[STA211] Ajustement d'une loi de Weibull sur des données de durée de vie d'un composant industriel, avec censures à droite

Simulation:

Question 1:

Calculons la fonction $F^{-1}(p|\alpha,\kappa)$:

Soit $y = F(x) = 1 - exp(-\alpha x^{\kappa})$,

Alors on a:

$$\begin{aligned} exp(-\alpha x^{\kappa}) &= 1 - y \\ \Rightarrow x^{\kappa} &= -\frac{\log(1-y)}{\alpha} \\ \Rightarrow x &= \left(-\frac{\log(1-y)}{\alpha}\right)^{1/\kappa} \end{aligned}$$

Ainsi:

$$F^{-1}(p|\alpha,\kappa) = \left(-\frac{\log(1-y)}{\alpha}\right)^{1/\kappa}$$

Nous allons maintenant utiliser la méthode d'inversion générique qui nous donne le résultat suivant:

Soit $U \sim \mathcal{U}(0,1)$, $X = F^{-1}(U)$

Alors, X est distribuée selon F:

$$X = F^{-1}(U) \sim F$$

Aussi, en appliquant ce résultat à notre exercice on a:

$$X \sim \mathcal{W}(\alpha,\kappa)$$

En prenant simplement:

$$X = F^{-1}(U|\alpha,\kappa)$$

où $U \sim \mathcal{U}(0,1)$.

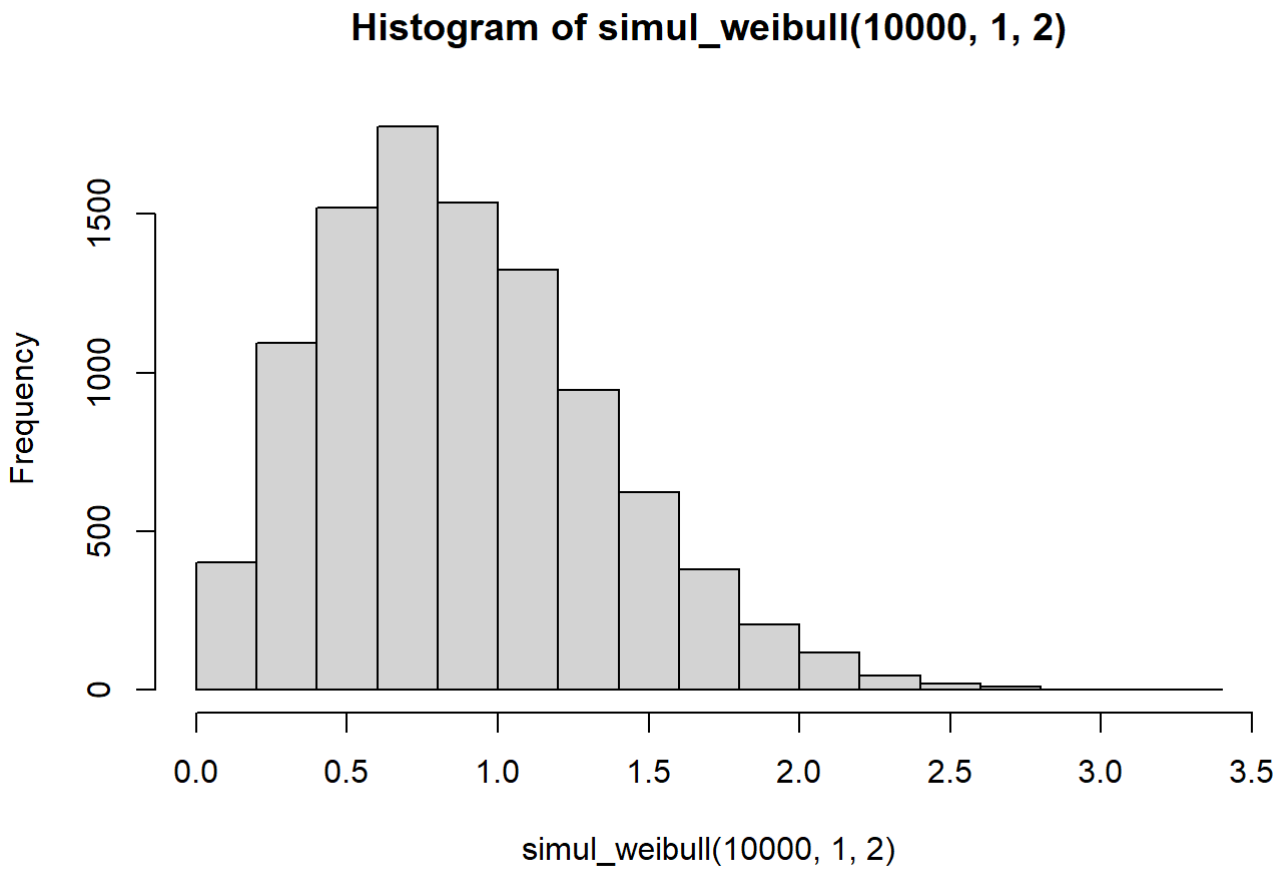
On peut donc en déduire un algorithme de simulation de cette loi basée sur l'inversion générique, on génère les réalisations d'une loi uniforme $\mathcal{U}(0,1)$ auxquelles on applique simplement la fonction $F^{-1}(u|\alpha,\kappa)$, ce qui permet d'obtenir des réalisations simulées selon la loi de Weibull.

On programme en R, une fonction permettant de réaliser cette simulation et qui nous renvoie n observations issue d'une loi de Weibull de paramètres α et κ.

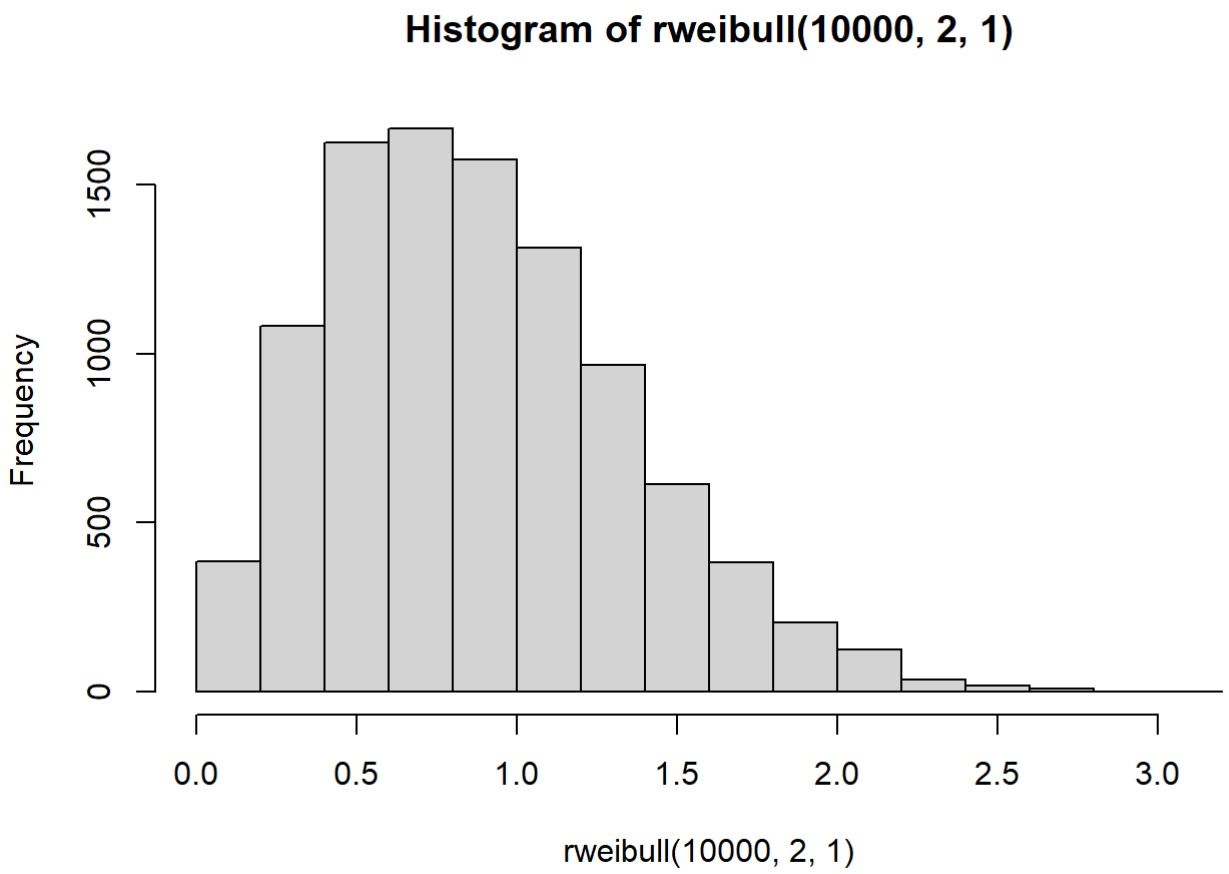
```
simul_weibull <- function(n,alpha,kappa)
{
u = runif(n,0,1) ; #simulation d'une uniforme [0,1]
x = (-log(1-u)/alpha)^{1/kappa}
return (x)
}
```

On vérifie cette fonction en comparant ces observations avec celles issues de la fonction *rweibull*, on obtient les figures suivantes pour $n = 10000$, $\alpha = 1$ et $\kappa = 2$:

```
hist(simul_weibull(10000,1,2))
```



