TP 2 (Corrigé) - Statistique bayésienne

Franck Corset

Master 2 - SSD

Cas Gaussien

On suppose que $X \sim \mathcal{N}(\mu, \sigma^2)$

On prend comme loi a priori sur μ, $\mathcal{N}(\mu_0, \sigma_0^2)$ et sur σ^2 une loi inverse gamma de paramètres a et b.

Mettre en place un programme permettant de calculer les estimateurs bayésiens de ce modèle.

On sait que les lois a posteriori sont :

- $\pi(\mu|\sigma^2) = \mathcal{N}(\dfrac{\sigma_0^2 \sum x_i + \sigma^2 \mu_0}{n\sigma_0^2 + \sigma^2}; \dfrac{\sigma^2 \sigma_0^2}{n\sigma_0^2 + \sigma^2})$
- $\pi(\sigma^2|\mu) = IG(a + \frac{n}{2}; b + \dfrac{1}{2} \sum (x_i - \mu)^2)$

```
require(MCMCpack)
```

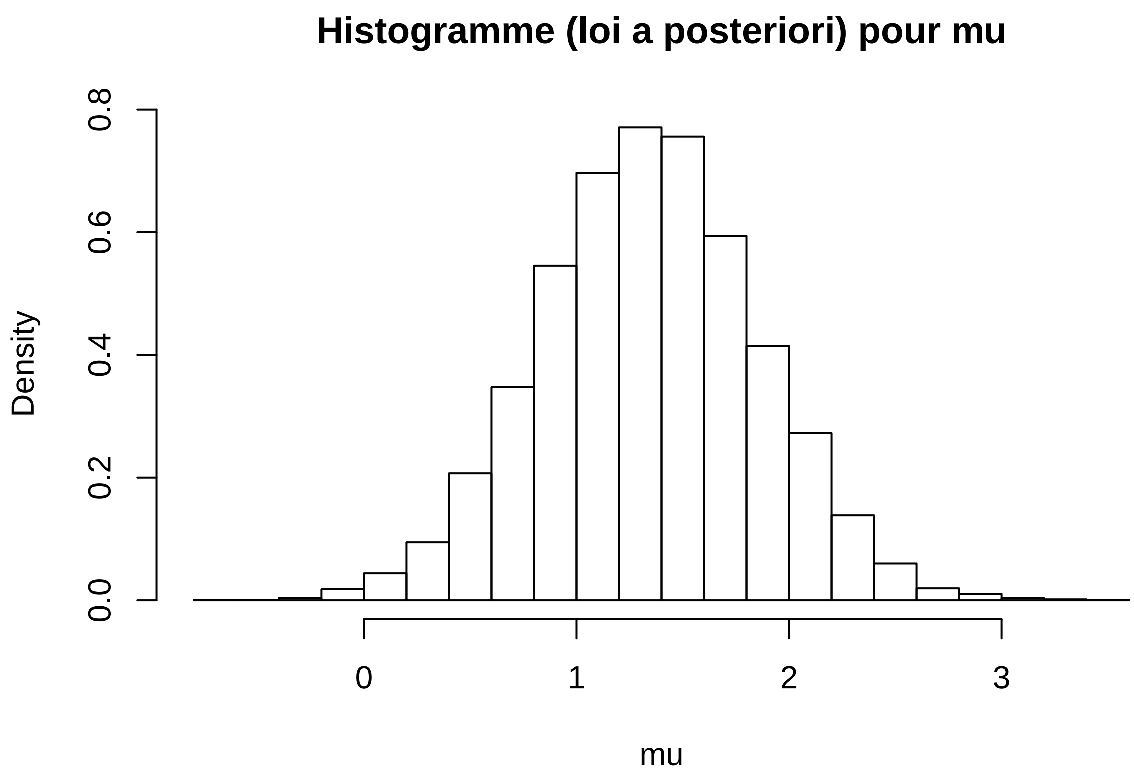
```
## Loading required package: MCMCpack

## Loading required package: coda

## Loading required package: MASS

## ##
## ## Markov Chain Monte Carlo Package (MCMCpack)

## ## Copyright (C) 2003-2016 Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park

## ##
## ## Support provided by the U.S. National Science Foundation

## ## (Grants SES-0350646 and SES-0350613)
## ##
```

```
mu.true<-2
sig2.true<-4
par.true<-c(mu.true,sig2.true)
n<-10 # petite taille d'échantillon



# Simulation d'un échantillon
ech<-rnorm(n,mu.true,sqrt(sig2.true))
est.mle<-cbind(mean(ech),var(ech))
```

