Simulação Estocástica via Cadeias de Markov II

Ricardo Ehlers
ehlers@icmc.usp.br

Departamento de Matemática Aplicada e Estatística
Universidade de São Paulo

Metropolis-Hastings - Exemplos

Amostrador de Gibbs

Exemplo. Uma variável aleatória discreta X assume valores no conjunto $S = \{, \dots, -2, -1, 0, 1, 2, \dots\}$ com probabilidades,

$$P(X = i) = C \; \left(i - \frac{1}{2}\right)^4 e^{-3|i|} \cos^2(i), \;\; i \in S.$$

Neste caso a constante C não é possivel de ser obtida analiticamente,

$$C = \left[\sum_{i=-\infty}^{\infty} \left(i - \frac{1}{2}\right)^4 e^{-3|i|} \cos^2(i)\right]^{-1}.$$

Propõe-se então um algoritmo de Metropolis-Hastings para simular valores de X.

Usando um passeio aleatório simétrico a matriz de transição Q é tal que,

$$q(i,j) = \begin{cases} 1/2, & j = i+1 \\ 1/2, & j = i-1 \\ 0, & \text{caso contrário.} \end{cases}$$

A probabilidade de aceitação fica,

$$\alpha(i,j) = \min\left\{1, \frac{\left(j - \frac{1}{2}\right)^4 e^{-3|j|} \cos^2(j)}{\left(i - \frac{1}{2}\right)^4 e^{-3|i|} \cos^2(i)}\right\},$$

pois $q(i,j) = q(j,i)$.

Funções em R para simular a cadeia de Markov.

```
> prob <- function(x) ((x-0.5)^4)*exp(-3*abs(x))*cos(x)^2

> metro <- function(i,n) {
+     y = seq(n)
+     for (k in 1:n) {
+         u= runif(1)
+         j= i+sample(c(-1,1),1)
+         r= prob(j)/prob(i)
+         new= if (r>=1) j else {ifelse(u<r,j,i)}
+         i=new
+         y[k]=i
+     }
+     return(y)
+ }
```

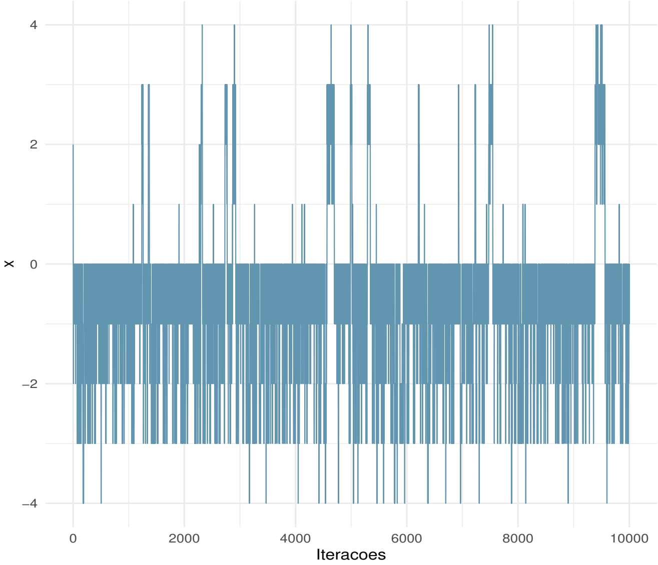
Comandos do R para simular a cadeia de Markov.

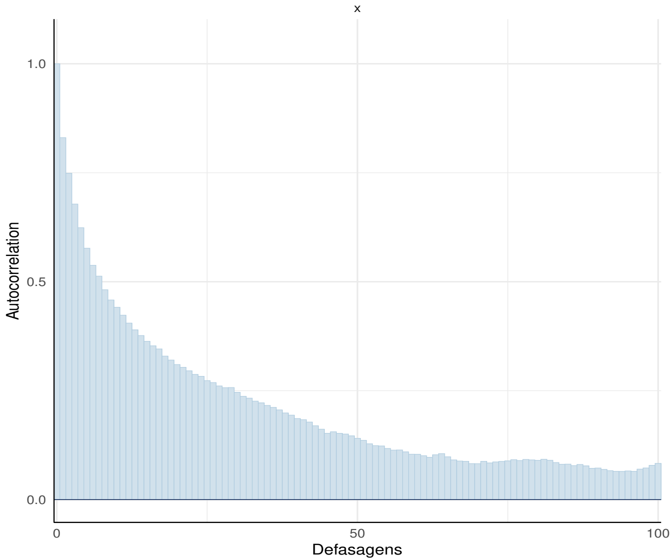
```
> niter= 10000
> nburn= 0
> x = metro(i=1,n=niter)
> xx= as.matrix(x[(nburn+1):niter])
> colnames(xx)="x"
```

