Computational statistics
Week 4

Léa Bresson

Exercices:

- Exercice 6.22
- Exercice 7.9
- Exercice 8.6 - R
- Exercice 9.4
- Exercice 10.10 - R
- Exercice 14.4
- Exercice 14.8

1 Exercice 6.22

A random walk on the non-negative integers $I = (0, 1, 2, ...)$ can be constructed in the following way. For $0 < p < 1$, let $Y_0, Y_1, ..$ be iid random variables with $P(Y_i = 1) = p$ and $P(Y_i = -1) = 1 - p$, and $X_k = \sum_{i=0}^{k} Y_i$. Then, (Xn) is a Markov chain with transition probabilities $P(X_{i+1} = j + 1|X_i = j) = p$, $P(X_{i+1} = j - 1|X_i = j) = 1 - p$, but we make the exception that $P(X_{i+1}) = 1|X_i = 0) = p$ and $\mathbf{P}(X_{i+1} = 0|X_i = 0) = 1 - p$.

a) Show that (X_n) is a Markov chain.

On a $\forall n \in \mathbb{N}^* : X_n = \sum_{i=0}^{k} Y_k = X_{n-1} + Y_n$

$$\begin{aligned} P(X_n = k|X_0, ..., X_{n-1}) &= P(X_{n-1} + Y_n = k|X_0, ..., X_{n-1}) \\ &= P(X_{n-1} + Y_n = k|X_{n-1}) = P(X_n = k|X_{n-1}) \end{aligned}$$

Donc (X_n) est une chaine de Markov.

b) Show that (X_n) is also irreducible.

Soient $n_1, n_2 \in \mathbb{N}^*$:

- si $n_1 > n_2 : p(n_1, n_2) = p^{n2-n1} > 0$
- si $n_1 < n_2 : p(n_1, n_2) = (1 - p)^{n2-n1} > 0$
- si $n_1 = n_2 \neq 0 : p(n_1, n_2) = 2p(1 - p)$

- si $n_1 = n_2 = 0 : p(0,0) = 1 - p$

Deux états de la chaine communiquent toujours: la chaine est irréductible.

c) Show that the invariant distribution of the chain is given by $a_k = \left(\frac{p}{1-p}\right)^k a_0, k = 1, 2...$ where a_k is the probability that the chain is at k and a_0 is arbitrary. For what values of p and a_0 is this a probability distribution?

On note a la distribution invariante: on a alors $aP = a$ avec P la matrice de transition de (X_n).

$$P = \begin{pmatrix} 1-p & p & 0 & ... \\ 1-p & 0 & p & ... \\ 0 & 1-p & 0 & ... \\ ... & ... & ... & ... \end{pmatrix}$$

Donc $P^T a^T = a^T \leftrightarrow \begin{cases} a_0(1-p) + a_1(1-p) = a_0 \\ \forall k > 0, a_{k+1} = a_k p + (1-p)a_{k+2} \end{cases} \leftrightarrow \begin{cases} a_1 = \frac{p}{1-p} a_0 \\ \forall k > 0, a_{k+1} = a_k p + (1-p)a_{k+2} \end{cases}$

(a_k) est donc solution de la relation de recurrence: $(1-p)a_{k+2} - a_{k+1} + pa_k = 0$.

L'équation caractéristique est : $(1-p)X^2 - X + p = 0$. Les solutions de cette équation sont : $x1 = 1$ et $x2 = \frac{p}{1-p}$.

On peut donc écrire : $a_k = \lambda_1(1)^k + \lambda_2(\frac{p}{1-p})^k$ avec $\lambda_1, \lambda_2 \in \mathbb{R}$.

L'évaluation de l'équation en $k = 0$ et $k = 1$ donne : $a_0 = \lambda_1 + \lambda_2$ et $a_1 = \lambda_1 + \lambda_2 \frac{p}{1-p} = \frac{p}{1-p} a_0$. Ainsi on a $\lambda_1 = 0$ et $\lambda_2 = a_0$.

Par conséquent, la mesure invariante de la chaine est donnée par :

$$a_k = \left(\frac{p}{1-p}\right)^k a_0.$$

Par ailleurs, a est une probabilité si elle somme à 1: $\sum_k a_k = 1 \leftrightarrow a_0 \sum(\frac{p}{1-p})^k = 1 \leftrightarrow a_0 \frac{1-p}{1-2p} = 1 \leftrightarrow a_0 = \frac{1-2p}{1-p}$. Et la série converge si $\frac{p}{1-p} < 1 \leftrightarrow p < \frac{1}{2}$.

Il faut donc $a_0 = \frac{1-2p}{1-p}$ et $p < \frac{1}{2}$.

d) If $\sum a_k < \infty$, show that the invariant distribution is also the stationary distribution of the chain; that is, the chain is ergodic.

$$
\begin{aligned}
P(X_{n+1}=j) &= \sum_{i=j}^{\infty} P(X_{n+1}=j|X_n=i)P(X_n=1) \\
&= P(X_{n+1}=j|X_n=j-1)P(X_n=j-1) + P(X_{n+1}=j|X_n=j+1)P(X_n=j+1) \\
&= pa_j + (1-p)a_{j+1} = a_j
\end{aligned}
$$

Par conséquent a est stationnaire, ce qui prouve que la chaine est ergodique.

2 Exercice 7.9

(Tierney 1994) Consider a version of [A.25] based on a "bound" M on f/g that is not a uniform bound; that is, $f(x)/g(x) > M$ for some x.

a) If an Accept–Reject algorithm uses the density g with acceptance probability $f(y)/Mg(y)$, show that the resulting variables are generated from $\tilde{f}(x) \propto min(f(x), Mg(x))$, instead of f.

