

# Entendendo a Inferência Bayesiana

Métodos Computacionalmente Intensivos

Ana Beatriz Tozzo Martins      Edson Antonio Alves da Silva

Orientador: Prof. PhD. Paulo Justiniano Ribeiro Jr.

18 de outubro de 2007

Este trabalho refere-se ao estudo da apostila do curso de Introdução à Inferência Bayesiana <sup>1</sup> reproduzindo (muitas vezes textualmente) aspectos teóricos e resolvendo exercícios e exemplos presentes no material. Foi utilizado também a apostila de Métodos Computacionalmente Intensivos em Estatística <sup>2</sup>.

## 1 Monte Carlo Simples

A idéia do método é escrever a integral que se deseja calcular, por exemplo,

$$I = \int_a^b g(\theta) d\theta$$

como o valor esperado

$$I = (b - a) \int_a^b g(\theta) \frac{1}{b - a} d\theta = (b - a) E[g(\theta)]$$

onde  $\theta$  é uma variável aleatória com distribuição  $U(a; b)$ . A estimativa de Monte Carlo

$$\hat{I} = (b - a) \frac{1}{n} \sum_{i=1}^n g(\theta_i)$$

pode ser obtida usando o seguinte algoritmo

1. Gerar  $\theta_1, \dots, \theta_n$  da distribuição  $U(a; b)$ ;
2. Calcular  $g(\theta_1), \dots, g(\theta_n)$ ;
3. Calcular a média amostral  $\bar{g} = \sum_{i=1}^n \frac{g(\theta_i)}{n}$ ;
4. Calcular  $\hat{I} = (b - a) \times \bar{g}$ .

**Exemplo 1:** Seja  $X$  uma variável aleatória com distribuição exponencial de parâmetro  $\lambda = 1$ , ou seja,  $f(x) = e^{-x}$ ,  $x > 0$ . Calcular  $P(1 \leq x \leq 3) = P(x \leq 3) - P(x \leq 1)$ .

**Solução:**

<sup>1</sup>Ehlers, R.S. (2005) *Introdução à Inferência Bayesiana*. Departamento de Estatística, UFPR. Disponível <http://leg.est.ufpr.br/~ehlers/notas/bayes2006>. Acesso em: agosto de 2007.

<sup>2</sup>Ehlers, R.S. (2003) *Métodos Computacionalmente Intensivos em Estatística*, UFPR. Disponível <http://leg.est.ufpr.br/~ehlers/notas/mci.pdf>. Acesso em: agosto de 2007.

- Usando a função `pexp` do R vamos calcular:

$$P(1 \leq x \leq 3) = \int_1^3 e^{-x} \quad (1)$$

```
> int.exp = pexp(3, 1) - pexp(1, 1)
> int.exp

[1] 0.3180924
```

- Usando simulação de Monte Carlo vamos calcular a estimativa de Monte Carlo dada por:

$$(3 - 1) \frac{1}{10} \sum_{i=1}^{10} e^{-x} \quad x : 1, \dots, 3 \quad (2)$$

Os passos necessários são:

- Gerar  $n = 10$  valores  $(\lambda_i)$  de uma distribuição uniforme no intervalo  $(1; 3)$ ;
- Calcular  $g(\lambda_i)$ ;
- Calcular a média  $\bar{g}$  dos  $g(\lambda_i)$ ;
- Calcular  $(3 - 1) \times \bar{g}$ .

```
> n = 10
> x = runif(n, 1, 3)
> g = exp(-x)
> (int.exp = (3 - 1) * mean(g))

[1] 0.3156973
```

Como se pode observar, a precisão na estimativa foi de 2 casas decimais.

- Escrevendo uma função geral para o R, cujos argumentos serão o número de simulações e limites de integração (intervalo que se pretende calcular a integral).

```
> int.exp = function(n, a, b) {
+   x = runif(n, a, b)
+   g = exp(-x)
+   int.exp = (b - a) * mean(g)
+   return(int.exp)
+ }
```

Para o cálculo de  $P(1 \leq x \leq 3)$  com  $n = 20$  simulações fazemos:

```
> int.exp(20, 1, 3)

[1] 0.3477491
```

Observamos que a precisão foi de uma casa decimal em relação aos valores obtidos anteriormente.

Uma vantagem em escrever uma função é que podemos repetir facilmente os cálculos. Por exemplo, para obter 20 resultados, cada um com 10 simulações no intervalo  $(1; 3)$  fazemos:

```
> m=NULL # garante que a variável m esteja totalmente vazia
> for (i in 1:20){
+   m = c(m,int.exp(10,1,3))
+ }
> m
```

```
[1] 0.2698334 0.3327875 0.3037126 0.2612778 0.2880306 0.2747677 0.3650898
[8] 0.2802789 0.3272915 0.3412805 0.3277631 0.3048185 0.3085896 0.3780798
[15] 0.2055650 0.3920031 0.3175781 0.3608137 0.3136632 0.3908758
```

```
> summary(m)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.2056  0.2861  0.3156  0.3172  0.3462  0.3920
```

Logo, aumentando-se o número de simulações observou-se que a precisão da estimativa 0.3172 foi de 2 casas decimais como nos itens (a) e (b).

Podemos, também, calcular a esperança de uma função  $g(x)$  da v.a.  $X$  cuja f.d.p. é  $p(x)$ . Para isto basta simular valores de  $p(x)$  e calcular a  $E(X)$ .

```
> n = 1000
> x = rexp(n, 1)
> mean(x)
```

```
[1] 1.033371
```

Iremos novamente calcular  $P(1 \leq x \leq 3)$

```
> sum(x > 1 & x < 3)/n
```

```
[1] 0.317
```

Novamente observamos que a precisão foi de 2 casas decimais.

O que foi feito aqui? Simplesmente calculamos a proporção (frequência relativa) dos valores simulados que “caíram” no intervalo (1; 3), ou seja, a probabilidade procurada.

- Variância do Estimador de Monte Carlo.