Para fazer gráficos com valores simulados e autocorrelações usaremos os pacotes bayesplot e ggplot2.

```
> library(bayesplot)
> library(ggplot2)
```

10000 valores simulados de X via Metropolis-Hastings.



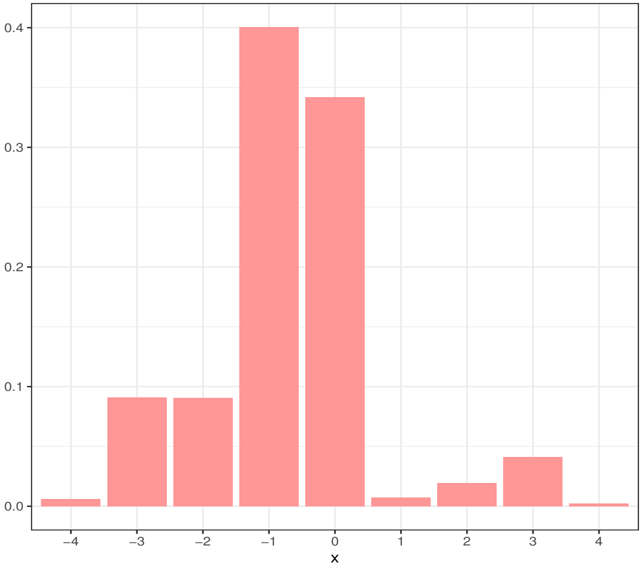

x
1.0
0.5
0.0
Autocorrelation
0
50
100
Defasagens

```
> bayesplot_theme_set(theme_minimal())
> color_scheme_set("blue")
> mcmc_trace(as.matrix(xx)) + labs(x="Iteracoes")

> mcmc_acf_bar(xx,lags=100) + labs(x="Defasagens")
```

Dados estes valores simulados da cadeia de Markov podemos obter aproximações para as probabilidades,

$$P(X = k) \approx \frac{1}{n}\sum_{t=1}^{n} I(X_t = k), \;\; k = \ldots, 0, 1, 2, \ldots$$

Distribuição aproximada de X via Metropolis-Hastings.



Distribuição aproximada de X via Metropolis-Hastings.

```
> tab= table(xx)
> sp= data.frame(tab/niter)
> ggplot(data=sp,aes(x=xx,y=Freq))+
+     geom_bar(stat="identity", fill="#FF9999")+
+     theme_bw() + labs(x="x",y="")
```

Exemplo. Seja uma variável aleatória Y com distribuição de Poisson cuja média X é uma variável aleatória uniforme discreta em $\{1, 2, \ldots, 20\}$. Ou seja,

$$\begin{aligned} Y|X = x &\sim \text{Poisson}(x) \\ P(X = x) &= 1/20, \; x = 1, 2, \ldots, 20. \end{aligned}$$

Deseja-se obter a distribuição de X condicional a um valor observado $Y = y$.