L'algorithme d'Acceptation Rejet consiste à :

- tirer $Y \sim g$
- tirer $U \sim \mathcal{U}_{[0,1]}$
- Accepter $X = Y$ comme tirage aléatoire de densité de probabilité f si $U < \frac{f(y)}{Mg(y)}$

Or d'après l'énoncé, on sait qu'il existe un ensemble A où: $\forall y \in A, \frac{f(y)}{g(y)} > M$. Ainsi, la variable X simulé par l'algorithme AR a pour fonction de répartition :

$$
\begin{aligned}
\mathbb{P}(X \leq x) &= \mathbb{P}\Big(Y \leq y, U \leq \frac{f(y)}{Mg(y)}\Big) \\
&= \mathbb{P}\Big(Y \leq y, U \leq min(1, \frac{f(y)}{Mg(y)})\Big) \\
&= \int_{-\infty}^{y}\int_{0}^{m} g(x)dxdu \text{ en notant } m = min(1, \frac{f(y)}{Mg(y)}) \\
&= \int_{-\infty}^{y}(\int_{0}^{m} du)g(x)dx \text{ d'après Fubini-Tonelli} \\
&= \int_{-\infty}^{y} min(1, \frac{f(x)}{Mg(x)})g(x)dx \\
&= \int_{-\infty}^{y} min(g(x), \frac{f(x)}{M})dx \\
&= \frac{1}{M}\int_{-\infty}^{y} min(Mg(x), f(x))dx \\
&= \frac{1}{M}\int_{-\infty}^{y} \tilde{f}(x)dx
\end{aligned}
$$

Ainsi la variable X générée par l'algorithme AR suit la loi $min(f(x), Mg(x))$ et non la loi f comme voulu.

b) Show that this error can be corrected, for instance by using the Metropolis–Hastings algorithm:

1. Generate $Y_t \sim \tilde{f}$.
2. Accept with probability

$$P(X^{(t+1)} = y_t | x^{(t)}, y_t) = \begin{cases} \min\left\{1, \frac{f(y_t)g(x^{(t)})}{g(y_t)f(x^{(t)})}\right\} & \texttt{if} \quad \frac{f(y_t)}{g(y_t)} > M \\ \min\left\{1, \frac{Mg(x^{(t)})}{f(x^{(t)})}\right\} & \texttt{otherwise.} \end{cases}$$

to produce a sample from f.

Pour corriger cette erreur, on utilise $\tilde{f}$ comme densité instrumentale dans le cadre de l'algorithme de Metropolis Hastings; la densité cible est f. La condition de convergence est $\forall y, \tilde{f}(y) > 0 \Rightarrow \forall y, g(y) > 0$ and $f(y) > 0$; on la suppose vérifiée.

Le critère de l'algorithme de Metropolis-Hastings utilisé ici est :

$$\begin{aligned} \rho(x,y) &= min\Big(1, \frac{f(y)\tilde{f}(x)}{f(x)\tilde{f}(y)}\Big) \\ &= min\Big(1, \frac{f(y)}{f(x)} \frac{min(Mg(x), f(x))}{min(Mg(y), f(y))}\Big) \end{aligned}$$

On distingue deux cas :

Si $f(y) \leq Mg(y)$:

$$\begin{aligned} \rho(x,y) &= min\Big(1, \frac{f(y)}{f(x)} \frac{min(Mg(x), f(x))}{f(y)}\Big) \\ &= min\Big(1, min(\frac{Mg(x)}{f(x)}, 1)\Big) \\ &= min\Big(1, \frac{Mg(x)}{f(x)}\Big) \end{aligned}$$

Si $f(y) \geq Mg(y)$:

$$\begin{aligned} \rho(x,y) &= min\Big(1, \frac{f(y)}{f(x)} \frac{min(Mg(x), f(x))}{Mg(y)}\Big) \\ &= min\Big(1, min(\frac{g(x)f(y)}{f(x)g(y)}, \frac{f(y)}{Mg(y)})\Big) \\ &= min\Big(1, \frac{g(x)f(y)}{f(x)g(y)}\Big) \text{ car } \frac{f(y)}{Mg(y)} \geq 1 \end{aligned}$$

La variable aléatoire simulée admet bien f pour densité.

3 Exercice 8.6 - R

Reproduce the comparison of Problem 8.5 in the case of (a) the gamma distribution and (b) the Poisson distribution.

a) the gamma distribution

Nous générons 10,000 variables aléatoires suivant une loi Gamma(3,3) grâce au Slice Sampler.

A chaque itération nous simulons:

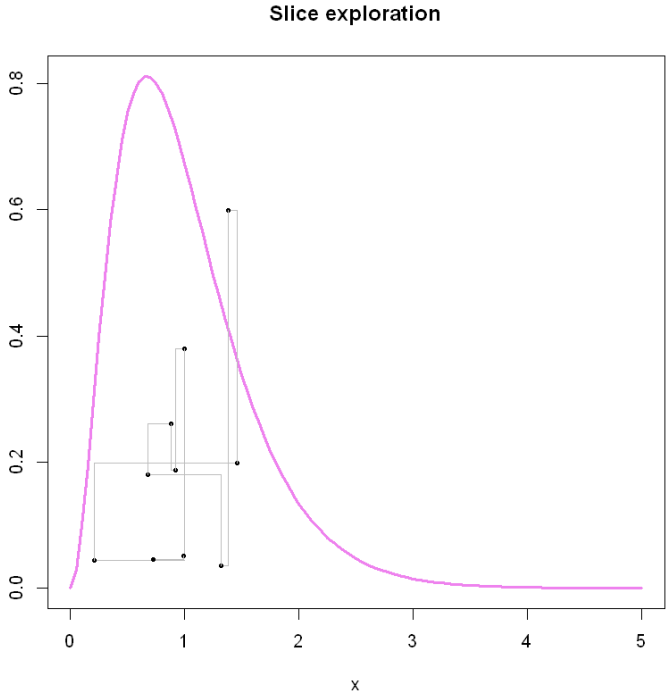
$u_{t+1} \sim \mathbb{U}[0, f(x_t)]$ et $x_{t+1} \sim \mathbb{U}[x \text{ tq } f(x) \geq u_{t+1}]$ avec f la densité d'une loi Gamma.