É importante ter uma idéia do erro cometido nesta aproximação, ou seja, o Erro de Monte Carlo. Esse erro é obtido medindo-se a variância empírica do estimador de Monte Carlo dada por:

$$v = \frac{1}{n^2} \sum_{i=1}^n (g(x_i) - \bar{g})^2 = \frac{1}{n} \left( \sum_{i=1}^n \frac{(g(x_i) - \bar{g})^2}{n} \right).$$

Sabemos que

$$Var(X) = E[X - E(X)]^2 = \int (x - E[X])^2 \cdot f(x) dx$$

onde  $(x - E[X])^2$  é uma função da variável aleatória  $X$ . Assim,

```
> (VX = mean((x - mean(x))^2))
```

```
[1] 1.089248
```

```
> v = mean(VX/n)
> ep = sqrt(v)
> ep
```

```
[1] 0.03300376
```

## 2 Monte Carlo via Função de Importância

Segundo Ehler(2006), em muitas situações pode ser muito custoso ou mesmo impossível simular valores da distribuição a *posteriori*. Neste caso, pode-se recorrer à uma função  $q(\theta)$  que seja de fácil amostragem, usualmente chamada de função de importância. O procedimento é comumente chamado de amostragem por importância. Se  $q(\theta)$  for uma função de densidade definida no mesmo espaço variação de  $\theta$  então:

$$I = \int \left( \frac{g(\theta)p(\theta)}{q(\theta)} \right) q(\theta) d\theta = E \left[ \frac{g(\theta)p(\theta)}{q(\theta)} \right]$$

onde a esperança agora é com respeito a distribuição  $q(\theta)$ . Assim, se dispomos de uma amostra aleatória  $\theta_1, \dots, \theta_n$  tomada da distribuição  $q(\theta)$  o estimador de Monte Carlo da integral acima fica:

$$\hat{I} = \frac{1}{n} \sum_{i=1}^n \frac{g(\theta_i)p(\theta_i)}{q(\theta_i)}$$

**Exemplo 2:** Tomemos uma única observação de uma v.a.  $X \sim N(\theta; 1)$  sendo  $\theta$  desconhecido. A experiência ou conhecimento prévio do parâmetro  $\theta$  como média da distribuição de  $X$  leva a supor que  $\theta \sim Cauchy(0; 1)$ .

### Solução

- Determinação da distribuição *posteriori*.

- $g(x|\theta) \propto e^{-0,5(x-\theta)^2}$ .
- $p(\theta) = \frac{1}{\pi(1+\theta^2)}$ .
- $p(\theta|x) = \frac{\left(\frac{1}{\pi(1+\theta^2)}\right) \frac{1}{\sqrt{2\pi}} e^{-0,5(x-\theta)^2}}{\int \left(\frac{1}{\pi(1+\theta^2)}\right) \frac{1}{\sqrt{2\pi}} e^{-0,5(x-\theta)^2} d\theta}$ .
- $E[\theta|x] = \frac{\int \theta \left(\frac{1}{1+\theta^2}\right) e^{-(x-\theta)^2} d\theta}{\int \left(\frac{1}{1+\theta^2}\right) e^{-(x-\theta)^2} d\theta}$ .
- $Var(\theta|x) = E[\theta^2|x] - (E[\theta|x])^2$ .

- Gráfico da priori e da verossimilhança.

```
> options(width=80)
> x=rnorm(1,2,1) # gera valor de x para theta = 2
> par(mfrow=c(1,1), mar=c(3.5,3.5,0.5, 0.5), mgp=c(2, .8, 0))
> curve(dnorm(x,2.666545,1),lty=1, from=-3, to=8, ylab='', xlab=expression(theta))
> curve(dcauchy(x,0,1),from=-3, to=8, add=T, lty=2)
> legend(4,0.35,legend=c('priori (Cauchy)', 'veross.(Normal)'), lty=c(2,1))
```

- Estimativa pontual de  $\theta$  (média a posteriori).

Obtenção de  $E[\theta|x]$ :

- a) Gerar  $\theta_1, \dots, \theta_n$  ( $n = 1000$ ), independentes, da distribuição  $N(x; 1)$ ;
  - > n = 1000
  - > set.seed(234)
  - > x = rnorm(1, 2, 1)
  - > theta = rnorm(n, x, 1)

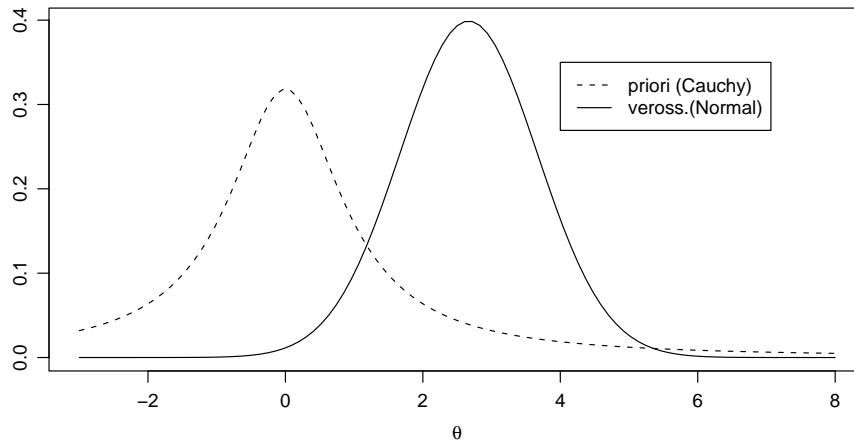


Figura 1: Distribuição a priori (Cauchy(0;1) e verossimilhança (Normal[2,7 ;1])

b) Calcular  $g_i = \frac{\theta_i}{1 + \theta_i^2}$  e  $g_i^* = \frac{1}{1 + \theta_i^2}$ ;

```
> g.num = theta/(1 + theta^2)
> g.den = 1/(1 + theta^2)
```

c) Calcular  $\hat{E}(\theta|x) = \frac{\sum_{i=1}^n g_i}{\sum_{i=1}^n g_i^*}$ .

```
> media.theta = mean(g.num)/mean(g.den)
> media.theta
```

```
[1] 1.877075
```

- Variância do estimador de  $\theta$ .

Determina-se aqui  $Var(\theta|x) = E[\theta^2|x] - (E[\theta|x])^2$ .

a) cálculo de  $g_i = \frac{\theta_i^2}{1 + \theta_i^2}$ ;

```
> g.num2 = theta^2/(1 + theta^2)
```

b) cálculo de  $E[\theta^2|x]$ ;

```
> media.theta2 = mean(g.num2)/mean(g.den)
```

c) cálculo de  $E[\theta^2|x] - E^2[\theta|x]$ .

```
> var.theta = media.theta2 - (media.theta)^2
> var.theta
```

```
[1] 1.065832
```

- Gráfico da distribuição a posteriori de  $\theta$ .

a) criar um intervalo de variação para  $\theta$ ;

b) para cada valor  $\theta$  do intervalo, calcular  $p(\theta|x)$  na sua forma reduzida (sem a constante normalizadora);

c) construir o gráfico do resultado ( $\theta \in (-2; 5)$ ).

```
> par(mar = c(4, 4, 2, 0.5), mgp = c(3, 0.8, 0))
> curve((1/(pi * (1 + media.theta^2))) * ((1/sqrt(2 * pi)) * exp(-0.5 *
+ (x - media.theta)^2)), from = -2, to = 5, ylab = expression(p(theta/x)),
+ xlab = expression(theta), las = 1)
```

- Generalização do problema para  $k$  observações.

