Caso práctico del reto 3. MCMC

Anabel Forte, Gonzalo García-Donato y Joaquín Martínez-Minaya

Índice

1. Introducción. MCMC al rescate

Ya hemos estudiado qué son los métodos MCMC y cómo pueden utilizarse para simular de la distribución a posteriori cuando esta no tiene una forma conocida.

En este caso práctico recuperamos los ejemplos 1 y 2 de los apuntes para obtener una simulación de la distribución posterior. Además, aprenderemos a valorar visualmente si hemos alcanzado la convergencia a la distribución estacionaria simulando varias cadenas de cada parámetro, que inicializaremos en valores diferentes.

2. Metropolis-Hastings. Cuando una distribución es desconocida

Si recordáis, en el ejemplo 1 el modelo asumido para las observaciones $Y_1, Y_2, \ldots, Y_n$ era normal de media μ y de varianza σ^2, que supondremos conocida ($\sigma = 1$). Además, las observaciones son independientes, y por tanto la función de verosimilitud es

$$L(\mu, \mathbf{y}) = \prod_{i=1}^{n} \text{normal}(y_i \mid \mu, \sigma) = \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} \exp\{-\frac{1}{2\sigma^2}(y_i - \mu)^2)\}, \quad \mu \in \mathbb{R}.$$

La distribución a priori para μ que es conjugada es también la distribución normal, pero en este problema se utiliza la distribución de Cauchy.

$$p(\mu) = \frac{1}{\pi}\frac{1}{1+\mu^2}, \quad \mu \in \mathbb{R}.$$

Supongamos que observamos los siguientes datos[1]:

```
##  [1] 1.519432 4.577169 2.043256 2.079995 1.002358 2.727704 2.684651 2.371745
##  [9] 2.893536 3.428015
```

Para obtener la distribución a posteriori aplicamos la fórmula implícita del teorema de Bayes (con el símbolo de proporcional $\propto$):

$$p(\mu \mid \mathbf{y}) \propto \prod_{i=1}^{n} \exp\{-\frac{1}{2\sigma^2}(y_i - \mu)^2)\} \times \frac{1}{1+\mu^2}.$$

Esta distribución a posteriori no es una distribución Cauchy, sino que se trata de una distribución que no pertenece a ninguna familia habitual, y difícilmente podemos aspirar a tener una forma explícita para ella.

Cuando estamos en una situación como esta, en la que debemos simular de una distribución desconocida para un único parámetro (no hay posibilidad de mirar las condicionales completas ni de hacer nada más), se puede usar el método Metropolis-Hastings (MH).

[1] Para ver si lo estamos haciendo bien, hemos simulado los datos de una normal donde la media es 3. Vamos a comprobar si la posterior se centra en dicho valor.

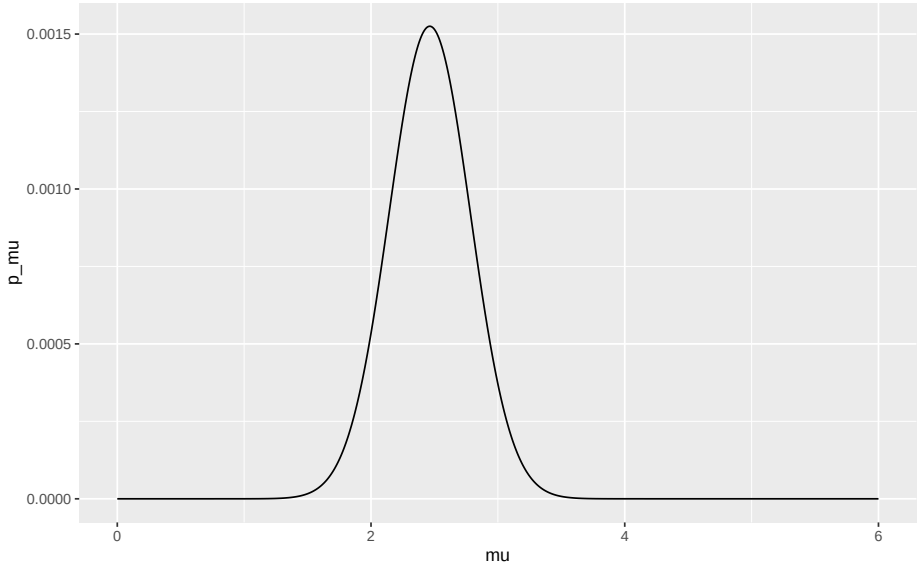
Para poder utilizar MH debemos pensar en proponer una distribución de la que simular, y para ello resulta interesante ver cuál es la forma de la distribución de la que queremos simular. Es importante destacar que esta figura tiene la forma, pero le falta la constante, es decir, no integrará 1.

```
pstar <- function(y, mu) {
   (1 / (1 + mu ^ 2)) * prod(exp(-(y - mu) ^ 2 / 2))
}

mu <- seq(0, 6, length.out = 1000)
p_mu <- unlist(lapply(mu, pstar, y = y))

data <- data.frame(mu = mu, p_mu = p_mu)

ggplot(data, aes(x = mu, y = p_mu)) + geom_line()
```