Note que,

$$\begin{aligned} P(X = x|Y = y) &= \frac{P(Y = y|X = x)\; P(X = x)}{P(Y = y)} \\ &= C\; P(Y = y|X = x) \\ &= C\; \frac{e^{-x} x^y}{y!}. \end{aligned}$$

Usando um passeio aleatório com matriz de transição Q tal que,

$$\begin{aligned} q(i,i+1) &= 1/2, \;\; i=1,\ldots,19 \\ q(i,i-1) &= 1/2, \;\; i=2,\ldots,20 \\ q(i,i) &= 1/2, \;\; i=1,20 \\ q(i,j) &= 0, \;\; \text{caso contrário.} \end{aligned}$$

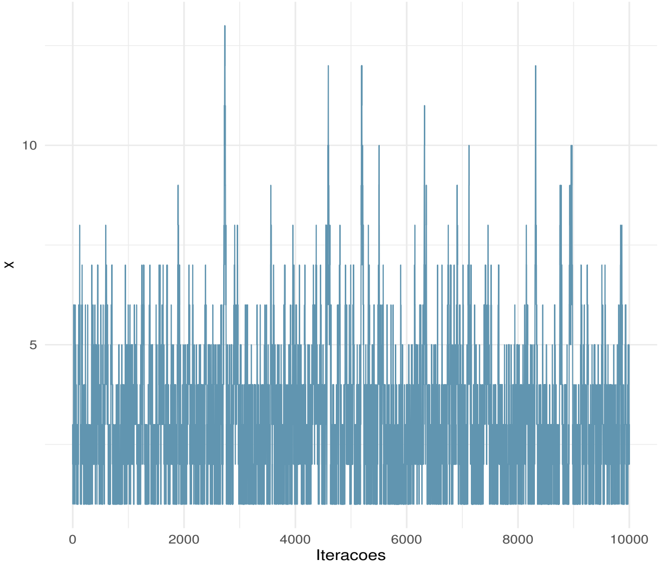
A probabilidade de aceitação fica,

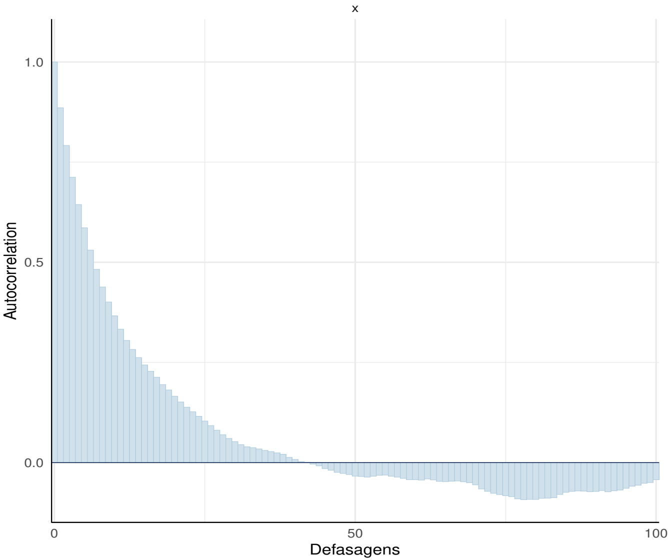
$$\alpha(i,j) = \min\left\{1, \frac{e^{-j} j^{y}}{e^{-i} i^{y}} \; \frac{q(j,i)}{q(i,j)}\right\}$$

Funções em R para executar o algoritmo anterior.

```
> prob <- function(x,y) exp(-x) * x^y

> metro1 <- function(y,i,n) {
+ x = array(0,n)
+ for (k in 1:n) {
+     if (i== 1) j= i + sample(c(0, 1),1)
+     if (i==20) j= i + sample(c(0,-1),1)
+     if (i>1 & i<20) j= i + sample(c(-1,1),1)
+     r= prob(j,y)/prob(i,y)
+     u= runif(1)
+     new= if (r>=1) j else {ifelse(u<r,j,i)}
+     i= new
+     x[k]=i
+ }
+ return(x)
+ }
```

10000 valores simulados de $X|Y = 2$ via Metropolis-Hastings.