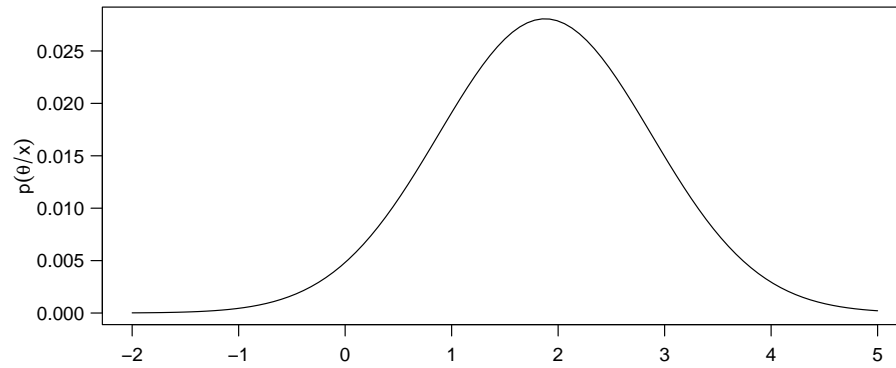


Figura 2: Distribuição da *posteriori* na forma analítica sem a constante normalizadora.

a) distribuição *posteriori*;

$$p(\theta|\mathbf{x}) \propto \frac{1}{(1 + \theta^2)} e^{-0,5 k (\bar{x} - \theta)^2}$$

b) simulando  $k = 50$  observações de uma população  $N(\theta = 2; \sigma^2 = 1)$ ;

```
> x = rnorm(50, 2, 1)
```

c) simulando 1000 observações de  $\theta \sim N(\bar{x}; \sigma^2/\sqrt{k} = 1/\sqrt{50})$ ;

```
> n = 1000
```

```
> theta = rnorm(n, mean(x), 1/sqrt(50))
```

d) cálculo de  $g_i = \frac{\theta_i}{1 + \theta_i^2}$  e  $g_i^* = \frac{1}{1 + \theta_i^2}$ ;

```
> g.num = theta/(1 + theta^2)
```

```
> g.den = 1/(1 + theta^2)
```

e) cálculo de  $\hat{E}(\theta|x) = \frac{\sum_{i=1}^n g_i}{\sum_{i=1}^n g_i^*}$ ;

```
> media.theta = mean(g.num)/mean(g.den)
```

```
> media.theta
```

```
[1] 2.072455
```

f) cálculo de  $E[\theta^2|x]$ ;

```
> g.num2 = theta^2/(1 + theta^2)
```

```
> media.theta2 = mean(g.num2)/mean(g.den)
```

g) cálculo de  $E[\theta^2|x] - E^2[\theta|x]$ .

```
> var.theta = media.theta2 - (media.theta)^2
```

```
> var.theta
```

```
[1] 0.02010174
```

### 3 Método de Aceitação-Rejeição

O método consiste em gerar um valor  $\theta^*$  de uma distribuição auxiliar  $q(\theta)$  (costuma-se utilizar a priori) e aceitar esse valor como sendo da distribuição *a posteriori* com probabilidade  $p(\theta|\mathbf{x})/Aq(\theta)$ , onde  $A$  é uma constante finita tal que  $p(\theta|\mathbf{x}) < Aq(\theta)$ . Para efeitos práticos, toma-se  $A$  como sendo o valor máximo da função de verossimilhança, ou seja,  $A = p(\mathbf{x}|\hat{\theta})$  onde  $\hat{\theta}$  é o estimador de máxima verossimilhança de  $\theta$ . Assim, a probabilidade de aceitação e rejeição se torna  $p(\mathbf{x}|\theta^*)/p(\mathbf{x}|\hat{\theta})$ . O método também funciona se usarmos a versão não normalizada da *posteriori* dada por:  $p(\theta|\mathbf{x}) \propto p(\mathbf{x}|\theta)p(\theta)$ . Se o processo de maximizar a função de verossimilhança for complexo, sugere-se o método da reamostragem ponderada (a seguir).

O algoritmo para gerar valores *a posteriori* é:

1. Gerar um valor  $\theta^*$  da distribuição a priori;
2. Gerar  $u \sim U(0; 1)$ ;
3. Aceitar  $\theta^*$  como um valor da *posteriori* se  $u < p(\mathbf{x}|\theta^*)/p(\mathbf{x}|\hat{\theta})$ , caso contrário, rejeitar  $\theta^*$  e retornar ao primeiro passo.

**Exemplo 3:** Tomemos uma única observação de uma v.a.  $X \sim N(\theta; 1)$  sendo  $\theta$  desconhecido. A experiência ou conhecimento prévio do parâmetro  $\theta$  como média da distribuição de  $X$  leva a supor que  $\theta \sim Cauchy(0; 1)$ . Suponha, agora, que  $X_1, \dots, X_n \sim N(\theta; 1)$  e assumamos uma priori *Cauchy*(0; 1) para  $\theta$ .

Sabemos que:

- $p(\mathbf{x}|\theta) \propto \exp\left\{-\frac{n}{2}(\bar{x} - \theta)^2\right\}$  é a função de verossimilhança;
- $\hat{\theta} = \bar{x}$  é o EMV de  $\theta$ ;
- a probabilidade de aceitação será:

$$\frac{p(\mathbf{x}|\theta^*)}{p(\mathbf{x}|\hat{\theta})} = \frac{\exp\left\{-\frac{n}{2}(\bar{x} - \theta^*)^2\right\}}{\exp\left\{-\frac{n}{2}(\bar{x} - \bar{x})^2\right\}} = \exp\left\{-\frac{n}{2}(\bar{x} - \theta^*)^2\right\}.$$

A aplicação do algoritmo fica:

