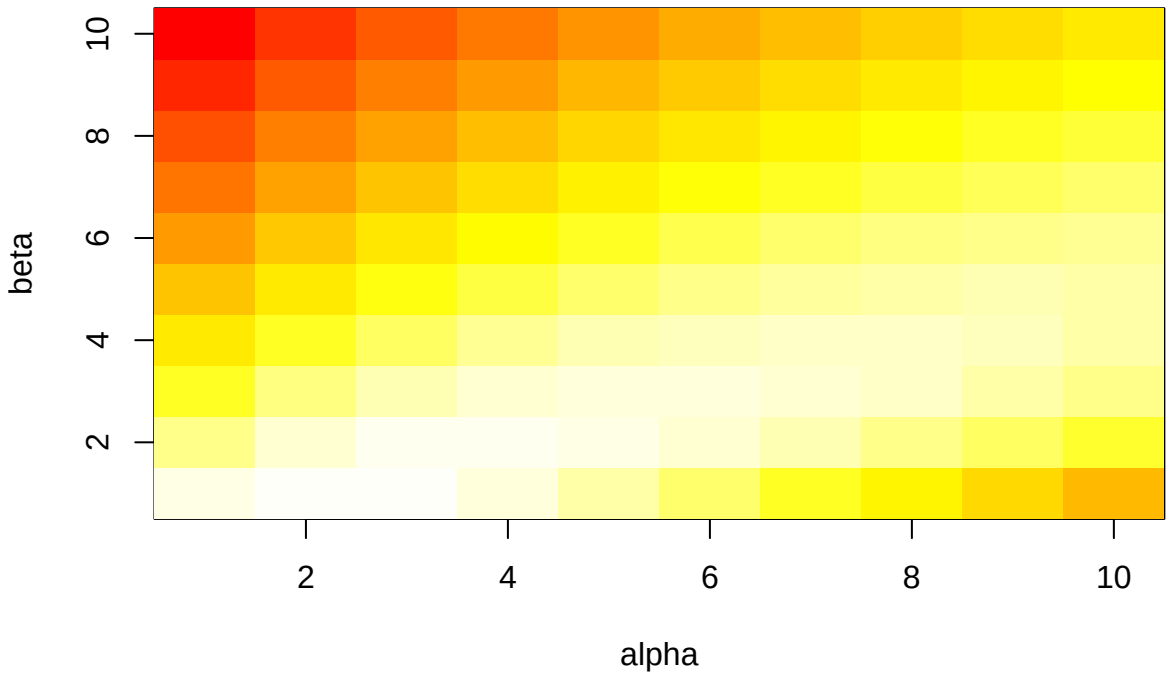
On remarque d'après les variations de couleurs, que le carré le plus clair est le carré correspondant au point $\theta_0 = (3, 2)$. C'est donc le maximum de vraisemblance pour notre échantillon.

```r
gamma_2 = rgamma(25, 2, 1)

hist(gamma_2, breaks = 10, prob = TRUE)
```



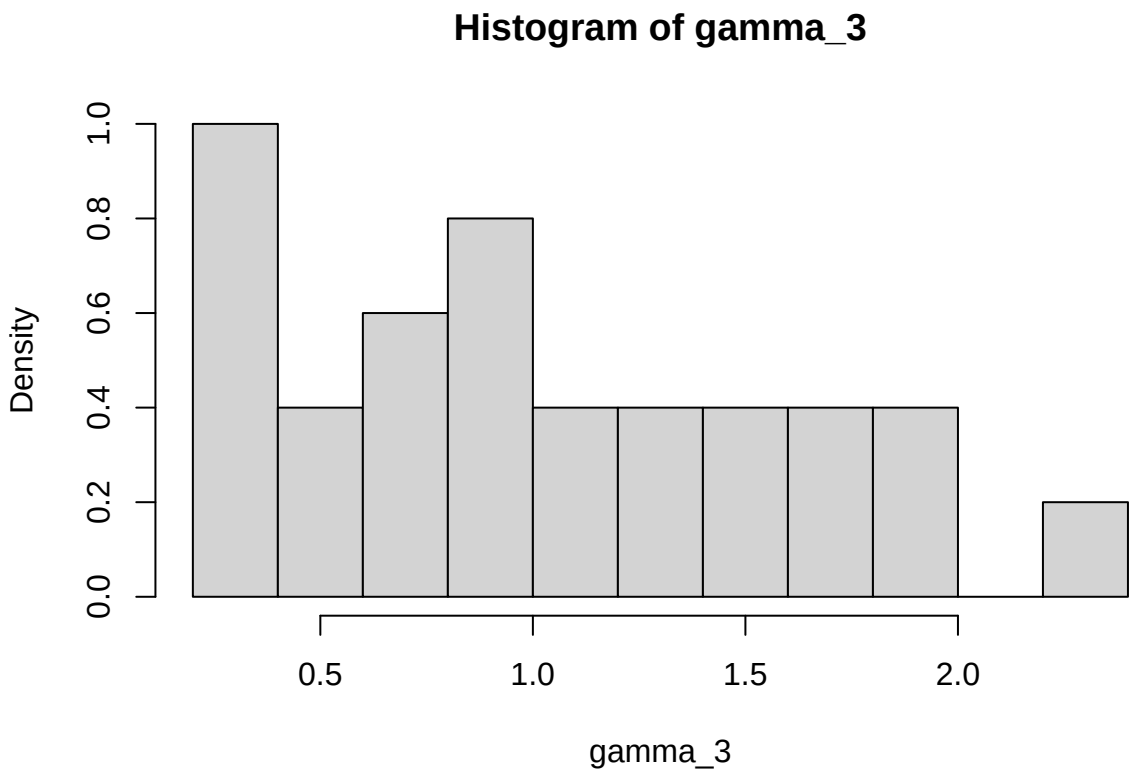

```r
lvg2 = matrix(data = 1, nrow = 10, ncol = 10)
for (i in 1:10) {
  for (j in 1:10) {
    lvg2[i, j] = log_vraisemblance_gamma(alpha[i], beta[j], gamma_2)
  }
}
```

```r
image(x = alpha, y = beta, z = lvg2, col = heat.colors(100))
```