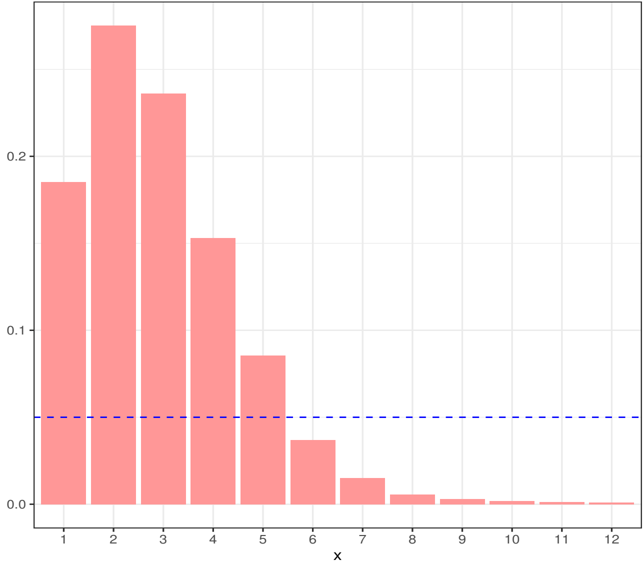



x
Autocorrelation
1.0
0.5
0.0
0
50
100
Defasagens

Comandos do R.

```
> niter= 10000
> nburn= 0
> x = metro1(y=2,i=1,n=niter)
> xx= as.matrix(x[(nburn+1):niter])
> colnames(xx)="x"

> bayesplot_theme_set(theme_minimal())
> color_scheme_set("blue")
> mcmc_trace(as.matrix(xx)) + labs(x="Iteracoes")

> mcmc_acf_bar(xx,lags=100) + labs(x="Defasagens")
```

Distribuição aproximada de $X|Y = 2$ via Metropolis-Hastings.



A figura mostra as probabilidades de X antes de observar Y (linha azul tracejada) e após observar $Y = 2$.

Comandos do R.

```
> tab=table(xx)
> sp <- data.frame(tab/niter)
> ggplot(data=sp,aes(x=xx,y=Freq)) +
+     geom_bar(stat="identity", fill="#FF9999") +
+     theme_bw() + labs(x="x",y="") +
+     geom_hline(yintercept=0.05,colour="blue",lty=2)
```

Exemplo. Sejam $Y_1, \ldots, Y_n$ independentes tais que,

$$\begin{aligned} Y_i | X = x &\sim \text{Poisson}(x) \\ X &\sim \text{Uniforme}\{1, 2, \ldots, 20\} \end{aligned}$$

Como no exemplo anterior,

$$P(X = x) = 1/20, \; x = 1, 2, \ldots, 20$$

e deseja-se obter a distribuição de X (não observável) condicional aos valores observados $Y_1 = y_1, \ldots, Y_n = y_n$.

Definindo-se os vetores $\mathbf{Y} = (Y_1, \ldots, Y_n)$ e $\mathbf{y} = (y_1, \ldots, y_n)$ segue que,

$$
\begin{aligned}
P(X = x|\mathbf{Y} = \mathbf{y}) &= \frac{P(\mathbf{Y} = \mathbf{y}|X = x)\ P(X = x)}{P(\mathbf{Y} = \mathbf{y})} \\
&= C\ P(\mathbf{Y} = \mathbf{y}|X = x) \\
&= C\ \prod_{i=1}^{n} P(Y_i = y_i|X = x) = C\ \prod_{i=1}^{n} \frac{e^{-x} x^{y_i}}{y_i!} \\
&= C\ \frac{e^{-nx} x^{\sum_{i=1}^{n} y_i}}{\prod_{i=1}^{n} y_i!}.
\end{aligned}
$$

O algoritmo anterior pode ser adaptado somente alterando a probabilidade de aceitação.

Usando o mesmo passeio aleatório com matriz de transição Q tal que,

$$
\begin{aligned}
q(i, i+1) &= 1/2, \;\; i = 1, \ldots, 19 \\
q(i, i-1) &= 1/2, \;\; i = 2, \ldots, 20 \\
q(i, i) &= 1/2, \;\; i = 1, 20 \\
q(i, j) &= 0, \;\; \text{caso contrário.}
\end{aligned}
$$

A probabilidade de aceitação fica,

$$
\alpha(i, j) = \min \left\{ 1, \frac{e^{-nj} j^{\sum_{i=1}^{n} y_i}}{e^{-ni} i^{\sum_{i=1}^{n} y_i}} \; \frac{q(j, i)}{q(i, j)} \right\}
$$