- a) simulando uma amostra de 50 observações de  $X \sim N(\theta = 2; 1)$  e obtendo o EMV de  $\theta$ ;  

```
> x = rnorm(50, 2, 1)
> x.bar = mean(x)
```
- b) simulando 1000 valores de  $\theta^*$  a partir da distribuição a priori (*Cauchy*(0; 1));  

```
> n = 1000
> theta = rcauchy(n, 0, 1)
```
- c) simulando 1000 valores  $u \sim U(0; 1)$ .  

```
> u = runif(n, 0, 1)
```
- d) calculando o vetor com as probabilidades para aceitação de cada valor de  $\theta^*$  simulado  

```
> prob = exp(-0.5 * 50 * (theta - x.bar)^2)
```
- e) contando os valores de `prob` que são maiores que o respectivo  $u$  simulado;

```
> soma = sum(u < prob)
```

f) determinando a taxa de aceitação.

```
> (taxa = soma/n)
```

```
[1] 0.028
```

g) a figura 3 mostra o gráfico da priori e da verossimilhança.

```
> options(width = 80)
> par(mar = c(3.5, 3.5, 2, 0.5), mgp = c(2, 0.8, 0))
> curve(dnorm(x, x.bar, 1/sqrt(50)), from = -2, to = 3, ylab = "",
+       xlab = expression(theta))
> curve(dcauchy(x, 0, 1), from = -2, to = 3, add = T, lty = 2)
> legend(-1, 2, legend = c("priori", "veross."), lty = c(2, 1))
> rug(theta)
```

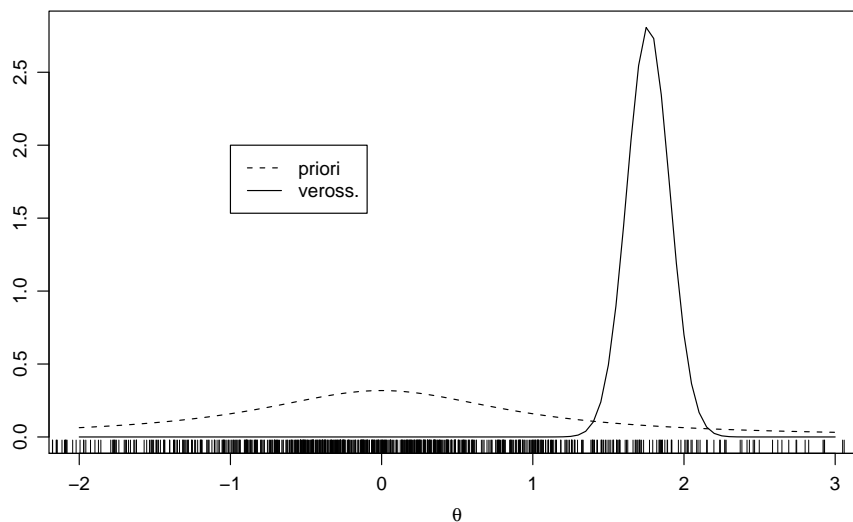


Figura 3: Distribuição da priori e da verossimilhança

Neste exemplo, a taxa de aceitação foi muito baixa (2.8%), significando que, de 1000 valores de  $\theta^*$  simulados, apenas 28 foram aceitos.



## 4 Reamostragem Ponderada

O método consiste em gerar os valores de uma distribuição auxiliar, sem a necessidade da maximização da verossimilhança. A desvantagem do método é que os valores obtidos serão apenas aproximadamente distribuídos segundo a *posteriori*. O algoritmo consiste basicamente de:

1. Gerar valores  $\theta_1, \dots, \theta_n$  da distribuição a priori;
2. Calcular os pesos  $w_1, \dots, w_n$ . De uma forma geral, esses pesos são calculados como:

$$w_i = \frac{p(\theta_i|\mathbf{x})/q(\theta_i)}{\sum_{j=1}^n p(\theta_j|\mathbf{x})/q(\theta_j)}, \quad i : 1 \dots, n$$

Se tomarmos a priori como a densidade auxiliar, ou seja,  $q(\theta) = p(\theta)$ , o peso se reduz a:

$$w_i = \frac{p(\mathbf{x}|\theta_i)}{\sum_{j=1}^n p(\mathbf{x}|\theta_j)}, \quad i : 1 \dots, n;$$

3. Reamostrar valores com probabilidades  $w_1, \dots, w_n$ .

**Exemplo 4:** Em um modelo de regressão linear simples temos que  $y_i \sim N(\beta x_i; 1)$ . Os dados observados são  $\mathbf{y} = (-2, 0, 0, 0, 2)$  e  $\mathbf{x} = (-2, -1, 0, 1, 2)$ , e usamos uma priori vaga  $N(0; 4)$  para  $\beta$ . Faça inferência sobre  $\beta$  obtendo uma amostra da *posteriori* usando reamostragem ponderada. Compare com a estimativa de máxima verossimilhança  $\hat{\beta} = 0,8$ .

**Solução:**

- A figura 4 mostra o diagrama de dispersão com a reta de regressão linear ajustada por mínimos quadrados:

```
> par(mar = c(3.5, 3.5, 0.5, 0.5), mgp = c(2, 0.8, 0))
> plot(c(-2, -1, 0, 1, 2), c(-2, 0, 0, 0, 2), xlab = "X", ylab = "Y")
> abline(lm(c(-2, -1, 0, 1, 2) ~ c(-2, 0, 0, 0, 2)))
```

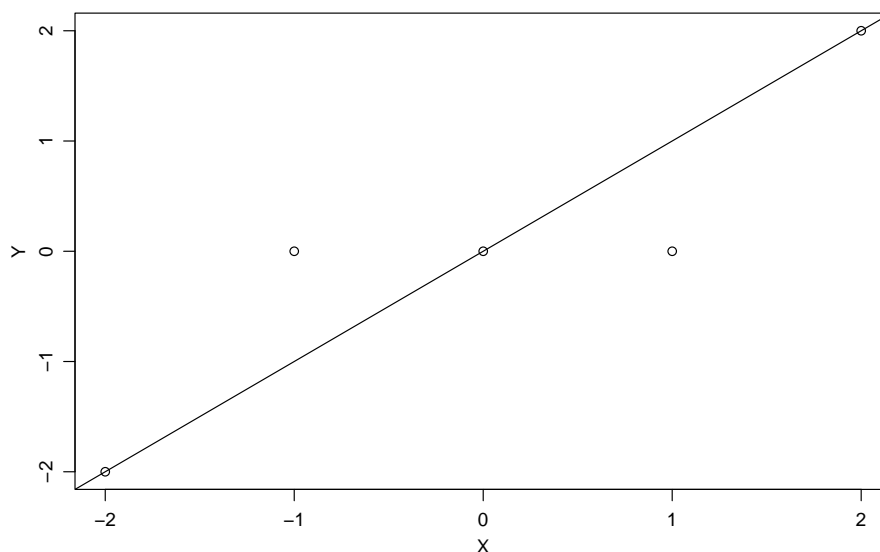


Figura 4: H-scatterplot de  $X$  versus  $Y$  e reta de regressão

- Aplicando o algoritmo:

