TP 5 Statistiques

Adib Habbou, Adel Kebli, Clark Ji

11/03/2022

Normalité asymptotique de l'EMV et intervalle de confiance

Question 1 :

```
norm_10 = rnorm(10, 2, 1)
```

```
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)
lvn = matrix(data = 1, nrow = 10, ncol = 10)
for (i in 1:10) {
  for (j in 1:10) {
    lvn[i, j] = log_vraisemblance_norm(mu[i], sigma[j], norm_10)
  }
}
```

```
estim = optim(par = c(10, 10),
              fn = function (theta) {log_vraisemblance_norm(theta[1], theta[2], norm_10)},
              method = "L-BFGS-B", lower = c(-Inf, 0.1), control = list(fnscale = -1))$par
estim_mu = estim[1]
estim_sigma = estim[2]
```

On utilise le maximum de vraisemblance pour trouver des estimateurs pour μ et σ. Les valeurs trouvées sont les suivantes :

```
estim_mu
```

```
[1] 1.93956
```

```
estim_sigma
```

```
[1] 0.8464397
```

Déterminons un intervalle de confiance pour μ précis à 95%

Soit $X = (X_1, ...X_n)$ avec $\forall i \in [\![1, n]\!],\ X_i \sim \mathcal{N}(\mu, \sigma^2)$ avec σ connu.

On a alors:

$$\overline{X_n} = \frac{1}{n}\sum_{i=1}^{n} X_i \sim \mathcal{N}\left(\mu, \frac{\sigma^2}{n}\right)$$

On sait donc que

$$Z = \sqrt{n}\frac{\overline{X_n} - \mu}{\sigma} \sim \mathcal{N}(0, 1)$$

Soit Φ la fonction de répartition de la loi $\mathcal{N}(0, 1)$

Soient Q_α les quantiles d'ordres α tels que $\Phi(Q_\alpha) = \alpha$

Comme $Z \sim \mathcal{N}(0, 1)$, on a donc

$$\mathbb{P}\left(Q_{\frac{\alpha}{2}} \leq Z \leq Q_{1-\frac{\alpha}{2}}\right) = 1 - \alpha$$

soit en utilisant la propriété $Q_{1-\alpha} = -Q_\alpha$

On a donc

$$\mathbb{P}\left(-Q_{1-\frac{\alpha}{2}} \leq Z \leq Q_{1-\frac{\alpha}{2}}\right) = 1 - \alpha$$

On cherche un intervalle de confiance pour μ précis à 95%, donc on prend $\alpha = 0,05$

On a alors

$$\mathbb{P}\left(-Q_{0,975} \leq Z \leq Q_{0,975}\right) = 0,95$$

soit

$$\mathbb{P}\left(-\Phi^{-1}(0,975) \leq Z \leq \Phi^{-1}(0,975)\right) = 0,95$$

soit

$$\mathbb{P}\left(-\Phi^{-1}(0,975) \leq \sqrt{n}\frac{\overline{X_n} - \mu}{\sigma} \leq \Phi^{-1}(0,975)\right) = 0,95$$

soit

$$\mathbb{P}\left(\overline{X_n} - \frac{\sigma}{\sqrt{n}}\Phi^{-1}(0,975) \leq \mu \leq \overline{X_n} + \frac{\sigma}{\sqrt{n}}\Phi^{-1}(0,975)\right) = 0,95$$

On a donc finalement l'intervalle de confiance à 95% pour μ,

$$\mathcal{I}_n(\sigma) = \left[\overline{X_n} - \frac{\sigma}{\sqrt{n}}\Phi^{-1}(0,975), \overline{X_n} + \frac{\sigma}{\sqrt{n}}\Phi^{-1}(0,975)\right]$$

```
moyenne_empirique <- function (norm) {
  n = length(norm)
  somme = 0
  for (x in norm) {
    somme = somme + x
  }
  return (somme / n)
}
```

```
xN = moyenne_empirique(norm_10)
z_alpha = qnorm(0.975, mean = 0, sd = 1)
coeff = z_alpha * sqrt((estim_sigma ^ 2) / length(norm_10))
borne_inf = xN - coeff
borne_sup = xN + coeff
```

L'intervalle trouvé est le suivant :

```
c(borne_inf,borne_sup)
```

```
[1] 1.414940 2.464179
```

Cet intervalle est cohérent puisque μ vaut :

```
estim_mu
```

```
[1] 1.93956
```

Question 2 :

```
cpt_10 = 0
estim_MV_mu = matrix(data = 1, nrow = 1, ncol = 100)
estim_MV_sigma = matrix(data = 1, nrow = 1, ncol = 100)
for (i in seq(1, 100, 1)) {
  norm_10 = rnorm(10, 2, 1)
  estim = optim(par = c(10, 10),
                fn = function (theta) {log_vraisemblance_norm(theta[1], theta[2], norm_10)},
                method = "L-BFGS-B", lower = c(-Inf, 0.1), control = list(fnscale = -1))$par
  estim_mu = estim[1]
  estim_sigma = estim[2]
  estim_MV_mu[1, i] = estim_mu
  estim_MV_sigma[1, i] = estim_sigma
  coeff = z_alpha * sqrt((estim_sigma ^ 2) / length(norm_10))
  xN = moyenne_empirique(norm_10)
  borne_inf = xN - coeff
  borne_sup = xN + coeff
  if (borne_inf <= 2 && borne_sup >= 2) {
    cpt_10 = cpt_10 + 1
  }
}
cpt_10 / 100
```

```
[1] 0.93
```

La valeur obtenue confirme presque la couverture de 95%. En effet, notre compteur calcule notre couverture empirique basée sur un intervalle de confiance construit autour de notre estimateur. Or, l'estimateur ne tend pas forcément vers la valeur exacte (même s'il s'en rapporche). De plus, la taille de l'intervalle de confiance diminue lorsque la taille de l'échantillon augmente. C'est ce qu'on va pouvoir observer à la question 3. Cela veut donc dire que pour obtenir une couverture de 95% il faudrait que sur 100 essais, l'estimateur tende vers le vrai paramètre pour une taille d'échantillon assez grande et ceci 95 fois.

Déterminons l'information de Fisher pour la loi normale.