Pour résoudre $f(x) \geq u_{t+1}$, je fais appel à la fonction *uniroot* de R.

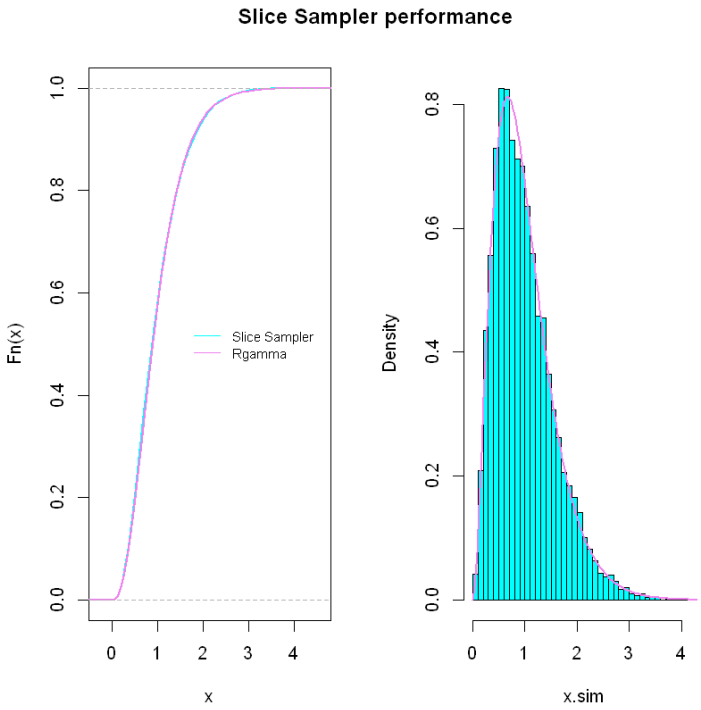
Algorithme Slice Sampler:

```
In [6]: nsim = 10000
        alpha = 3
        beta = 3

        slice_sampler = function(nsim, A){
            u = rep(NA, nsim + 1)
            x = rep(NA, nsim + 1)
            x[1] = 1
            for (i in 1:nsim){
                u[i + 1] = runif(1, 0, dgamma(x[i], shape=alpha, rate=beta))
                x[i + 1] = runif(1, A(u[i + 1], x[i])[1], A(u[i + 1], x[i])[2])
            }
            return(x=x)
        }

        A0 = function(u, x){
            c(uniroot(function(x) dgamma(x, shape = alpha, rate = beta) - u, c(0, x))$root,
                uniroot(function(x) dgamma(x, shape = alpha, rate = beta) - u, c(x,10^10))$root)
        }

        x.sim = slice_sampler(nsim, A0)
        u.sim = slice_sampler(nsim, A0)$u
```

```
In [7]: # Step representation
        curve(dgamma(x, alpha, beta), xlim = c(0, 5), main = "Slice exploration for a gamma",
        ylab = "", col = "violet", lwd = 3)

        for (i in 1:10) {
        points(x.sim[i], u.sim[i], pch = 19, cex = 0.5)
        segments(x.sim[i], u.sim[i], x.sim[i], u.sim[i + 1], col = "gray")
        segments(x.sim[i], u.sim[i + 1], x.sim[i + 1], u.sim[i + 1], col = "gray")
        }
        points(x.sim[i + 1], u.sim[i + 1], pch = 19, cex = 0.5)
```




```
In [8]: par(mfrow = c(1, 2))
        plot(ecdf(x.sim), verticals = TRUE, col.points = "blue", col = "cyan", lwd = 2, main="")
        lines(ecdf(rgamma(nsim, alpha, beta)), verticals = TRUE, col = "violet", lwd = 2)
        legend("right", legend = c("Slice Sampler", "Rgamma"), lty=1, col=c("cyan","violet")
        , bty = "n", cex=.75)
        title(main="Slice Sampler performance", outer=TRUE, line=-2)

        hist(x.sim, freq = F, breaks = 50, main = "", col="cyan")
        curve(dgamma(x, alpha, beta), add = T, col = "violet", lwd=2)
        legend("right", legend = c("Slice Sampler", "Rgamma"), lty=1, col=c("cyan","violet")
        , bty = "n", cex=.75)
```



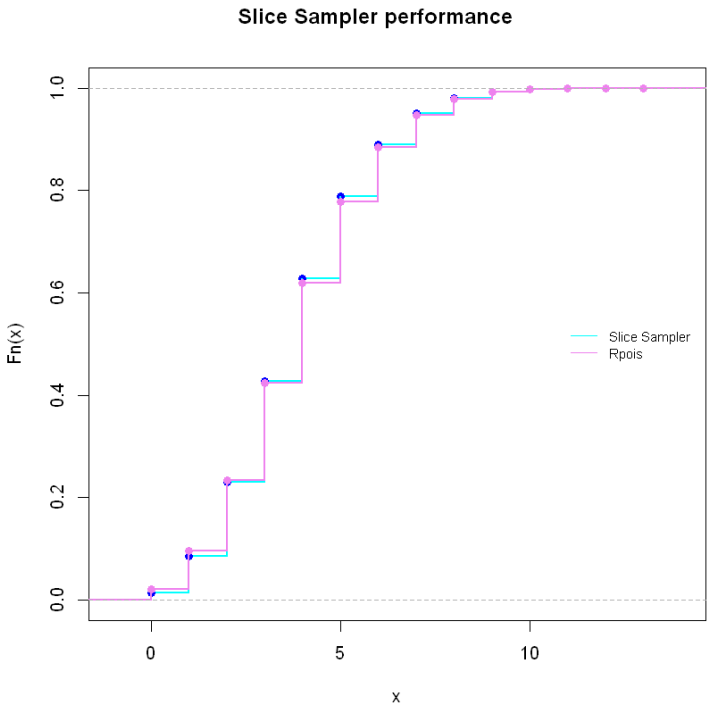
La fonction de répartition donnée par l'algorithme Slice Slamper est quasiment confondue avec la fonction de répartition obtenue grâce à la fonction de R "*rgamma*". Cela atteste de la très bonne performance de l'algorithme Slice Slamper.