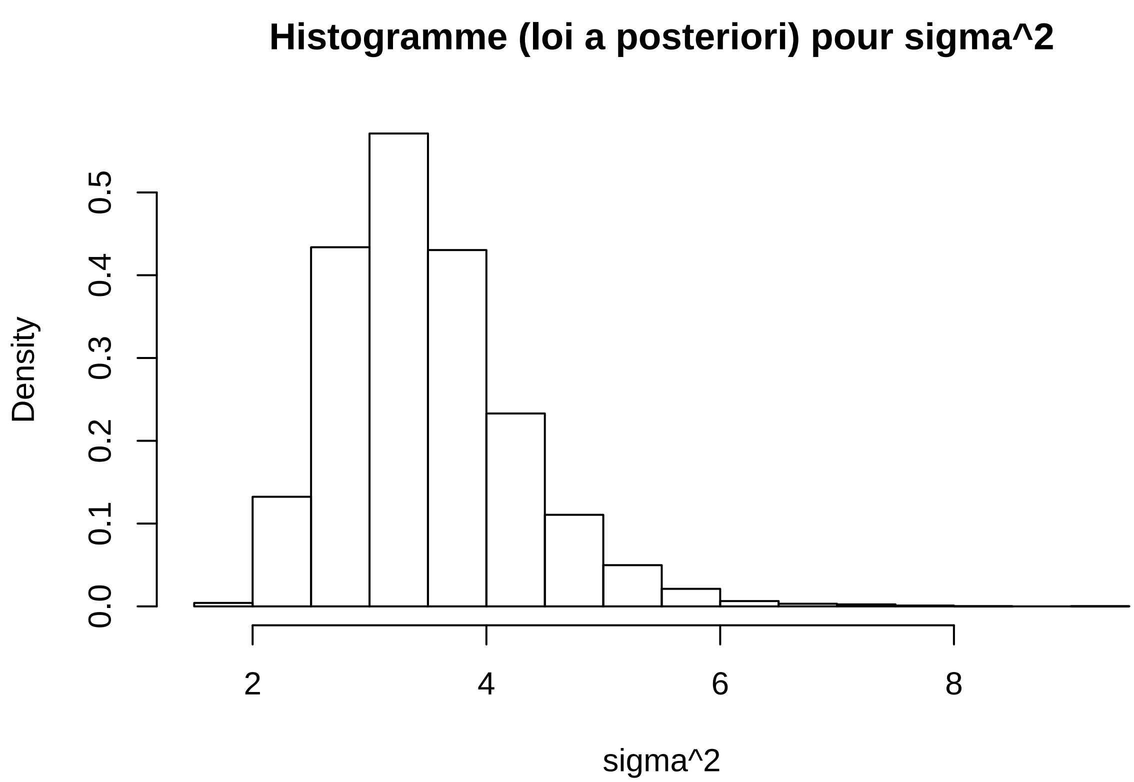
```r
mu.mle<-est.mle[1]
sig2.mle<-est.mle[2]


# Définition des hyperparamètres
## Pour mu
mu0<-2   # Notre expert est bon
sig2_0<-1

## Pour sigma2
moy.prior.sig2<- 4
var.prior.sig2<- 1

a<-2+moy.prior.sig2^2/var.prior.sig2  # Calcul des hyperparamètres de la loi IG
b<-moy.prior.sig2*(a-1)              # à partir de la moyenne et variance a priori


# Algo de Gibbs

K<- 10000
mu.c <- mu.mle     # valeur courante du paramètre
sig2.c <- sig2.mle   # On choisit ici l'estimation par maximum de vraisemblance

mu.bay <-rep(0,K)   # initialisation du vecteur où l'on va stocker les valeurs des paramètres
sig2.bay <-rep(0,K)

for(k in 1:K){
  # On commence par simuler selon la loi a posteriori de mu sachant sigma^2
    mu.bay[k] <- rnorm(1,(sig2_0*sum(ech)+sig2.c*mu0)/(n*sig2_0+sig2.c),
                       sqrt(sig2.c*sig2_0/(n*sig2_0+sig2.c)))
    mu.c<-mu.bay[k]

    # On simule sigma2

    sig2.bay[k]<-rinvgamma(1,shape =a+n/2,scale=1/2*sum((ech-mu.c)^2)+b)
    sig2.c<- sig2.bay[k]
}
ech.bay<-cbind(mu.bay,sig2.bay)

est.bay<-colMeans(ech.bay)

est.bay.mu<-est.bay[1]
est.bay.sig2<-est.bay[2]

res<-data.frame(mu=c(mu.true,mu.mle,est.bay.mu),sig2=c(sig2.true,sig2.mle,est.bay.sig2),
                row.names=c("True","MLE","Bayésien"))
res
```

```
##                 mu     sig2
## True     2.000000 4.000000
## MLE      1.128219 1.554098
## Bayésien 1.341588 3.475956
```

```
hist(ech.bay[,1],freq=F, xlab="mu",main = "Histogramme (loi a posteriori) pour mu")
```




```
hist(ech.bay[,2],freq=F, xlab="sigma^2",main = "Histogramme (loi a posteriori) pour sigma^2")
```



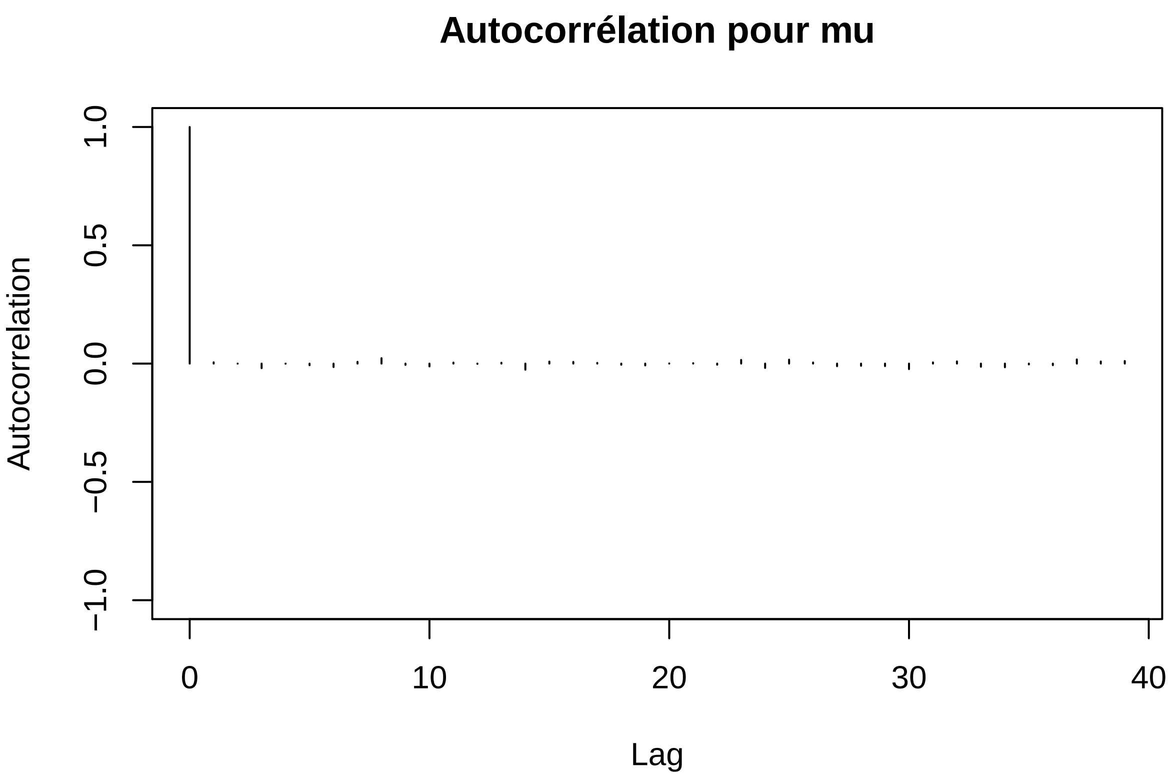
```
# Intervalle de crédibilité à 95%
alpha<-0.05
ic.mu <- quantile(ech.bay[,1],c(alpha/2,1-alpha/2))
ic.mu
```

```
##     2.5%    97.5%
## 0.335284 2.340713
```