- a) Entrando com os valores observados de  $X$  e  $Y$ ;
- ```
> x = c(-2, -1, 0, 1, 2)
> y = c(-2, 0, 0, 0, 2)
```
- b) Gerando 1000 valores  $\beta_i \sim N(0; 2^2)$  da distribuição priori;
- ```
> n = 1000
> beta = rnorm(n, 0, 2)
```
- c) Calculando a verossimilhança  $l(\theta_i) = p(\mathbf{x}|\theta_i)$ ;
- ```
> l = sapply(beta, function(b) exp(-0.5 * (sum((y - b * x)^2))))
```
- d) Calculando os pesos  $w_i$ ;
- ```
> w = 1/sum(l)
```
- e) Reamostrando 500  $\beta$ 's com probabilidade  $w$ . Serão escolhidos os 500 valores de  $\beta$  correspondentes às 500 maiores probabilidades  $w$ ;
- ```
> m = 500
> beta.resample = sample(beta, size = m, rep = T, prob = w)
```
- f) Visualizando graficamente o resultado da reamostragem (figura 5).
- ```
> hist(beta.resample, main = "")
```

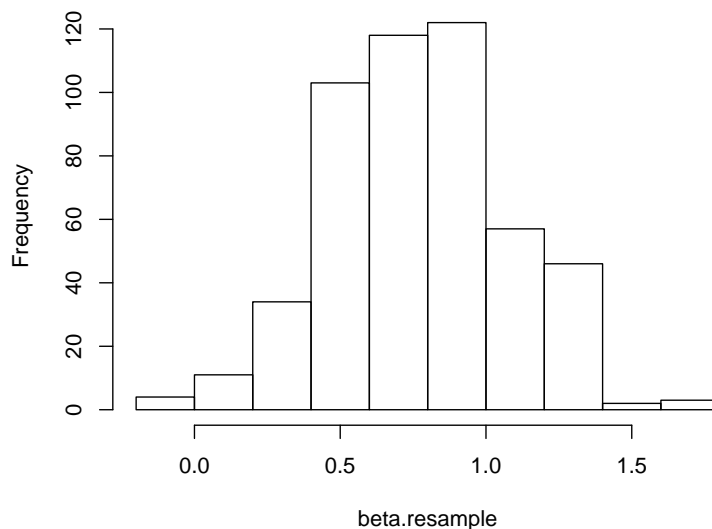


Figura 5: Histograma dos valores reamostrados

Neste exemplo, o estimador  $\hat{\beta}$  de  $\beta$  será a média dos valores reamostrados dado por  $\hat{\beta} = 0.7665$ . Podemos ainda visualizar graficamente os valores reamostrados comparados com a distribuição a priori, conforme a figura 6:

```
> options(width = 80)
> curve(dnorm(x, 0, 2), from = -3, to = 3, ylab = "priori", xlab = expression(beta))
> rug(beta.resample)
```

**Exemplo 5:** Para o mesmo modelo do exemplo 4 e os mesmos dados suponha agora que a variância é desconhecida, i.e.,  $y_i \sim N(\beta x_i; \sigma^2)$ . Usamos uma priori hierárquica para  $(\beta; \sigma^2)$ , i.e.,  $\beta | \sigma^2 \sim N(0; \sigma^2)$  e  $\sigma^{-2} \sim \text{Gama}(0, 1; 0, 1)$ .

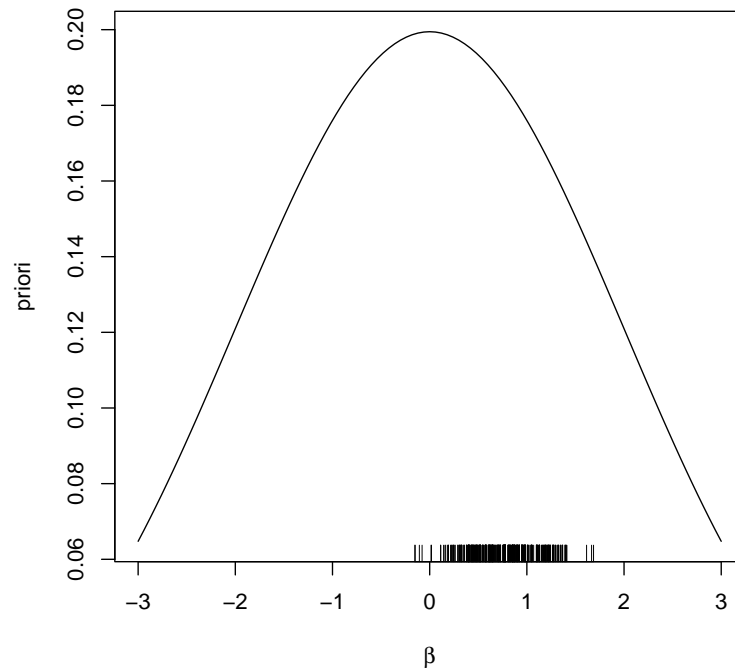


Figura 6: Valores reamostrados comparados com a distribuição a priori

- (a) Obtenha uma amostra da *posteriori* de  $(\beta; \sigma^2)$  usando reamostragem ponderada. Neste problema,  $\theta = (\beta; \sigma^2)$ . Assim, a priori hierárquica será dada por  $p(\beta; \sigma^2) = p(\beta|\sigma^2)p(\sigma^2)$ . Será adotado  $\beta|\sigma^2 \sim N(0; \sigma^2)$  e  $\sigma^{-2} \sim \text{Gamma}(0, 1; 0, 1)$ .
- (b) Estime  $\beta$  e  $\sigma^2$  usando uma aproximação para a média a *posteriori*. Compare com as estimativas de máxima verossimilhança.
- (c) Baseado na amostra do item (a), faça um histograma das distribuições marginais de  $\beta$  e  $\sigma^2$ .