b) the Poisson distribution

Nous procédons comme précédemment à la différence que nous sommes désormais en présence d'une loi discrète, la loi de Poisson. La fonction *uniroot* est encore une fois appelée pour déterminer l'intervalle sur lequel les variables suivant une loi de Poisson seront générées

```
In [9]: slice_samplerb = function(nsim, target, L, U) {
            x.sim = rep(NA, nsim)
            u.sim = rep(NA, nsim)
            x.sim[1] = 1
            f = function(x, u.sim, lambda) target(x, lambda) - u.sim
            for (i in 1:(nsim - 1)) {
                u.sim[i + 1] = runif(1, min = 0, max = target(x.sim[i], lam))
                if (f(0, u.sim[i + 1], lam) < 0) {
                    A = c(ceiling(L(u.sim[i + 1], x.sim[i], lam)), floor(U(u.sim[i + 1], x.sim[i], lam)))
                } else {
                    A = c(0, U(u.sim[i + 1], x.sim[i], lam))
                }
                x.sim[i + 1] = sample(A[1]:A[2], 1)
            }
            return(x.sim)
        }


        lam = 4
        target = function(x, lambda) {
            ifelse(x >= 0, exp(-lambda) * lambda^x/factorial(x), 0)
        }
        L = function(u.sim, x.sim, lambda) {
            uniroot(function(x) target(x, lambda = lam) - u.sim, c(0, x.sim))$root
        }
        U = function(u.sim, x.sim, lambda) {
            uniroot(function(x) target(x, lambda = lam) - u.sim, c(x.sim, 100))$root
        }

        x2.sim = slice_samplerb(nsim, target, L, U)
```

```
In [10]: plot(ecdf(x2.sim), verticals = TRUE, col.points = "blue", col = "cyan", lwd = 2, main="")
         lines(ecdf(rpois(nsim, lam)), verticals = TRUE, col = "violet", lwd = 2)
         legend("right", legend = c("Slice Sampler", "Rpois"), lty=1, col=c("cyan","violet"), bty =
          "n", cex=.75)
         title(main="Slice Sampler performance", outer=TRUE, line=-2)
```



4 Exercice 9.4

In the case of the two-stage Gibbs sampler, the relationship between the Gibbs and Metropolis–Hastings algorithms becomes particularly clear. If we have the bivariate Gibbs sampler $X \sim f(x|y)$ and $Y \sim f(y|x)$, consider the X chain alone and show:

a) $K(x, x') = g(x|x') = \int \int f(x'|y) f(y|x) dy$

L'algorithme Gibbs sampler en 2 étapes consiste à générer, pour t=1,2,...:

- $Y_t \sim f_{Y|X}(.|x_{t-1})$
- $X_t \sim f_{X|Y}(.|y_t)$

Les séquences (X_t, Y_t), (X_t) et (Y_t) sont alors des chaines de Markov.

L'algorithme de Metropolis-Hastings sur la chaine (X_t) consiste à tirer $x' \sim K(x, x') = g(x|x')$ avec $K(x, x')$ le noyeau de transition de la chaine (X_t).

On a alors:

$$
\begin{aligned}
K(x, x') &= f(x'|x) \\
&= \frac{f(x, x')}{f(x)} \\
&= \int \frac{f(x', x|y)f(y)}{f(x)} dy \\
&= \int f(x'|y) \frac{f(x|y)f(y)}{f(x)} dy \\
&= \int f(x'|y) f(y|x) dy
\end{aligned}
$$

Pour conclure, on a bien : $K(x, x') = g(x|x') = \int \int f(x'|y) f(y|x) dy$

b) $\rho = min\left(\frac{f(x')/g(x'|x)}{f(x)/g(x|x')}, 1\right)$**, where** $f()$ **is the marginal distribution;**

On génère, grâce à l'algorithme de Metropolis-Hastings, $X' \sim g(x'|x)$.

Ainsi, par définition, on accepte $X = x'$ avec probabilité $\rho = min\left(\frac{f(x')/g(x'|x)}{f(x)/g(x|x')}, 1\right)$ et on conserve $X = x$ avec probabilité 1-ρ.

Ici f, la distribution stationnaire, est la densité marginale de X. En effet,

$$
\begin{aligned}
\int f(x) g(x'|x) dx &= \int f(x) \int f(x'|y) f(y|x) dy dx \text{ d'après la question précèdente} \\
&= \int f(y) f(x'|y) dy = f(x')
\end{aligned}
$$

On voit que l'algorithme Gibbs sampling est en fait un cas particulier de l'algorithme de Metropolis-Hastings où pour chaque étape la densité candidate est la densité conditionelle et la distribution cible est la densité marginale.

c) $f(x')/g(x'|x) = f(x)/g(x|x')$**, so** $\rho = 1$ **and the Metropolis–Hastings proposal is always accepted.**

Nous allons montrer que $f(x')g(x|x') = f(x)g(x'|x)$.

$$
\begin{aligned}
f(x')g(x|x') &= f(x)\int f(x'|y)f(y|x)dy \\
&= f(x)\int \frac{f(x',y)}{f(y)}\frac{f(y,x)}{f(x)}dy \\
&= \int \frac{f(x',y)}{f(y)}f(y,x)dy \\
&= \int f(x',y)f(x|y)dy \\
&= \int f(x)f(y|x')f(x|y)dy \\
&= f(x)\int f(y|x')f(x|y)dy \\
&= f(x)g(x'|x)
\end{aligned}
$$

Ainsi $\rho = 1$ et on accepte toujours x' dans l'algorithme de Metropolis-Hastings.