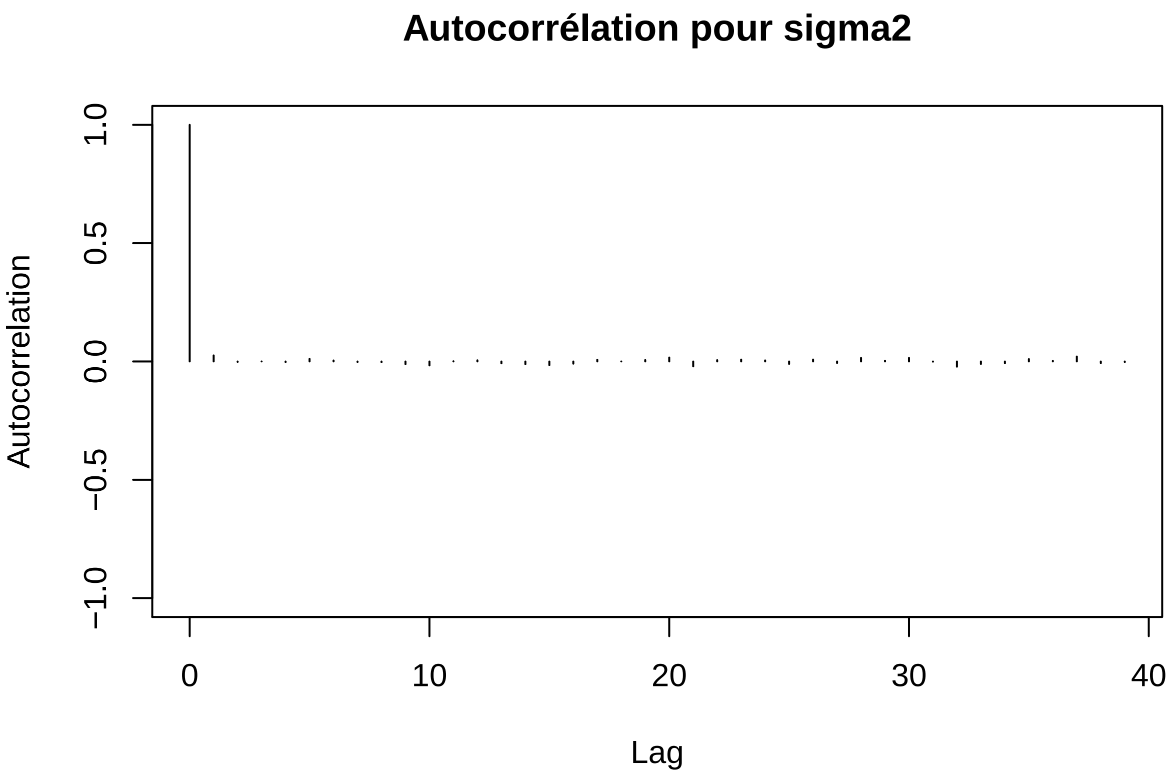
```
ic.sig2 <- quantile(ech.bay[,2],c(alpha/2,1-alpha/2))
ic.sig2
```

```
##     2.5%    97.5%
## 2.312453 5.284494
```

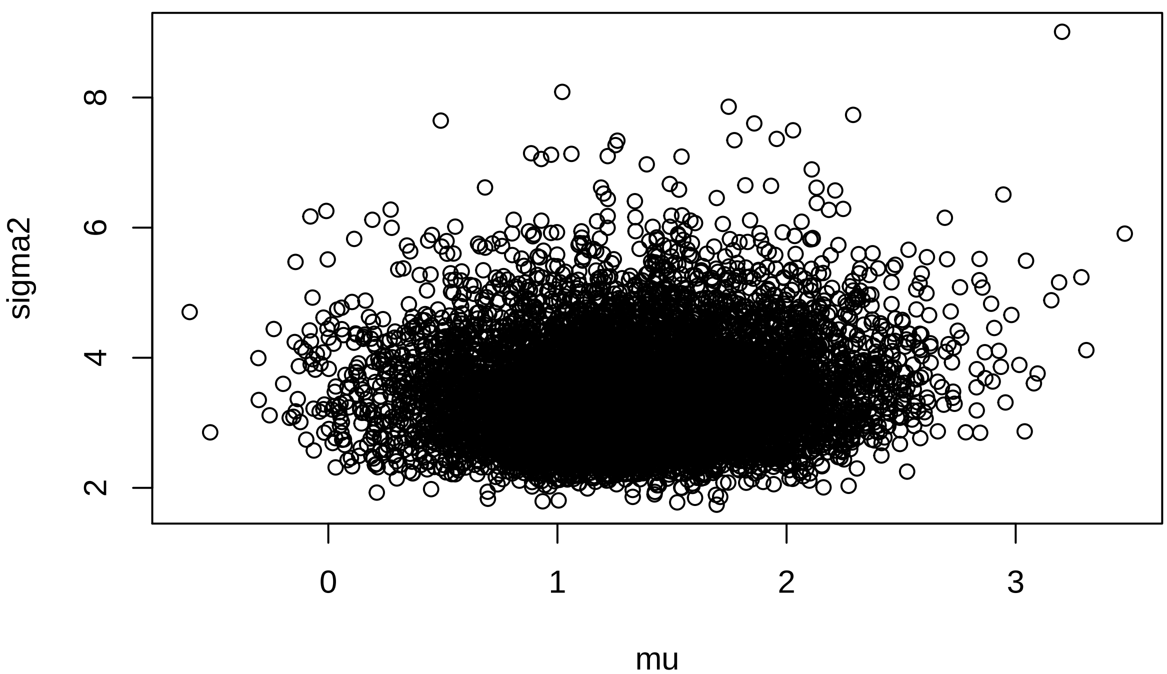
```
# Analyse des sorties de l'algo MCMC
autocorr.plot(ech.bay[-c(1:2000),1],auto.layout = F,main="Autocorrélation pour mu")
```