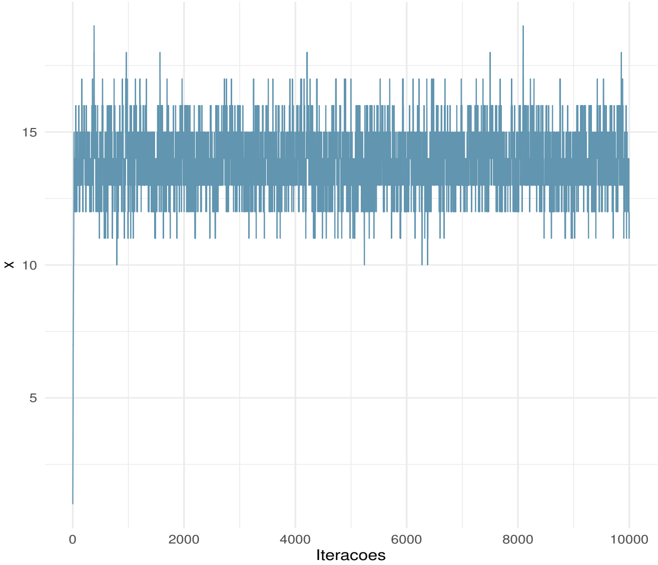
Funções em R para executar o algoritmo anterior.

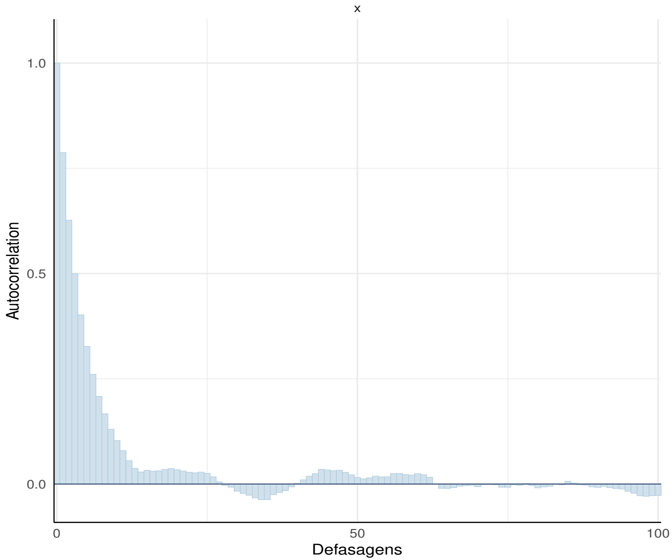
```
> prob <- function(x,y,n) exp(-n*x) * x^(sum(y))

> metro2 <- function(y,i,niter) {
+ x = array(0,niter)
+ n = length(y)
+ for (k in 1:niter) {
+     if (i== 1) j= i + sample(c(0, 1),1)
+     if (i==20) j= i + sample(c(0,-1),1)
+     if (i>1 & i<20) j= i + sample(c(-1,1),1)
+     r= prob(j,y,n)/prob(i,y,n)
+     u= runif(1)
+     new= if (r>=1) j else {ifelse(u<r,j,i)}
+     i= new
+     x[k]=i
+ }
+ return(x)
+ }
```

Exemplo. No exemplo anterior foram gerados 10 valores de $Y \sim$ Poisson(15) e o algoritmo foi usado para aproximar a distribuição de $X|\mathbf{Y} = \mathbf{y}$.

```
> n = 10
> lambda=15
> y = rpois(n,lambda)
> y
 [1] 13 17 22 15 16 17 20  6 12 14
```

10000 valores simulados de $X|\mathbf{Y} = \mathbf{y}$ via Metropolis-Hastings.