5 Exercice 10.10

The clinical mastitis data described in Example 10.15 is the number of cases observed in 127 herds of dairy cattle (the herds are adjusted for size). The data are given in Table 10.3.

a) Verify the conditional distributions for λ_i and β_i.

Le modèle de l'exemple 10.15 est le suivant:

$$X_i \sim \mathcal{P}(\lambda_i) \qquad \lambda_i \sim \mathcal{G}(\alpha, \beta_i) \qquad \beta_i \sim \mathcal{G}(a, b)$$

Par définition,

- la densité d'une Loi de Poisson est $\mathbb{P}(X = k) = \frac{\lambda^k}{k!}e^{-\lambda}$,
- la densité d'une loi Gamma est $f(x;\alpha,\beta) = x^{\alpha-1}\frac{\beta^\alpha\, e^{-\beta x}}{\Gamma(\alpha)}$.

Ainsi, la vraisemblance du modèle est :

$$L(x;a,b,\alpha,\beta,\lambda) = \prod_{i=1}^{m}\frac{\lambda_i^{x_i}}{x_i!}e^{-\lambda_i}$$

On a:

$$\pi(\lambda_i|x,\alpha,\beta_i) \propto L(x;a,b,\alpha,\beta,\lambda)\lambda_i^{\alpha-1}\frac{\beta_i^\alpha\, e^{-\beta_i\lambda_i}}{\Gamma(\alpha)} \propto \frac{\lambda_i^{x_i}}{x_i!}e^{-\lambda_i}\lambda_i^{\alpha-1}\frac{\beta_i^\alpha\, e^{-\beta_i\lambda_i}}{\Gamma(\alpha)}$$

et

$$\pi(\beta_i|x,\alpha,\lambda_i) \propto L(x;a,b,\alpha,\beta,\lambda)\lambda_i^{\alpha-1}\frac{\beta_i^\alpha\, e^{-\beta_i\lambda_i}}{\Gamma(\alpha)}\beta_i^{a-1}\frac{b^a\, e^{-b\,\beta_i}}{\Gamma(a)} \propto \frac{\beta_i^\alpha\, e^{-\beta_i\lambda_i}}{\Gamma(\alpha)}\beta_i^{a-1}\frac{b^a\, e^{-b\,\beta_i}}{\Gamma(a)}$$

Par conséquent:

$$\pi(\lambda_i|x,\alpha,\beta_i) \propto \lambda_i^{x_i+\alpha-1} e^{-\lambda_i(1+\beta_i)} = \mathcal{G}(x_i+\alpha, 1+\beta_i)$$

et :

$$\pi(\beta_i|x,\alpha,\lambda_i,a,b) \propto \beta_i^{\alpha+a-1} e^{-(b+\lambda_i)\beta_i} = \mathcal{G}(a+\alpha, \lambda_i+b)$$

Pour conclure on a bien :

$$\lambda_i \sim \mathcal{G}(x_i+\alpha, 1+\beta_i)$$

et :

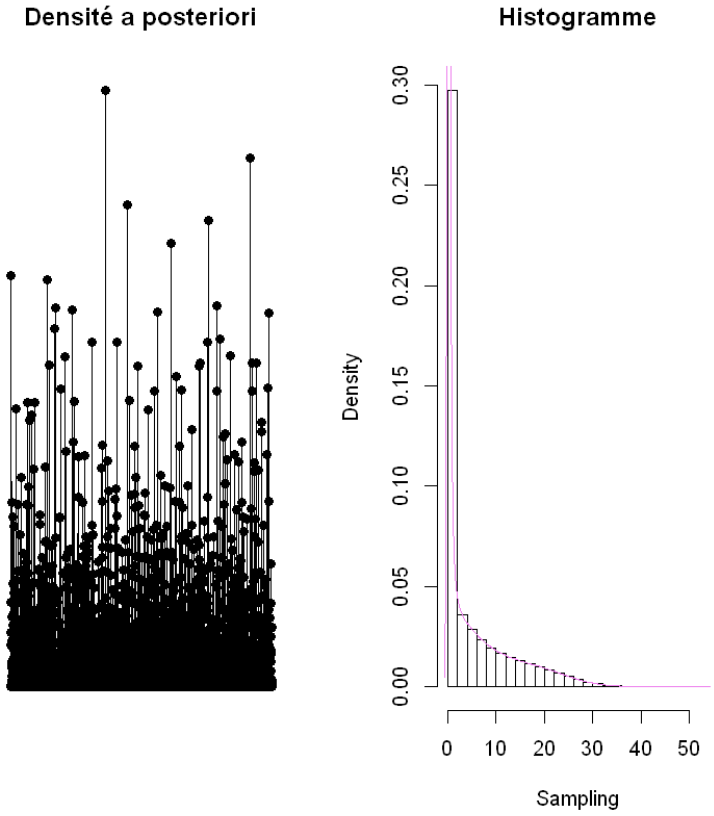
$$\beta_i \sim \mathcal{G}(a+\alpha, \lambda_i+b)$$

b) For these data, implement the Gibbs sampler and plot the posterior density $\pi(\lambda_1|x,\alpha)$. (Use the values α = .1, a = b = 1.)