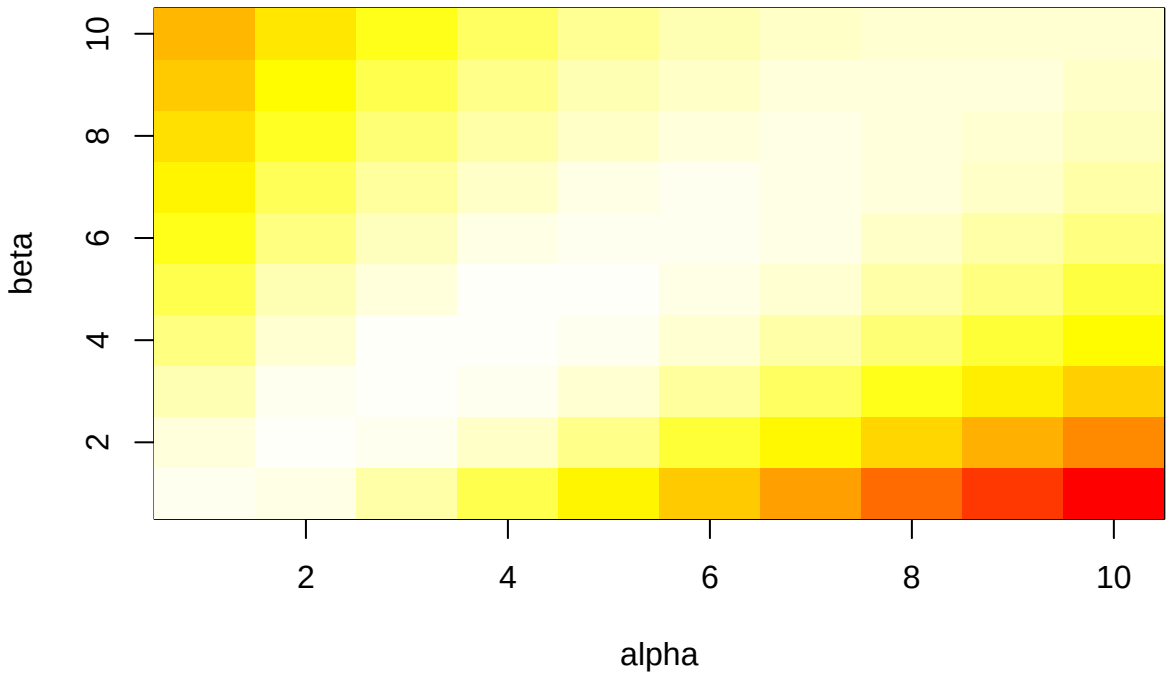
On remarque d'après les variations de couleurs, que le carré le plus clair est le carré correspondant au point $\theta_0 = (2, 1)$. C'est donc le maximum de vraisemblance pour notre échantillon.

```r
gamma_3 = rgamma(25, 2, 2)

hist(gamma_3, breaks = 10, prob = TRUE)
```




```r
lvg3 = matrix(data = 1, nrow = 10, ncol = 10)
for (i in 1:10) {
  for (j in 1:10) {
    lvg3[i, j] = log_vraisemblance_gamma(alpha[i], beta[j], gamma_3)
  }
}
```

```r
image(x = alpha, y = beta, z = lvg3, col = heat.colors(100))
```

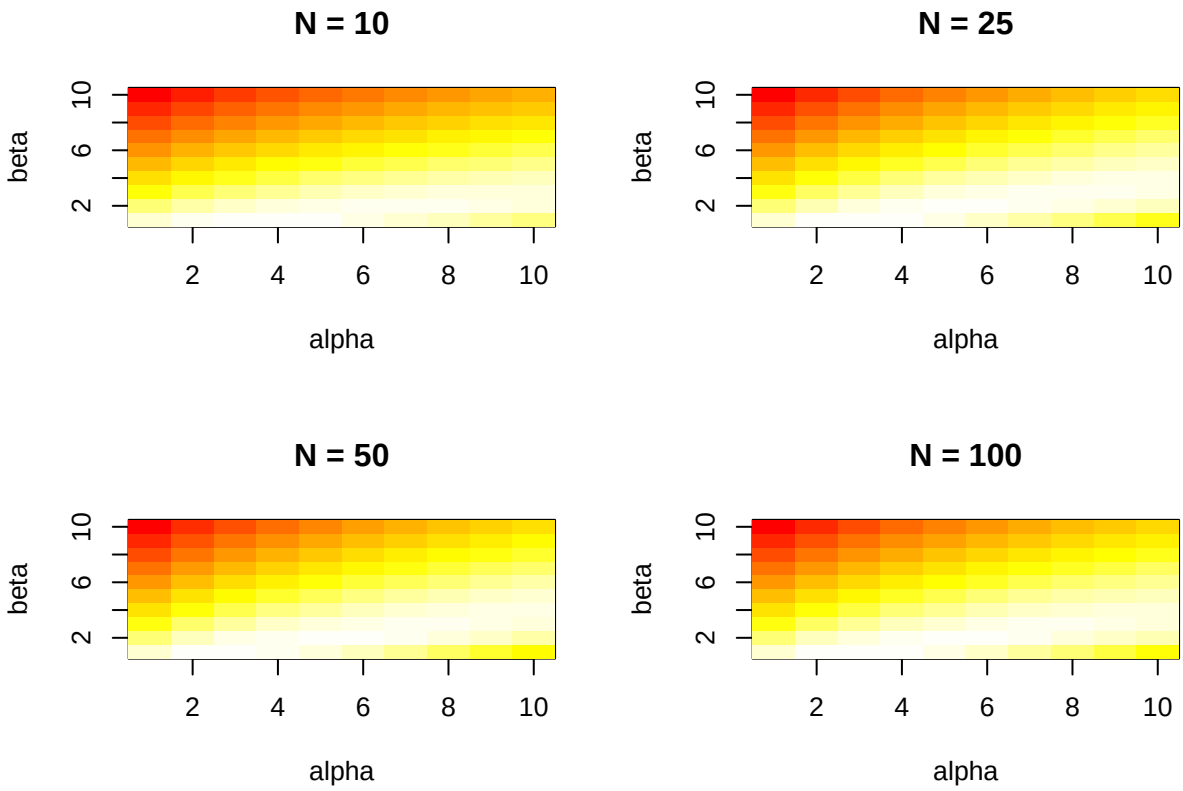


On remarque d'après les variations de couleurs, que le carré le plus clair est le carré correspondant au point $\theta_0 = (2, 2)$. C'est donc le maximum de vraisemblance pour notre échantillon.

Question 10 :