Si nos fijamos en el gráfico, vemos que la forma es muy similar a la de una normal, por lo que podríamos utilizar esta distribución como propuesta. En concreto, para cada nuevo valor de la cadena $\mu^{(k)}$, la propuesta será simular de una normal centrada en el valor $\mu^{(k-1)}$ y con varianza 1, esto es, $\mu^{(prop)} \sim \text{normal}(\mu^{(k-1)}, 1)$. El valor de α sobre la base de la que decidiremos si quedarnos con $\mu^{(prop)}$ como el k-ésimo valor de la cadena será:

$$\alpha = \min\left(1, \frac{p^\star(\mu^{(prop)})\, q(\mu^{(t-1)}|\mu^{(prop)})}{p^\star(\mu^{(t-1)})\, q(\mu^{(prop)}|\mu^{(t-1)})}\right)$$

En R, la programación del método quedaría así:

```
mh <- function(inits, n.iter, sig) {
   #Inicializamos los valores
   result <- rep(0, n.iter)#vector que guardará la cadena
```

```
   n.acep <- 0 #número de valores aceptados
   result[1] <- inits #valor inicial de la cadena

   for (t in 2:n.iter) {
      mu.prop <- rnorm(1, mean = result[t - 1], sd = 1)

      ratio <-
         pstar(y, mu.prop) * dnorm(result[t - 1], mean = mu.prop, sd = sig) /
         (pstar(y, result[t - 1]) * dnorm(mu.prop, mean = result[t - 1]))

      alpha <- min(1, ratio) #Calculamos alpha

      u <- runif(1, 0, 1) # Simulamos un valor de una uniforme
      if (u <= alpha) {
         #cuando u<=alpha
         result[t] <- mu.prop #nos quedamos con el valor
         n.acep <- n.acep + 1 #aumentamos el número de aceptados
      }
      else
         result[t] <- result[t - 1]
      #si u>alpha se queda con el valor previo de la cadena
   }
   cat("Rate of Acceptance:", n.acep / n.iter, "\n")
   return(result)
}
```

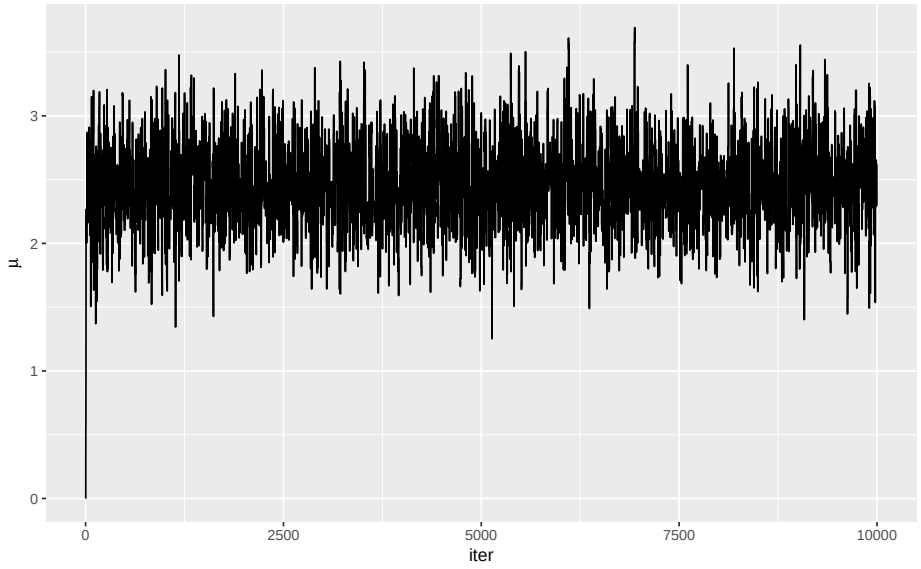
Vamos a probar esta simulación con 10.000 iteraciones y valor inicial $x^{(0)} = 0$.

```
n.iter <- 10000
sims <- mh(inits = 0, n.iter = n.iter, 1)
```

```
## Rate of Acceptance: 0.3674
```