Soit $X \sim \mathcal{N}(\mu, \sigma^2)$

L'information de Fisher est définie par

$$\mathcal{I}_X(\mu, \sigma^2) = -\mathbb{E}_X(\ \nabla_2(\ \mathcal{L}(X, \mu, \sigma)\ )\ )$$

X a pour densité

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

Donc la log-vraisemblance de cette densité est

$$\mathcal{L}(x, \mu, \sigma) = -\frac{1}{2}\ln(2\pi) - \frac{1}{2}\ln(\sigma^2) - \frac{(x-\mu)^2}{2\sigma^2}$$

On en déduit alors

$$\frac{\partial \mathcal{L}}{\partial \mu} = \frac{x-\mu}{\sigma^2}$$

$$\frac{\partial \mathcal{L}}{\partial (\sigma^2)} = -\frac{1}{2\sigma^2} + \frac{(x-\mu)^2}{2\sigma^4} = \frac{1}{2\sigma^2}\left(\frac{(x-\mu)^2}{\sigma^2} - 1\right)$$

On a alors

$$\frac{\partial^2 \mathcal{L}}{\partial \mu^2} = -\frac{1}{\sigma^2}$$

$$\frac{\partial^2 \mathcal{L}}{\partial \mu \partial (\sigma)^2} = -\frac{(x-\mu)}{\sigma^4}$$

$$\frac{\partial^2 \mathcal{L}}{\partial (\sigma^2)^2} = -\frac{1}{2\sigma^4}\left(\frac{(x-\mu)^2}{\sigma^2} - 1\right) - \frac{(x-\mu)^2}{\sigma^4} \times \frac{1}{2\sigma^2} = -\frac{(x-\mu)^2}{2\sigma^6} + \frac{1}{2\sigma^4} - \frac{(x-\mu)^2}{2\sigma^6} = \frac{1}{2\sigma^4} - \frac{(x-\mu)^2}{\sigma^6}$$

On a alors

$$\nabla_2(\ \mathcal{L}(X, \mu, \sigma)\ ) = \begin{pmatrix} \frac{\partial^2 \mathcal{L}}{\partial \mu^2} & \frac{\partial^2 \mathcal{L}}{\partial \mu \partial \sigma} \\ \\ \frac{\partial^2 \mathcal{L}}{\partial \mu \partial \sigma} & \frac{\partial^2 \mathcal{L}}{\partial (\sigma^2)^2} \end{pmatrix}$$

soit

$$\nabla_2(\ \mathcal{L}(X, \mu, \sigma)\ ) = \begin{pmatrix} -\frac{1}{\sigma^2} & -\frac{(x-\mu)}{\sigma^4} \\ \\ -\frac{(x-\mu)}{\sigma^4} & \frac{1}{2\sigma^4} - \frac{(x-\mu)^2}{\sigma^6} \end{pmatrix}$$

On a donc

$$\mathbb{E}_X(\ \nabla_2(\ \mathcal{L}(X, \mu, \sigma)\ )\ ) = \begin{pmatrix} -\mathbb{E}_X\left(\frac{1}{\sigma^2}\right) & -\mathbb{E}_X\left(\frac{(x-\mu)}{\sigma^4}\right) \\ \\ -\mathbb{E}_X\left(\frac{(x-\mu)}{\sigma^4}\right) & \mathbb{E}_X\left(\frac{1}{2\sigma^4} - \frac{(x-\mu)^2}{\sigma^6}\right) \end{pmatrix}$$

Or

$$\mathbb{E}_X(x - \mu) = \mathbb{E}_X(x) - \mu = 0$$

et par définition de la variance

$$\mathbb{E}_X(\ (x-\mu)^2\ ) = \sigma^2$$

donc

$$\mathbb{E}_X(\ \nabla_2(\ \mathcal{L}(X, \mu, \sigma)\ )\ ) = \begin{pmatrix} -\frac{1}{\sigma^2} & 0 \\ \\ 0 & \frac{1}{2\sigma^4} - \frac{\sigma^2}{\sigma^6} \end{pmatrix}$$

soit

$$\mathbb{E}_X(\ \nabla_2(\ \mathcal{L}(X,\mu,\sigma)\ )\ ) = \begin{pmatrix} -\frac{1}{\sigma^2} & 0 \\ 0 & \frac{1}{2\sigma^4} - \frac{1}{\sigma^4} \end{pmatrix}$$

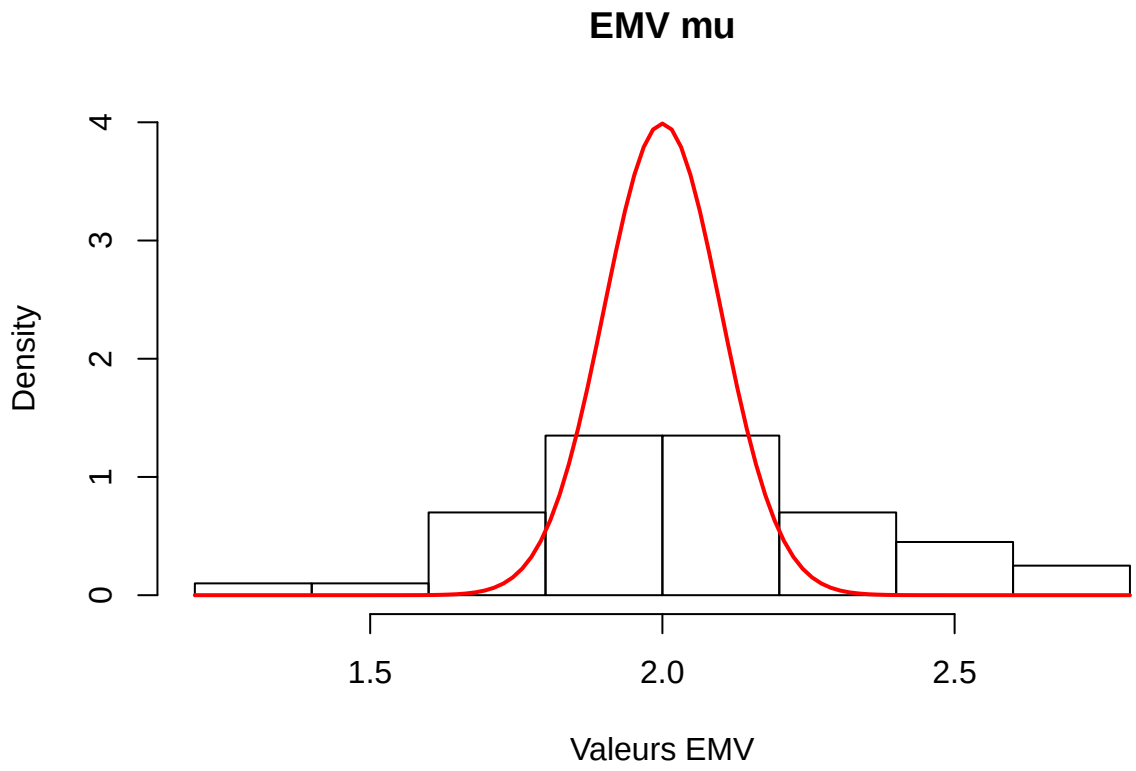
soit

$$\mathbb{E}_X(\ \nabla_2(\ \mathcal{L}(X,\mu,\sigma)\ )\ ) = \begin{pmatrix} -\frac{1}{\sigma^2} & 0 \\ 0 & -\frac{1}{2\sigma^4} \end{pmatrix}$$