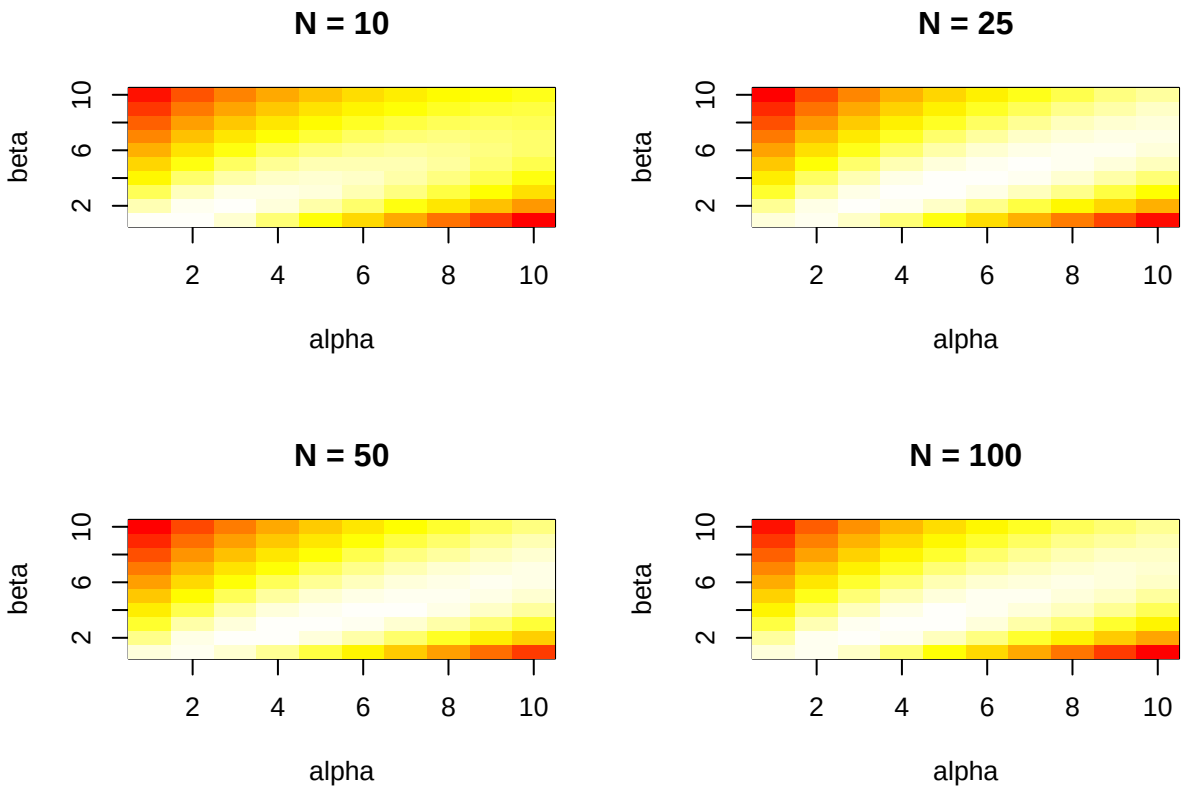
```
gamma_0_10 = rgamma(10, 3, 1)
gamma_0_25 = rgamma(25, 3, 1)
gamma_0_50 = rgamma(50, 3, 1)
gamma_0_100 = rgamma(100, 3, 1)

lvg0_10 = matrix(data = 1, nrow = 10, ncol = 10)
lvg0_25 = matrix(data = 1, nrow = 10, ncol = 10)
lvg0_50 = matrix(data = 1, nrow = 10, ncol = 10)
lvg0_100 = matrix(data = 1, nrow = 10, ncol = 10)

for (i in 1:10) {
  for (j in 1:10) {
    lvg0_10[i, j] = log_vraisemblance_gamma(alpha[i], beta[j], gamma_0_10)
  }
}
for (i in 1:10) {
  for (j in 1:10) {
    lvg0_25[i, j] = log_vraisemblance_gamma(alpha[i], beta[j], gamma_0_25)
  }
}
for (i in 1:10) {
  for (j in 1:10) {
    lvg0_50[i, j] = log_vraisemblance_gamma(alpha[i], beta[j], gamma_0_50)
  }
}
for (i in 1:10) {
  for (j in 1:10) {
    lvg0_100[i, j] = log_vraisemblance_gamma(alpha[i], beta[j], gamma_0_100)
  }
}
```

```r
par(mfrow = c(2, 2))

image(x = alpha, y = beta, z = lvg0_10, col = heat.colors(100), main = "N = 10")
image(x = alpha, y = beta, z = lvg0_25, col = heat.colors(100), main = "N = 25")
image(x = alpha, y = beta, z = lvg0_50, col = heat.colors(100), main = "N = 50")
image(x = alpha, y = beta, z = lvg0_100, col = heat.colors(100), main = "N = 100")
```



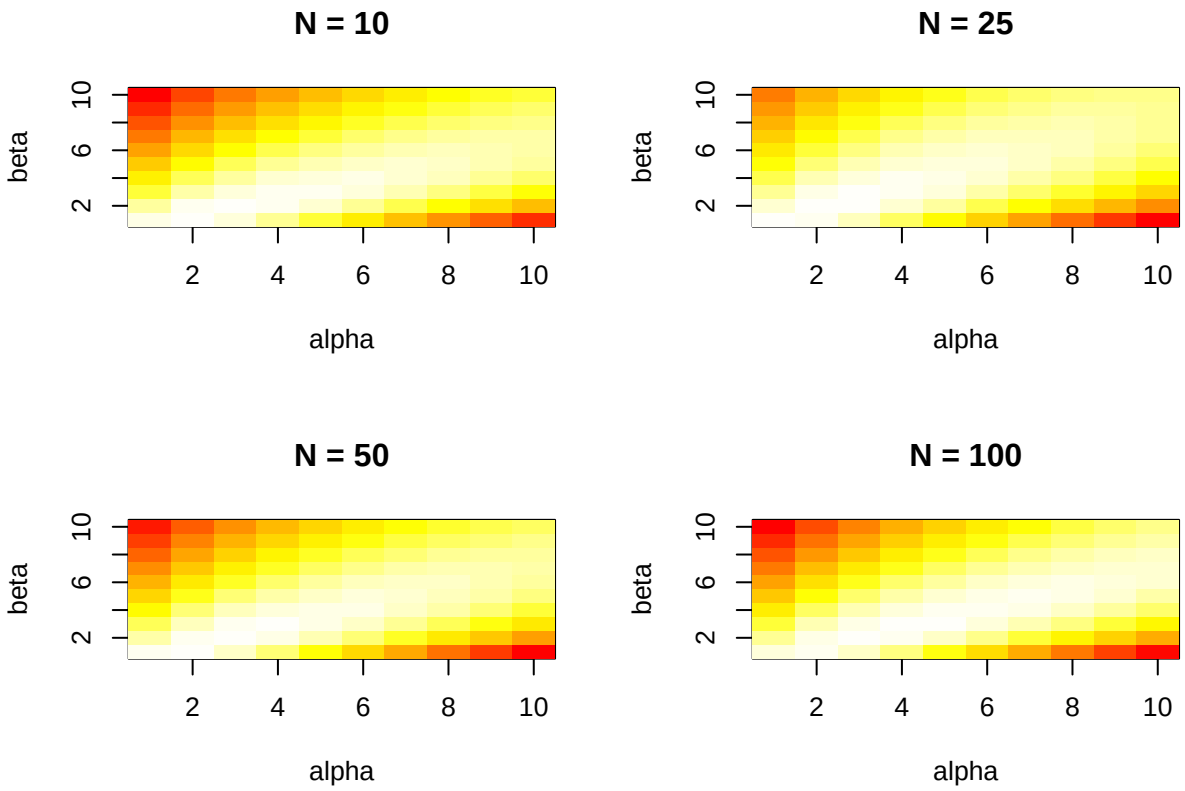
```r
gamma_1_10 = rgamma(10, 3, 2)
gamma_1_25 = rgamma(25, 3, 2)
gamma_1_50 = rgamma(50, 3, 2)
gamma_1_100 = rgamma(100, 3, 2)

lvg1_10 = matrix(data = 1, nrow = 10, ncol = 10)
lvg1_25 = matrix(data = 1, nrow = 10, ncol = 10)
lvg1_50 = matrix(data = 1, nrow = 10, ncol = 10)
lvg1_100 = matrix(data = 1, nrow = 10, ncol = 10)

for (i in 1:10) {
  for (j in 1:10) {
    lvg1_10[i, j] = log_vraisemblance_gamma(alpha[i], beta[j], gamma_1_10)
  }
}
for (i in 1:10) {
  for (j in 1:10) {
    lvg1_25[i, j] = log_vraisemblance_gamma(alpha[i], beta[j], gamma_1_25)
  }
}
for (i in 1:10) {
  for (j in 1:10) {
    lvg1_50[i, j] = log_vraisemblance_gamma(alpha[i], beta[j], gamma_1_50)
  }
}
for (i in 1:10) {
  for (j in 1:10) {
    lvg1_100[i, j] = log_vraisemblance_gamma(alpha[i], beta[j], gamma_1_100)
  }
}
```

```r
par(mfrow = c(2, 2))

image(x = alpha, y = beta, z = lvg1_10, col = heat.colors(100), main = "N = 10")
image(x = alpha, y = beta, z = lvg1_25, col = heat.colors(100), main = "N = 25")
image(x = alpha, y = beta, z = lvg1_50, col = heat.colors(100), main = "N = 50")
image(x = alpha, y = beta, z = lvg1_100, col = heat.colors(100), main = "N = 100")
```