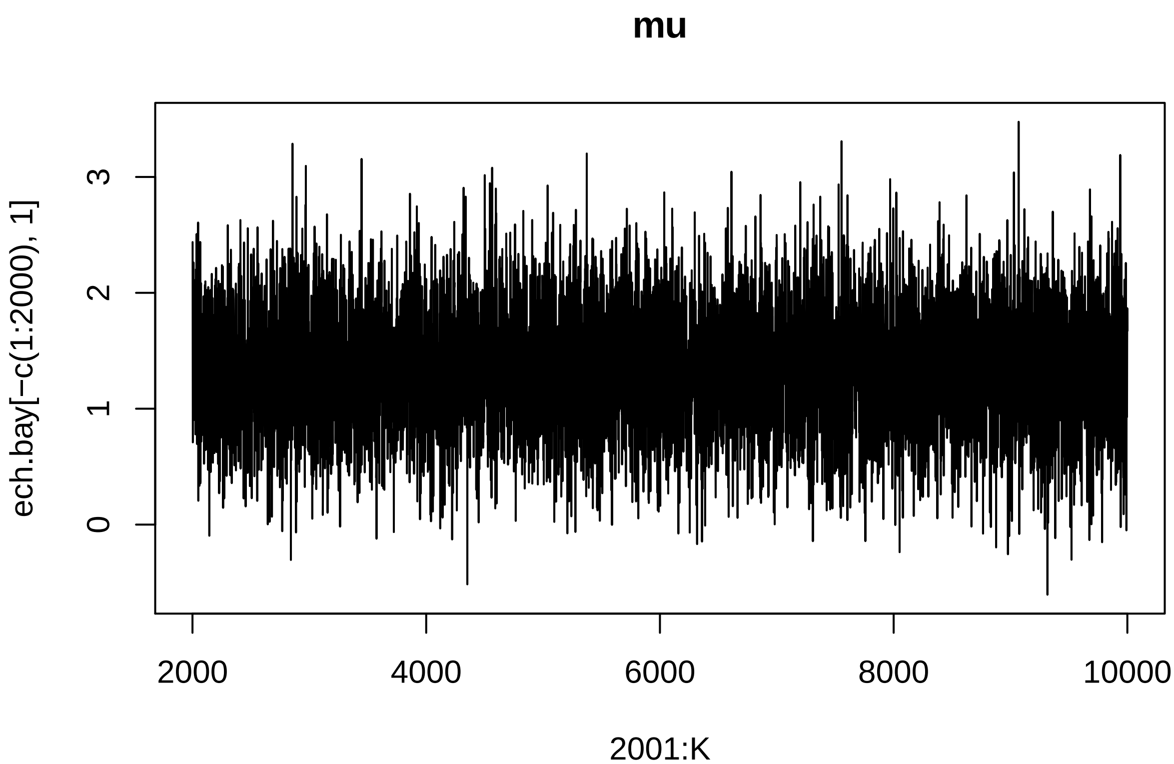
```
autocorr.plot(ech.bay[-c(1:2000),2],auto.layout = F,main="Autocorrélation pour sigma2")
```



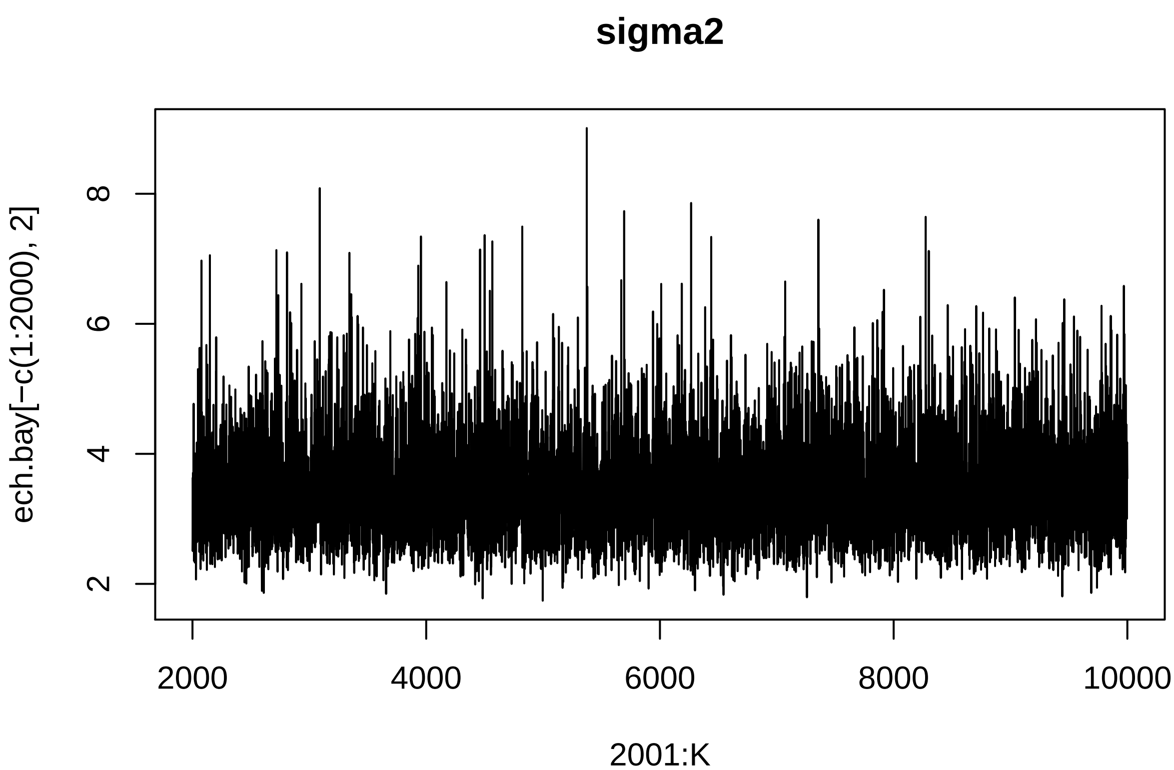

```
plot(ech.bay[-c(1:2000),1],ech.bay[-c(1:2000),2],xlab="mu",ylab="sigma2")
```



```
plot(2001:K,ech.bay[-c(1:2000),1],type="l",main="mu")
```




```
plot(2001:K,ech.bay[-c(1:2000),2],type="l",main="sigma2")
```