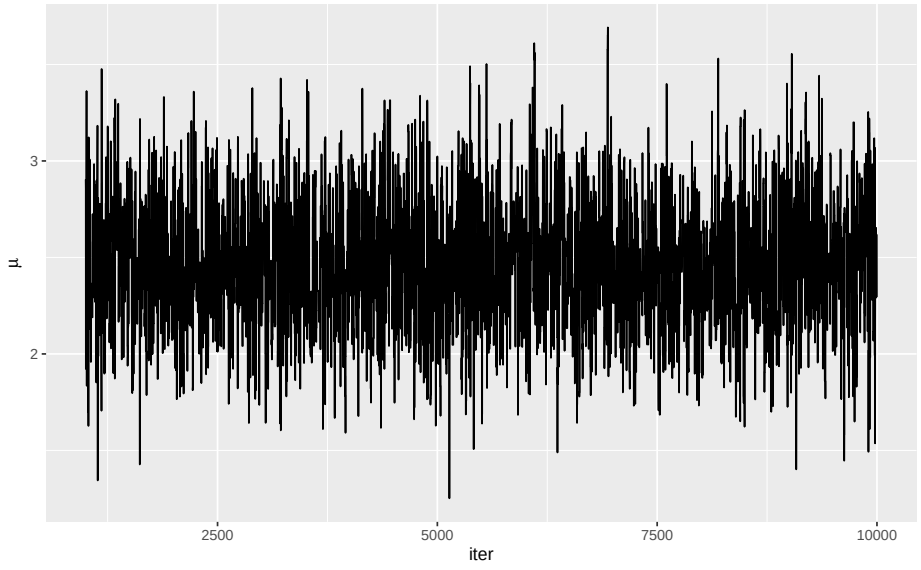
Si la dibujamos, observamos que los valores obtenidos, excepto para las primeras simulaciones, pronto se centran en un valor de 2.5 (recordemos que el valor de la media a partir del que se habían simulado los datos era 3, y que la previa Cauchy se centra en 0) y que varían en torno a este valor.

```
data <- data.frame(sims = sims, x = 1:n.iter)
ggplot(data, aes(y = sims, x = x)) + geom_line()+
   xlab("iter")+ylab(expression(mu))
```



Eliminaremos la parte inicial, alrededor de un 10 % de la cadena, que actuará como burning, y nos queda:

```
data <-
   data.frame(sims = sims[(0.1 * n.iter):n.iter],
              x = (0.1 * n.iter):n.iter)
ggplot(data, aes(y = sims, x = x)) + geom_line()+
   xlab("iter")+ylab(expression(mu))
```



Pero tener una sola cadena no es lo más apropiado, porque no sabemos qué habría pasado si la simulación hubiese comenzado en un punto diferente del espacio muestral. Vamos a simular entonces dos más, empezando en valores aleatorios de una normal (0,1).

```
sims2 <- mh(inits = rnorm(1), n.iter = n.iter, 1)
```