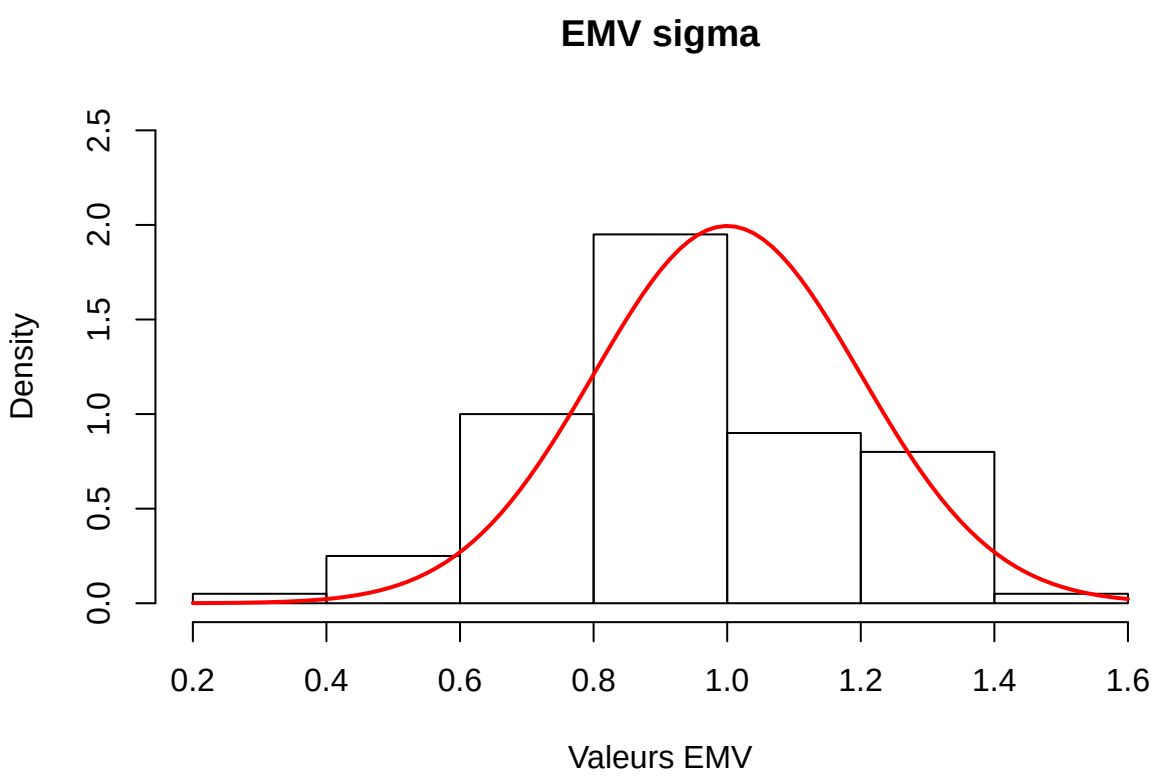
On a donc finalement l'information de Fisher :

$$\mathcal{I}_X(\mu,\sigma^2) = -\mathbb{E}_X(\ \nabla_2(\ \mathcal{L}(X,\mu,\sigma)\ )\ ) = \begin{pmatrix} \frac{1}{\sigma^2} & 0 \\ 0 & \frac{1}{2\sigma^4} \end{pmatrix}$$

```
hist(estim_MV_mu, prob = TRUE, xlab = "Valeurs EMV", main = "EMV mu",ylim=c(0,4))
curve(dnorm(x, 2, 0.1), add=TRUE, col="red", lwd=2)
```




```
hist(estim_MV_sigma, prob = TRUE, xlab = "Valeurs EMV", main = "EMV sigma",ylim=c(0,2.5))
curve(dnorm(x, 1, 0.2), add=TRUE, col="red", lwd=2)
```



Théoriquement, on devrait remarquer que la distribution de notre estimateur du maximum de vraisemblance de μ correspond à la distribution d'une loi normale $\mathcal{N}(2, \frac{1}{10})$ et que la distribution de notre estimateur du maximum de vraisemblance pour σ correspond à la distribution d'une loi normale $\mathcal{N}(1, \frac{2}{10})$.

Question 3 :

```
cpt_20 = 0
for (i in seq(1, 100, 1)) {
  norm_10 = rnorm(20, 2, 1)
  estim = optim(par = c(10, 10),
                fn = function (theta) {log_vraisemblance_norm(theta[1], theta[2], norm_10)},
                method = "L-BFGS-B", lower = c(-Inf, 0.1), control = list(fnscale = -1))$par
  estim_mu = estim[1]
  estim_sigma = estim[2]
  coeff = z_alpha * sqrt((estim_sigma ^ 2) / length(norm_10))
  xN = moyenne_empirique(norm_10)
  borne_inf = xN - coeff
  borne_sup = xN + coeff
  if (borne_inf <= 2 && borne_sup >= 2) {
    cpt_20 = cpt_20 + 1
  }
}
cpt_20 / 100
```

```
[1] 0.94
```

```r
cpt_50 = 0
for (i in seq(1, 100, 1)) {
  norm_10 = rnorm(50, 2, 1)
  estim = optim(par = c(10, 10),
                fn = function (theta) {log_vraisemblance_norm(theta[1], theta[2], norm_10)},
                method = "L-BFGS-B", lower = c(-Inf, 0.1), control = list(fnscale = -1))$par
  estim_mu = estim[1]
  estim_sigma = estim[2]
  coeff = z_alpha * sqrt((estim_sigma ^ 2) / length(norm_10))
  xN = moyenne_empirique(norm_10)
  borne_inf = xN - coeff
  borne_sup = xN + coeff
  if (borne_inf <= 2 && borne_sup >= 2) {
    cpt_50 = cpt_50 + 1
  }
}
cpt_50 / 100
```

```
[1] 0.93
```

```r
cpt_100 = 0
for (i in seq(1, 100, 1)) {
  norm_10 = rnorm(100, 2, 1)
  estim = optim(par = c(10, 10),
                fn = function (theta) {log_vraisemblance_norm(theta[1], theta[2], norm_10)},
                method = "L-BFGS-B", lower = c(-Inf, 0.1), control = list(fnscale = -1))$par
  estim_mu = estim[1]
  estim_sigma = estim[2]
  coeff = z_alpha * sqrt((estim_sigma ^ 2) / length(norm_10))
  xN = moyenne_empirique(norm_10)
  borne_inf = xN - coeff
  borne_sup = xN + coeff
  if (borne_inf <= 2 && borne_sup >= 2) {
    cpt_100 = cpt_100 + 1
  }
}
cpt_100 / 100
```

```
[1] 0.92
```

Les valeurs obtenues pour les couvertures à travers nos compteurs confirment la couverture de 95% comme prévu à la question précédente puisque si $n \to +\infty$, la largeur de l'intervalle de confiance tend vers 0, mais l'estimateur converge vers la valeur exacte du modèle avec $(\widehat{\mu}, \widehat{\sigma})$, qui diffère de celle du modèle avec (μ, σ).

La taille de l'échantillon a donc une influence sur l'inférence asymptotique puisque plus la taille est grande plus la largeur de notre intervalle de confiance diminue et plus notre couverture augmente en se rapprochant de 95%.