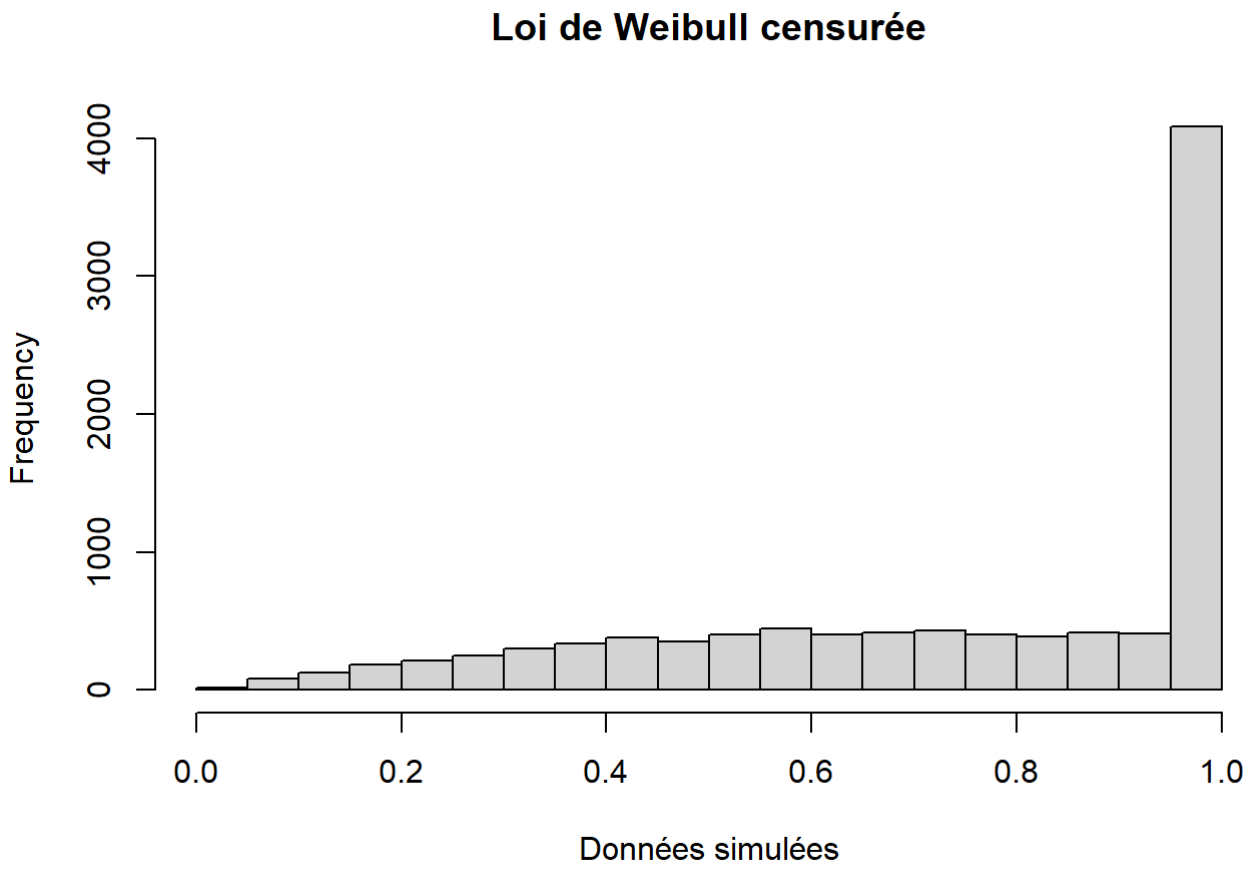

```
hist(rweibull(10000,2,1))
```



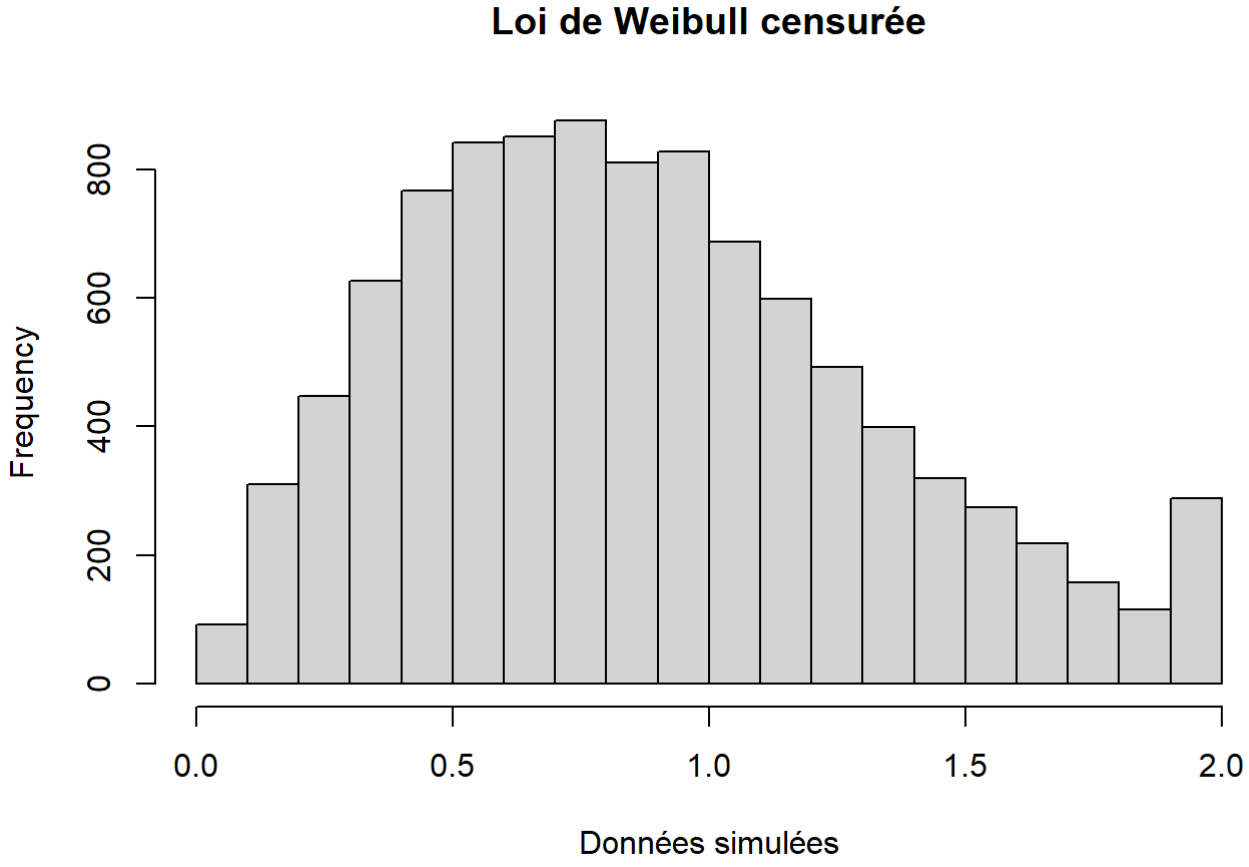
Notre fonction semble fonctionner correctement après comparaison des histogrammes.

Question 2: Dans cette question, nous réalisons une fonction simulant n observations de la loi de Weibull de paramètres α et κ donnés, censurée au-dessus d'un niveau t_0 donné, la fonction renverra le vecteur des données simulées ainsi qu'un nombre p correspondant au nombre d'observations non censurées (triées au début du vecteur).

```
simul_weibull_censur <- function(n,alpha,kappa,t0)
{
  x = simul_weibull(n,alpha,kappa)
  x=sort(as.numeric(lapply(x,function(xi){ifelse(xi<=t0,xi,t0)})))
  p=length(which(x<t0))
  return (list(x,p))
}

hist(simul_weibull_censur(10000,1,2,1)[[1]], main="Loi de Weibull censurée", xlab="Données si
mulées")
```




```
hist(simul_weibull_censur(10000,1,2,2)[[1]], main="Loi de Weibull censurée", xlab="Données si
mulées")
```



Les histogrammes semblent cohérent, en effet on observe dans les deux cas les observations censurées, par une augmentation du nombres de réalisations valant 1 sur la première figure (car censure à 1) et d'une augmentation du nombres de réalisations valant 2 sur la seconde figure (car censure à 2) par rapport à la loi de Weibull calculées précédemment.

#Calcul de la vraisemblance:

Question 3:

La densité de notre variable aléatoire est donnée par :

$$f(t) = \alpha\kappa t^{\kappa-1} exp(-\alpha t^{\kappa})$$

La log-vraisemblance d'une observation non censurée t_i est simplement donnée par la formule:

$$log(t_i, \alpha, \kappa, p) = log(\alpha) + log(\kappa) + (\kappa - 1)log(t) - \alpha t^{\kappa}$$

Pour une censure à droite, la log-vraisemblance est égale à la fonction de queue de la loi de Weibull :

$$log(t_i, \alpha, \kappa, p) = -\alpha t_0^{\kappa}$$

D'où,

$$\begin{aligned} l(t_{1:n}|\alpha, \kappa) &= \sum_{i=1}^{p} log(t_i, \alpha, \kappa, p) + \sum_{i=p+1}^{n} log(t_i, \alpha, \kappa, p) \\ &= \sum_{i=1}^{p} (log(\alpha) + log(\kappa) + (\kappa - 1)log(t_i) - \alpha t_i^{\kappa}) + \sum_{i=p+1}^{n} -\alpha t_0^{\kappa} \\ &= p(log(\alpha) + log(\kappa)) + \sum_{i=1}^{p} (\kappa - 1)log(t_i) - \sum_{i=1}^{p} \alpha t_i^{\kappa} - (n-p)\alpha t_0^{\kappa} \end{aligned}$$

Donc, la log-vraisemblance dans le modèle de Weibull dont les $n - p$ premières données sont censurées à droites, s'écrit:

$$l(t_{1:n}|\alpha, \kappa) = p(log(\alpha) + log(\kappa)) + \sum_{i=1}^{p} (\kappa - 1)log(t_i) - \sum_{i=1}^{n} \alpha t_i^{\kappa}$$

Question 4:

Calculons l'expression exacte du gradient de la log-vraisemblance:

La dérivée par rapport au paramètre α vaut:

$$\frac{\partial l}{\partial \alpha} = \frac{p}{\alpha} - \sum_{i=1}^{n} t_i^{\kappa}$$

La dérivée par rapport au paramètre κ vaut:

$$\frac{\partial l}{\partial \kappa} = \frac{p}{\kappa} + \sum_{i=1}^{p} log(t_i) - \alpha\kappa \sum_{i=1}^{n} t_i^{\kappa-1}$$

Ainsi, le gradient de la log-vraisemblance a pour expression:

$$\nabla l = \begin{pmatrix} \frac{p}{\alpha} - \sum_{i=1}^{n} t_i^{\kappa} \\ \frac{p}{\kappa} + \sum_{i=1}^{p} log(t_i) - \alpha\kappa \sum_{i=1}^{n} t_i^{\kappa-1} \end{pmatrix}$$

La matrice hessienne a l'expression suivante :

$$\begin{pmatrix} \frac{\partial^2 l}{\partial \alpha^2} & \frac{\partial^2 l}{\partial \alpha \partial \kappa} \\ \frac{\partial^2 l}{\partial \kappa \partial \alpha} & \frac{\partial^2 l}{\partial \kappa^2} \end{pmatrix} = \begin{pmatrix} -\frac{p}{\alpha^2} & -\kappa \sum_{i=1}^{n} t_i^{\kappa-1} \\ -\kappa \sum_{i=1}^{n} t_i^{\kappa-1} & -\frac{p}{\kappa^2} - \alpha \sum_{i=1}^{n} t_i^{\kappa-1} - \alpha\kappa(\kappa-1) \sum_{i=1}^{n} t_i^{\kappa-2} \end{pmatrix}$$

Approche fréquentiste

Question 5 Les estimateurs de maximum de vraisemblance vérifient les relations suivantes

$$\frac{\partial l}{\partial \alpha} = 0$$

et:

$$\frac{\partial l}{\partial \kappa} = 0$$

(De plus, le hessien est définie négatif pour les estimateurs du maximum de vraisemblance.) On peut déduire de l'annulation de la dérivée par rapport à α la relation suivante:

$$\frac{p}{\hat{\alpha}_{MLE}} - \sum_{i=1}^{n} t_i^{\hat{\kappa}_{MLE}} = 0$$

D'où l'on conclut que l'estimateur α_{MLE} peut se déduire de l'estimateur du maximum de vraisemblance κ_{MLE} à l'aide de la formule suivante:

$$\hat{\alpha}_{MLE} = \frac{p}{\sum_{i=1}^{n} t_i^{\hat{\kappa}_{MLE}}}$$

Par ailleurs, sachant que c'est cette valeur de α qui maximise la vraisemblance, on peut désormais maximiser la vraisemblance en remplaçant α par $\frac{p}{\sum_{i=1}^{n} t_i^{\kappa}}$.