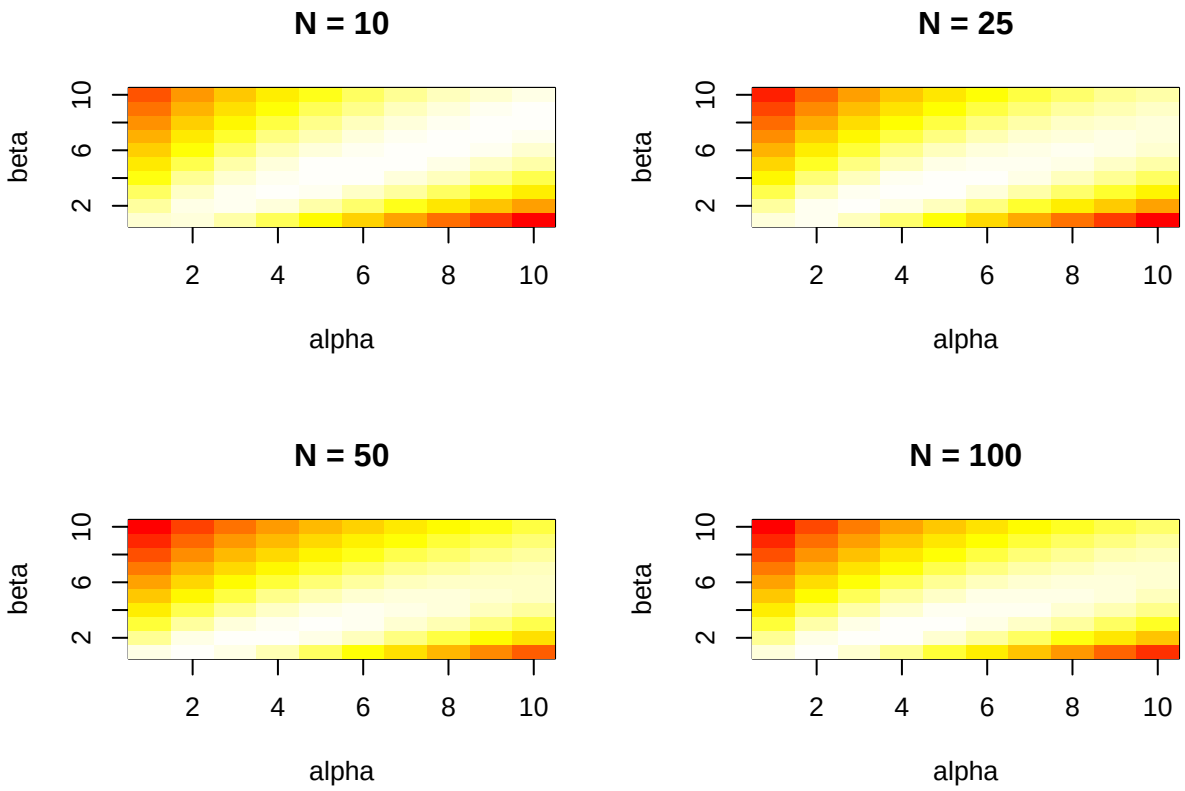
```
gamma_2_10 = rgamma(10, 3, 2)
gamma_2_25 = rgamma(25, 3, 2)
gamma_2_50 = rgamma(50, 3, 2)
gamma_2_100 = rgamma(100, 3, 2)

lvg2_10 = matrix(data = 1, nrow = 10, ncol = 10)
lvg2_25 = matrix(data = 1, nrow = 10, ncol = 10)
lvg2_50 = matrix(data = 1, nrow = 10, ncol = 10)
lvg2_100 = matrix(data = 1, nrow = 10, ncol = 10)

for (i in 1:10) {
  for (j in 1:10) {
    lvg2_10[i, j] = log_vraisemblance_gamma(alpha[i], beta[j], gamma_2_10)
  }
}
for (i in 1:10) {
  for (j in 1:10) {
    lvg2_25[i, j] = log_vraisemblance_gamma(alpha[i], beta[j], gamma_2_25)
  }
}
for (i in 1:10) {
  for (j in 1:10) {
    lvg2_50[i, j] = log_vraisemblance_gamma(alpha[i], beta[j], gamma_2_50)
  }
}
for (i in 1:10) {
  for (j in 1:10) {
    lvg2_100[i, j] = log_vraisemblance_gamma(alpha[i], beta[j], gamma_2_100)
  }
}
```

```r
par(mfrow = c(2, 2))

image(x = alpha, y = beta, z = lvg2_10, col = heat.colors(100), main = "N = 10")
image(x = alpha, y = beta, z = lvg2_25, col = heat.colors(100), main = "N = 25")
image(x = alpha, y = beta, z = lvg2_50, col = heat.colors(100), main = "N = 50")
image(x = alpha, y = beta, z = lvg2_100, col = heat.colors(100), main = "N = 100")
```



```
gamma_3_10 = rgamma(10, 3, 2)
gamma_3_25 = rgamma(25, 3, 2)
gamma_3_50 = rgamma(50, 3, 2)
gamma_3_100 = rgamma(100, 3, 2)

lvg3_10 = matrix(data = 1, nrow = 10, ncol = 10)
lvg3_25 = matrix(data = 1, nrow = 10, ncol = 10)
lvg3_50 = matrix(data = 1, nrow = 10, ncol = 10)
lvg3_100 = matrix(data = 1, nrow = 10, ncol = 10)

for (i in 1:10) {
  for (j in 1:10) {
    lvg3_10[i, j] = log_vraisemblance_gamma(alpha[i], beta[j], gamma_3_10)
  }
}
for (i in 1:10) {
  for (j in 1:10) {
    lvg3_25[i, j] = log_vraisemblance_gamma(alpha[i], beta[j], gamma_3_25)
  }
}
for (i in 1:10) {
  for (j in 1:10) {
    lvg3_50[i, j] = log_vraisemblance_gamma(alpha[i], beta[j], gamma_3_50)
  }
}
for (i in 1:10) {
  for (j in 1:10) {
    lvg3_100[i, j] = log_vraisemblance_gamma(alpha[i], beta[j], gamma_3_100)
  }
}
```

```r
par(mfrow = c(2, 2))

image(x = alpha, y = beta, z = lvg3_10, col = heat.colors(100), main = "N = 10")
image(x = alpha, y = beta, z = lvg3_25, col = heat.colors(100), main = "N = 25")
image(x = alpha, y = beta, z = lvg3_50, col = heat.colors(100), main = "N = 50")
image(x = alpha, y = beta, z = lvg3_100, col = heat.colors(100), main = "N = 100")
```