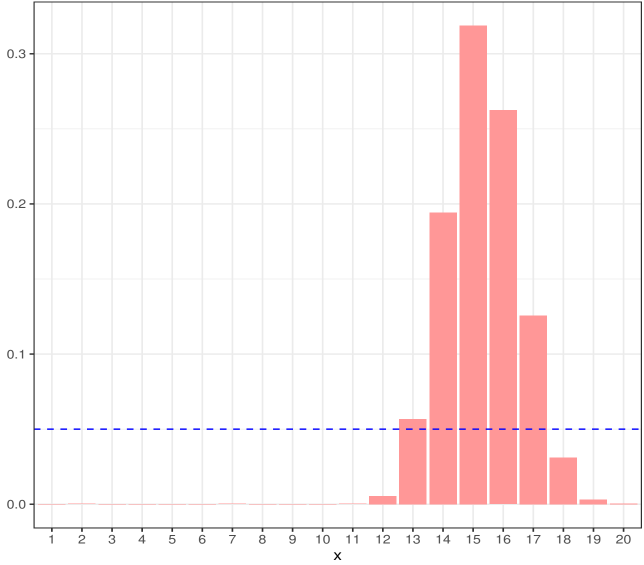



x
1.0
0.5
0.0
Autocorrelation
0
50
100
Defasagens

Comandos do R.

```
> niter= 10000
> nburn= 0
> x = metro2(y,i=1,n=niter)
> xx= as.matrix(x[(nburn+1):niter])
> colnames(xx)="x"

> bayesplot_theme_set(theme_minimal())
> color_scheme_set("blue")
> mcmc_trace(as.matrix(xx)) + labs(x="Iteracoes")

> mcmc_acf_bar(xx,lags=100) + labs(x="Defasagens")
```

Distribuição aproximada de $X|\mathbf{Y} = \mathbf{y}$ via Metropolis-Hastings.



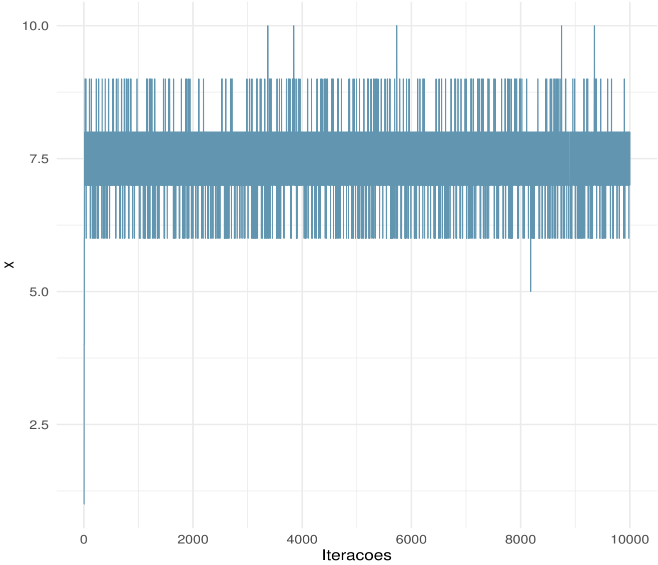
Exemplo. Número anual de casamentos por 1000 habitantes na Itália de 1936 a 1951. Qual a distribuição do número médio de casamentos após observar estes dados?

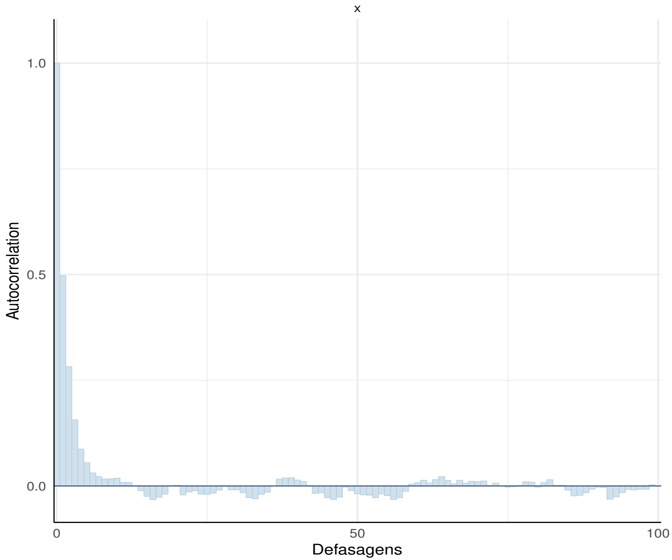
```
> y = c(7,8,9,7,7,6,6,5,5,7,9,10,8,8,8,7)
```