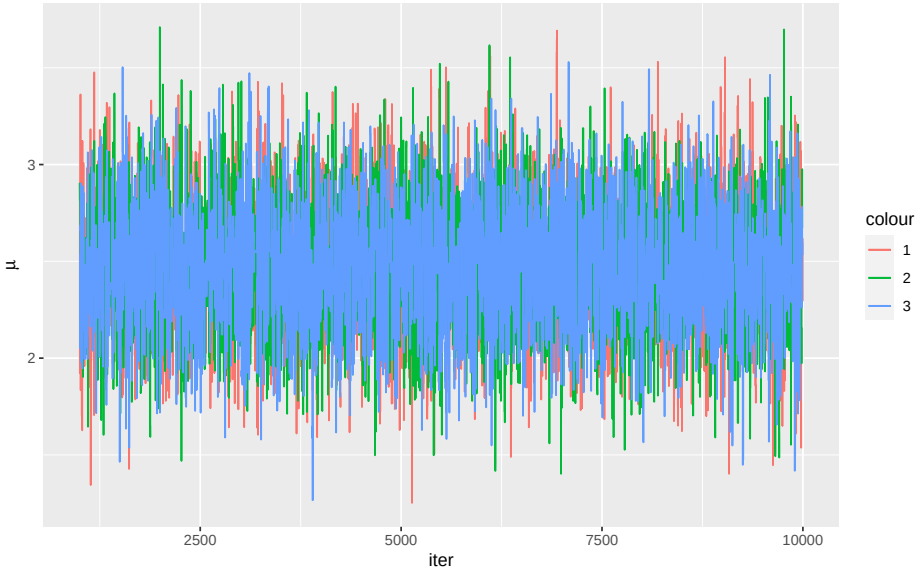
```
## Rate of Acceptance: 0.3598
```

```
sims3 <- mh(inits = rnorm(1), n.iter = n.iter, 1)
```

```
## Rate of Acceptance: 0.3547
```

Antes de ver la convergencia de las tres cadenas es importante comentar que un valor aceptable de esta se situará entre 0.25 y 0.5, lo que observamos que sucede en este caso. Seguidamente, vemos cómo se mezclan las cadenas.

```
data <-
   data.frame(
      sims1 = sims[(0.1 * n.iter):n.iter],
      sims2 = sims2[(0.1 * n.iter):n.iter],
      sims3 = sims3[(0.1 * n.iter):n.iter],
      x = (0.1 * n.iter):n.iter
   )
ggplot(data, aes(y = sims1, x = x)) + geom_line(aes(colour = "1")) +
   geom_line(aes(y =sims2, colour = "2")) +
   geom_line(aes(y = sims3, colour = "3"))+
   xlab("iter")+ylab(expression(mu))
```



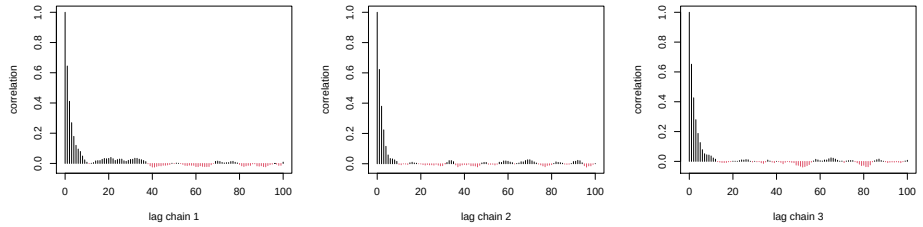
Efectivamente, observamos que hay convergencia y las tres cadenas, inicializadas en puntos diferentes, llevan a una simulación aleatoria de la misma distribución, esto es, a la distribución estacionaria que, en nuestro caso, es la distribución a posteriori para μ.

Otro de los diagnósticos que es importante observar es la autocorrelación. Es decir, el hecho de que