Estimation de la covariance asymptotique :

Question 4 :

```
weibull_10 = rweibull(10, 1.5, 3)
```

Soit un échantillon iid $X = (X_1, ..., X_n)$ avec $\forall i \in [\![1, n]\!], X_i \sim Weibull(a, b)$

$\forall i \in [\![1, n]\!], X_i$ a pour densité $f(x_i, a, b) = \frac{a}{b} \left(\frac{x_i}{b}\right)^{a-1} e^{-\left(\frac{x_i}{b}\right)^a}$

La vraisemblance vaut alors

$$L(X_1, ..., X_n, a, b) = \prod_{i=1}^{n} \frac{a}{b} \left(\frac{x_i}{b}\right)^{a-1} e^{-\left(\frac{x_i}{b}\right)^a}$$

En passant à la log-vraisemblance, on obtient alors

$$\mathcal{L}(X_1, ..., X_n, a, b) = \sum_{i=1}^{n} \left(\ln\left(\frac{a}{b}\right) + (a-1) \ln\left(\frac{x_i}{b}\right) - \left(\frac{x_i}{b}\right)^a \right)$$

soit finalement

$$\mathcal{L}(X_1, ..., X_n, a, b) = n \ln\left(\frac{a}{b}\right) + (a-1) \sum_{i=1}^{n} \ln\left(\frac{x_i}{b}\right) - b^{-a} \sum_{i=1}^{n} (x_i)^a$$

```
log_vraisemblance_weibull <- function (a, b, x) {
  n = length(x)
  somme_x = 0
  somme_lnx = 0
  for (i in 1:n) {
    somme_x = somme_x + x[i] ^ a
    somme_lnx = somme_lnx + log(x[i] / b)
  }
  return ( ( (n * log(a/b)) - (b^(-a) * somme_x) + ((a-1) * somme_lnx) ) )
}
```

```
estim = optim(par = c(1, 1),
              fn = function (theta) {log_vraisemblance_weibull(theta[1], theta[2], weibull_10)},
              method = "L-BFGS-B", lower = c(10^-4, 10^-4), control = list(fnscale = -1),
              hessian = TRUE)
estim_a = estim$par[1]
estim_b = estim$par[2]
hessian_10 = estim$hessian
estim_a
```

```
[1] 1.367925
```

```
estim_b
```

```
[1] 3.566369
```

```
hessian_10
```

-9.373815	1.193575
1.193575	-1.471202

On remarque que nos estimateurs de a et b se rapprochent de 1.5 et 3.0 mais en sont quand même assez éloignés, cela est principalement dû à la taille de notre échantillon qui est uniquement de 10.

```
inv_hessian_10 = inv(-1 * hessian_10)
inv_hessian_10
```

0.1189700	0.0965195
0.0965195	0.7580217

```
z_alpha = qweibull(0.975, 1.5, 3)
borne_inf_a = estim_a - (inv_hessian_10[1, 1] / sqrt(10)) * z_alpha
borne_sup_a = estim_a + (inv_hessian_10[1, 1] / sqrt(10)) * z_alpha
borne_inf_b = estim_b - (inv_hessian_10[2, 2] / sqrt(10)) * z_alpha
borne_sup_b = estim_b + (inv_hessian_10[2, 2] / sqrt(10)) * z_alpha
```

L'intervalle trouvé pour l'estimateur de a est le suivant :

```
c(borne_inf_a,borne_sup_a)
```

```
[1] 1.098469 1.637382
```

L'intervalle trouvé pour l'estimateur de b est le suivant :

```
c(borne_inf_b,borne_sup_b)
```

```
[1] 1.849518 5.283219
```

On remarque que nos vrais paramètres se trouve dans l'intervalle de confiance pour a et b.

Question 5 :

```
cpt_a = 0
cpt_b = 0

a_vect = c()
b_vect = c()

for (i in seq(1, 100, 1)) {
  weibull_10 = rweibull(10, 1.5, 3)
  estim = optim(par = c(5, 5),
                fn = function (theta) {log_vraisemblance_weibull(theta[1], theta[2], weibull_10)},
```

```
                method = "L-BFGS-B", lower = c(10^-6, 10^-6), control = list(fnscale = -1),
                hessian = TRUE)

  hessian_10 = estim$hessian
  estim_a = estim$par[1]
  estim_b = estim$par[2]

  a_vect = append(a_vect,estim_a)
  b_vect = append(b_vect,estim_b)

  inv_hessian_10 = inv(-1 * hessian_10)
  z_alpha = qweibull(0.975,estim_a,estim_b)

  borne_inf_a = mean(a_vect) - (sqrt(inv_hessian_10[1, 1]) / sqrt(10)) * z_alpha
  borne_sup_a = mean(a_vect) + (sqrt(inv_hessian_10[1, 1]) / sqrt(10)) * z_alpha
  borne_inf_b = mean(b_vect) - (sqrt(inv_hessian_10[2, 2]) / sqrt(10)) * z_alpha
  borne_sup_b = mean(b_vect) + (sqrt(inv_hessian_10[2, 2]) / sqrt(10)) * z_alpha

  if (borne_inf_a <= 1.5 && borne_sup_a >= 1.5) {
    cpt_a = cpt_a + 1
  }
  if (borne_inf_b <= 3 && borne_sup_b >= 3) {
    cpt_b = cpt_b + 1
  }
}

cpt_a / 100
```

```
[1] 0.95
```

```
cpt_b / 100
```

```
[1] 0.99
```

La valeur empirique n'est pas proche de la valeur théorique, notre compteur est loin d'être égal à 95. Par le même raisonnement qu'à la question 2 et 3 on peut justifier ce résultat pour la loi de Weibull.