**Solução item (a):**

- i) Simulando valores dos parâmetros  $\sigma^2$  e  $\beta|\sigma^2$  da priori conjunta;

```
> inv.sigma2 = rgamma(n, 0.1, 0.1)
> sigma2 = 1/inv.sigma2
> beta = rnorm(n, sqrt(sigma2))
> theta = cbind(beta, sigma2)
```

- ii) Calculando a verossimilhança;

```
> options(width = 80)
> l = sapply(1:1000, function(i) (theta[i, 2]^(-length(y)/2)) *
+   exp(-0.5 * sum(((y - theta[i, 1] * x)^2)/theta[i, 2])))
```

- iii) Calculando os pesos  $w_i$ ;

```
> w = 1/sum(l)
```

- iv) Reamostrando 500  $\theta = (\beta, \sigma^2)$  com probabilidade  $w$ . Serão escolhidos os 500 valores de  $\theta$  correspondentes às 500 maiores probabilidades  $w$ ;

```
> ind = sample(1:1000, size = m, rep = T, prob = w)
> theta.resample = theta[ind, ]
```

**Solução item (b):**

As estimativas obtidas das médias marginais serão:

```
> options(width = 80)
> theta.hat = as.numeric(c(round(mean(theta.resample[, 1]), 4),
+   round(mean(theta.resample[, 2]), 4)))
```

$\hat{\beta}$	0.8092
$\hat{\sigma}^2$	0.7166

**Solução item (c):**

```
> par(mfrow = c(1, 2))
> hist(theta.resample[, 1], main = "", xlab = "")
> hist(theta.resample[, 2], main = "", xlab = "")
```

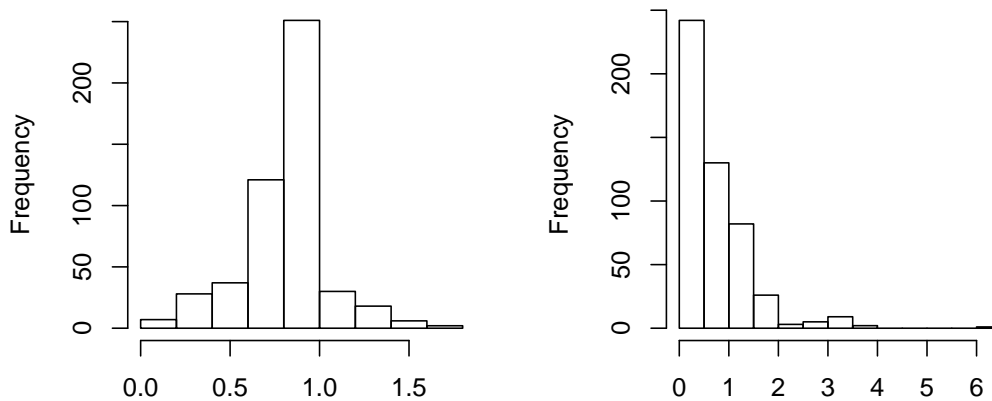


Figura 7: Histograma dos valores reamostrados para  $\beta$  (esquerda) e  $\sigma^2$  (direita)

## 5 Monte Carlo via Cadeias de Markov - MCMC

A idéia do método consiste em utilizar técnicas de simulação iterativa (repetidas) baseadas em Cadeias de Markov para gerar valores correlacionados. Os métodos mais utilizados são: algoritmo de Metropolis-Hastings e Amostrador de Gibbs. Esses métodos realizam um “passeio” aleatório no espaço paramétrico até que o parâmetro convirja para uma distribuição de interesse.

### 5.1 Algoritmo de Metropolis-Hastings

O algoritmo de Metropolis-Hastings pode ser desenvolvido nos seguintes passos:

1. Escolher o número  $n$  de etapas (número de simulações ou número de passos da cadeia).
2. Iniciar o contador de iterações  $t$ , fazendo  $t = 1$ .
3. Especificar um valor inicial para  $\theta^{(t)}$ .
4. Gerar um valor  $\theta'$  de uma distribuição proposta  $q(\theta'|\theta)$ . Esta distribuição proposta pode depender do estado atual da cadeia.
5. Gerar  $u \sim U(0; 1)$  (priori).
6. Determinar a probabilidade de aceitação  $\alpha(\theta; \theta')$  dada por:

$$\alpha(\theta; \theta') = \min \left\{ 1; \frac{p(x|\theta') p(\theta') q(\theta|\theta')}{p(x|\theta) p(\theta) q(\theta'|\theta)} \right\}$$

onde  $p(x|\theta)$  é a distribuição de interesse.

7. Se  $u < \alpha(\theta; \theta')$  então o valor de  $\theta'$  é aceito e  $\theta^{(t)} = \theta'$ . Caso contrário  $\theta'$  é rejeitado e  $\theta^{(t)} = \theta^{(t-1)}$ .
8. Se  $t = n$  (é o último passo) ir para o passo 11.
9. Incrementar o contador  $t$  fazendo  $t = t + 1$ .
10. Voltar para o passo 4.
11. FIM.

**Exemplo 6:** Em uma certa população de animais sabe-se que cada animal pode pertencer a uma dentre 4 linhagens genéticas com probabilidades

$$p_1 = \frac{1}{2} + \frac{\theta}{4}; \quad p_2 = \frac{1-\theta}{4}; \quad p_3 = \frac{1-\theta}{4}; \quad p_4 = \frac{\theta}{4}$$

sendo  $0 < \theta < 1$  um parâmetro desconhecido. Para qualquer  $\theta \in (0; 1)$  é fácil verificar que  $p_i > 0$ ,  $i = 1, 2, 3, 4$  e  $p_1 + p_2 + p_3 + p_4 = 1$ . Observando-se  $n$  animais dentre os quais  $y_i$  pertencem à linhagem  $i$ , o vetor aleatório  $Y = (y_1, y_2, y_3, y_4)$  tem distribuição multinomial com parâmetros  $n, p_1, p_2, p_3, p_4$  e portanto,

$$\begin{aligned} p(y|\theta) &= \frac{n!}{y_1! y_2! y_3! y_4!} p_1^{y_1} p_2^{y_2} p_3^{y_3} p_4^{y_4} \\ &\propto (2 + \theta)^{y_1} (1 - \theta)^{y_2 + y_3} \theta^{y_4} \end{aligned}$$

Foram observados 197 animais com a quantidade deles nas diferentes categorias dado por  $y = (125, 18, 20, 34)$ . Gerar uma cadeia de Markov com 1000 valores do parâmetro  $\theta$  que definirá as proporções  $p_1, p_2, p_3, p_4$ . Descartar os 100 primeiros valores (amostra de aquecimento).