Tout comme pour le cas normal, on peut dire que plus la taille de l'échantillon est grande, plus le calcul de la log-vraisemblance est précis. Par conséquent on arrive à voir beaucoup plus nettement le carré le plus clair sur la représenation graphique de la surface de la log-vraisemblance. Cela nous permet de déterminer de manière plus précise le maximum de vraisemblance.

Question 11 :

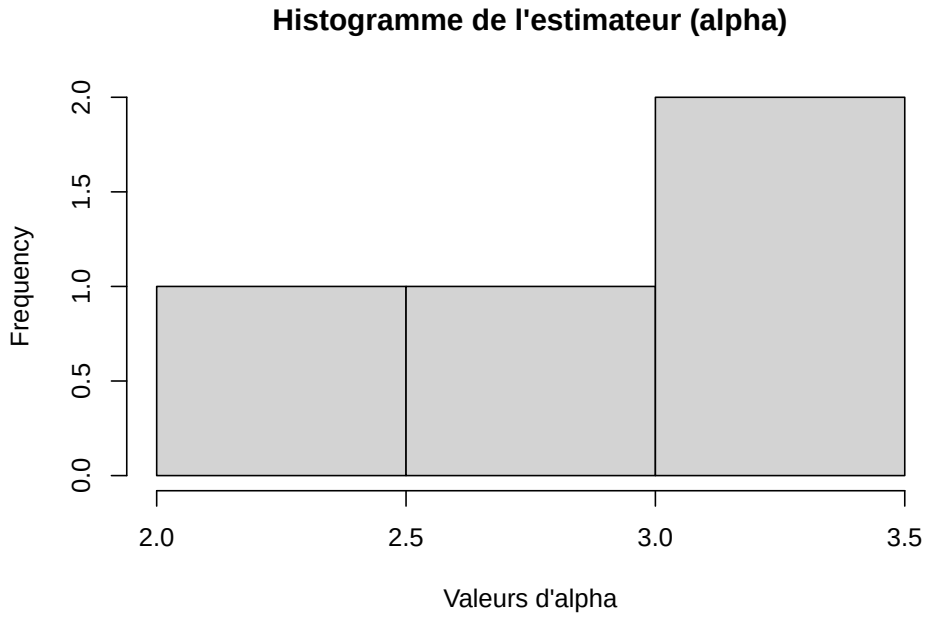
```
optim(par = c(1, 1), fn = function (theta)
  {log_vraisemblance_gamma(theta[1], theta[2], gamma_0)}, method = "L-BFGS-B", lower = c(0, 0),
  control = list(fnscale = -1))$par
```

```
[1] 3.350924 1.199954
```

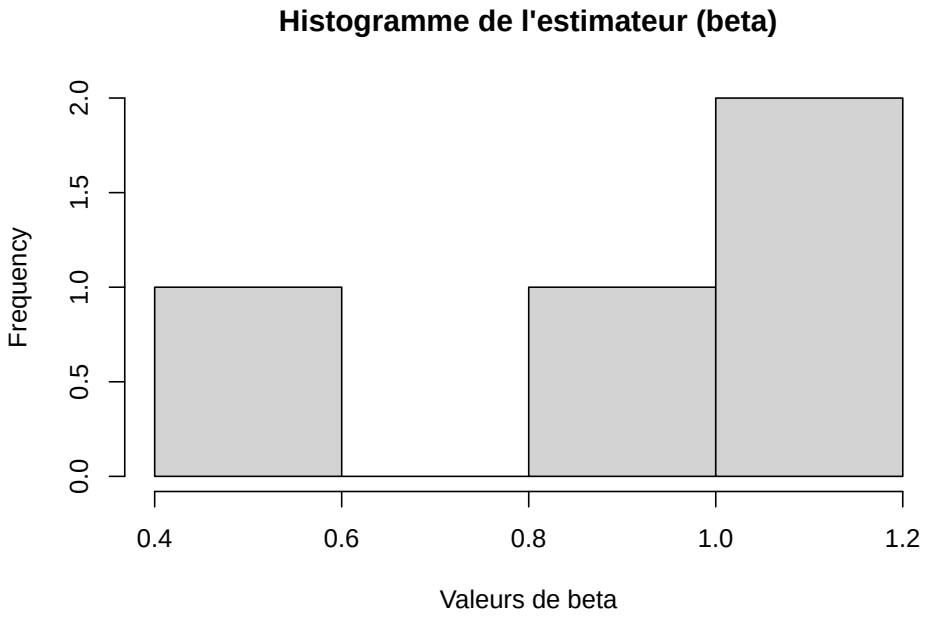
Question 12 :

```
emvg_0_10 = optim(par = c(1, 1), fn = function (theta)
  {log_vraisemblance_gamma(theta[1], theta[2], gamma_0_10)},
  method = "L-BFGS-B", lower = c(0, 0), control = list(fnscale = -1))$par

emvg_0_25 = optim(par = c(1, 1), fn = function (theta)
  {log_vraisemblance_gamma(theta[1], theta[2], gamma_0_25)},
  method = "L-BFGS-B", lower = c(0, 0), control = list(fnscale = -1))$par

emvg_0_50 = optim(par = c(1, 1), fn = function (theta)
  {log_vraisemblance_gamma(theta[1], theta[2], gamma_0_50)},
  method = "L-BFGS-B", lower = c(0, 0), control = list(fnscale = -1))$par

emvg_0_100 = optim(par = c(1, 1), fn = function (theta)
  {log_vraisemblance_gamma(theta[1], theta[2], gamma_0_100)},
  method = "L-BFGS-B", lower = c(0, 0), control = list(fnscale = -1))$par
```

```
hist(c(emvg_0_10[1], emvg_0_25[1], emvg_0_50[1], emvg_0_100[1]), prob = FALSE, breaks = 4,
     xlab = "Valeurs d'alpha", main = "Histogramme de l'estimateur (alpha)")
```