al simular un punto a partir de otro anterior, estos no dependan excesivamente el uno del otro. Para comprobarlo podemos estudiar la correlación del vector de simulaciones de una cadena (eliminando el burning) con el mismo vector, pero con un lapso de k iteraciones. Veámoslo con k desde 0 (donde la correlación es obviamente 1) hasta 100.

```
par(mfrow = c(1, 3))
corr <- 1
for (k in 1:100) {
   aux <-
      cor(data$sims1[1:(length(data$sims1) - k)],
          data$sims1[(k + 1):length(data$sims1)])
   corr <- cbind(corr, aux)
}
plot(0:100,
     corr,
     "h",
     col = (1 + (corr <= 0)),
     xlab = "lag chain 1",
     ylab = "correlation")
corr <- 1
for (k in 1:100) {
   aux <-
      cor(data$sims2[1:(length(data$sims2) - k)],
          data$sims2[(k + 1):length(data$sims2)])
   corr <- cbind(corr, aux)
}
plot(0:100,
     corr,
     "h",
     col = (1 + (corr <= 0)),
     xlab = "lag chain 2",
     ylab = "correlation")
corr <- 1
for (k in 1:100) {
   aux <-
      cor(data$sims3[1:(length(data$sims3) - k)],
          data$sims3[(k + 1):length(data$sims3)])
   corr <- cbind(corr, aux)
```

```
}
plot(0:100,
     corr,
     "h",
     col = (1 + (corr <= 0)),
     xlab = "lag chain 3",
     ylab = "correlation")
```

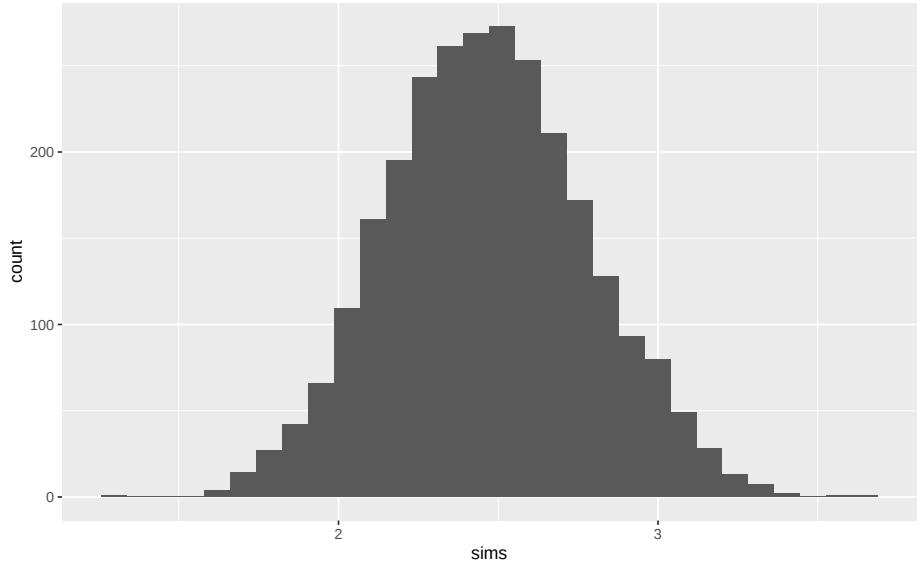


A la vista de estas gráficas, vemos que la autocorrelación decae para valores separados por diez iteraciones, por lo que parece razonable que nos quedemos con una de cada diez iteraciones. A esto se lo llama «adelgazar» la cadena (*thin* en inglés), y lo que estamos estableciendo es que n.thin= 10.

```
index <- seq(1,length(data$sims1), by = 10)
data <- data[index,]
```

Para terminar, podemos unir los valores seleccionados de cada una de las cadenas y tener un vector de simulaciones con el que completar la inferencia a posteriori sobre el parámetro μ mediante gráficas o resúmenes numéricos, como los que mostramos a continuación:

```
sims_final <- data.frame(sims =c(data$sims1,data$sims2,data$sims3))
ggplot(sims_final, aes(x=sims)) + geom_histogram()
```



```r
summary(sims_final$sims)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.266   2.255   2.457   2.467   2.676   3.617
```

3. Gibbs Sampling

El método Gibbs aparece cuando debemos simular de una distribución multivariante (para varios parámetros a la vez). Estas distribuciones son muy difíciles de calcular y no suelen tener una forma conocida. Para ilustrar el proceso recurriremos al segundo de los ejemplos que estudiábamos en el documento sobre métodos MCMC.