Aussi, pour trouver $\hat{\kappa}_{MLE}$ il suffit de résoudre le problème d'optimisation en 1D suivant:

$$\hat{\kappa}_{MLE} = argmax_{\kappa}(l(t_{1:n}|\frac{p}{\sum_{i=1}^{n} t_i^{\kappa}}, \kappa))$$

Question 6: On utilise la fonct *optimize* pour trouver la solution du problème précédent et on écrit la fonction *maxvraise* qui permet de calculer les estimateurs du maximum de vraisemblance de (α,κ):

```
weibullcensLogLik<-function(alpha,kappa,t,p)
{
  return (p*(log(alpha)+log(kappa))+(kappa-1)*sum(log(t)[1:p])-alpha*sum(t^kappa))
}
maxvraise<-function(t,p){
  kappaopt= optimize(function(kappa){return (weibullcensLogLik(p/sum(t^kappa),kappa,t,p))},in
terval = seq(from=0, to=20,length.out = 100), maximum = TRUE)$maximum
  alpha=p/sum(t^kappaopt)
  return (list(alpha,kappaopt))
}
```

Question 7: On peut maintenant estimer (α,κ) par maiximum de vraisemblance 100 fois, à partir de 100 jeu de données simulés selon la loi de Weibull censurée à t_0:

```
alpha=5
kappa=2
t0= (-log(1-0.6)/alpha)^(1/kappa)
n=1000
alphaest=c()
kappaest=c()
for(i in 1:100)
{
  x=simul_weibull_censur(n,alpha,kappa,t0)
  t=x[[1]]
  p=x[[2]]
  abmaxvr=maxvraise(t,p)
  alphaest=c(alphaest,abmaxvr[[1]])
  kappaest=c(kappaest,abmaxvr[[2]])
}
```

On calcule ensuite le biais de l'estimateur de alpha et de kappa:

```
biais_alpha=mean(alphaest)-alpha
biais_kappa=mean(kappaest)-kappa
biais_alpha
```

```
## [1] 0.04092626
```

```
biais_kappa
```

```
## [1] 0.003671704
```

Puis la variance des deux estimateurs:

```
var_alpha<-mean((alphaest-mean(alphaest))^2)
var_beta<-mean((kappaest-mean(kappaest))^2)
var_alpha
```

```
## [1] 0.1948572
```

```
var_beta
```

```
## [1] 0.005862209
```

Et enfin, on calcule leur coefficient de variation:

```
cv_alpha<-mean((alphaest-mean(alphaest))^2)/mean(alphaest)
cv_beta<-mean((kappaest-mean(kappaest))^2)/mean(kappaest)
cv_alpha
```

```
## [1] 0.03865504
```

```
cv_beta
```

```
## [1] 0.002925733
```

On retrouve bien un biais proche de zéro ce qui est logique pour un estimateur du maximum de vraisemblance (asymptotiquement sans biais). De plus, la variance semble tendre vers 0 ce qui est aussi cohérent avec notre EMV. Le coefficient de variation est relativement faible ce qui traduit une dispersion assez faible. La méthode de simulation semble efficace

Approche bayésienne

Question 8:

Montrer que si la loi a priori de α est la distribution de Gamma $\mathcal{G}(a,b)$, alors la loi a posteriori conditionnelle de α sachant κ, notée $\pi(\alpha|\kappa,t_{1:n},p)$ est encore une loi Gamma, dont on précisera les paramètres.

Le théorème de Bayes nous donne la relation suivante : $\pi(\alpha|\kappa,t_{1:n},p) \propto \mathcal{L}(t_{1:n}|\alpha,\kappa,p) \times \pi(\alpha)$

Supposons que :

$$\pi(\alpha) \propto \alpha^{a-1}e^{-b\alpha}$$

Alors, la densité a posteriori de λ est proportionnelle vérifie $\pi(\alpha|\kappa,t_{1:n},p) \propto \mathcal{L}(t_{1:n}|\alpha,\kappa,p)\alpha^{a-1}e^{-b\alpha}$

Or $\mathcal{L}(t_{1:n}|\alpha,\kappa,p) = \alpha^p \times \kappa^p \prod_{i=1}^p t_i^{\kappa-1} \times e^{-\sum_{i=1}^n \alpha t_i^\kappa} \propto a^p \times e^{-\sum_{i=1}^n \alpha t_i^\kappa}$

D'où $\pi(\alpha) \propto \alpha^{a-1}e^{-b\alpha} \times a^p e^{-\sum_{i=1}^n \alpha t_i^\kappa}$

D'où $\pi(\alpha) \propto \alpha^{a+p-1}e^{-\alpha(b+\sum_{i=1}^n t_i^\kappa)}$

Ce qui montre que la loi a posteriori suit une loi Gamma $\mathcal{G}(a+p, b+\sum_{i=1}^n t_i^\kappa)$

Dans la suite, on prendra de même une loi a priori de type Gamma pour κ, de paramètres c et d, et on utilisera le choix "faiblement informatif" suivant : a = b = c = d = 10−3.

Question 9

Montrons que la densité marginale a posteriori de κ est proportionnelle à :

$$\pi(\kappa|t_{1:n},p) \propto \pi(\kappa)\kappa^p \prod_{i=1}^p t_i^{\kappa-1}(b+\sum_{i=1}^n t_i^\kappa)^{-(a+p)}$$

On a la relation suivante :

$$\pi(\kappa|,t_{1:n},\alpha,p)\pi(\alpha|t_{1:n},p) = \pi(\kappa|t_{1:n},p)$$

D'où

$$\pi(\kappa|,t_{1:n},p) = \frac{\pi(\kappa,\alpha,t_{1:n},p)}{\pi(\alpha|t_{1:n},p)}$$

On applique alors le théorème de Bayes :

$$\pi(\kappa,\alpha|t_{1:n},p) \propto \mathcal{L}(t_{1:n}|\kappa,\alpha)\pi(\kappa)\pi(\alpha)$$

D'où

$$\pi(\kappa|t_{1:n},p) \propto \frac{\pi(\alpha)\pi(\kappa)\mathcal{L}(t_{1:n}|\kappa,\alpha)}{\pi(\alpha|\kappa,t_{1:n})}$$

$$\pi(\kappa|t_{1:n},p) \propto \frac{\pi(\kappa)\alpha^{a-1}e^{-b\alpha}\kappa^p\prod_{i=1}^p t_i^{\kappa-1}e^{-\alpha\sum_{i=1}^n t_i^\kappa}}{\alpha^{a+p-1}e^{-\alpha(b+\sum_{i=1}^n t_i^\kappa)}(b+\sum_{i=1}^n t_i^\kappa)^{a+p}}$$

D'où

$$\pi(\kappa|t_{1:n},p) \propto \pi(\kappa)\kappa^p\prod_{i=1}^p t_i^{\kappa-1}(b+\sum_{i=1}^n t_i^\kappa)^{-(a+p)}$$

(a)

Montrons en supposant $t_{max} = \max t_i > 1$ les équivalents suivants (à une constante multiplicative près):

$$\pi(\kappa|t_{1:n},p) \sim_{\kappa\to 0} \pi(\kappa)\kappa^p$$

Lorsque $\kappa \to 0$ on a

$$\prod_{i=1}^p t_i^{\kappa-1} \to \prod_{i=1}^p t_i^{-1}$$

Et

$$(b+\sum_{i=1}^n t_i^\kappa)^{-(a+p)} \to (b+n)^{-(a+p)}$$

Ces termes ne dépendent plus de κ, d'où le résultat souhaité :

$$\pi(\kappa|t_{1:n},p) \sim_{\kappa\to 0} \pi(\kappa)\kappa^p$$

Montrons désormais que:

$$\pi(\kappa|t_{1:n},p) \sim_{\kappa\to \infty} \pi(\kappa)t_{max}^{\kappa(a+n)}$$

Ce qui revient à montrer que pour $\kappa \to \infty$ on a

$$\frac{\pi(\kappa|t_{1:n},p)}{\pi(\kappa)t_{max}^{\kappa(a+p)}} \to 1$$

Pour $\kappa \to \infty$ on a

$$\lim_{\kappa\to\infty} \frac{\pi(\kappa|t_{1:n},p)}{\pi(\kappa)\prod_{i=1}^p t_i^{\kappa-1}t_{max}^{\kappa(a+p)}} = \lim_{\kappa\to\infty}(\frac{b+\sum_{i=1}^n t_i^\kappa}{t_{max}^\kappa})^{-(a+p)} = 1$$

Donc:

$$\pi(\kappa|t_{1:n},p) \sim \pi(\kappa)\prod_{i=1}^p t_i^{\kappa-1}t_{max}^{\kappa(a+p)}$$

Déterminons les conditions sur e et f pour que le quotient avec $g(\kappa|e,f)$ reste borné. On veut que les limites au voisinage de 0 et $+\infty$ soient nulles pour avoir le rapport $\frac{\pi(\kappa|t_{1:n})}{g(\kappa)}$ soit borné sur $\mathbb{R}$

$$\frac{\pi(\kappa|t_{1:n})}{g(\kappa)} \sim \frac{\kappa^{c-1}e^{-d\kappa}\kappa^p}{\kappa^{e-1}e^{-f\kappa}}$$

On doit avoir $e \leq p + c$ pour avoir $\frac{\pi(\kappa|t_{1:n})}{g(\kappa)}$ borné au $\mathcal{V}(0)$

On obtient la condition sur f en raisonnant de manière analogue pour $\kappa \to \infty$ On doit avoir le rapport $\frac{\pi(\kappa|t_{1:n})}{g(\kappa)}$ borné au $\mathcal{V}(\infty)$

$$\frac{\pi(\kappa|t_{1:n})}{g(\kappa)} \sim \frac{\kappa^{c-1}e^{-d\kappa}\kappa^{p}e^{-\kappa(a+p)\log(t_{max})}e^{(\kappa-1)\sum_{i=1}^{p}\log(t_i)}}{\kappa^{e-1}e^{-f\kappa}}$$

On doit avoir $f < d + (a+p)\log(t_{max}) - \sum_{i=1}^{p}\log(t_i)$ ou
$f = d + (a+p)\log(t_{max}) - \sum_{i=1}^{p}\log(t_i)$ et $e \geq c$ pour avoir $\frac{\pi(\kappa|t_{1:n})}{g(\kappa)}$ borné au $\mathcal{V}(\infty)$

(b) On effectue le calcul de $\hat{\kappa}_{MAP}$ à l'aide de la fonction *optimize*:

```
a=1e-3
b=1e-3
c=1e-3
d=1e-3
p=6
n=10
t=sort(df[,2])
neg_log_post = function(kappa){
   - dgamma(kappa, c, d, log=TRUE) + (a+p) * log((b + sum(t^kappa))) - p * log(kappa) - (kapp
a-1) *  sum(log(t[1:p]))
}


o = optimize(neg_log_post, interval=c(0.1, 10) )
hat_kappa_MAP = o$minimum
hat_kappa_MAP
```

```
## [1] 1.052583
```

Par la suite, on calcule $\hat{\sigma}^2_{\hat{\kappa}_{MAP}}$:

```
library(numDeriv)
```

```
## Warning: package 'numDeriv' was built under R version 4.0.3
```

```
hat_prec_kappa = hessian(neg_log_post, hat_kappa_MAP)
hat_v_kappa = 1/hat_prec_kappa
hat_v_kappa
```

```
##          [,1]
## [1,] 0.169822
```

(c)

On choisit la loi instrumentale $g \sim \mathcal{G}(e, f)$ de manière à ce qu'on ait les relations suivantes pour fixé l'espérance et la variance de la loi comme demandé dans l'énoncé:

$$\frac{e}{f} = \hat{\kappa}_{MAP}$$

et

$$\frac{e}{f^2} = \hat{\sigma}^2_{MAP}$$

On obtient

$$\frac{\hat{\kappa}_{MAP}}{\hat{\sigma}^2_{MAP}} = f$$

et

$$\frac{\hat{\kappa}^2_{MAP}}{\hat{\sigma}^2_{MAP}} = e$$

```
e = hat_kappa_MAP^2 / hat_v_kappa
f = hat_kappa_MAP / hat_v_kappa
# verification des conditions :
if (( e <= p + c ) & ((f < (a+p)*log(max(t))+d-sum(log(t)[1:p]))|((e >= c)&(f == (a+p)*log(ma
x(t))+d-sum(log(t)[1:p])))){
 print("loi instrumentale conforme")
}else{
 print("loi instrumentale non conforme")
}
```

```
## [1] "loi instrumentale non conforme"
```