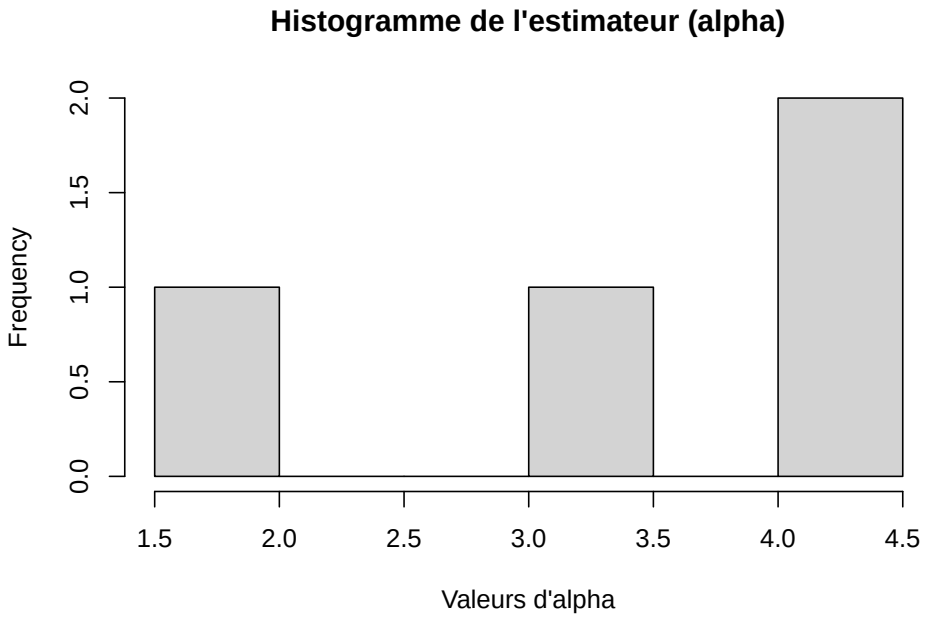


```r
hist(c(emvg_0_10[2], emvg_0_25[2], emvg_0_50[2], emvg_0_100[2]), prob = FALSE, breaks = 4,
     xlab = "Valeurs de beta", main = "Histogramme de l'estimateur (beta)")
```

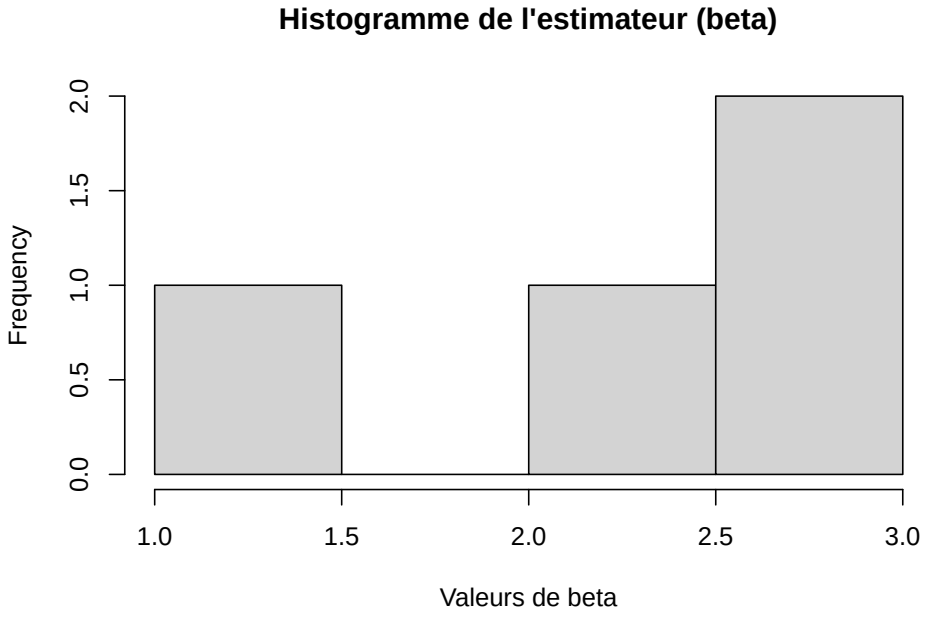



```r
emvg_1_10 = optim(par = c(1, 1), fn = function (theta)
  {log_vraisemblance_gamma(theta[1], theta[2], gamma_1_10)},
  method = "L-BFGS-B", lower = c(0, 0), control = list(fnscale = -1))$par

emvg_1_25 = optim(par = c(1, 1), fn = function (theta)
  {log_vraisemblance_gamma(theta[1], theta[2], gamma_1_25)},
  method = "L-BFGS-B", lower = c(0, 0), control = list(fnscale = -1))$par

emvg_1_50 = optim(par = c(1, 1), fn = function (theta)
  {log_vraisemblance_gamma(theta[1], theta[2], gamma_1_50)},
  method = "L-BFGS-B", lower = c(0, 0), control = list(fnscale = -1))$par

emvg_1_100 = optim(par = c(1, 1), fn = function (theta)
  {log_vraisemblance_gamma(theta[1], theta[2], gamma_1_100)},
  method = "L-BFGS-B", lower = c(0, 0), control = list(fnscale = -1))$par
```

```r
hist(c(emvg_1_10[1], emvg_1_25[1], emvg_1_50[1], emvg_1_100[1]), prob = FALSE, breaks = 4,
     xlab = "Valeurs d'alpha", main = "Histogramme de l'estimateur (alpha)")
```

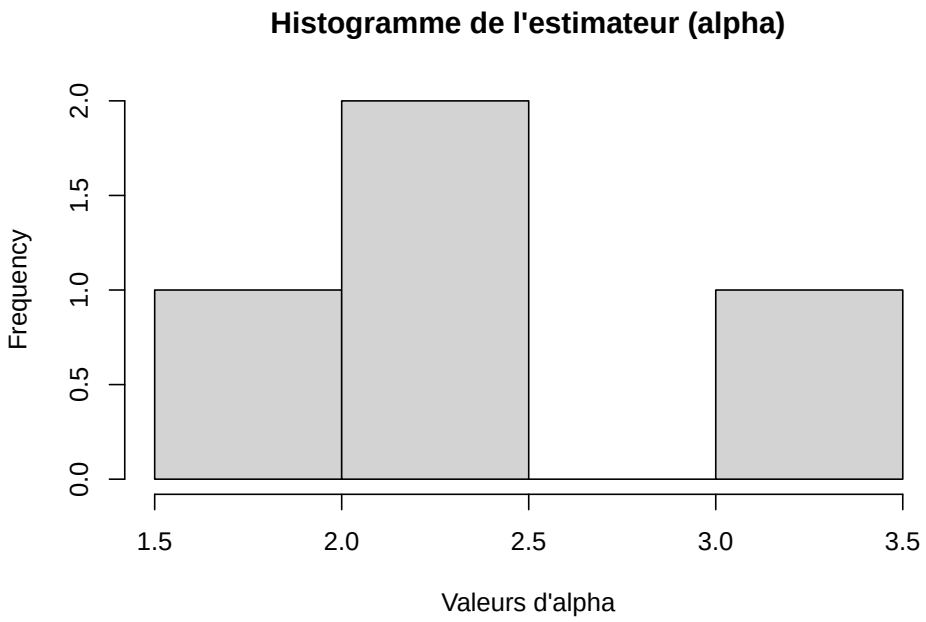



```r
hist(c(emvg_1_10[2], emvg_1_25[2], emvg_1_50[2], emvg_1_100[2]), prob = FALSE, breaks = 4,
     xlab = "Valeurs de beta", main = "Histogramme de l'estimateur (beta)")
```