Assume-se que $Y_1, \ldots, Y_n$ são independentes tais que,

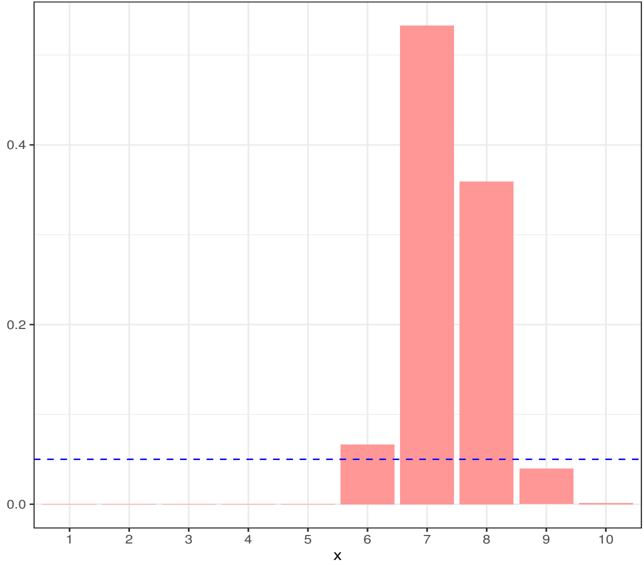
$$\begin{aligned} Y_i|X = x &\sim \text{Poisson}(x), \;\; i = 1, \ldots, 16 \\ X &\sim \text{Uniforme}\{1, 2, \ldots, 20\} \end{aligned}$$

10000 valores simulados de $X|\mathbf{Y} = \mathbf{y}$ via Metropolis-Hastings.




x
Autocorrelation
1.0
0.5
0.0
0
50
100
Defasagens

Distribuição aproximada de $X|\mathbf{Y} = \mathbf{y}$ via Metropolis-Hastings.



Exemplo. No exemplo anterior pode ser mais razoável assumir que $X \sim \text{Poisson}(\lambda)$ com $\lambda > 0$ conhecido e truncada em $\{1, 2, \ldots\}$. Neste caso,

$$
\begin{aligned}
P(X = x) &= \frac{e^{-\lambda}\lambda^x}{x!} \frac{1}{P(X \geq 1)} \\
&= \frac{e^{-\lambda}\lambda^x}{x!} \frac{1}{1 - P(X = 0)} \\
&= \frac{e^{-\lambda}\lambda^x}{x!} \frac{1}{1 - e^{-\lambda}}, \; x = 1, 2, \ldots
\end{aligned}
$$

X não pode assumir o valor 0, por isso a distribuição deve ser truncada.

Portanto,

$$
\begin{aligned}
P(X=x|\mathbf{Y}=\mathbf{y}) &= \frac{P(\mathbf{Y}=\mathbf{y}|X=x)\ P(X=x)}{P(\mathbf{Y}=\mathbf{y})} \\
&= C\ P(\mathbf{Y}=\mathbf{y}|X=x)\ P(X=x) \\
&= C\ \prod_{i=1}^{n} P(Y_i=y_i|X=x)\ P(X=x) \\
&= C\ \frac{e^{-nx}x^t}{\prod_{i=1}^{n} y_i!}\frac{e^{-\lambda}\lambda^x}{x!}\frac{1}{1-e^{-\lambda}},\ x=1,2,\ldots
\end{aligned}
$$

sendo $t=\sum_{i=1}^{n} y_i$ ou, equivalentemente

$$
P(X=x|\mathbf{Y}=\mathbf{y}) = C^*\ e^{-nx}x^t\ \frac{\lambda^x}{x!},\ x=1,2,\ldots
$$

Amostrador de Gibbs

Seja $\mathbf{X} = (X_1, \ldots, X_d)$ um vetor aleatório discreto com função de probabilidade $p(\mathbf{x})$ especificada a menos de uma constante,

$$p(x_1, \ldots, x_d) = C \; h(x_1, \ldots, x_d),$$

sendo $h(\cdot)$ conhecida e C uma constante desconhecida.