Implémentation du Gibbs sampler:

```
In [1]: cm_data = c(0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1,
                    1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3,
                    3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 5,
                    5, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 6, 6,
                    6, 7, 7, 7, 7, 7, 7, 8, 8, 8, 8, 8, 9, 9, 9,
                    10, 10, 10, 10, 11, 11, 11, 11, 11, 11, 11, 12, 12, 12, 12,
                    13, 13, 13, 13, 13, 14, 14, 15, 16, 16, 16, 16, 17, 17, 17,
                    18, 18, 18, 19, 19, 19, 19, 20, 20, 21, 21, 22, 22, 22, 22,
                    23, 25, 25, 25, 25, 25, 25)
          data = sort(cm_data)

          # initialisation
          m = length(data)
          alpha = 0.1
          a = 1
          b = 1

          # algorithme
          Gibbs_Sampler = function(data, alpha, a, b, n){
            Res = matrix(nrow = m*2, ncol = n)
            Res[ ,1] = 0
            former_vals = Res[,1]
            mapping = function(former_values){
              lambdai = rgamma(n= m, shape = data + alpha, rate = 1 + former_vals[(m+1):(2*m)])
              betai = rgamma(n=m, shape = a + alpha, rate = b + lambdai)
              former_vals = c(lambdai, betai)
              return(c(lambdai, betai))
            }
            return(apply(Res,2,mapping))
          }

          n = 10000
          Sampling = Gibbs_Sampler(data, alpha, a, b, n)
```

```
In [2]: par(mfrow = c(1, 2))
        plot(1:n, Sampling[1,], pch=16, axes=F, xlab="", ylab="", type="o",col="black", main="Densité a posteri
        hist(Sampling, freq=FALSE, main="Histogramme")
        lines(density(Sampling), col = "violet")
```



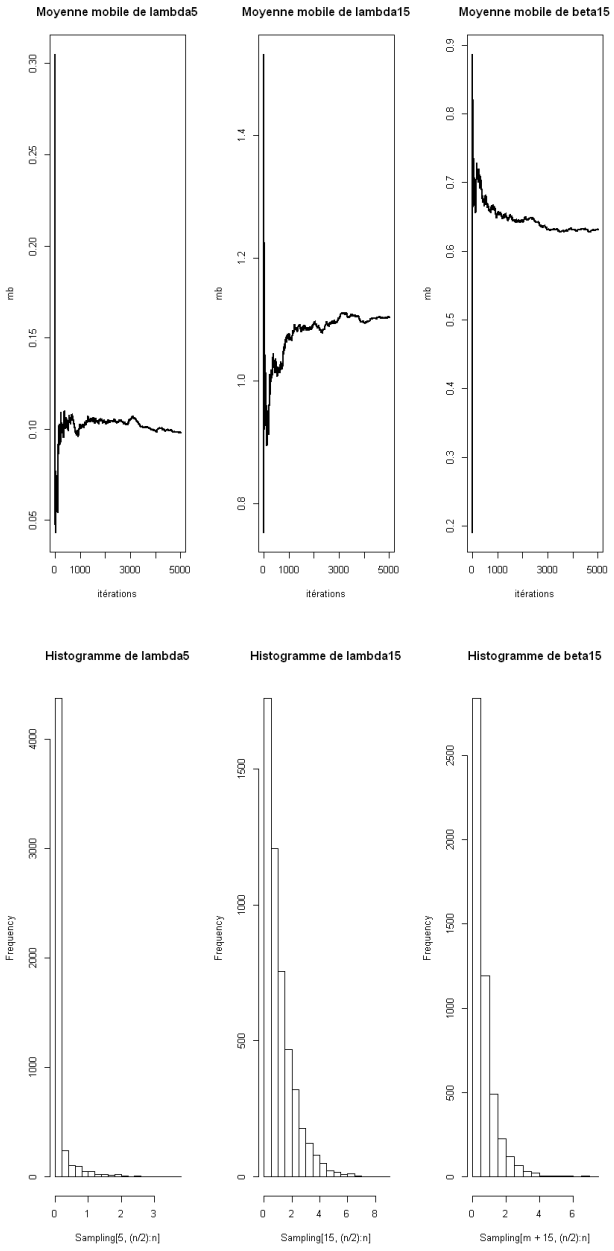
c) Make histograms and monitor the convergence of λ5, λ15, and β15.

Rapellons que

- λ_5 correspond aux troupeaux avec 0 cas de Mastisis,
- λ_{15} et β_{15} correspond aux troupeaux avec 1 cas de Mastisis.

```
In [2]: par(mfrow = c(1, 3))
        mb = cumsum(Sampling[5,(n/2):n])/seq(along=Sampling[5,(n/2):n])
        plot(mb, main="Moyenne mobile de lambda5",xlab="itérations", type="l", lwd=2)
        mb = cumsum(Sampling[15,(n/2):n])/seq(along=Sampling[15,(n/2):n])
        plot(mb, main="Moyenne mobile de lambda15",xlab="itérations", type="l", lwd=2)
        mb = cumsum(Sampling[m+15,(n/2):n])/seq(along=Sampling[m+15,(n/2):n])
        plot(mb, main="Moyenne mobile de beta15",xlab="itérations", type="l", lwd=2)
```