Se trata de un modelo sencillo de punto de cambio en el que las primeras m observaciones $Y_1, \ldots, Y_m$ se distribuyen Poisson de media λ (desconocido), mientras que las m siguientes se distribuyen también Poisson, pero de media $\lambda\delta$ y donde $\delta \in (1, 2)$ es desconocido. Todas las observaciones son independientes.

$$Y_i \mid \lambda \sim \text{Pois}(\lambda),\ i = 1, 2, \ldots, m \qquad Y_i \mid \lambda, \delta \sim \text{Pois}(\lambda\delta),\ i = m+1, m+2, \ldots, 2m.$$

Como ya habíamos comentado, esta situación podría corresponderse con un análisis estadístico de calidad en el que se observan defectos diariamente de media λ. A partir del día m se ha probado un sistema de ahorro en la sensibilidad de la producción que puede haber aumentado el número de defectos en cierta proporción, multiplicándolos por δ. Valores de δ de aproximadamente 1 testificarían que no ha habido cambios en la media de defectos, mientras que valores superiores de δ implicarían muchos más defectos que en la configuración inicial de la producción. Además, se sabe que δ puede tomar un valor máximo de 2, indicando que se ha duplicado el número de defectos.

La función de verosimilitud para $\mathbf{y} = (y_1, \ldots, y_m, y_{m+1}, \ldots, y_{2m})$ se puede escribir en forma compacta como

$$L(\lambda, \delta, \mathbf{y}) \propto e^{-m\lambda(1+\delta)} \lambda^{S_1+S_2} \delta^{S_2},$$

donde S_1 y S_2 representan la suma del número de defectos en las primeras m observaciones o en las m siguientes respectivamente. Este escenario se parece al proceso con observaciones Poisson y distribución a priori Gamma, pero no coincide exactamente. De hecho, es difícil pensar en distribuciones a priori que fueran conjugadas.

En cualquier caso, suponemos que las distribuciones a priori son $p(\lambda) = \text{Gamma}(\lambda \mid a, b)$, e independientemente $p(\delta) = \text{unif}(\delta \mid 1, 2)$. Esta última distribución sobre δ refleja que no tenemos información a priori para distinguir ningún valor de δ en su intervalo inicial. Los parámetros de los que depende la información a priori para λ (a y b) se asignan utilizando información de expertos, y

en este caso concreto supondremos que $a = 3$ y $b = 1$, indicando que la media a priori para λ es 3, con una varianza de 3.

Después de observar los datos tenemos que la suma de defectos en las $m = 10$ primeras observaciones fue de $S_1 = 35$, mientras que en el segundo periodo fue de $S_2 = 47$.

La distribución a posteriori es

$$p(\lambda, \delta \mid \mathbf{y}) \propto e^{-10\lambda(1+\delta)}\, \lambda^{35+47}\, \delta^{47}\, \lambda^{3-1}\, e^{-\lambda}, \qquad \lambda > 0,\ \delta \in (1, 2).$$

Esta distribución a posteriori no es del tipo gamma multiplicada por uniforme como la previa, y por tanto la previa de este problema no es conjugada. Evidentemente, que la previa no tenga esta propiedad no es ningún problema en sí mismo, lo que es un gran inconveniente es que la distribución a posteriori no tiene una forma reconocible. Como en el ejemplo anterior, no tiene una forma explícita y su forma implícita no es demasiado útil para obtener resúmenes, o lo que es lo mismo, inferencias sobre los parámetros desconocidos.

Cuando tenemos un problema multivariante de este tipo es lógico pensar en el procedimiento de Gibbs Sampling, para el que nos fijaremos en si las distribuciones condicionales tienen forma conocida. Esto es, si cuando condicionamos un valor de δ concreto, $p(\lambda \mid \delta, \mathbf{y})$ es conocida y si cuando condicionamos un valor de λ concreto, $p(\delta \mid \lambda, \mathbf{y})$ es conocida.

En el caso de λ observamos que

$$p(\lambda \mid \delta, \mathbf{y}) \propto e^{-10\lambda(1+\delta)}\, \lambda^{35+47}\, \lambda^{3-1}\, e^{-\lambda} = e^{-(10(1+\delta)+1)\lambda} \lambda^{85-1}, \quad \lambda > 0.$$

Esto nos está indicando que, conocido el valor de δ, λ se comporta como una gamma con parámetros $a = 92$ y $b = 10(1 + \delta) + 2$. En este escenario nos bastará con simular de esta distribución utilizando la función que R tiene disponible para ello.