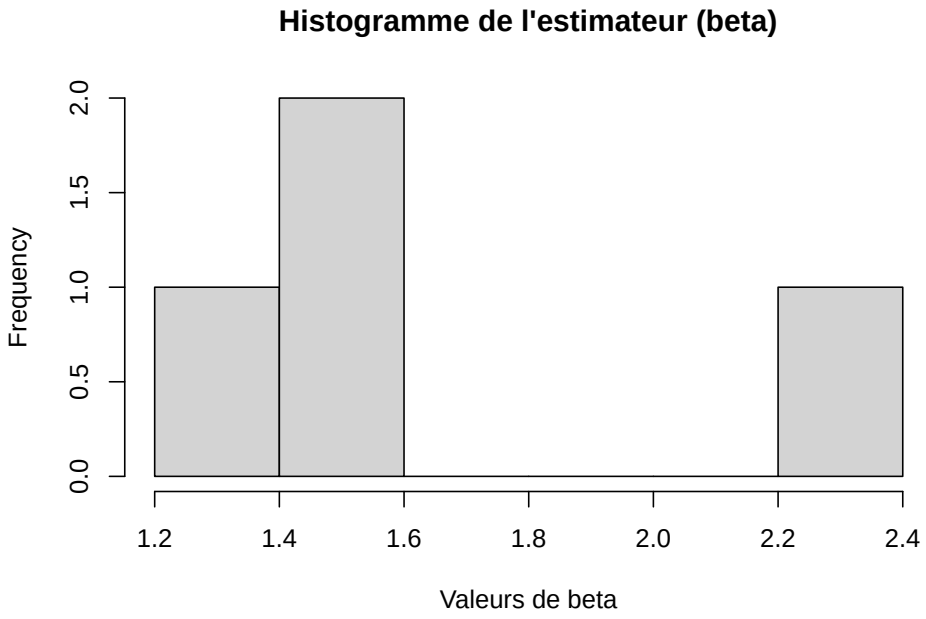


```r
emvg_2_10 = optim(par = c(1, 1), fn = function (theta)
  {log_vraisemblance_gamma(theta[1], theta[2], gamma_2_10)},
  method = "L-BFGS-B", lower = c(0, 0), control = list(fnscale = -1))$par

emvg_2_25 = optim(par = c(1, 1), fn = function (theta)
  {log_vraisemblance_gamma(theta[1], theta[2], gamma_2_25)},
  method = "L-BFGS-B", lower = c(0, 0), control = list(fnscale = -1))$par

emvg_2_50 = optim(par = c(1, 1), fn = function (theta)
  {log_vraisemblance_gamma(theta[1], theta[2], gamma_2_50)},
  method = "L-BFGS-B", lower = c(0, 0), control = list(fnscale = -1))$par

emvg_2_100 = optim(par = c(1, 1), fn = function (theta)
  {log_vraisemblance_gamma(theta[1], theta[2], gamma_2_100)},
  method = "L-BFGS-B", lower = c(0, 0), control = list(fnscale = -1))$par
```

```r
hist(c(emvg_2_10[1], emvg_2_25[1], emvg_2_50[1], emvg_2_100[1]), prob = FALSE, breaks = 4,
     xlab = "Valeurs d'alpha", main = "Histogramme de l'estimateur (alpha)")
```

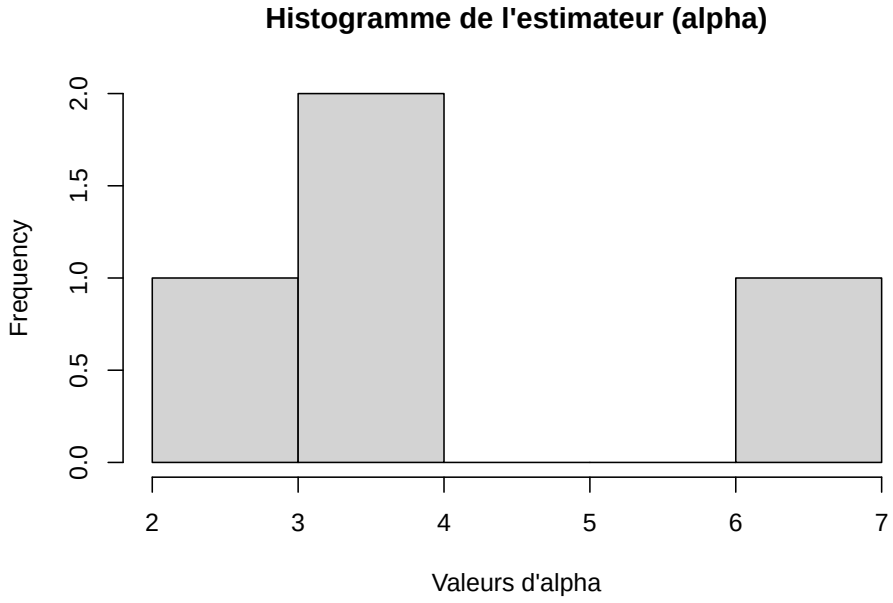


```r
hist(c(emvg_2_10[2], emvg_2_25[2], emvg_2_50[2], emvg_2_100[2]), prob = FALSE, breaks = 4,
     xlab = "Valeurs de beta", main = "Histogramme de l'estimateur (beta)")
```

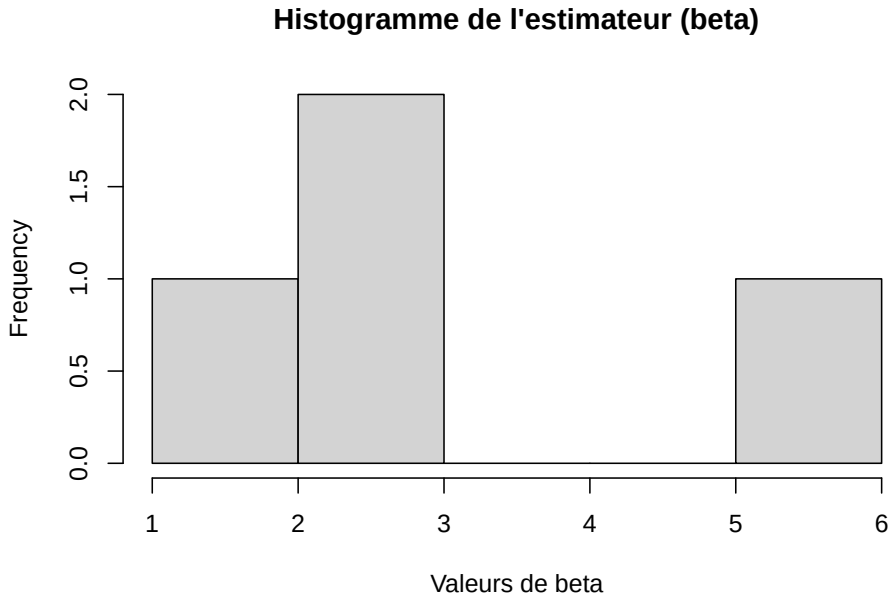



```r
emvg_3_10 = optim(par = c(1, 1), fn = function (theta)
  {log_vraisemblance_gamma(theta[1], theta[2], gamma_3_10)},
  method = "L-BFGS-B", lower = c(0, 0), control = list(fnscale = -1))$par

emvg_3_25 = optim(par = c(1, 1), fn = function (theta)
  {log_vraisemblance_gamma(theta[1], theta[2], gamma_3_25)},
  method = "L-BFGS-B", lower = c(0, 0), control = list(fnscale = -1))$par

emvg_3_50 = optim(par = c(1, 1), fn = function (theta)
  {log_vraisemblance_gamma(theta[1], theta[2], gamma_3_50)},
  method = "L-BFGS-B", lower = c(0, 0), control = list(fnscale = -1))$par

emvg_3_100 = optim(par = c(1, 1), fn = function (theta)
  {log_vraisemblance_gamma(theta[1], theta[2], gamma_3_100)},
  method = "L-BFGS-B", lower = c(0, 0), control = list(fnscale = -1))$par
```

```r
hist(c(emvg_3_10[1], emvg_3_25[1], emvg_3_50[1], emvg_3_100[1]), prob = FALSE, breaks = 4,
     xlab = "Valeurs d'alpha", main = "Histogramme de l'estimateur (alpha)")
```




```r
hist(c(emvg_3_10[2], emvg_3_25[2], emvg_3_50[2], emvg_3_100[2]), prob = FALSE, breaks = 4,
     xlab = "Valeurs de beta", main = "Histogramme de l'estimateur (beta)")
```