(d)

```
func = function(kappa){
return(-neg_log_post(kappa) - dgamma(kappa,e,f,log=TRUE))
#-neg_log_post(kappa)-dgamma(kappa,e,f,log=TRUE)
}
```

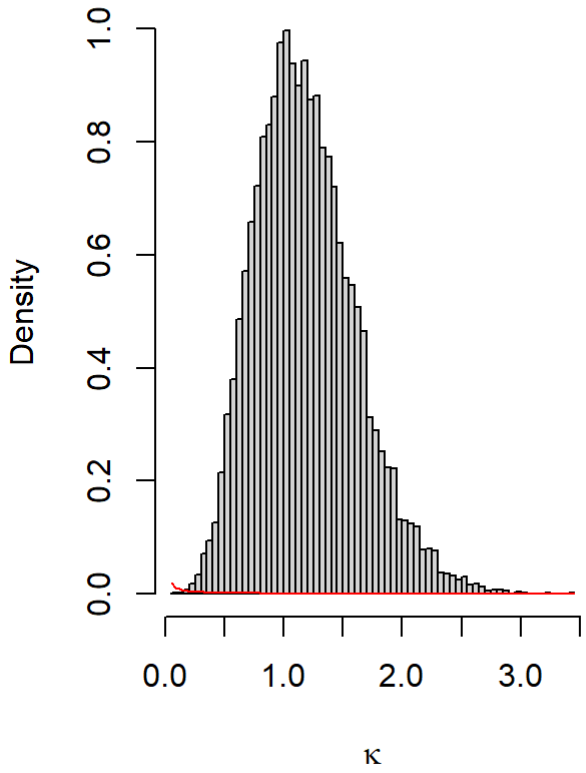
```
o=optimize(func,interval=c(0.1,10), maximum=TRUE)
logM = o$objective
M=exp(logM)
M
```

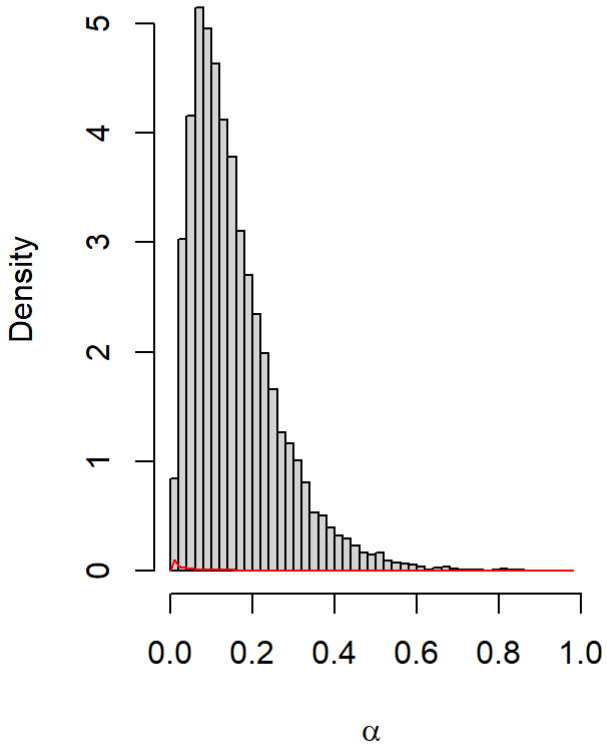
```
## [1] 1.247696e-12
```

Question 10

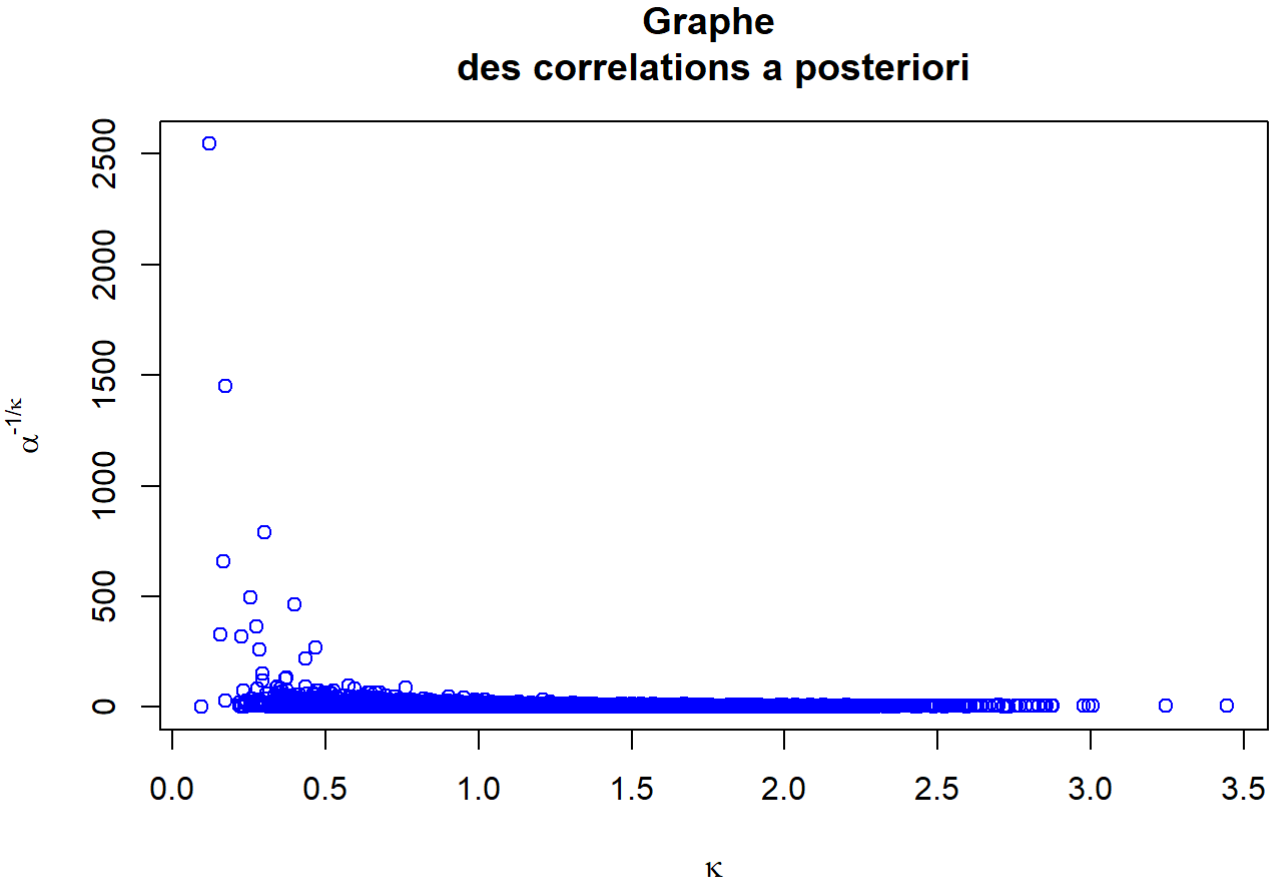
```
acceptation_rejet_kappa <-function(n_simu)
{
  n_accept=0
  n_generer=0
  X <-c()
  while (n_accept<n_simu)
  {
    n_generer=n_generer+1
    Xi <-rgamma(1,e,f)
    ui <-runif(1,0,1)
    pi =exp(-neg_log_post(Xi)) #il faut calculer pi(k | ti,p)

    Ri <- pi/(M*dgamma(Xi,e,f))
    if(ui<=Ri) #si ui est inferieur au rapport alors on conserve Xi
    {
      n_accept=n_accept+1
      X <-c(X,Xi)
    }
  }
  return (list(X=X, taux_acceptation= n_simu/n_generer))
}

nsimu=10000

kappa_posteriori=acceptation_rejet_kappa(nsimu)
alpha_posteriori=c()

for(kappa in kappa_posteriori$X )
{
  alpha=rgamma(1,a+p,b+sum(t^kappa))
  alpha_posteriori=c(alpha_posteriori,alpha)
}
```

```
par(mfrow=c(1,2))
hist(kappa_posteriori$X, breaks="Scott", probability=TRUE, main="", xlab=TeX('$\\kappa$'))
curve(dgamma(x,c,d), col='red', add=TRUE)

hist(alpha_posteriori, breaks="Scott", probability=TRUE, main="", xlab=TeX('$\\alpha$'))
curve(dgamma(x,a,b), col='red', add=TRUE)
```






```
plot(kappa_posteriori$X,alpha_posteriori^{-1/kappa_posteriori$X}, col='blue', main="Graphe
des correlations a posteriori", xlab=TeX("$\\kappa$"), ylab=TeX("$\\alpha^{-1/\\kappa}$"))
```