En el caso de δ, la distribución condicional sería

$$p(\delta \mid \lambda, \mathbf{y}) \propto e^{-10\lambda\delta} \delta^{47}, \quad \delta \in (1, 2).$$

Si nos fijamos, se trata del núcleo de una distribución gamma de parámetros $a = 48$ y $b = 10\lambda$. Sin embargo, esta distribución no puede considerarse directamente, puesto que δ solo está definido en el intervalo (1, 2). Lo que haremos en este caso es simular de la distribución gamma señalada, pero nos quedaremos solo con los valores que estén en el intervalo correspondiente. A este proceso se lo denomina «truncar» la distribución, y lo llevaremos a cabo utilizando un bucle *while*.

En resumen, el procedimiento que vamos a seguir es el siguiente:

1. Definimos unos valores iniciales $(\lambda^{(0)}, \delta^{(0)})$.
2. Para $t = 1, \ldots, n.iter$:
   - Simular $\lambda^{(t)}$ condicionando $\delta^{(t-1)}$.

- Simular $\delta^{(t)}$ condicionando el $\lambda^{(t)}$ que acabamos de simular.

En R nos quedaría:

```
gibbs <- function(n.iter, delta_ini, lambda_ini){
   result <- data.frame(delta =rep(0, n.iter), lambda =rep(0, n.iter))
   result$delta[1] <- delta_ini
   result$lambda[1]<- lambda_ini #valor inicial de la cadena

   for (t in 2:n.iter) {
   l_aux = rgamma(1,shape=85,rate=10*(result$delta[t-1]+1)+1)
   d_aux =0
   while(d_aux < 1 || d_aux >2){
      d_aux <- rgamma(1, shape=48, rate=10*l_aux)
   }

   result$delta[t] <- d_aux
   result$lambda[t] <- l_aux
   }
   return(result)
}
```

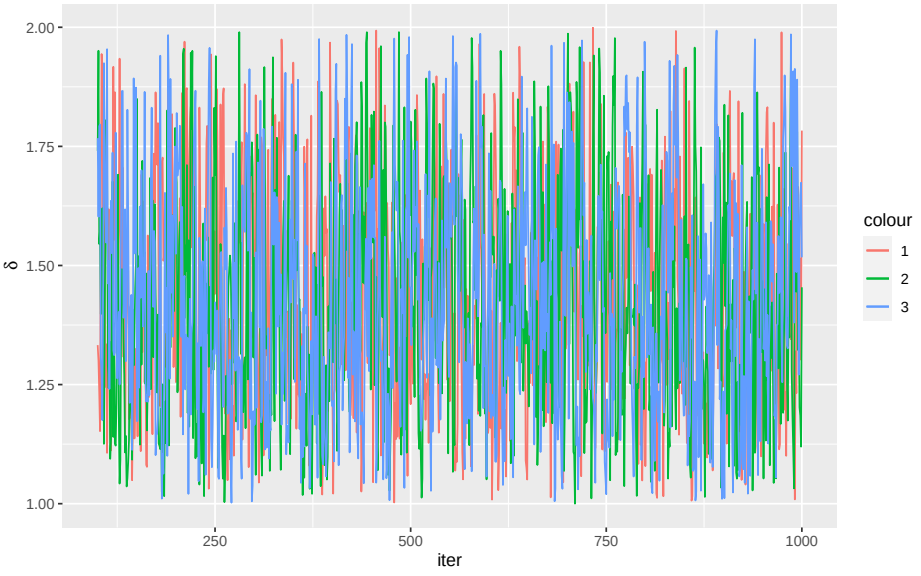
Utilizando está función calculamos tres cadenas, inicializándolas en valores relacionados con la información a priori disponible. Esto es, valores en el intervalo (1, 2) para δ, y en el intervalo (2, 4) para λ.

```
n.iter <- 1000
sims <- gibbs(n.iter, delta_ini = runif(1,1,2), lambda_ini = runif(1,2,4))
sims2<- gibbs(n.iter, delta_ini = runif(1,1,2), lambda_ini = runif(1,2,4))
sims3 <- gibbs(n.iter, delta_ini = runif(1,1,2), lambda_ini = runif(1,2,4))
```