On remarque que plus la taille de l'échantillon est grande, plus la valeur obtenue pour l'estimateur est proche de la valeur théorique. Pour des échantillons de petite taille on a des valeurs trop éloignées, mais pour des échantillons de grande taille, on arrive à des valeurs très proches de la valeur théorique.

Application aux données sur l'ozone

Question 13 :

```
summer_ozone <- read.csv("summer_ozone.csv")
winter_ozone <- read.csv("winter_ozone.csv")
summer_neuil <- summer_ozone$NEUIL
summer_rur <- summer_ozone$RUR.SE
winter_neuil <- winter_ozone$NEUIL
winter_rur <- winter_ozone$RUR.SE
```

```
emvn_SN = optim(par = c(10, 10), fn = function (theta)
  {log_vraisemblance_norm(theta[1], theta[2], summer_neuil)},
  method = "L-BFGS-B", lower = c(-Inf, 0.1), control = list(fnscale = -1))$par

emvn_SR = optim(par = c(10, 10), fn = function (theta)
  {log_vraisemblance_norm(theta[1], theta[2], summer_rur)},
  method = "L-BFGS-B", lower = c(-Inf, 0.1), control = list(fnscale = -1))$par

emvn_WN = optim(par = c(10, 10), fn = function (theta)
  {log_vraisemblance_norm(theta[1], theta[2], winter_neuil)},
  method = "L-BFGS-B", lower = c(-Inf, 0.1), control = list(fnscale = -1))$par

emvn_WR = optim(par = c(10, 10), fn = function (theta)
  {log_vraisemblance_norm(theta[1], theta[2], winter_rur)},
  method = "L-BFGS-B", lower = c(-Inf, 0.1), control = list(fnscale = -1))$par
```

La densité de la loi log-normale est :

$$f(x, \mu, \sigma) = \frac{1}{x\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{\ln(x)-\mu}{\sigma}\right)^2}$$

Pour n échantillons iid, on a alors la vraisemblance qui est égale à:

$$L(x_1, ..., x_n, \mu, \sigma) = \prod_{i=1}^{n} \frac{1}{x_i\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{\ln(x_i)-\mu}{\sigma}\right)^2}$$

$$L(x_1, ..., x_n, \mu, \sigma) = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n \exp\left(-\frac{1}{2}\sum_{i=1}^{n}\left(\frac{\ln(x_i)-\mu}{\sigma}\right)^2\right) \prod_{j=1}^{n}\frac{1}{x_j}$$

En passant à la log-vraisemblance, on a finalement:

$$\mathcal{L}(x_1, ..., x_n, \mu, \sigma) = -\frac{n}{2}\ln(2\pi) - n\ln(\sigma) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(\ln(x_i)-\mu)^2 - \sum_{i=1}^{n}\ln(x_i)$$

```r
log_vraisemblance_log_norm <- function (mu, sigma, x) {
  n = length(x)
  somme_lx = 0
  somme_lx_u = 0
  for (i in 1:n) {
    if (x[i] == 0) x[i] = 1
    somme_lx = somme_lx + log(x[i])
    somme_lx_u = somme_lx_u + ((log(x[i]) - mu) ^ 2)
  }
  return ((-n / 2) * (log(2 * pi)) + (-n) * log(sigma)
          - (1 / 2 * (sigma ^ 2)) * somme_lx_u - somme_lx)
}
```

```r
emvln_SN = optim(par = c(1, 1), fn = function (theta)
  {log_vraisemblance_log_norm(theta[1], theta[2], summer_neuil)},
  method = "L-BFGS-B", lower = c(-Inf, 0.01), control = list(fnscale = -1))$par

emvln_SR = optim(par = c(1, 1), fn = function (theta)
  {log_vraisemblance_log_norm(theta[1], theta[2], summer_rur)},
  method = "L-BFGS-B", lower = c(-Inf, 0.01), control = list(fnscale = -1))$par

emvln_WN = optim(par = c(1, 1), fn = function (theta)
  {log_vraisemblance_log_norm(theta[1], theta[2], winter_neuil)},
  method = "L-BFGS-B", lower = c(-Inf, 0.01), control = list(fnscale = -1))$par

emvln_WR = optim(par = c(1, 1), fn = function (theta)
  {log_vraisemblance_log_norm(theta[1], theta[2], winter_rur)},
  method = "L-BFGS-B", lower = c(-Inf, 0.01), control = list(fnscale = -1))$par
```

On remarque que dans l'échantillon winter_neuil, il y a des valeurs nulles. On modifie donc notre log vraisemblance de manière à changer les 0 en 1 pour pouvoir faire les calculs nécessaires pour la loi log normale.

```r
emvn_SN - emvln_SN
```

```
[1] 82.30250 30.96572
```

```r
emvn_SR - emvln_SR
```

```
[1] 88.43112 26.57399
```

```r
emvn_WN - emvln_WN
```

```
[1] 35.63971 19.19698
```

```r
emvn_WR - emvln_WR
```

```
[1] 53.58478 18.82349
```

La différence est considèrable entre la log-vraisemblance des échantillons en considérant une loi noramle ou une loi log normale. On trouve des valeurs bien plus grande pour la loi normale que pour la loi log normale.

On préfère ainsi la loi normale pour sa facilité d'interprétation. Son espérance, sa variance et son écart-type sont beaucoup plus simples et rapides à calculer que pour la loi log normale, qui elle peut s'avérer utile pour certains modèles où elle colle parfaitement à l'échantillon. Ce qui n'est pas le cas pour nos échantillons ici.