Loi de Weibull

Soit $X \sim \mathcal{W}(\eta, \beta)$ de densité à support sur $\mathbb{R}^+$

$$f(x) = \frac{\beta}{\eta}\left(\frac{x}{\eta}\right)^{\beta-1} e^{-\left(\frac{x}{\eta}\right)^{\beta}}$$

On prend une loi uniforme sur $[1, 5]$ pour β et une loi inverse gamma pour η.

Ecrire la vraisemblance et donner la loi a posteriori. Mettre en oeuvre un programme permettant de calculer l'estimateur bayésien. Le comparer à l'estimateur du maximum de vraisemblance.

```
require(MCMCpack)
n<-10
eta.true<-1000
beta.true<-1.5
par.true<-c(eta.true,beta.true)

# Simulation de l'échantillon
ech<-rweibull(n,shape=beta.true,scale=eta.true)

# Estimation par Maximum de vraisemblance
logVraiWeib<-function(x,donnees){
  # x[1] = eta
  # x[2] = beta
  # donnees = échantillon
  n<-length(donnees)
  return(n*log(x[2]/x[1]) + (x[2]-1)*sum(log(donnees/x[1]))
        -sum((donnees/x[1])^x[2]))
}
est.mle <- optim(c(200,2),function(x)logVraiWeib(x,donnees=ech),
                 control = list(fnscale=-1))$par
```

```
## Warning in log(x[2]/x[1]): production de NaN

## Warning in log(x[2]/x[1]): production de NaN

## Warning in log(x[2]/x[1]): production de NaN

## Warning in log(x[2]/x[1]): production de NaN

## Warning in log(x[2]/x[1]): production de NaN

## Warning in log(x[2]/x[1]): production de NaN
```

```
eta.mle<-est.mle[1]
beta.mle<-est.mle[2]
est.mle
```

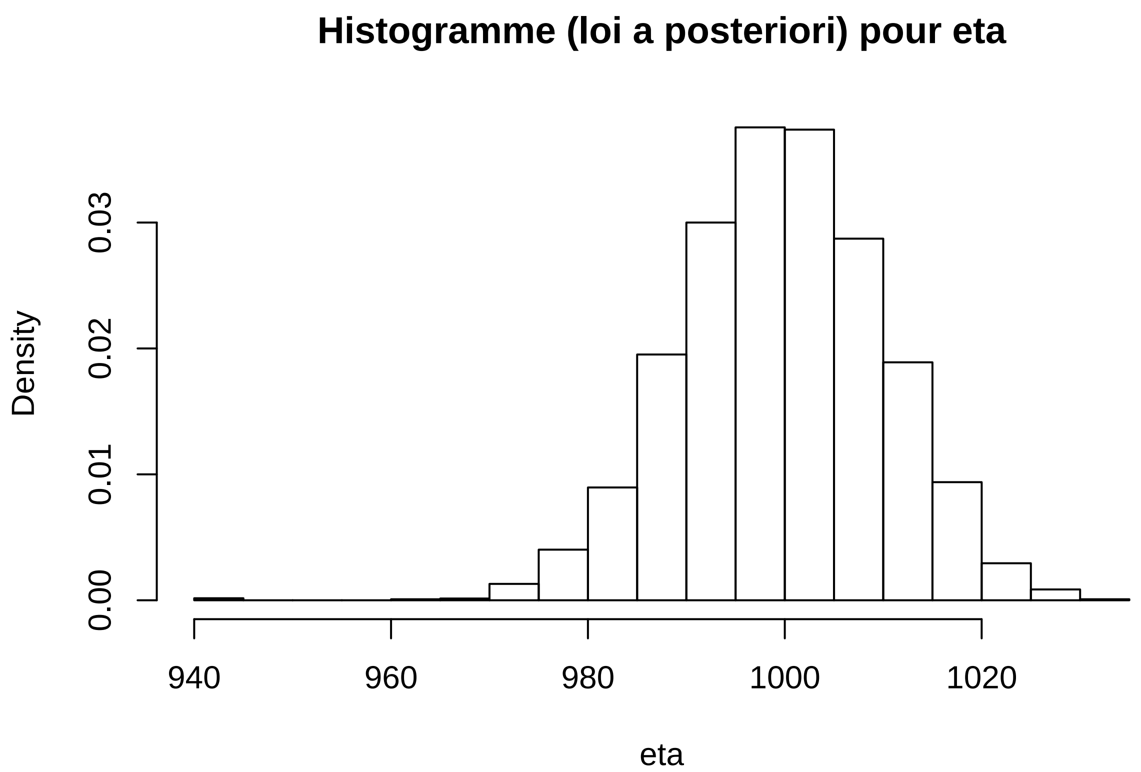
```
## [1] 937.321966   2.838384
```

```
# Loi a priori
## Pour eta
moy.prior.sig2<- 1000
var.prior.sig2<- 100

a<-2+moy.prior.sig2^2/var.prior.sig2  # Calcul des hyperparamètres de la loi IG
b<-moy.prior.sig2*(a-1)              # à partir de la moyenne et variance a priori

## Pour beta
beta.inf<-1
beta.sup<-3


# Calcul de la log loi a posteriori (sans les constantes)

logPost<-function(x,donnees,hyperpara=c(a,b,beta.inf,beta.sup)){
  # x[1] = eta
  # x[2] = beta
  # donnees = échantillon
  # hyperpara[1] = a
  # hyperpara[2] = b
  # hyperpara[3] = beta.inf
  # hyperpara[4] = beta.sup
  n<-length(donnees)
  res<-ifelse((x[2]<hyperpara[3])||(x[2]>hyperpara[4]),-Inf,
              logVraiWeib(x,donnees = ech)-(a+1)*log(x[1])-b/x[1])
  return(res)
}


# Algo de Gibbs
# On passe tout en log car sinon pb numérique (vraisemblance =  0)

# Initialisation
eta.c<-est.mle[1]
beta.c<-est.mle[2]

# Paramètre de l'algo
K<-10000
sig.eta<-50
sig.beta<-1


eta.bay <-rep(0,K)   # initialisation du vecteur où l'on va stocker les valeurs des paramètres
beta.bay <-rep(0,K)

for(k in 1:K){
  # Etape eta
  # On veut simuler selon pi(eta/beta(k-1))
  # Etape Metroplis-Hastings avec une loi normale
  eta.star <- rnorm(1,eta.c,sqrt(sig.eta))
```