Question 6 :

```
cpt_a = 0
cpt_b = 0

a_vect = c()
b_vect = c()

for (i in seq(1, 100, 1)) {
  weibull_10 = rweibull(20, 1.5, 3)
  estim = optim(par = c(5, 5),
                fn = function (theta) {log_vraisemblance_weibull(theta[1], theta[2], weibull_10)},
                method = "L-BFGS-B", lower = c(10^-6, 10^-6), control = list(fnscale = -1),
```

```
                hessian = TRUE)

  hessian_10 = estim$hessian
  estim_a = estim$par[1]
  estim_b = estim$par[2]

  a_vect = append(a_vect,estim_a)
  b_vect = append(b_vect,estim_b)

  inv_hessian_10 = inv(-1 * hessian_10)
  z_alpha = qweibull(0.975,estim_a,estim_b)

  borne_inf_a = mean(a_vect) - (sqrt(inv_hessian_10[1, 1]) / sqrt(20)) * z_alpha
  borne_sup_a = mean(a_vect) + (sqrt(inv_hessian_10[1, 1]) / sqrt(20)) * z_alpha
  borne_inf_b = mean(b_vect) - (sqrt(inv_hessian_10[2, 2]) / sqrt(20)) * z_alpha
  borne_sup_b = mean(b_vect) + (sqrt(inv_hessian_10[2, 2]) / sqrt(20)) * z_alpha

  if (borne_inf_a <= 1.5 && borne_sup_a >= 1.5) {
    cpt_a = cpt_a + 1
  }
  if (borne_inf_b <= 3 && borne_sup_b >= 3) {
    cpt_b = cpt_b + 1
  }
}

cpt_a / 100
```

```
[1] 0.99
```

```
cpt_b / 100
```

```
[1] 0.99
```

```
cpt_a = 0
cpt_b = 0

a_vect = c()
b_vect = c()

for (i in seq(1, 100, 1)) {
  weibull_10 = rweibull(50, 1.5, 3)
  estim = optim(par = c(5, 5),
                fn = function (theta) {log_vraisemblance_weibull(theta[1], theta[2], weibull_10)},
                method = "L-BFGS-B", lower = c(10^-6, 10^-6), control = list(fnscale = -1),
                hessian = TRUE)

  hessian_10 = estim$hessian
  estim_a = estim$par[1]
  estim_b = estim$par[2]

  a_vect = append(a_vect,estim_a)
  b_vect = append(b_vect,estim_b)
```

```
  inv_hessian_10 = inv(-1 * hessian_10)
  z_alpha = qweibull(0.975,estim_a,estim_b)

  borne_inf_a = mean(a_vect) - (sqrt(inv_hessian_10[1, 1]) / sqrt(50)) * z_alpha
  borne_sup_a = mean(a_vect) + (sqrt(inv_hessian_10[1, 1]) / sqrt(50)) * z_alpha
  borne_inf_b = mean(b_vect) - (sqrt(inv_hessian_10[2, 2]) / sqrt(50)) * z_alpha
  borne_sup_b = mean(b_vect) + (sqrt(inv_hessian_10[2, 2]) / sqrt(50)) * z_alpha

  if (borne_inf_a <= 1.5 && borne_sup_a >= 1.5) {
    cpt_a = cpt_a + 1
  }
  if (borne_inf_b <= 3 && borne_sup_b >= 3) {
    cpt_b = cpt_b + 1
  }
}

cpt_a / 100
```

```
[1] 1
```

```
cpt_b / 100
```

```
[1] 1
```

```
cpt_a = 0
cpt_b = 0

a_vect = c()
b_vect = c()

for (i in seq(1, 100, 1)) {
  weibull_10 = rweibull(100, 1.5, 3)
  estim = optim(par = c(5, 5),
                fn = function (theta) {log_vraisemblance_weibull(theta[1], theta[2], weibull_10)},
                method = "L-BFGS-B", lower = c(10^-6, 10^-6), control = list(fnscale = -1),
                hessian = TRUE)

  hessian_10 = estim$hessian
  estim_a = estim$par[1]
  estim_b = estim$par[2]

  a_vect = append(a_vect,estim_a)
  b_vect = append(b_vect,estim_b)

  inv_hessian_10 = inv(-1 * hessian_10)
  z_alpha = qweibull(0.975,estim_a,estim_b)

  borne_inf_a = mean(a_vect) - (sqrt(inv_hessian_10[1, 1]) / sqrt(100)) * z_alpha
  borne_sup_a = mean(a_vect) + (sqrt(inv_hessian_10[1, 1]) / sqrt(100)) * z_alpha
  borne_inf_b = mean(b_vect) - (sqrt(inv_hessian_10[2, 2]) / sqrt(100)) * z_alpha
  borne_sup_b = mean(b_vect) + (sqrt(inv_hessian_10[2, 2]) / sqrt(100)) * z_alpha
```

```
  if (borne_inf_a <= 1.5 && borne_sup_a >= 1.5) {
    cpt_a = cpt_a + 1
  }
  if (borne_inf_b <= 3 && borne_sup_b >= 3) {
    cpt_b = cpt_b + 1
  }
}

cpt_a / 100
```

```
[1] 0.97
```

```
cpt_b / 100
```

```
[1] 0.99
```

En effectuant le même raisonnement qu'à la question 3 on arrive donc à dire que pour une loi de Weibull également la taille de l'échantillon a une influence sur l'inférence asymptotique puisque plus la taille est grande plus la largeur de notre intervalle de confiance va devenir petite et donc on se rapproche de la couverture de 95% souhaitée.

On remarque également que pour un échantillon de taille 1000, on se rapporche encore plus de la couverture théorique puisqu'on obtient que des valeurs supérieures ou égale à 95%.

Méthode Delta :

Question 7 :

```
deriv <- function(g, par) {
  d = length(par)
  ep = 0.0001
  eps = ep*par
  Dg = rep(0,d)
  for(i in 1 : d){
    par_t = par
    par_t[i] = par_t[i] + eps[i]
    Dg[i] = (g(par_t) - g(par)) / eps[i]
  }
  return (Dg)
}
```

```
gamma = rgamma(200, 2, 2)
```

```
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)
}
```

Pour $\Phi = \frac{\alpha}{\beta}$

On a

$$\mathbb{V}(\Phi) = \mathbb{V}\left(\frac{\alpha}{\beta}\right) = \left(\frac{\alpha}{\beta}\right)^2 \mathbb{V}(1) = \left(\frac{\alpha}{\beta}\right)^2$$

d'où

$$\sigma(\Phi) = \sqrt{\mathbb{V}(\Phi)} = \frac{\alpha}{\beta}$$

Pour $[a, b] \subseteq \mathbb{R}_+$ et $[c, d] \subseteq \mathbb{R}_+$

et pour $\alpha \in [a, b]$ et $\beta \in [c, d]$

on a $\frac{1}{\beta} \in \left[\frac{1}{d}, \frac{1}{c}\right]$

d'où finalement $\Phi = \frac{\alpha}{\beta} \in \left[\frac{a}{d}, \frac{b}{c}\right]$

```
estim = optim(par = c(1, 1),
              fn = function (theta) {log_vraisemblance_gamma(theta[1], theta[2], gamma)},
              method = "L-BFGS-B", lower = c(10^-2, 10^-2), control = list(fnscale = -1),
              hessian = TRUE)
hessian_g = -estim$hessian
estim_alpha = estim$par[1]
estim_beta = estim$par[2]
estim_phi = estim_alpha / estim_beta
estim_phi
```

```
[1] 1.020096
```

```
g <- function (alpha, beta ) {
    return (alpha / beta)
}
```

```
thetag = matrix(data = 1, nrow = 1, ncol = 2)
thetag[1,1] = deriv(function (a) {g(a,estim_beta)},estim_alpha)[1]
thetag[1,2] = deriv(function (b) {g(estim_alpha,b)},estim_beta)[1]
thetag
```

0.5268961	-0.5374307

```
sd_gamma = thetag%*%inv(hessian_g)%*%t(thetag)
sd_gamma
```

0.0026869

```
z_alpha = qgamma(0.975, 2, 2)
borne_inf_phi = estim_phi - (sd_gamma / sqrt(200)) * z_alpha
borne_sup_phi = estim_phi + (sd_gamma / sqrt(200)) * z_alpha
```

L'intervalle trouvé pour l'estimateur de Φ est le suivant :

```
c(borne_inf_phi,borne_sup_phi)
```

```
[1] 1.019566 1.020625
```

L'estimateur étant égal à :

```
estim_phi
```

```
[1] 1.020096
```

On a bien l'estimateur dans l'intervalle de confiance.