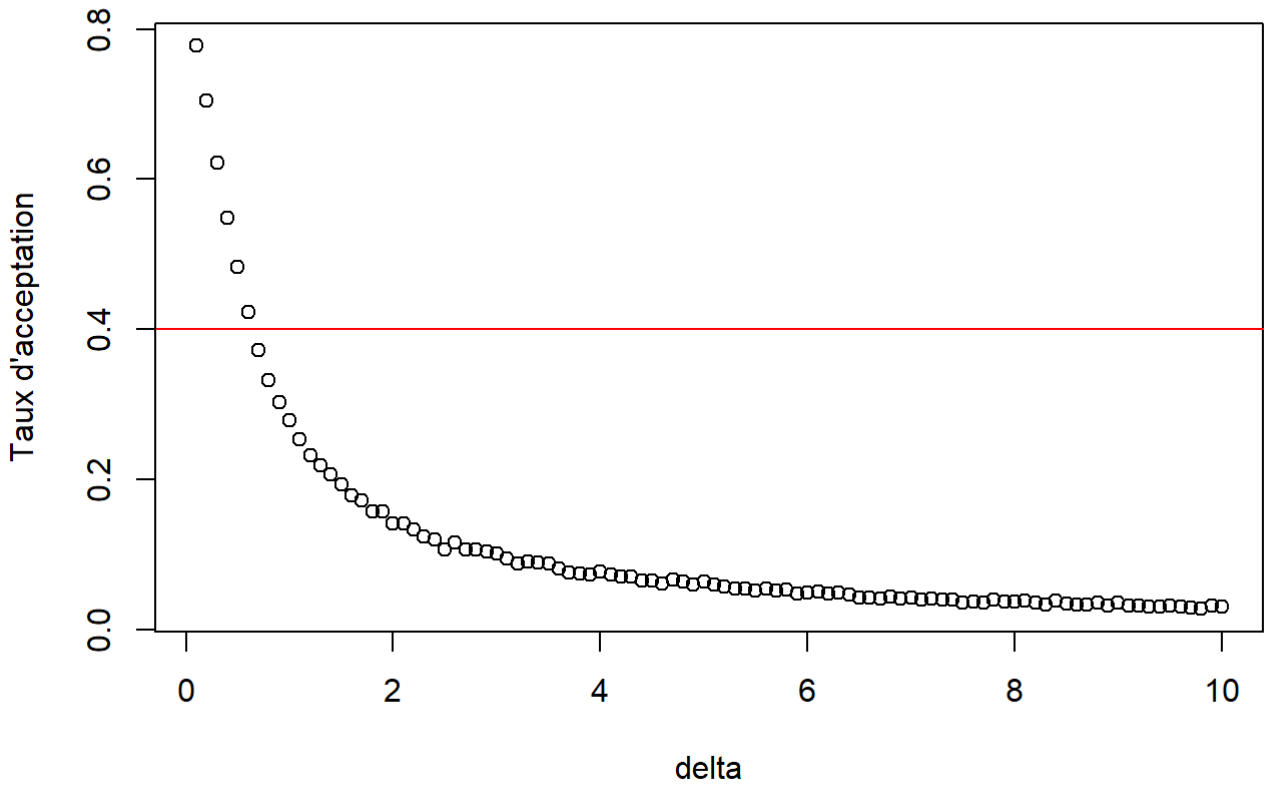


Question 11:

```
library("coda")
```

```
## Warning: package 'coda' was built under R version 4.0.5
```

```
dweibull<-function(alpha,kappa,t,p)
{
  return (((alpha*kappa)^p)*((prod(t[1:p]))^(kappa-1))*(exp(-alpha*sum(t^kappa))))
}
pi <- function(x)
{
  return(dgamma(x,c,d)*(x^p)*exp((x-1)*sum(log(t)))*(b+sum(t^x))^(-(a+p)))
}

MCMC<-function(t,p,iter,sdnorm,alpha0,kappa0){
  ##Taux d'acceptation
  accept=vector("numeric",iter)
  ##Valeur courante
  alphac=alpha0
  kappac=kappa0
  ##Création de la chaîne de Markov
  alpha_chain=alphac
  kappachain=kappac
  for(i in 1:(iter-1))
  {
    ##Mise à jour
    ##Gibbs
    alphacandidat=rgamma(1,a+p,b+sum(t^kappac))
    #Metropolis-Hastings
    kappacandidat=rnorm(1,mean=kappac,sd=sdnorm)
    r=(dgamma(kappacandidat,c,d)/dgamma(kappac,c,d))*(dweibull(alphacandidat,kappacandidat,t,
p)/dweibull(alphac,kappac,t,p))*(dnorm(kappac,mean=kappacandidat,sd=sdnorm)/dnorm(kappacandid
at,mean=kappac,sd=sdnorm))
    #r=(pi(kappacandidat)/pi(kappac))*(dnorm(kappac,mean=kappacandidat,sd=sdnorm)/dnorm(kappa
candidat,mean=kappac,sd=sdnorm))
    u=runif(1,0,1)
    minr1=min(r,1)
    if((!is.nan(minr1))&(u<minr1))
    {
      accept[i]=1
      kappac=kappacandidat
    }
    alphac=alphacandidat
    ##Stockage des données
    alpha_chain=c(alpha_chain,alphac)
    kappachain=c(kappachain,kappac)
    }
  return (list(acceptrate=mean(accept),result=mcmc(matrix(c(kappachain,alpha_chain),nrow=ite
r,ncol=2,byrow = FALSE))))
}


G=10000
delta=seq(0.1,10,by=0.1)
acceptrates=lapply(delta,function(sdnorm){return (MCMC(t,p,G, sdnorm,1,1)$acceptrate)})

plot(delta,unlist(acceptrates),ylab="Taux d'acceptation")
abline(h=0.4,col='red')
```



On obtient la valeur de delta suivante permettant d'obtenir un taux d'acceptation de 40%:

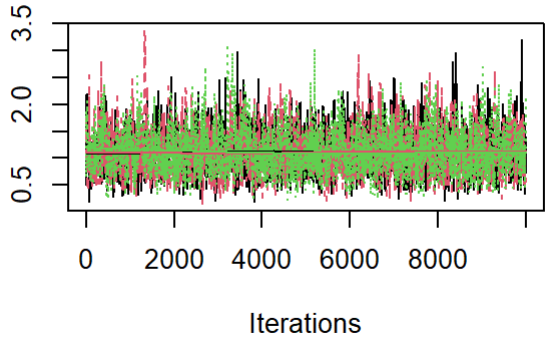
```
acceptrates[which.min(abs(0.4-as.numeric(acceptrates)))]
```

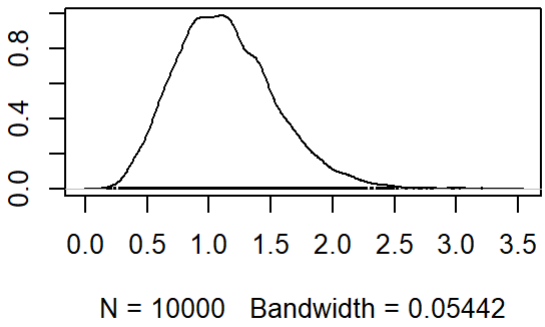
```
## [[1]]
## [1] 0.4228
```

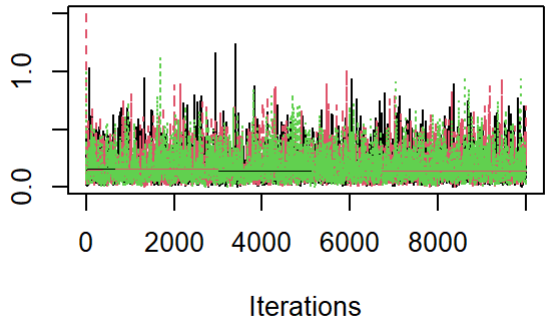
```
deltaopt=delta[which.min(abs(0.4-as.numeric(acceptrates)))]
```

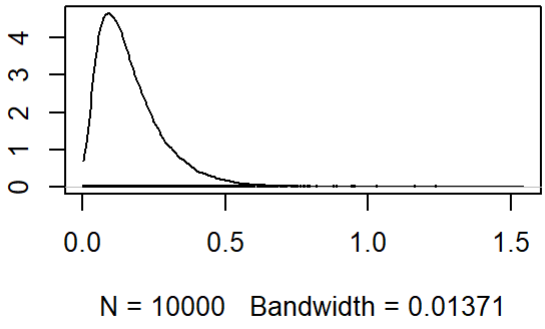
Question 13

```
chain1 <- MCMC(t,p,10000, deltaopt,0.5,0.5)$result
chain2 <- MCMC(t,p,10000, deltaopt,1.5,1)$result
chain3 <- MCMC(t,p,10000, deltaopt,1,1)$result
mcmcchains <- mcmc.list(list(chain1,chain2,chain3))
plot(mcmcchains)
```





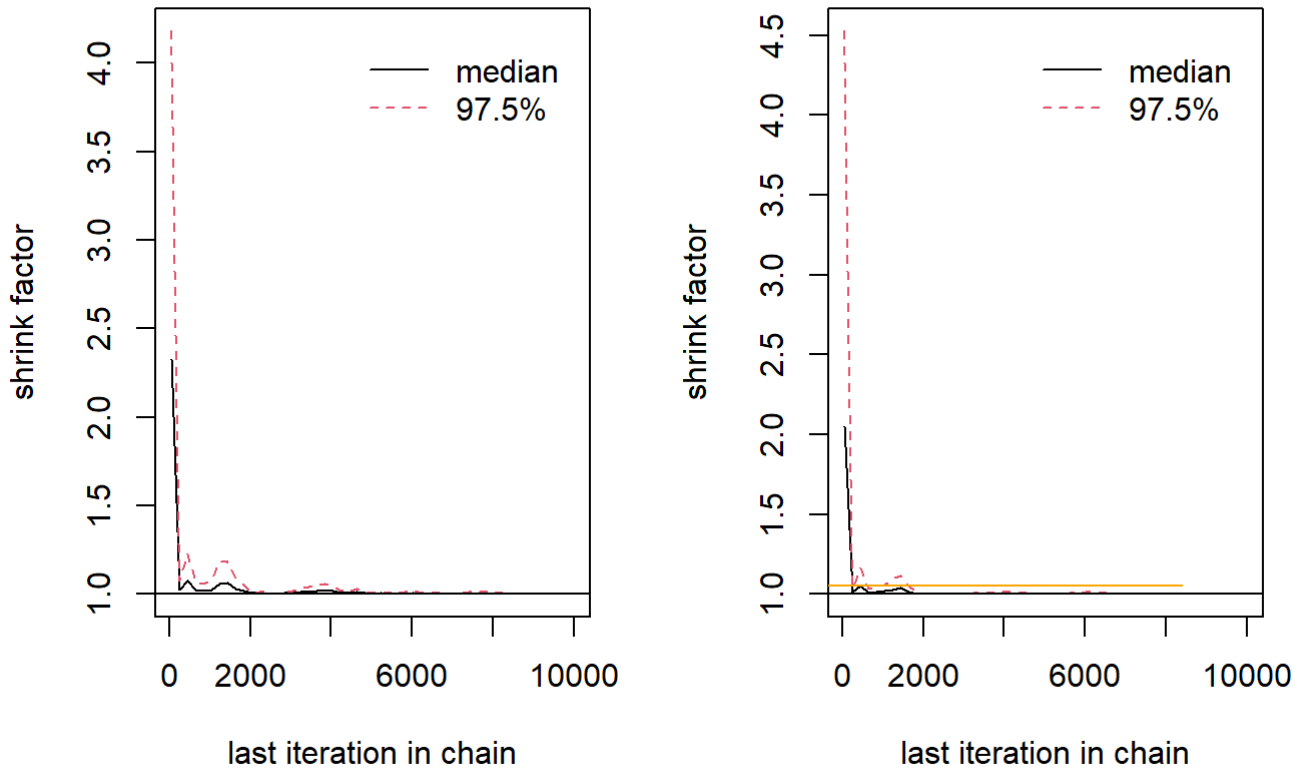





```
gelman.diag(mcmcchains)
```

```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]          1          1
## [2,]          1          1
##
## Multivariate psrf
##
## 1
```

```
gelman.plot(mcmcchains)
abline(h=1.05, col="orange")
```



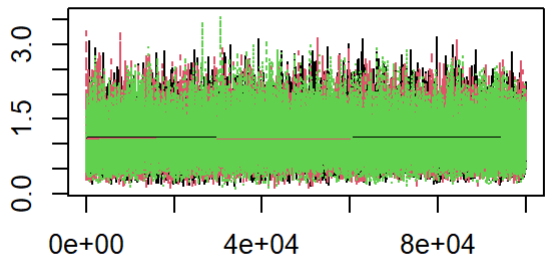
Tout d'abord, on peut remarquer que les chaînes se mélangent correctement, ainsi il n'y a pas d'indice de non convergence de l'algorithme.

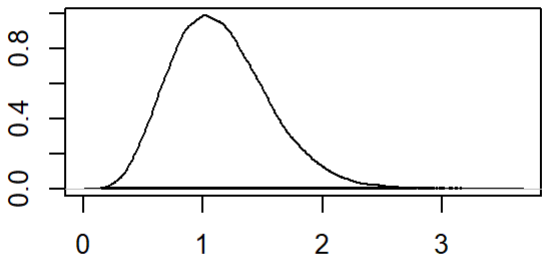
De plus, on ne peut pas vraiment conclure sur la convergence avec le critère de Gelman-Rubin, en effet il semble qu'il y ait stabilité mais il y a quelques pics pour les premières itérations. Il y a peut être un très léger problème de convergence au départ car en appliquant le critère de Gelman-Rubin, on doit obtenir stabilité et avoir R<1.05. De plus, il faut noter que la vitesse de convergence dépend du point de départ. Une solution est alors d'augmenter le nombre d'itérations comme nous le faisons ci-après avec 100 000 itérations.