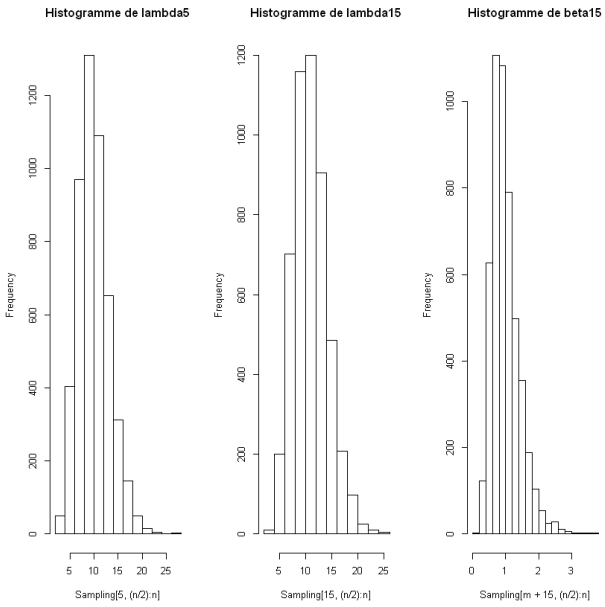
```
In [2]: par(mfrow = c(1, 3))
        hist(Sampling[5,(n/2):n], main = "Histogramme de lambda5")
        hist(Sampling[15,(n/2):n], main = "Histogramme de lambda15")
        hist(Sampling[m+15,(n/2):n], main = "Histogramme de beta15")
```



d) Investigate the sensitivity of your answer to the specification of α, a and b.

- Effet d'une hausse de α:

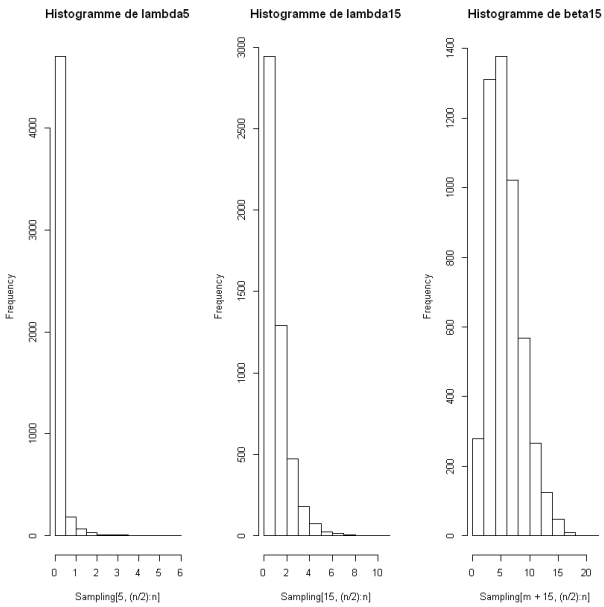
```
In [3]: alpha = 10

        Sampling = Gibbs_Sampler(data,alpha,a,b,n)

        par(mfrow = c(1, 3))
        hist(Sampling[5,(n/2):n], main = "Histogramme de lambda5")
        hist(Sampling[15,(n/2):n], main = "Histogramme de lambda15")
        hist(Sampling[m+15,(n/2):n], main = "Histogramme de beta15")
```



On observe que le mode de la distribution postérieure sera plus grand pour tous les paramètres.

- Effet d'une hausse de a:

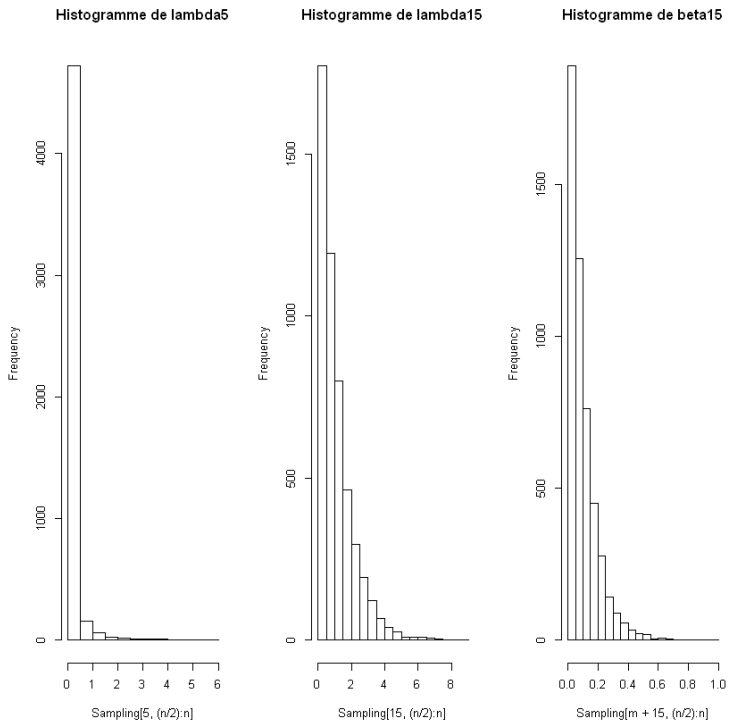
```
In [4]: alpha = 0.1
        a = 10
        b = 1

        Sampling = Gibbs_Sampler(data,alpha,a,b,n)

        par(mfrow = c(1, 3))
        hist(Sampling[5,(n/2):n], main = "Histogramme de lambda5")
        hist(Sampling[15,(n/2):n], main = "Histogramme de lambda15")
        hist(Sampling[m+15,(n/2):n], main = "Histogramme de beta15")
```



La paramètre β est davantage impacté que le paramètre λ: le mode de la distribution est plus grand.

- *Effet d'une hausse de b:*

```
In [5]: alpha = 0.1
        a = 1
        b = 10

        Sampling = Gibbs_Sampler(data,alpha,a,b,n)

        par(mfrow = c(1, 3))
        hist(Sampling[5,(n/2):n], main = "Histogramme de lambda5")
        hist(Sampling[15,(n/2):n], main = "Histogramme de lambda15")
        hist(Sampling[m+15,(n/2):n], main = "Histogramme de beta15")
```



Une variabion de b a un très faible impact sur les distributions.

Pour conclure nous avons vu que les distributions sont sensibles aux choix des paramètres a et α. Il faut donc les choisir soigneusement.