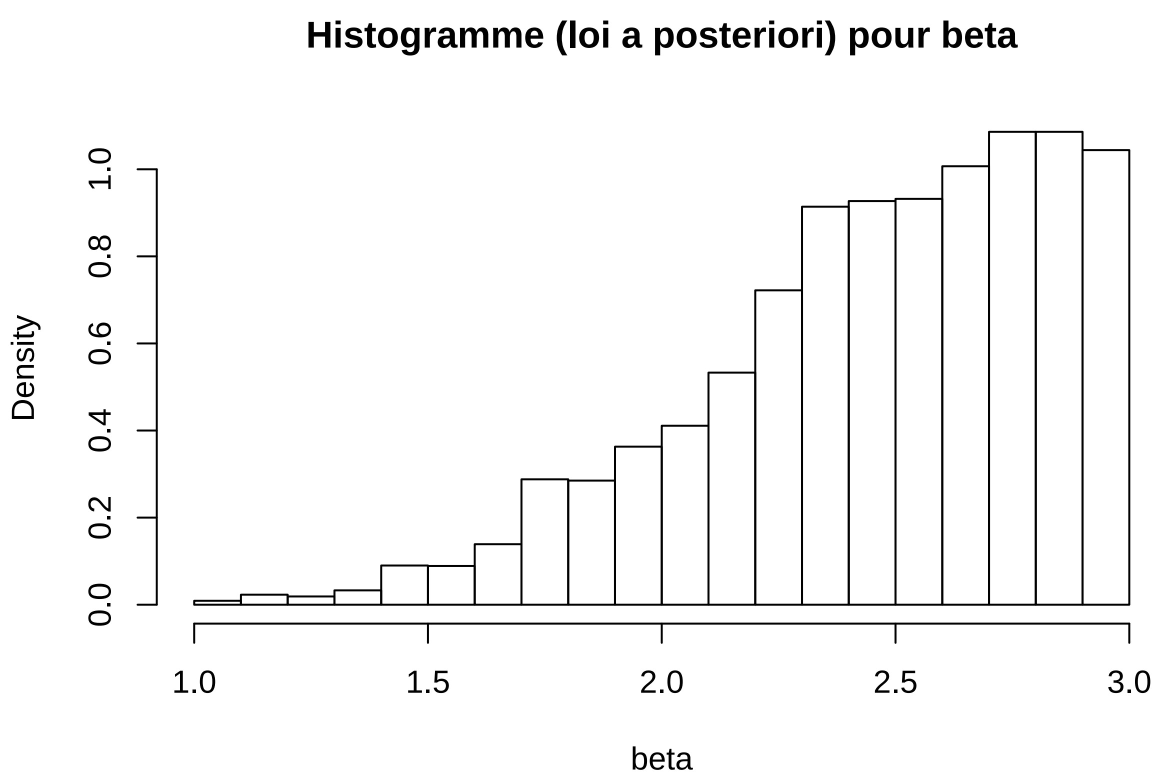
```r
  # Calcul de logr.eta
  logr.eta<-logPost(x = c(eta.star,beta.c),donnees=ech)-logPost(x =
        c(eta.c,beta.c),donnees=ech)
  # On accepte la nouvelle valeur avec proba = min(r,1)
  # Pour cela on tire une loi uniforme
  u<-runif(1)
  eta.bay[k]<-eta.star*(log(u)<=logr.eta)+eta.c*(log(u)>logr.eta)
  eta.c<-eta.bay[k] # On remet à jour la valeur courante de eta

  # Etape beta
  # On veut simuler selon pi(beta|eta(k))
  # Etape Metroplis-Hastings avec une loi normale
  beta.star <- rnorm(1,beta.c,sqrt(sig.beta))
  # Calcul de logr.beta
  logr.beta<-logPost(x = c(eta.c,beta.star),donnees=ech)-logPost(x =
        c(eta.c,beta.c),donnees=ech)
  # On accepte la nouvelle valeur avec proba = min(r,1)
  # Pour cela on tire une loi uniforme
  u<-runif(1)
  beta.bay[k]<-beta.star*(log(u)<=logr.beta)+beta.c*(log(u)>logr.beta)
  beta.c<-beta.bay[k] # On remet à jour la valeur courante de beta

}
ech.bay<-cbind(eta.bay,beta.bay)
est.bay<-colMeans(ech.bay)


est.bay.eta<-est.bay[1]
est.bay.beta<-est.bay[2]

res<-data.frame(eta=c(eta.true,eta.mle,est.bay.eta),beta=c(beta.true,beta.mle,est.bay.beta),
                row.names=c("True","MLE","Bayésien"))
res
```

```
##                 eta     beta
## True      1000.0000 1.500000
## MLE        937.3220 2.838384
## Bayésien   999.6923 2.459870
```

```r
hist(ech.bay[,1],freq=F, xlab="eta",main = "Histogramme (loi a posteriori) pour eta")
```




```
hist(ech.bay[,2],freq=F, xlab="beta",main = "Histogramme (loi a posteriori) pour beta")
```




```
# Intervalle de crédibilité à 95%
alpha<-0.05
```