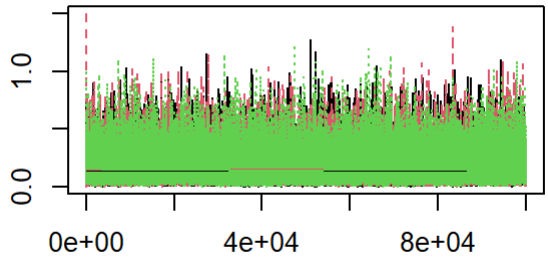
De plus, pour atteindre l'état stationnaire, on peut supposer d'après les courbes ci-dessus que l'on peut prendre à minima 2000 itérations.

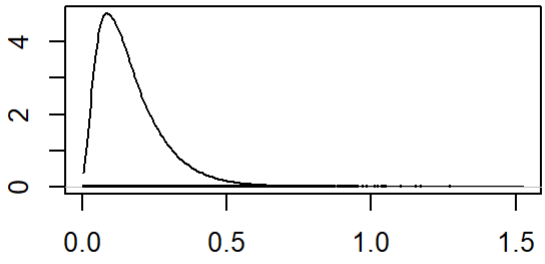
Par ailleurs, ci-dessous on augmente le nombre d'itérations pour vérifier l'amélioration de la convergence de l'algorithme:

```
chain1 <- MCMC(t,p,100000, deltaopt,0.5,0.5)$result
chain2 <- MCMC(t,p,100000, deltaopt,1.5,1)$result
chain3 <- MCMC(t,p,100000, deltaopt,1,1)$result
mcmcchains <- mcmc.list(list(chain1,chain2,chain3))
plot(mcmcchains, density=TRUE)
```





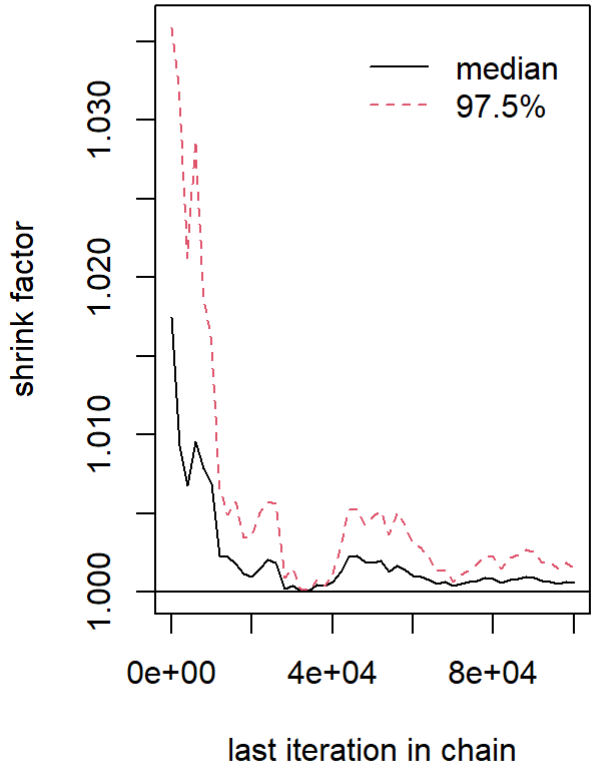


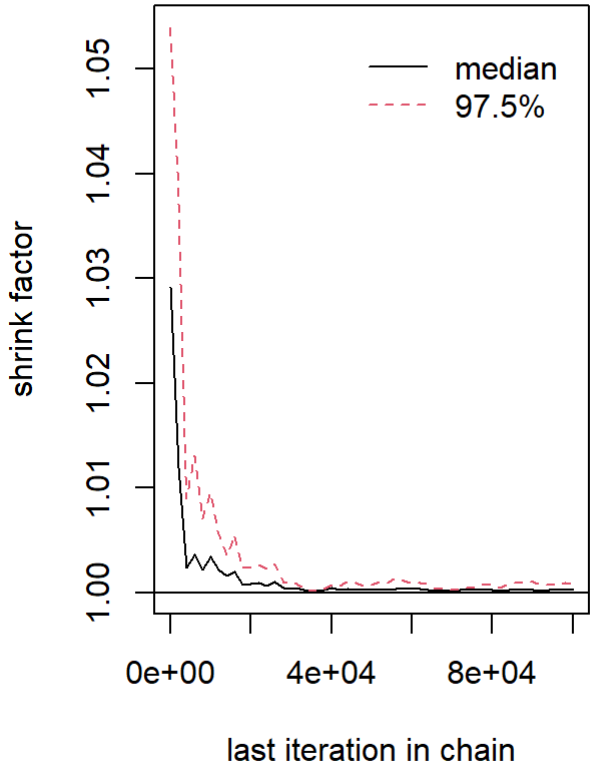



```
gelman.diag(mcmcchains)
```

```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]          1          1
## [2,]          1          1
##
## Multivariate psrf
##
## 1
```

```
gelman.plot(mcmcchains)
```





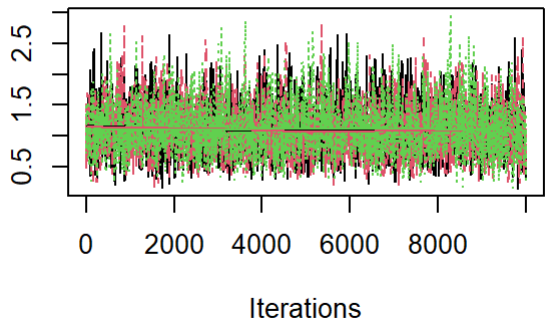
Tout d'abord, on peut remarquer que les chaînes se mélangent correctement, ainsi il n'y a pas d'indice de non convergence de l'algorithme.

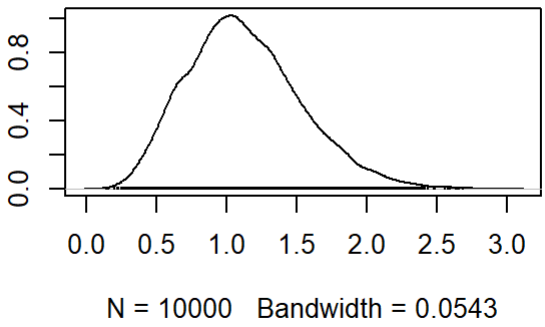
De plus, en utilisant le critère de Gelman-Rubin, on peut appliquer la règle standard car le graphe nous montre qu'il y a stabilité et on peut remarquer qu'on a R<1.05 à partir d'un certain rang, donc il n'y a pas de problème de convergence majeure désormais.

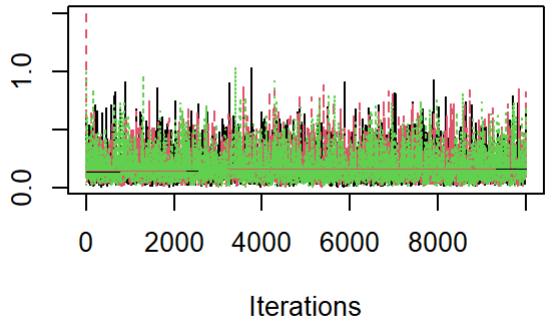
Question 14

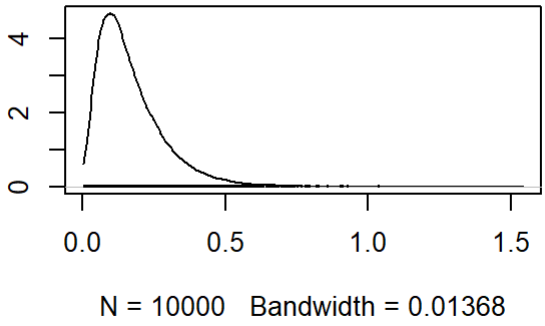
On resimule les 10000 réalisations dans un premier temps:

```
G1 <- 10000

chain1_G1 <- MCMC(t,p,10000, deltaopt,0.5,0.5)$result
chain2_G1 <- MCMC(t,p,10000, deltaopt,1.5,1)$result
chain3_G1 <- MCMC(t,p,10000, deltaopt,1,1)$result
mcmcchains <- mcmc.list(list(chain1_G1,chain2_G1,chain3_G1))
plot(mcmcchains)
```









On supprime désormais les itérations correspondant au temps de chauffe, et par ailleurs on effectue les différents calculs avec un temps de chauffe à 4000:

```
mcmctc_G1=mcmc.list(list(as.mcmc(chain1_G1[2001:10000,]),as.mcmc(chain2_G1[2001:10000,]),as.m
cmc(chain3_G1[2001:10000,])))
ESS_G1=effectiveSize(mcmctc_G1)
names(ESS_G1)=list("kappa","alpha")
ESS_G1
```

```
##    kappa    alpha
## 1670.200 3094.387
```

La taille de l'ESS est faible, c'est peut-être dû à une grande auto-corrélation pour nos différentes chaînes MCMC qui réduit fortement la taille de l'échantillon a posteriori.On augmente donc par la suite le nombre d'itérations G pour pouvoir calculer les densités a posteriori avec plus de précision, on effectue donc 100000 itérations, de plus on choisit cette fois un temps de chauffe à 25000 afin de ne conserver que les observations les plus précises:

```
mcmctc=mcmc.list(list(as.mcmc(chain1[25001:100000,]),as.mcmc(chain2[25001:100000,]),as.mcmc(c
hain3[25001:100000,])))

ESS=effectiveSize(mcmctc)

names(ESS)=list("kappa","alpha")

ESS
```

```
##    kappa    alpha
## 15284.96 25615.62
```

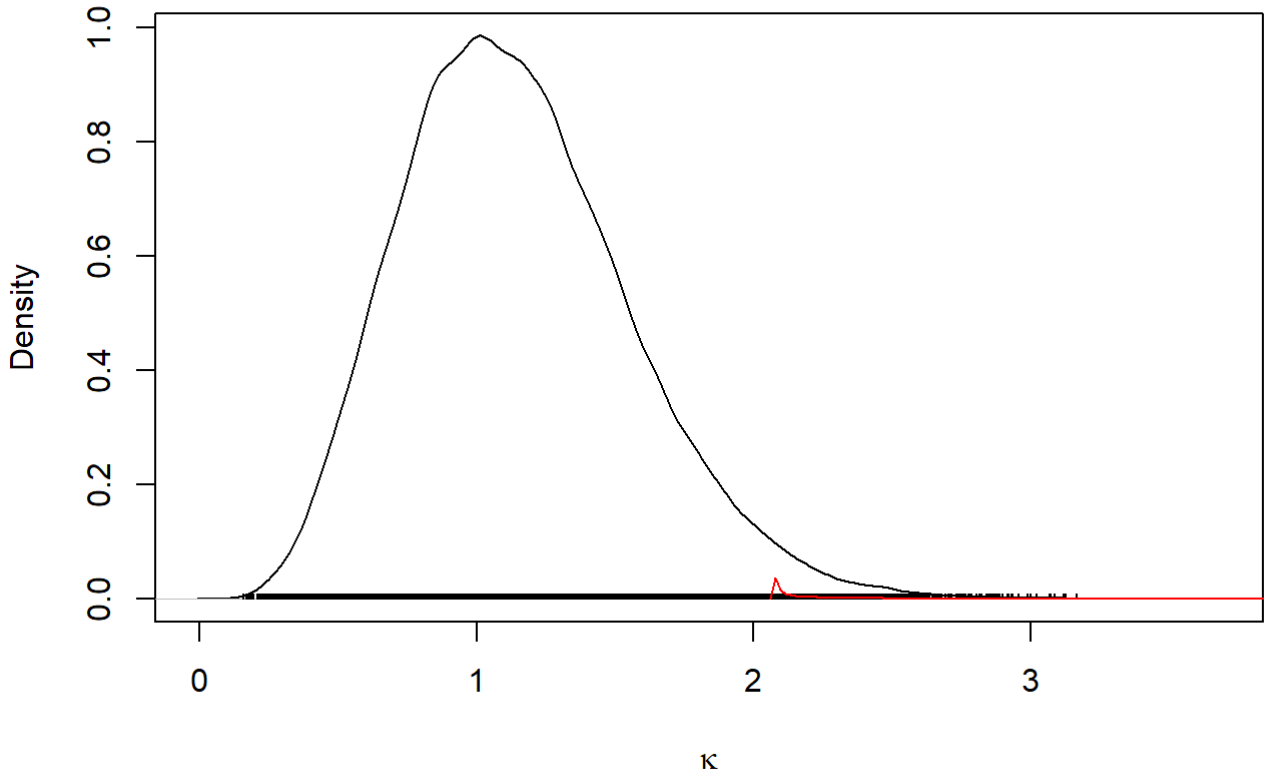
On peut remarquer que l' ESS est désormais supérieur à celui que l'on avait obtenu précédemment et on peut donc considérer que l'ESS est maintenant "satisfaisant".