**Solução:**

- Escolhemos a priori  $\theta \sim U(0; 1)$ . Assim,  $p(\theta) = 1, \forall \theta \in (0; 1)$  e a *posteriori* será dada por  $p(\theta|y) \propto (2 + \theta)^{y_1} (1 - \theta)^{y_2+y_3} \theta^{y_4}$ .

Tomando como distribuição porposta  $q(\theta)$ , também, a distribuição  $U(0; 1)$ , então,  $q(\theta) = 1 \forall \theta$ . Daí, a probabilidade de aceitação será dada por:

$$\begin{aligned} \alpha(\theta; \theta') &= \min \left\{ 1; \frac{p(y|\theta')}{p(y|\theta)} \right\} = \\ &= \min \left\{ 1; \left( \frac{2 + \theta'}{2 + \theta} \right)^{y_1} \left( \frac{1 - \theta'}{1 + \theta} \right)^{y_2+y_3} \left( \frac{\theta'}{\theta} \right)^{y_4} \right\} \end{aligned}$$

- Criando a função para o cálculo da probabilidade de interesse  $p(y|\theta)$ .

```
> p = function(teta, y) {
+   p = (2 + teta)^y[1] * (1 - teta)^(y[2] + y[3]) * teta^y[4]
+   return(p)
+ }
```

- Criando a função de implementação do algoritmo de simulação de Metropolis-Hastings (Cadeia de Markov);

```
> options(width=80)
> metr=function(n,y,p,start){
+   theta=matrix(NA, nrow=n)
+   theta[1] = start # valor inicial
+   sucess=0 # quant. de valores aceitos
+   for (i in 2:n) {
+     theta.prop = runif(1) # valor proposto
+     A = p(theta.prop,y)/p(theta[i-1],y)
+     prob=min(1,A)
+     theta.u=runif(1)
+     if (theta.u < prob) {
+       theta[i]=theta.prop
+       sucess=sucess+1
+     }
+     else theta[i]=theta[i-1]
+   }
+   taxa=sucess/n
+   return(list(theta=theta, taxa=round(taxa,2)))
+ }
```

- Simulando para um valor inicial  $\theta = 0.2$ .

```
> n = 1000
> y = c(125, 18, 20, 34)
> start = 0.2
> m = metr(n, y, p, start)
```

- Obtendo a taxa e aceitação e a média, excluindo as 100 primeiras simulações.

```
> options(width = 80)
> theta.hat = as.numeric(c(round(m$taxa, 2), round(mean(m$theta[101:1000]),
+   4)))
```

Taxa	0.17
$\hat{\mu}$	0.6251

### Exemplo 7:

Suponha que queiramos simular valores  $X \sim N(0; 1)$  propondo valores  $Y \sim N(x; \sigma^2)$ .

### Solução:

- Neste exemplo, a probabilidade de aceitação-rejeição fica:

$$\begin{aligned}\alpha(x; y) &= \min \left\{ 1; \frac{\pi(y) q(x|y)}{\pi(x) q(y|x)} \right\} \\ \alpha(x; y) &= \min \left\{ 1; \frac{\exp\{-0.5 y^2\} \exp\{(x-y)^2\}}{\exp\{-0.5 x^2\} \exp\{(y-x)^2\}} \right\} \\ \alpha(x; y) &= \min \left\{ 1; \exp \left\{ -\frac{1}{2} (y^2 - x^2) \right\} \right\}\end{aligned}\tag{3}$$

Como podemos notar na equação 3, houve uma simplificação no quociente das “propostas”, devido à simetria da função  $q(x|y)$  e  $q(y|x)$ . Esta é a situação de um processo denominado *random walk* no qual o algoritmo dá saltos aleatórios cobrindo mais eficientemente o espaço paramétrico em torno desse novo centro. Neste caso, taxas de aceitação entre 20 e 30

- Definindo uma função para executar o algoritmo.

```
> options(width=80)
> # proposta q() simetrica
> metrop = function(n,sigma){
+   x = matrix(NA, nrow=n)
+   x[1] = 0
+   sucess = 0
+   for (i in 2:n){
+     y = rnorm(1,x[i-1],sigma)
+     prob = min(1,exp(-0.5*(y^2-x[i-1]^2)))
+     u = runif(1)
+     if (u < prob) {
+       x[i] = y
+       sucess = sucess + 1
+     }
+     else x[i] = x[i-1]
+   }
+   return(list(x=x,taxa=round(sucess/n,4)))
+ }
```

Nesta função,  $n$  representa o número de simulações,  $\sigma$  é o desvio padrão da “proposta”,  $y$  é simulado da distribuição da “proposta” com média igual ao valor de  $x$  obtido no “passo” anterior e os  $x$  são os valores da distribuição que desejamos simular.

- Simulando 1000 valores com diferentes valores de  $\sigma$  da “proposta  $q()$ ”.

```
> n = 1000
> sigma = c(0.05, 2, 4)
> m1 = metrop(n, sigma[1])
> m2 = metrop(n, sigma[2])
> m3 = metrop(n, sigma[3])
> taxa.r = as.numeric(c(m1$taxa, m2$taxa, m3$taxa))
> cat("\n Taxa de aceitação", taxa.r, "\n")
```

Taxa de aceitação 0.988 0.504 0.306

- Construindo gráficos dos valores simulados.

```

> options(width = 80)
> par(mfrow = c(1, 3))
> plot(m1$x, type = "l", xlab = "iterações", ylab = expression(theta))
> acf(m1$x, main = "", xlab = "defasagens", ylab = "autocorrelações")
> qqnorm(m1$x[seq(100, n, 5)], pch = "*", main = "", xlab = "Quantil teorico",
+       ylab = "Quantil amostral")
> qqline(m1$x[seq(100, n, 5)], lwd = 2)

```

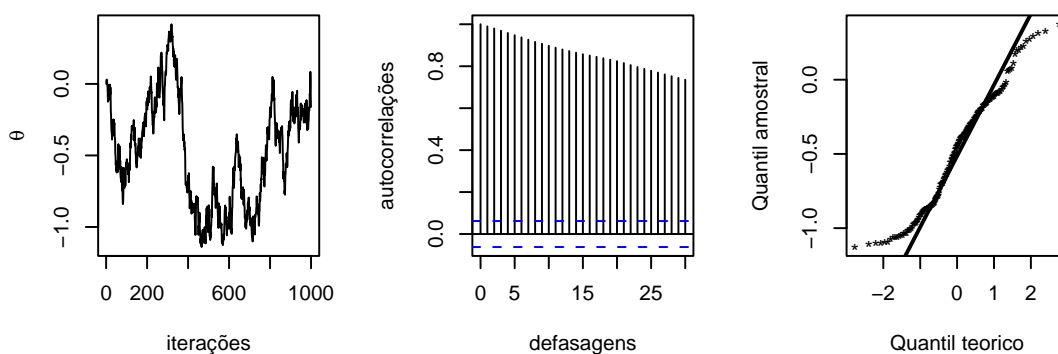


Figura 8: Iterações (esquerda), correlações (centro) de 1000 valores simulados usando o algoritmo de Metropolis-Hastings com  $\sigma = 0.05$  e taxa de aceitação de 98.8% e qqplot (direita) com *burning* dos 100 primeiros valores e *thinning* de 5 em 5.