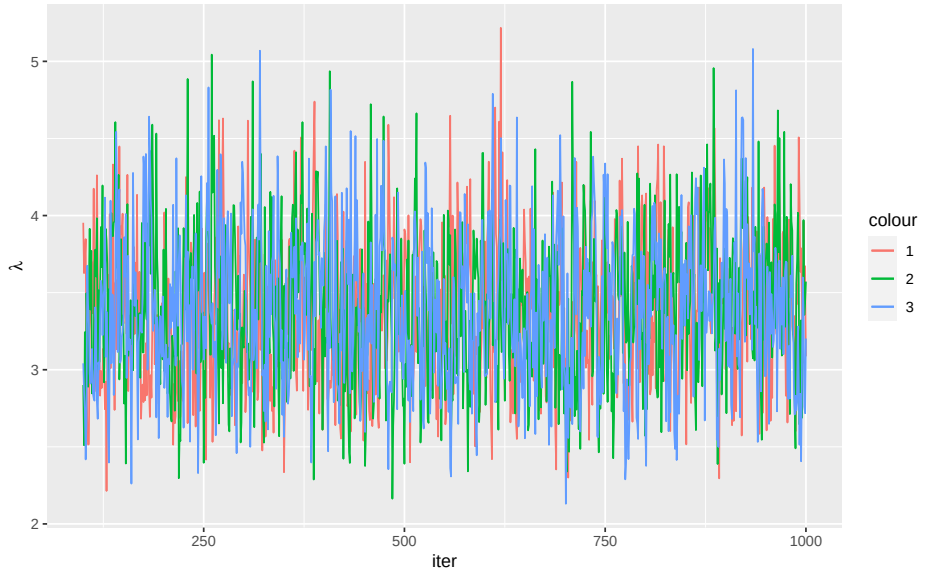
A continuación, eliminamos un 10 % de las simulaciones y nos quedamos con los valores restantes. Ahora, veamos si las cadenas han convergido.

```
data_delta <-
   data.frame(
      sims1 = sims$delta[(0.1 * n.iter):n.iter],
      sims2 = sims2$delta[(0.1 * n.iter):n.iter],
      sims3 = sims3$delta[(0.1 * n.iter):n.iter],
      x = (0.1 * n.iter):n.iter
   )
```

```
ggplot(data_delta, aes(y = sims1, x = x)) + geom_line(aes(colour = "1")) +
   geom_line(aes(y =sims2, colour = "2")) +
   geom_line(aes(y = sims3, colour = "3"))+
   xlab("iter")+ylab(expression(delta))
```




```
data_lambda<-
   data.frame(
      sims1 = sims$lambda[(0.1 * n.iter):n.iter],
      sims2 = sims2$lambda[(0.1 * n.iter):n.iter],
      sims3 = sims3$lambda[(0.1 * n.iter):n.iter],
      x = (0.1 * n.iter):n.iter
   )
ggplot(data_lambda, aes(y = sims1, x = x)) + geom_line(aes(colour = "1")) +
   geom_line(aes(y =sims2, colour = "2")) +
   geom_line(aes(y = sims3, colour = "3"))+
   xlab("iter")+ylab(expression(lambda))
```



Llegados a este punto, podríamos estudiar la autocorrelación de las cadenas para decidir si conviene utilizar cierto adelgazamiento, igual que hicimos con las cadenas resultantes del MH, y una vez tengamos las simulaciones para δ y λ podríamos realizar la inferencia a posteriori que más nos interesase. Por ejemplo, podríamos calcular la probabilidad de que δ sea mayor que 1.5 (por considerarlo un valor no admisible para el aumento del número de defectos). Para ello solo deberíamos contar cuántas simulaciones hay por encima de ese valor, del total de simulaciones utilizadas, es decir:

```
sims_delta <- c(data_delta$sims1,data_delta$sims2,data_delta$sims3)


sum(sims_delta >1.5)/length(sims_delta)
```

```
## [1] 0.382168
```