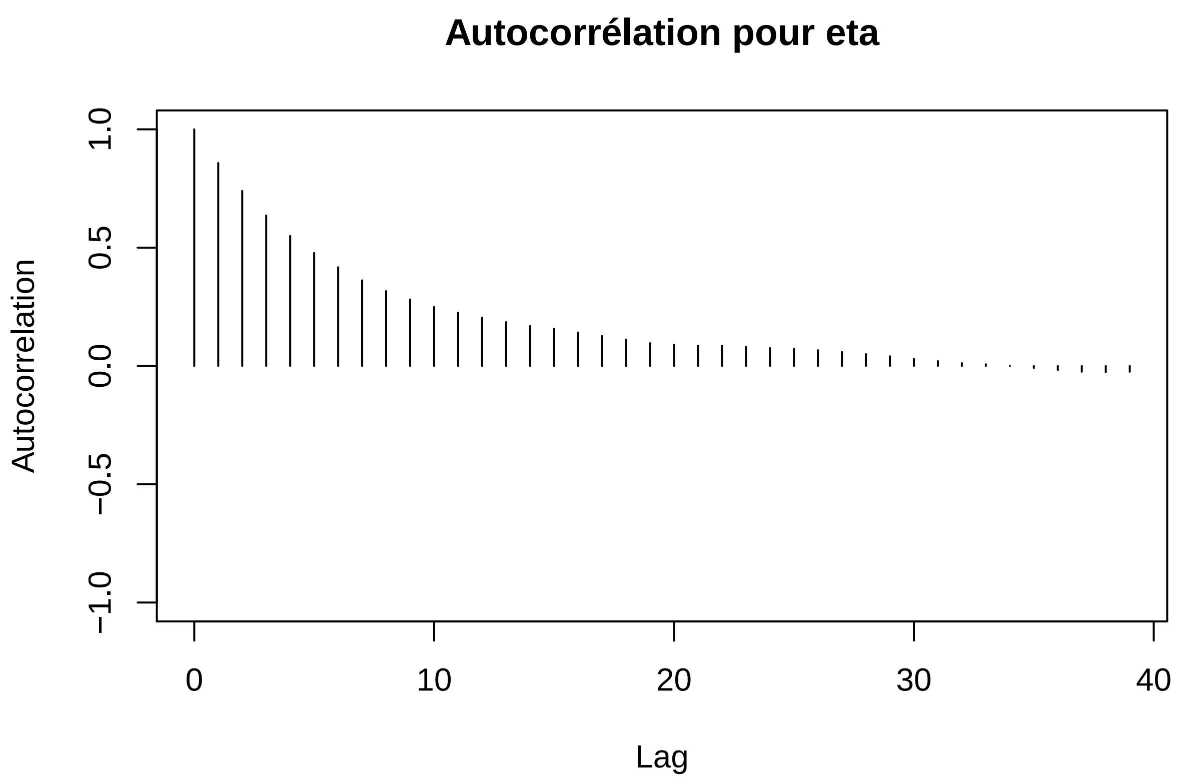
```
ic.eta <- quantile(ech.bay[,1],c(alpha/2,1-alpha/2))
ic.eta
```

```
##      2.5%     97.5%
##  979.3934 1019.4378
```

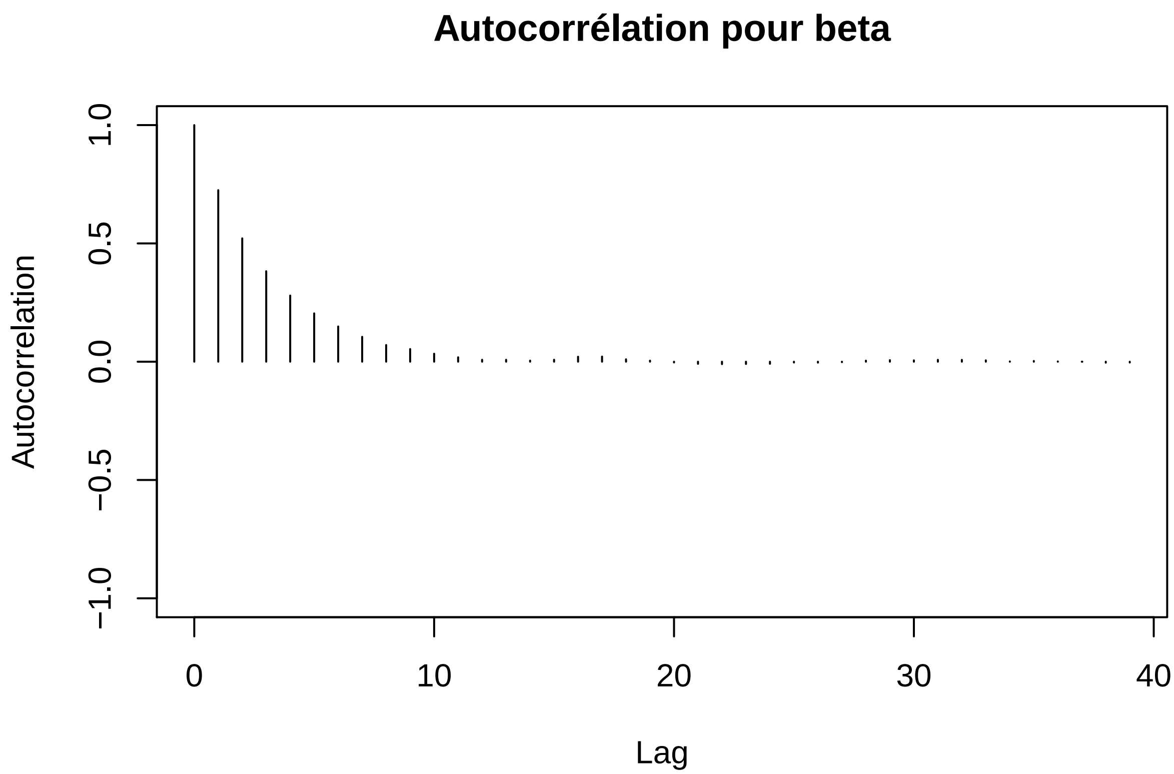
```
ic.beta <- quantile(ech.bay[,2],c(alpha/2,1-alpha/2))
ic.beta
```

```
##     2.5%    97.5%
## 1.593758 2.979364
```