Question 8 :

```
cauchy = rcauchy(200,10,0.1)
```

```
log_vraisemblance_cauchy <- function (x0, alpha, x) {
  n = length(x)
  somme_log = 0
  for (i in 1:n) {
    somme_log = somme_log + log(1+ ((x[i]-x0)^2) /alpha)
  }
  return (-n * log(alpha*pi) - somme_log )
}
```

```
estim = optim(par = c(10, 1),
              fn = function (theta) {log_vraisemblance_cauchy(theta[1], theta[2], cauchy)},
              method = "L-BFGS-B", lower = c(10^-4, 10^-1), control = list(fnscale = -1),
              hessian = TRUE)
hessian_c = -estim$hessian
estim_x0 = estim$par[1]
estim_alpha = estim$par[2]
```

Pour $X \sim Cauchy(x_0, \alpha)$

La densité de X est $f(x, x_0, \alpha) = \frac{1}{\pi}\left(\frac{\alpha}{(x-x_0)^2+\alpha^2}\right)$

On a donc

$$\mathbb{P}(X > 100) = \int_{100}^{+\infty} \frac{1}{\pi}\left(\frac{\alpha}{(x-x_0)^2+\alpha^2}\right) dx$$

soit

$$\mathbb{P}(X > 100) = \frac{1}{\pi\alpha}\int_{100}^{+\infty} \frac{1}{1+\left(\frac{x-x_0}{\alpha}\right)^2} dx$$

En effectuant le changement de variable $u = \frac{x-x_0}{\alpha}$, avec $dx = \alpha\ du$

On obtient

$$\mathbb{P}(X > 100) = \frac{1}{\pi\alpha} \int_{\frac{100-x_0}{\alpha}}^{+\infty} \frac{1}{1+u^2}\ \alpha\ dx$$

D'où

$$\mathbb{P}(X > 100) = \frac{1}{\pi} \Big[Arctan(u) \Big]_{\frac{100-x_0}{\alpha}}^{+\infty}$$

D'où finalement

$$\mathbb{P}(X > 100) = \frac{1}{\pi} \left(\frac{\pi}{2} - Arctan\left(\frac{100-x_0}{\alpha} \right) \right)$$

On peut finalement poser la fonction

$$G : (x_0, \alpha) \mapsto \frac{1}{\pi} \left(\frac{\pi}{2} - Arctan\left(\frac{100-x_0}{\alpha} \right) \right)$$

```
P_X100 <- function (x0, alpha) {
  return( ( (pi/2) - atan((100-x0)/alpha) ) / pi )
}

estim_P_X100 = P_X100(estim_x0,estim_alpha)
estim_P_X100
```

```
[1] 0.000353678
```

```
thetag = matrix(data = 1, nrow = 1, ncol = 2)
thetag[1,1] = deriv(function (x0) {P_X100(x0,estim_alpha)},estim_x0)[1]
thetag[1,2] = deriv(function (alpha) {P_X100(estim_x0,alpha)},estim_alpha)[1]
thetag
```

3.9e-06	0.0035368

```
sd_cauchy = thetag %*% inv(-hessian_c) %*% t(thetag)
sd_cauchy
```

0

```
z_alpha = qcauchy(0.975, 10, 0.1)
borne_inf_P_X100 = estim_P_X100 - (sd_cauchy / thetag %*% t(thetag)) * z_alpha
borne_sup_P_X100 = estim_P_X100 + (sd_cauchy / thetag %*% t(thetag)) * z_alpha
```

L'intervalle trouvé pour l'estimateur de $P(X > 100)$ est le suivant :

```
c(borne_inf_P_X100,borne_sup_P_X100)
```

```
[1] -0.0004678523  0.0011752083
```

L'estimateur étant égal à :

```
estim_P_X100
```

```
[1] 0.000353678
```

On a bien l'estimateur dans l'intervalle de confiance.

On effectue le même procédé pour le cas x tel que $P(X < x) = 0.99$:

```
P_X99 <- function (x0, alpha) {
  return( qcauchy(0.99,x0,alpha) )
}

estim_P_X99 = P_X99(estim_x0,estim_alpha)
estim_P_X99
```

```
[1] 13.18218
```

```
thetag = matrix(data = 1, nrow = 1, ncol = 2)
thetag[1,1] = deriv(function (x0) {P_X99(x0,estim_alpha)},estim_x0)[1]
thetag[1,2] = deriv(function (alpha) {P_X99(estim_x0,alpha)},estim_alpha)[1]
thetag
```

1	31.82052

```
sd_cauchy = thetag %*% inv(-hessian_c) %*% t(thetag)
sd_cauchy
```

0.0731686

```
borne_inf_P_X99 = estim_P_X99 - (sd_cauchy / thetag %*% t(thetag)) * z_alpha
borne_sup_P_X99 = estim_P_X99 + (sd_cauchy / thetag %*% t(thetag)) * z_alpha
```

L'intervalle trouvé pour l'estimateur de $P(X > 100)$ est le suivant :

```
c(borne_inf_P_X99,borne_sup_P_X99)
```

```
[1] 13.18136 13.18299
```

L'estimateur étant égal à :

```
estim_P_X99
```

```
[1] 13.18218
```

on a bien l'estimateur dans l'intervalle de confiance.

Bootstrap paramètrique :

Question 9 :

Déterminons le maximum de vraisemblance pour la moyenne d'une loi normale $\mathcal{N}(\theta, 1)$

Pour $X = (X_1, ..., X_n)$ avec $\forall i \in [\![1, n]\!], X_i \sim \mathcal{N}(\theta, 1)$

On a

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

soit

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

d'où

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

On a donc

$$\frac{\partial \mathcal{L}}{\partial \theta} = \sum_{i=1}^{n}(x_i - \theta) = -n\theta + \sum_{i=1}^{n} x_i = 0$$

$$\iff \widehat{\theta}^{MV} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Pour

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

On a

$$\widehat{\theta}^{MV} = max(\overline{x}, 0)$$

Pour montrer que le modèle est régulier, on reprend l'information de Fisher calculée précedemment, on avait avec $X \sim \mathcal{N}(\mu, \sigma^2)$

$$\mathcal{I}_X(\mu, \sigma^2) = \begin{pmatrix} \frac{1}{\sigma^2} & 0 \\ 0 & \frac{1}{2\sigma^4} \end{pmatrix}$$

On a donc

$$det(\mathcal{I}_X(\mu, \sigma^2)) = \frac{1}{2\sigma^6} \neq 0$$

On a finalement que $\mathcal{I}_X(\mu, \sigma^2))$ est inversible, et on en déduit donc que le modèle est régulier.

```
bt = rnorm(200, 2, 1)
```

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

```
[1] 1.985045
```

```
EMV = matrix(data = 1, nrow = 100, ncol = 1)
for (i in seq(1, 100, 1)) {
  norm_btp = rnorm(200, estim_mu, 2)
  EMV[i, 1] = optim(par = c(10, 10),
                    fn = function (theta) {log_vraisemblance_norm(theta[1], theta[2], norm_btp)},
                    method = "L-BFGS-B", lower = c(-Inf, 0.1), control = list(fnscale = -1))$par[1]
}
```

```
theta_moy = mean(EMV)
z_alpha = qnorm(0.975, mean = estim_mu, sd = 1)
coeff = z_alpha * sqrt((estim_mu ^ 2) / length(bt))
borne_inf = theta_moy - coeff
borne_sup = theta_moy + coeff
```

Notre estimateur de base pour le bootstrap paramètrique est :

```
estim_mu
```

```
[1] 1.985045
```

Notre moyenne empirique des estimateurs générés est :

```
theta_moy
```

```
[1] 1.990534
```

L'intervalle de confiance trouvé par bootstrap paramètrique est le suivant :

```
c(borne_inf,borne_sup)
```

```
[1] 1.436797 2.544270
```

L'estimateur est donc bien compris dans l'intervalle de confiance trouvé.

```
EMV = matrix(data = 1, nrow = 100, ncol = 1)
for (i in seq(1, 100, 1)) {
  norm_btnp = sample(bt, 100, replace = TRUE)
  EMV[i, 1] = optim(par = c(10, 10),
                    fn = function (theta) {log_vraisemblance_norm(theta[1], theta[2], norm_btnp)},
                    method = "L-BFGS-B", lower = c(-Inf, 0.1),
                    control = list(fnscale = -1))$par[1]
}
```

```
theta_moy = mean(EMV)
z_alpha = qnorm(0.975, mean = estim_mu, sd = 1)
coeff = z_alpha * sqrt((estim_mu ^ 2) / length(bt))
borne_inf = theta_moy - coeff
borne_sup = theta_moy + coeff
```

L'intervalle de confiance trouvé par bootstrap non paramètrique est le suivant :

```
c(borne_inf,borne_sup)
```

```
[1] 1.420017 2.527491
```

L'estimateur est donc bien compris dans l'intervalle de confiance trouvé.