6 Exercice 14.4

Consider a hidden Markov model with hidden Markov chain (X_t) on $(1, ..., \kappa)$, associated with a transition matrix P and observable Y_t such that

$$Y_t|X_t \sim f(y_t|x_t).$$

a) Show the actualization equations are given by

$$p(x_{1:t}|y_{1:t}) = \frac{f(y_t|x_t)p(x_{1:t}|y_{1:(t-1)})}{p(y_t|y_{1:(t-1)})}$$

and

$$p(x_{1:t}|y_{1:(t-1)}) = \mathbb{P}_{x_{t-1}x_t}p(x_{1:(t-1)}|y_{1:(t-1)}),$$

where $f(y_t|x_t)$ is the density $f_{\theta xt}(yt)$ and $\mathbb{P}_{nm}$ denotes the $(n, m) - th$ element of $\mathbb{P}$.

Grâce à la formule de Bayes, on peut écrire:

$$\begin{aligned} p(x_{1:t}|y_{1:t}) &= \frac{p(x_{1:t}, y_t|y_{1:(t-1)})}{p(y_t|y_{1:(t-1)})} \\ &= \frac{p(y_t|x_{1:t}, y_{1:(t-1)})p(x_{1:t}|y_{1:(t-1)})}{p(y_t|y_{1:(t-1)})} \\ &= \frac{f(y_t|x_t)p(x_{1:t}|y_{1:(t-1)})}{p(y_t|y_{1:(t-1)})} \end{aligned}$$

D'autre part, en utilisant la propriété de Markov de la chaine (X_t), on sait que :

$$\begin{aligned} p(x_t|y_{1:(t-1)}, x_{1:(t-1)}) &= p(x_t|x_{t-1}) \\ &= \mathbb{P}_{x_{t-1}x_t} \end{aligned}$$

On obtient bien :

$$\begin{aligned} p(x_{1:t}|y_{1:(t-1)}) &= p(x_t|y_{1:(t-1)}, x_{1:(t-1)})p(x_{1:(t-1)}y_{1:(t-1)}) \\ &= \mathbb{P}_{x_{t-1}x_t}p(x_{1:(t-1)}|y_{1:(t-1)}) \end{aligned}$$

b) Deduce from

$$p(x_{1:t}|y_{1:t}) = \frac{f(y_t|x_t)\mathbb{P}_{x_{t-1}x_t}}{p(y_t|y_{1:(t-1)})}p(x_{1:(t-1)}|y_{1:(t-1)})$$

that computation of the filtering density $p(x_{1:t}|y_{1:t})$ has the same complexity as the computation of the density $p(y_t|y_{1:(t-1)})$.

En combinant les deux équations précédentes, on trouve:

$$p(x_{1:t}|y_{1:t}) = \frac{f(y_t|x_t)\mathbb{P}_{x_{t-1}x_t}}{p(y_t|y_{1:(t-1)})}p(x_{1:(t-1)}|y_{1:(t-1)}).$$

Or :

$$\begin{aligned} p(y_t|y_{1:(t-1)}) &= \int_{\mathbb{X}^t} p(y_t, x_{1:t}|y_{1:(t-1)})\, dx_{1,t} \\ &= \int_{\mathbb{X}^t} p(y_t|x_{1:t})p(x_{1:t}|y_{1:(t-1)})\, dx_{1,t} \\ &= \int_{\mathbb{X}^t} f(y_t|x_t)p(x_t|x_{1:(t-1)}, y_{1:(t-1)})p(x_{1:(t-1)}|y_{1:(t-1)})\, dx_{1,t} \\ &= \int_{\mathbb{X}^t} f(y_t|x_t)\mathbb{P}_{x_{t-1}x_t}p(x_{1:(t-1)}|y_{1:(t-1)})\, dx_{1,t} \text{ en utilisant les résultats de la question précédente} \end{aligned}$$

Ainsi, $p(x_{1:t}|y_{1:t})$ et $p(y_t|y_{1:(t-1)})$ ont une complexité computationelle similaire.

c) Verify the propagation equation

$$p(x_t|y_{1:(t-1)}) = \sum_{x_{1:(t-1)}} p(x_{1:(t-1)}|y_{1:(t-1)})\mathbb{P}_{x_{t-1}xt}.$$

En utilisant, une nouvelle fois la formule de Bayes

$$\begin{aligned} p(x_t|y_{1:(t-1)} &= \frac{p(x_t, y_{1:(t-1)})}{p(y_{1:(t-1)})} \\ &= \sum_{x_{1:(t-1)}} \frac{p(x_t, y_{1:(t-1)})}{p(y_{1:(t-1)})} \\ &= \sum_{x_{1:(t-1)}} p(x_{1:(t-1)}|y_{1:(t-1)})\mathbb{P}_{x_{t-1}xt} \end{aligned}$$

Dans cette question nous avons utilisé une somme - c'est équivalent à une intégrale puisque la chaine de Markov est à espace d'état fini.

7 Exercice 14.8 (supplémentaire)

In the case where P is known and $f(y|x)$ is the density of the normal distribution $N(\mu x, \sigma^2 x)$, explicitly write the gradient equations of part (d). On a : $f(y|x) = \frac{1}{\sqrt{2\pi\sigma^2}} exp^{\frac{(y-\mu)^2}{2\sigma^2}}$

En dérivant on obtient :

- $\frac{\partial f(y|x)}{\partial \mu} = \frac{2y-2\mu}{2\sigma^2}\mu f(y|x)$
- $\frac{\partial f(y|x)}{\partial \sigma} = \frac{-f(y|x)}{\sigma} - \frac{(y-\mu)^2 f(y|x)}{2\sigma^3}$

D'où :

$$\nabla f(y|x) = \begin{pmatrix} \frac{2y-2\mu}{2\sigma^2}\mu f(y|x) \\ \frac{-f(y|x)}{\sigma} - \frac{(y-\mu)^2 f(y|x)}{2\sigma^3} \end{pmatrix}$$