Question 15

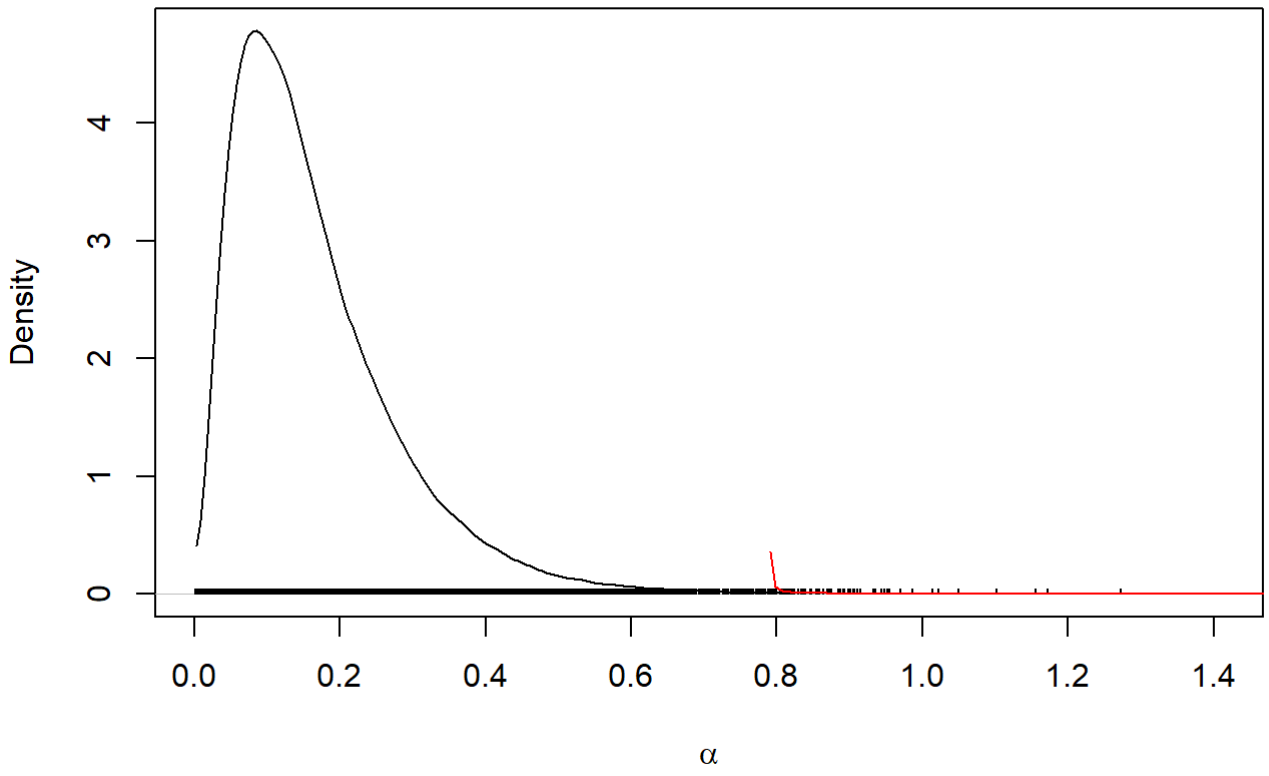
On représente ci-dessous les densités a priori et a posteriori pour chaque paramètre, ainsi que le graphe de corrélation a posteriori du couple $\left(\kappa, \alpha^{-1/\kappa}\right)$ puis on compare les résultats issus des deux algorithmes d'inférence bayésienne:

Voici le résultats en termes de densités a priori et a posteriori pour l'algorithme MCMC:

```
par(mfrow=c(1,2))
plot(mcmc.list(list(as.mcmc(chain1[25001:100000,1]),as.mcmc(chain2[25001:100000,1]),as.mcmc(c
hain3[25001:100000,1]))),trace=FALSE,xlab=TeX('$\\kappa$'),ylab="Density")
curve(dgamma(x,c,d), col='red', add=TRUE)
```




```
plot(mcmc.list(list(as.mcmc(chain1[25001:100000,2]),as.mcmc(chain2[25001:100000,2]),as.mcmc(c
hain3[25001:100000,2]))),trace=FALSE, xlab=TeX('$\\alpha$'),ylab="Density")
curve(dgamma(x,a,b), col='red', add=TRUE)
```

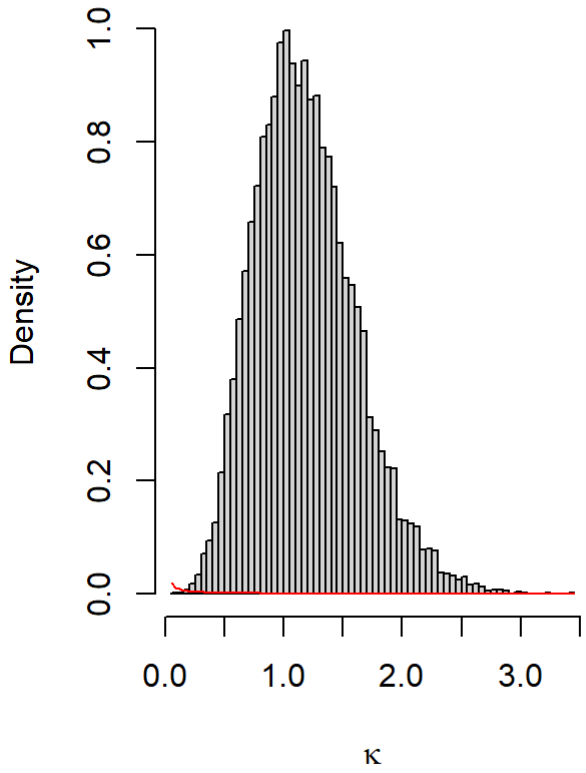


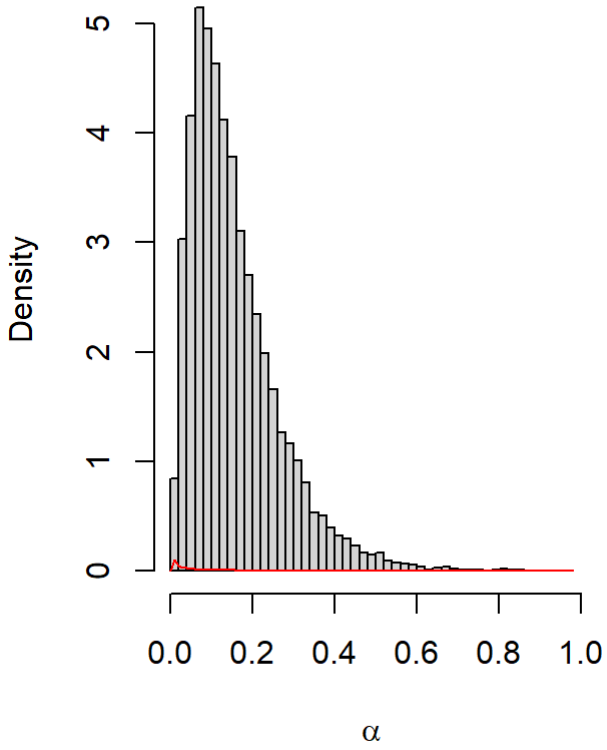
En traçant individuellement les résultats pour les trois chaines, on peut aussi voir qu'ils concordent c'est un bon signe de convergence.

Comparaisons des deux algorithmes d'inférences bayesiens:

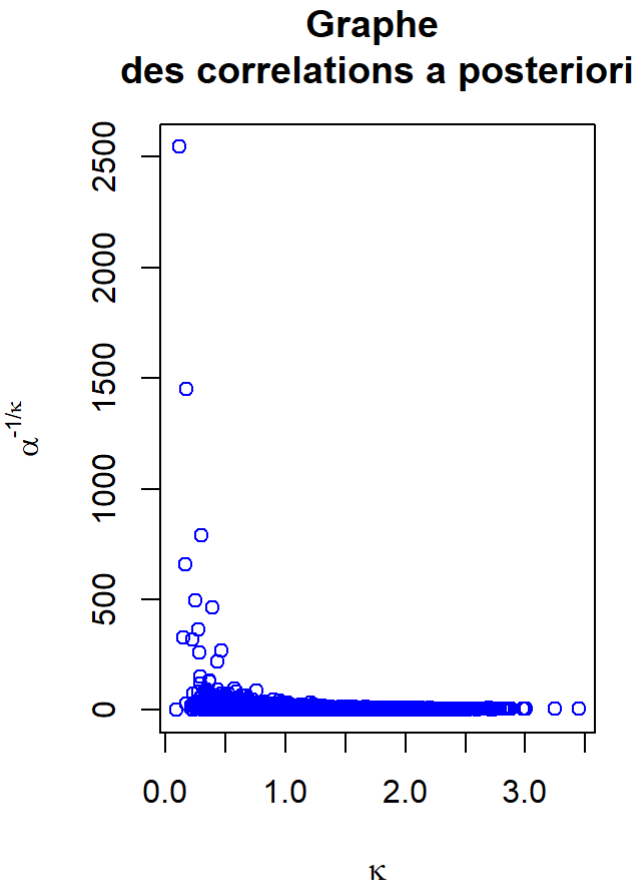
On a les résultats suivants:

```
par(mfrow=c(1,2))
hist(kappa_posteriori$X, breaks="Scott", probability=TRUE, main="", xlab=TeX('$\\kappa$'))
curve(dgamma(x,c,d), col='red', add=TRUE)

hist(alpha_posteriori, breaks="Scott", probability=TRUE, main="", xlab=TeX('$\\alpha$'))
curve(dgamma(x,a,b), col='red', add=TRUE)
```