Les deux méthodes nous permettent d'obtenir des intervalles de confiance de quasiment la même largeur et contenant tous les deux la bonne valeur.

On peut donc dire que les deux méthodes se valent et donnent des résultats cohérents. Le choix de l'utilisation de l'une ou l'autre dépend donc du cas d'usage, selon ce qu'on possède comme donnée de base pour le problème qu'on souhaite résoudre.

Question 10 :

On a déjà calculé l'intervalle de de confiance par normalité asymptotique à la question 7 :

L'intervalle trouvé pour l'estimateur de ϕ est le suivant :

```
c(borne_inf_phi,borne_sup_phi)
```

```
[1] 1.019566 1.020625
```

```
cpt_a = 0
phi_vect = c()

for (i in seq(1, 100, 1)) {
  gamma = rgamma(200, 2, 2)
  estim = optim(par = c(1, 1),
                fn = function (theta) {log_vraisemblance_gamma(theta[1], theta[2], gamma)},
                method = "L-BFGS-B", lower = c(10^-2, 10^-2), control = list(fnscale = -1),
                hessian = TRUE)
  hessian_g = (-1) * estim$hessian
  estim_alpha = estim$par[1]
  estim_beta = estim$par[2]

  phi_vect = append(phi_vect,estim_alpha/estim_beta)

  inv_hessian_g = inv(-1 * hessian_g)
  thetag = matrix(data = 1, nrow = 1, ncol = 2)
  thetag[1,1] = deriv(function (a) {g(a,estim_beta)},estim_alpha)
  thetag[1,2] = deriv(function (b) {g(estim_alpha,b)},estim_beta)
  sd_gamma = thetag%*%inv(hessian_g)%*%t(thetag)

  z_alpha = qgamma(0.975,estim_alpha,estim_beta)

  borne_inf_phi = mean(phi_vect) - (sqrt(sd_gamma) / sqrt(200)) * z_alpha
  borne_sup_phi = mean(phi_vect) + (sqrt(sd_gamma) / sqrt(200)) * z_alpha
  if (borne_inf_phi <= 1 && borne_sup_phi >= 1) {
    cpt_a = cpt_a + 1
  }
}

cpt_a / 100
```

```
[1] 0.88
```

On remarque une très nette variation de la couverture à chaque fois qu'on regénére un échantillon. Nous avons donc une couverture cohérente avec le raisonnement émis à la question 2 et 3.

```
bt = rgamma(200, 2, 2)

estim_alpha = optim(par = c(1, 1),
                    fn = function (theta) {log_vraisemblance_gamma(theta[1], theta[2], bt)},
                    method = "L-BFGS-B", lower = c(10^-2, 10^-2),
                    control = list(fnscale = -1))$par[1]

estim_beta = optim(par = c(1, 1),
                   fn = function (theta) {log_vraisemblance_gamma(theta[1], theta[2], bt)},
                   method = "L-BFGS-B", lower = c(10^-2, 10^-2),
                   control = list(fnscale = -1))$par[2]

EMV = matrix(data = 1, nrow = 100, ncol = 1)
for (i in seq(1, 100, 1)) {
  norm_btp = rgamma(200, estim_alpha, estim_beta)
  est_alpha = optim(par = c(1, 1),
                    fn = function (theta) {log_vraisemblance_gamma(theta[1], theta[2], norm_btp)},
                    method = "L-BFGS-B", lower = c(10^-2, 10^-2),
                    control = list(fnscale = -1))$par[1]
  est_beta = optim(par = c(1, 1),
                    fn = function (theta) {log_vraisemblance_gamma(theta[1], theta[2], norm_btp)},
                    method = "L-BFGS-B", lower = c(10^-2, 10^-2),
                    control = list(fnscale = -1))$par[2]
  EMV[i, 1] = est_alpha / est_beta
}

theta_moy = mean(EMV)
z_alpha = qgamma(0.975, estim_alpha,estim_beta)
coeff = z_alpha * sqrt(((estim_alpha/estim_beta) ^ 2) / length(bt))
borne_inf = theta_moy - coeff
borne_sup = theta_moy + coeff
```

On a donc pour intervalle de confiance :

```
c(borne_inf,borne_sup)
```

```
[1] 0.8267142 1.2213430
```

```
cpt_btp = 0


for (i in seq(1, 100, 1)) {
  EMV=c()
  btp = rgamma(200, estim_alpha, estim_beta)

  estim_alpha = optim(par = c(1, 1),
                      fn = function (theta) {log_vraisemblance_gamma(theta[1], theta[2], btp)},
```

```
                      method = "L-BFGS-B", lower = c(10^-4, 10^-4),
                      control = list(fnscale = -1))$par[1]
  estim_beta = optim(par = c(1,1),
                     fn = function (theta) {log_vraisemblance_gamma(theta[1], theta[2], btp)},
                     method = "L-BFGS-B", lower = c(10^-4, 10^-4),
                     control = list(fnscale = -1))$par[2]

  for (j in seq(1, 100, 1)) {
    EMV = append(EMV, (estim_alpha / estim_beta) )
    theta_moy = mean(EMV)
  }


  z_alpha = qgamma(0.975, estim_alpha,estim_beta)
  coeff = z_alpha * sqrt(((estim_alpha/estim_beta)^2) / length(btp))
  borne_inf = theta_moy - coeff
  borne_sup = theta_moy + coeff
  if (borne_inf <= 1 && borne_sup >= 1) {
    cpt_btp = cpt_btp + 1
  }

}

cpt_btp / 100
```

```
[1] 0.54
```

```
cpt_btnp = 0
for (i in seq(1, 100, 1)) {
  EMV=c()
  btnp = sample(rgamma(200, 2, 2))

  estim_alpha = optim(par = c(1, 1),
                      fn = function (theta) {log_vraisemblance_gamma(theta[1], theta[2], bt)},
                      method = "L-BFGS-B", lower = c(10^-4, 10^-4),
                      control = list(fnscale = -1))$par[1]
  estim_beta = optim(par = c(1, 1),
                     fn = function (theta) {log_vraisemblance_gamma(theta[1], theta[2], bt)},
                     method = "L-BFGS-B", lower = c(10^-4, 10^-4),
                     control = list(fnscale = -1))$par[2]




  for (j in seq(1, 100, 1)) {
    EMV = append(EMV, (estim_alpha / estim_beta) )
  }

  theta_moy = mean(EMV)
  z_alpha = qgamma(0.975, estim_alpha,estim_beta)
  coeff = z_alpha * sqrt(((estim_alpha/estim_beta) ^ 2) / length(bt))
  borne_inf = theta_moy - coeff
  borne_sup = theta_moy + coeff
```

```
  if (borne_inf <= 1 && borne_sup >= 1) {
    cpt_btnp = cpt_btnp + 1
  }


}
cpt_btnp / 100
```

```
[1] 1
```