Deseja-se simular valores do vetor $\mathbf{X}$.

Suponha que seja possivel simular o i-ésimo elemento de $\mathbf{X}$ dados os outros elementos com probabilidade,

$$P(X_i = x | X_1 = x_1, \ldots, X_{i-1} = x_{i-1}, X_{i+1} = x_{i+1}, \ldots, X_d = x_d),$$

$i = 1, \ldots, d$.

Será que isto é equivalente a simular um valor do vetor $\mathbf{X}$?

Podemos usar o algoritmo de Metropolis-Hastings de um modo bem particular para simular uma cadeia de Markov com distribuição estacionária $p(\mathbf{x})$.

1. Dado o estado atual $\mathbf{x}$ um elemento é selecionado ao acaso,
2. Se o i-ésimo elemento foi selecionado gere um valor da distribuição de $X_i|X_j = x_j, j \neq i$.
3. Se foi gerado $X_i = x$ o estado proposto da cadeia é,

$$\mathbf{y} = (x_1, \dots, x_{i-1}, x, x_{i+1}, \dots, x_n).$$

Temos então que,

$$q(\mathbf{x},\mathbf{y}) = \frac{1}{n}\ P(X_i = x | X_j = x_j, j \neq i) = \frac{1}{n}\ \frac{p(\mathbf{y})}{P(X_j = x_j, j \neq i)}.$$

O estado proposto **y** é aceito com probabilidade,

$$\alpha(\mathbf{x},\mathbf{y}) = \min\left\{1, \frac{p(\mathbf{y})}{p(\mathbf{x})}\frac{q(\mathbf{y},\mathbf{x})}{q(\mathbf{x},\mathbf{y})}\right\} = \min\left\{1, \frac{p(\mathbf{y})\ p(\mathbf{x})}{p(\mathbf{x})\ p(\mathbf{y})}\right\} = 1$$

- As propostas são sempre aceitas, i.e. a cadeia sempre muda de estado.

- As distribuições de $X_i|X_j = x_j, j \neq i$, $i = 1, 2, \ldots$ são chamadas *distribuições condicionais completas*.

- Para implementar o amostrador de Gibbs é necessário gerar valores das distribuições condicionais completas,

$$P(X_i = x_i|X_j = x_j, j \neq i) = \frac{p(\mathbf{x})}{P(X_j = x_j, j \neq i)}.$$

Exemplo. Seja um vetor aleatório discreto $\mathbf{X} = (X_1, X_2, X_3)$ com função de probabilidade $p(\mathbf{x})$ que admite a seguinte simplificação,

$$
\begin{aligned}
p(\mathbf{x}) &= p(x_1|x_2, x_3)p(x_2|x_3)p(x_3) \\
&= p(x_1|x_2)p(x_2)p(x_3).
\end{aligned}
$$

sendo $p(x_1|x_2)$, $p(x_2)$ e $p(x_3)$ conhecidas.

As probabilidades condicionais completas ficam,

$$
\begin{aligned}
p(x_1|x_2, x_3) &\propto p(x_1|x_2) \\
p(x_2|x_1, x_3) &\propto p(x_1|x_2)p(x_2) \\
p(x_3|x_1, x_2) &\propto p(x_3).
\end{aligned}
$$

Na prática os métodos MCMC requerem,

- Escolha do valor inicial da cadeia.
- Monitorar a convergência. Como decidir se a cadeia atingiu o equilibrio?
- A cadeia resultante precisa ser homogênea, irredutivel e aperiódica.

Em particular, no algoritmo de Metropolis-Hastings também é preciso especificar a matriz de transição Q.

Resumindo

- Dada uma variável aleatória cuja função massa de probabilidades é parcialmente conhecida vimos mais exemplos de como simular cadeias de Markov via Metropolis-Hastings.
- Foi introduzido a amostrador de Gibbs para simular vetores aleatórios via distribuições condicionais completas.
- Na próxima aula veremos como simular cadeias de Markov em espaços de estados continuos.