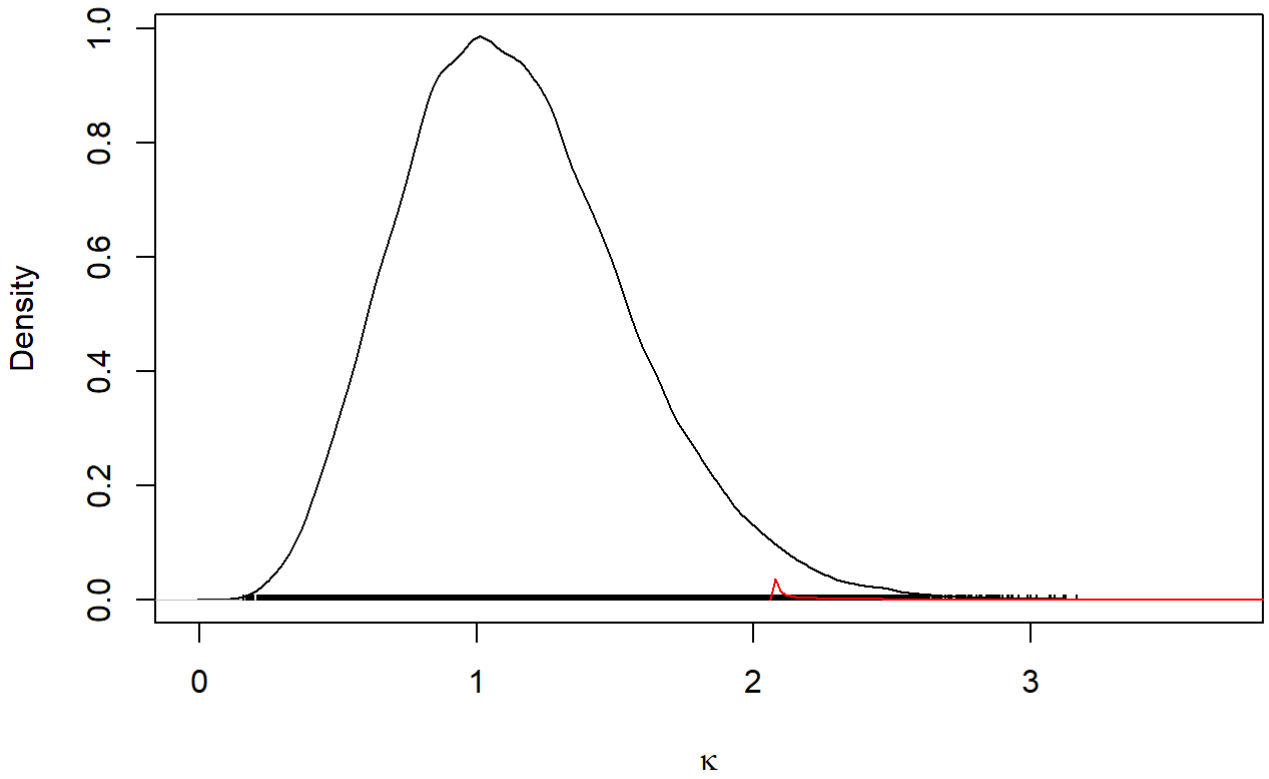




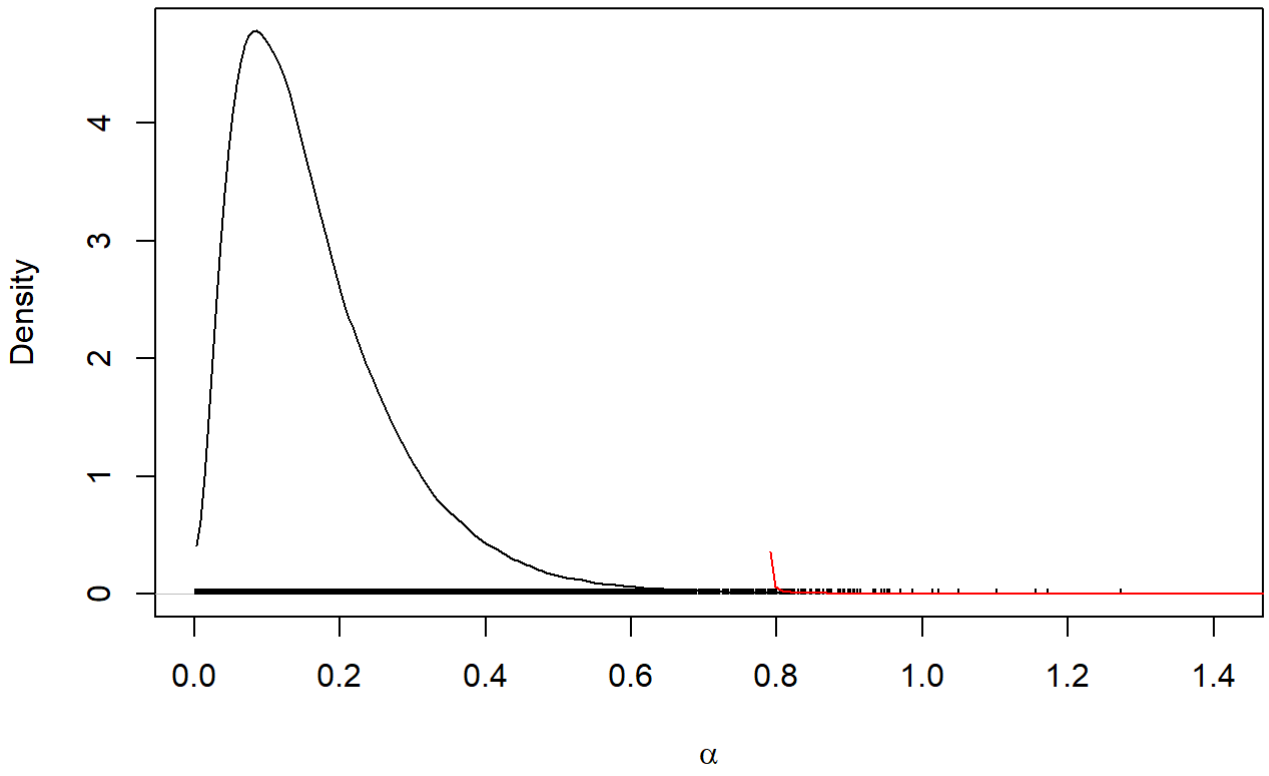

```
plot(kappa_posteriori$X,alpha_posteriori^{-1/kappa_posteriori$X}, col='blue', main="Graphe
des correlations a posteriori", xlab=TeX("$\\kappa$"), ylab=TeX("$\\alpha^{-1/\\kappa}$"))


par(mfrow=c(1,2))
```



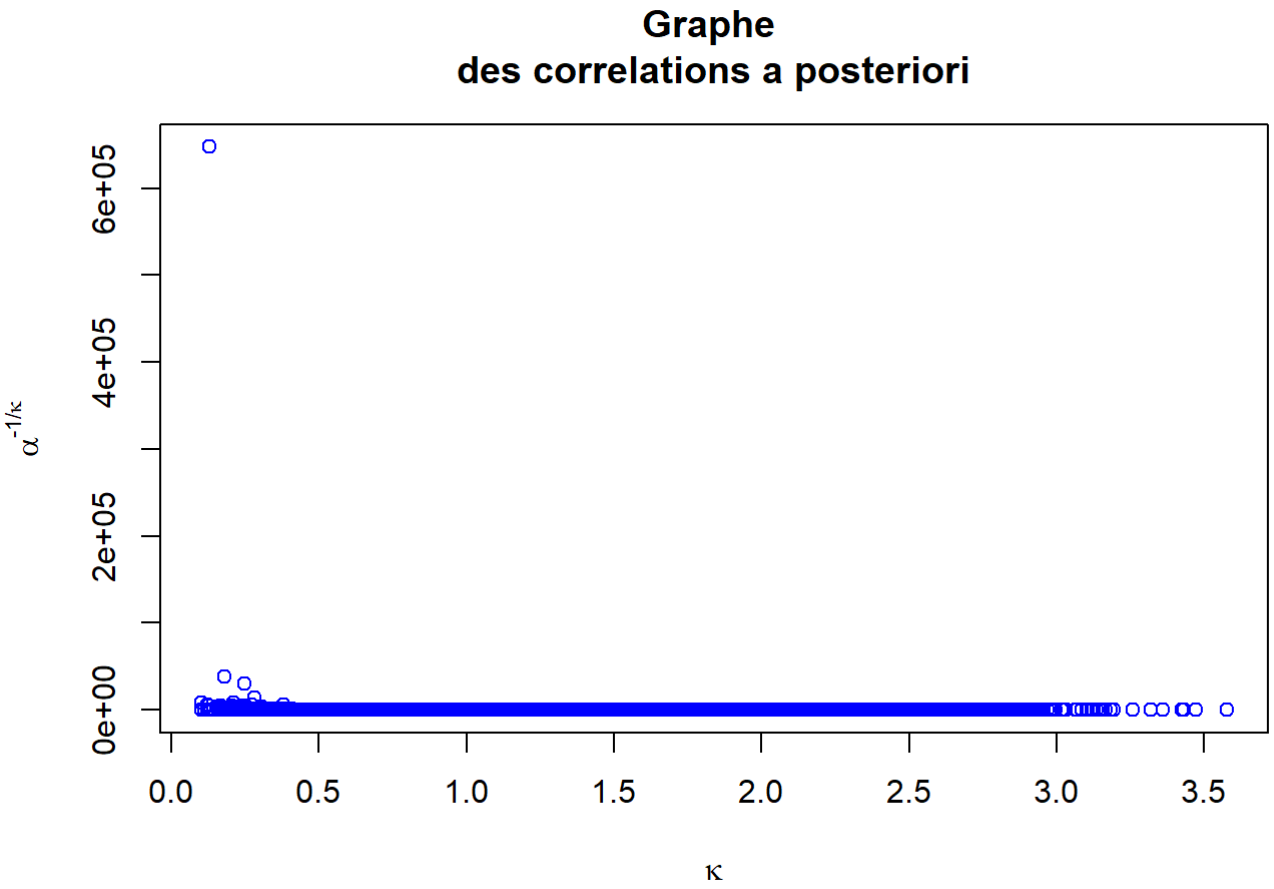

```
plot(mcmc.list(list(as.mcmc(chain1[25001:100000,1]),as.mcmc(chain2[25001:100000,1]),as.mcmc(c
hain3[25001:100000,1]))),trace=FALSE,xlab=TeX('$\\kappa$'),ylab="Density")
curve(dgamma(x,c,d), col='red', add=TRUE)
```




```
plot(mcmc.list(list(as.mcmc(chain1[25001:100000,2]),as.mcmc(chain2[25001:100000,2]),as.mcmc(c
hain3[25001:100000,2]))),trace=FALSE, xlab=TeX('$\\alpha$'),ylab="Density")
curve(dgamma(x,a,b), col='red', add=TRUE)
```




```
kappalist=mcmctc[,1]
kappalist=rbind(kappalist[[1]],kappalist[[2]],kappalist[[3]])
alphalist=mcmctc[,2]
alphalist=rbind(alphalist[[1]],alphalist[[2]],alphalist[[3]])



plot(as.numeric(kappalist),as.numeric(alphalist^{-1/kappalist}), col='blue', main="Graphe
des correlations a posteriori", xlab=TeX("$\\kappa$"), ylab=TeX("$\\alpha^{-1/\\kappa}$"))
```



On peut remarquer que les résultats renvoyés par les deux algorithmes sont globalement similaires que ce soit en terme de densité ou en terme de corrélations des variables. En effet, les moyennes et variances ainsi que l'aspect global est très semblable. Lorsque l'on compare l'application des deux algorithmes, le traitement des données nécessaire aux algorithmes MCMC est conséquent en réflexion et en calcul ce qui est moins le cas de la première approche bayésienne utilisant le principe d'acceptation-rejet. Egalement, du point de vue théorique, l'algorithme MCMC demande plus de réflexion pour être appréhendé. Toutefois, ces algorithmes ont tout de même des points communs (tests d'acceptation d'une valeur candidate à chaque itération). De plus, le fait qu'il y n'y ait pas à faire d'optimisation (contrairement à l'acceptation rejet) pour trouver de paramètre adéquat, le fait qu'il y ait des résultats de convergence théorique et enfin le fait qu'il y ait de nombreuses façons de vérifier ses résultats et la propriété de convergence en pratique font que l'on préfèrera utiliser l'algorithme MCMC (en plus des résultats convenables qu'il fournit ici).