Question 11 :

```
p = 0.5
mu = 1
melange_norm = p * rnorm(100, mu, 1) + (1 - p) * rnorm(100, 0, 1)
```

Nous allons procéder par bootstrap non paramétrique :

```
log_vraisemblance_multinormale <- function(mu,p,x) {
  n = length(x)
  somme_log = 0
  for (i in 1:n) {
    somme_log = somme_log + log(p*(exp(-(x[i]-mu)^2/2)) + (1-p)*(exp(-x[i]^2/2)))
  }
  return(somme_log)
}
```

Nous allons utiliser la log vraisemblance associée à la densité de notre mélange de lois normales.

```
EMV = matrix(data = 1, nrow = 100, ncol = 1)
for (i in seq(1, 100, 1)) {
  norm_multi = sample(melange_norm, 100, replace = TRUE)
  EMV[i, 1] = optim(par = c(1, 0.5),
                    fn = function (theta) {log_vraisemblance_norm(theta[1], theta[2], norm_multi)},
                    method = "L-BFGS-B", lower = c(-Inf, 10^-6), upper = c(+Inf, 1),
                    control = list(fnscale = -1))$par[1]
}

theta_moy = mean(EMV)
z_alpha = qnorm(0.975, mean = estim_mu, sd = 1)
coeff = z_alpha * sqrt((estim_mu ^ 2) / length(bt))
borne_inf_mu = theta_moy - coeff
borne_sup_mu = theta_moy + coeff
```

On obtient comme intervalle de confiance pour μ :

```
c(borne_inf_mu,borne_sup_mu)
```

```
[1] -0.02103338  1.08643999
```

La “vraie” valeur de μ qui est 1 est belle et bien comprise dans cet intervalle.

```
EMV = matrix(data = 1, nrow = 100, ncol = 1)
for (i in seq(1, 100, 1)) {
  norm_multi = sample(melange_norm, 100, replace = TRUE)
  EMV[i, 1] = optim(par = c(1, 0.5),
                    fn = function (theta) {log_vraisemblance_norm(theta[1], theta[2], norm_multi)},
                    method = "L-BFGS-B", lower = c(-Inf, 10^-6), upper = c(+Inf, 1),
                    control = list(fnscale = -1))$par[2]
}

theta_moy = mean(EMV)
z_alpha = qnorm(0.975, mean = estim_mu, sd = 1)
coeff = z_alpha * sqrt((estim_mu ^ 2) / length(bt))
borne_inf_p = theta_moy - coeff
borne_sup_p = theta_moy + coeff
```

On obtient comme intervalle de confiance pour p :

```
c(borne_inf_p,borne_sup_p)
```

```
[1] 0.1487496 1.2562230
```

La “vraie” valeur de p qui est 0.5 est belle et bien aussi comprise dans cet intervalle.

Par conséquent, l’application de la méthode de bootstrap non paramétrique pour trouver des intervalles de confiance est assez efficace même pour une loi non usuelle comme par exemple le mélange de lois normales qu’on vient d’étudier.

Question 12 :

```
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
```

```
summer = summer_rur - summer_neuil
winter = winter_rur - winter_neuil
```

```
EMVS = 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

EMVW = 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
```

```
summer_mu = EMVS[1]
summer_sigma = EMVS[2]
winter_mu = EMVW[1]
winter_sigma = EMVW[2]
```

```
z_90 = qnorm(0.95, mean = 0, sd = 1)
z_99 = qnorm(0.995, mean = 0, sd = 1)
```

```
summer_moy = mean(summer)

coeff_summer_mu = z_90 * sqrt((summer_mu ^ 2) / length(summer))
borne_inf_summer_mu = summer_moy - coeff_summer_mu
borne_sup_summer_mu = summer_moy + coeff_summer_mu

coeff_summer_sigma = z_90 * sqrt((summer_sigma ^ 2) / length(summer))
borne_inf_summer_sigma = summer_moy - coeff_summer_sigma
borne_sup_summer_sigma = summer_moy + coeff_summer_sigma
```

L'intervalle de confiance obtenu sur l'été pour μ est le suivant :

```
c(borne_inf_summer_mu,borne_sup_summer_mu)
```

```
[1] -0.2201535 12.6519254
```

```
winter_moy = mean(winter)

coeff_winter_mu = z_90 * sqrt((winter_mu ^ 2) / length(winter))
borne_inf_winter_mu = winter_moy - coeff_winter_mu
borne_sup_winter_mu = winter_moy + coeff_winter_mu

coeff_winter_sigma = z_90 * sqrt((winter_sigma ^ 2) / length(winter))
borne_inf_winter_sigma = winter_moy - coeff_winter_sigma
borne_sup_winter_sigma = winter_moy + coeff_winter_sigma
```

L'intervalle de confiance obtenu sur l'été pour σ est le suivant :

```
c(borne_inf_summer_sigma,borne_sup_summer_sigma)
```

```
[1] 3.916521 8.515251
```

```
summer_moy = mean(summer)

coeff_summer_mu = z_99 * sqrt((summer_mu ^ 2) / length(summer))
borne_inf_summer_mu = summer_moy - coeff_summer_mu
borne_sup_summer_mu = summer_moy + coeff_summer_mu

coeff_summer_sigma = z_99 * sqrt((summer_sigma ^ 2) / length(summer))
borne_inf_summer_sigma = summer_moy - coeff_summer_sigma
borne_sup_summer_sigma = summer_moy + coeff_summer_sigma
```

L'intervalle de confiance obtenu sur l'hiver pour μ est le suivant :

```
c(borne_inf_winter_mu,borne_sup_winter_mu)
```

```
[1] 15.49709 21.47052
```

```
winter_moy = mean(winter)

coeff_winter_mu = z_99 * sqrt((winter_mu ^ 2) / length(winter))
borne_inf_winter_mu = winter_moy - coeff_winter_mu
borne_sup_winter_mu = winter_moy + coeff_winter_mu

coeff_winter_sigma = z_99 * sqrt((winter_sigma ^ 2) / length(winter))
borne_inf_winter_sigma = winter_moy - coeff_winter_sigma
borne_sup_winter_sigma = winter_moy + coeff_winter_sigma
```

L'intervalle de confiance obtenu sur l'hiver pour σ est le suivant :

```
c(borne_inf_winter_sigma,borne_sup_winter_sigma)
```

```
[1] 16.18455 20.78305
```

On remarque très clairement qu'aucun intervalle de confiance ne contient zéro. Cela peut s'interpréter par le fait que nos deux paramètres μ et σ ne valent jamais zéro. On peut ainsi dire qu'on aura jamais la même moyenne du niveau de concentration d'ozone entre une zone urbaine et une zone rurale, même si on compare pour deux mêmes saisons.

Le résultat est tout à fait cohérent avec la réalité puisque les zones urbaines sont beaucoup plus polluées que les zones rurales, l'écart en niveau de concentration d'ozone est donc logique puisque la pollution augmente la concentration d'ozone dans l'air.