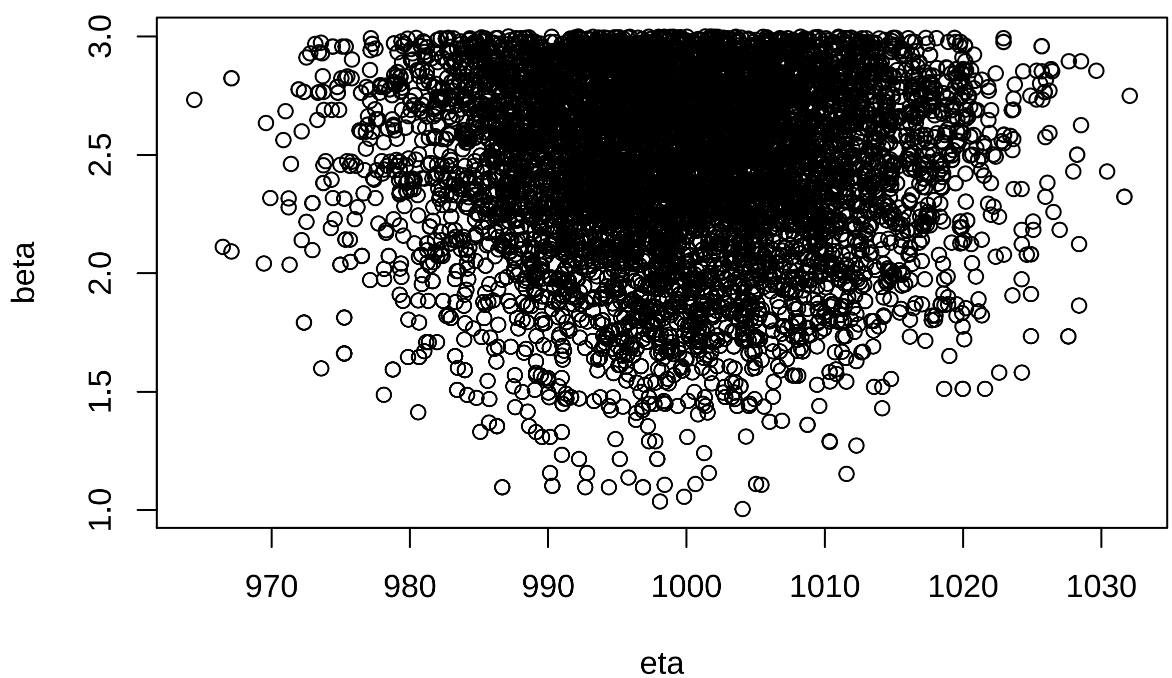
```
# Analyse des sorties de l'algo MCMC
autocorr.plot(ech.bay[-c(1:2000),1],auto.layout = F,main="Autocorrélation pour eta")
```



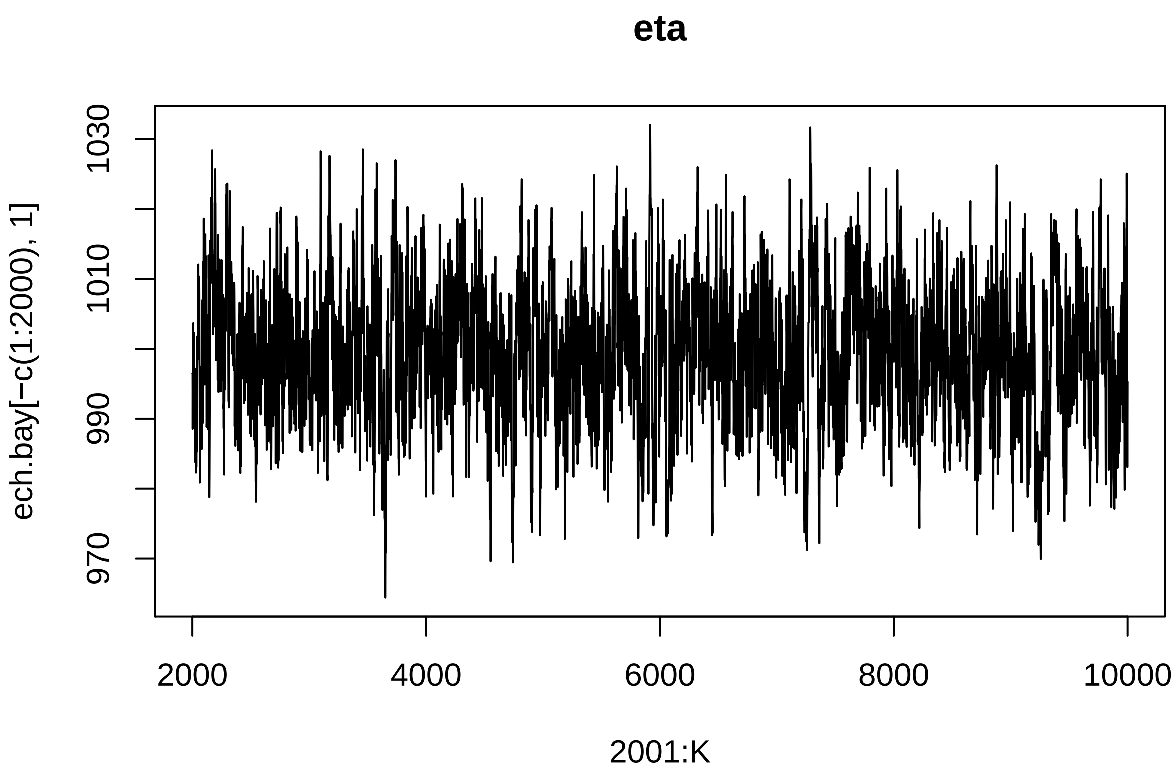

```
autocorr.plot(ech.bay[-c(1:2000),2],auto.layout = F,main="Autocorrélation pour beta")
```




```
plot(ech.bay[-c(1:2000),1],ech.bay[-c(1:2000),2],xlab="eta",ylab="beta")
```



```r
plot(2001:K,ech.bay[-c(1:2000),1],type="l",main="eta")
```




```r
plot(2001:K,ech.bay[-c(1:2000),2],type="l",main="beta")
```