A figura 8 à esquerda mostra a ausência de estacionariedade sugerindo a não-convergência da cadeia. A mesma figura, ao centro, indica uma alta autocorrelação serial.

O *thinning* é um procedimento que faz uma espécie de amostragem sistemática no conjunto de valores simulados da distribuição *a posteriori* do(s) parâmetro(s) de interesse. A idéia central consiste em tomar sequencialmente um valor em um intervalo de  $k$  simulados. Neste exemplo, os valores simulados da distribuição *a posteriori* deveriam corresponder valores de uma distribuição  $N(0;1)$ . Podemos verificar se a suposição é correta contruindo um gráfico qqnorm. A figura 8 (direita) ilustra o respectivo qqnorm de  $\theta|y$  com um *thinning* de 5 simulações e um *burning* das 100 primeiras observações onde notamos um afastamento da normalidade em ambas as caudas da distribuição.

```

> par(mfrow = c(1, 3))
> plot(m2$x, type = "l", xlab = "iterações", ylab = expression(theta))
> acf(m2$x, main = "", xlab = "defasagens", ylab = "autocorrelações")
> qqnorm(m2$x[seq(100, n, 5)], pch = "*", main = "", xlab = "Quantil teorico",
+       ylab = "Quantil amostral")
> qqline(m2$x[seq(100, n, 5)], lwd = 1)

```

A figura 9 à esquerda mostra estacionariedade sugerindo a convergência da cadeia. A mesma figura, ao centro, ainda indica uma autocorrelação serial, mas somente para os valores relativamente próximos (menores que o  $10^0$  valor anterior simulado).

```

> options(width = 80)
> par(mfrow = c(1, 3))
> plot(m3$x, type = "l", xlab = "iterações", ylab = expression(theta))
> acf(m3$x, main = "", xlab = "defasagens", ylab = "autocorrelações")
> qqnorm(m3$x[seq(100, n, 5)], pch = "*", main = "", xlab = "Quantil teorico",
+       ylab = "Quantil amostral")
> qqline(m3$x[seq(100, n, 5)], lwd = 1)

```



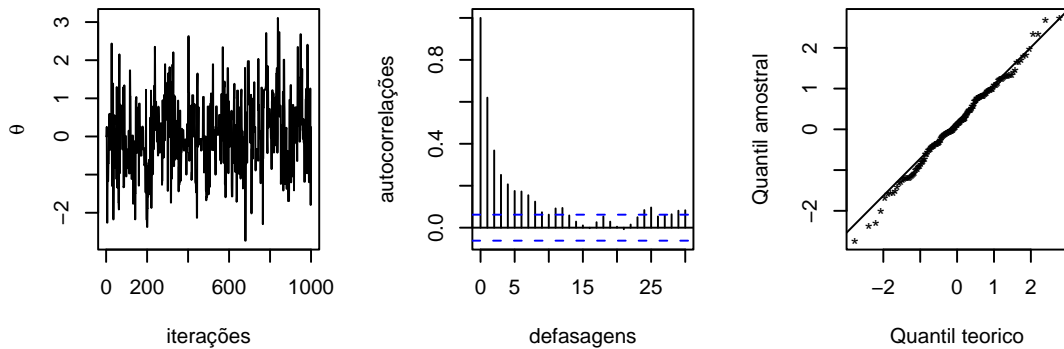


Figura 9: Iterações (esquerda), correlações (centro) de 1000 valores simulados usando o algoritmo de Metropolis-Hastings com  $\sigma = 2$  e taxa de aceitação de 50.4% e qqplot (direita) com *burning* dos 100 primeiros valores e *thinning* de 5 em 5.

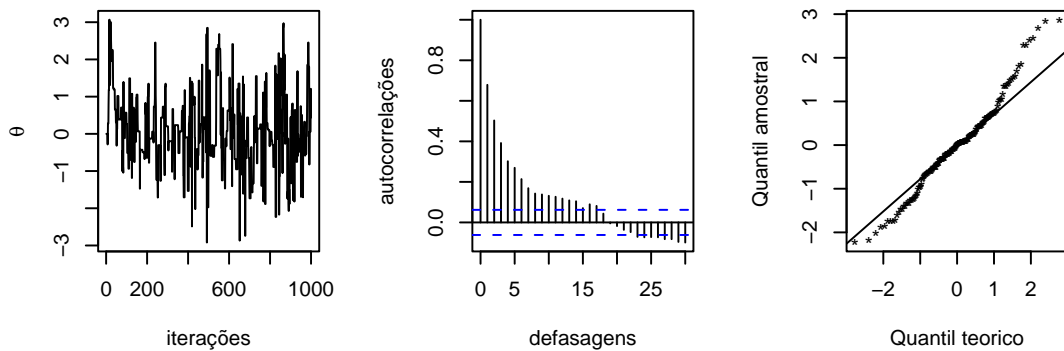


Figura 10: Iterações (esquerda), correlações (centro) de 1000 valores simulados usando o algoritmo de Metropolis-Hastings com  $\sigma = 4$  e taxa de aceitação de 30.6% e qqplot (direita) com *burning* dos 100 primeiros valores e *thinning* de 5 em 5.

A figura 10 à esquerda mostra estacionariedade indicando a convergência da cadeia. A mesma figura, à direita, ainda indica uma alta autocorrelação serial.

```
> par(mfrow = c(1, 3))
> hist(m1$x[seq(100, n, 5)], main = "", xlab = "")
> hist(m2$x[seq(100, n, 5)], main = "", xlab = "")
> hist(m3$x[seq(100, n, 5)], main = "", xlab = "")
```

- Determinando as estimativas da *posteriori*.

As estimativas de  $\theta$  em relação a cada variância atribuída à “proposta” com “burning” das 100 primeiras observações resulta em:

```
> theta.hat = as.numeric(c(round(mean(m1$x[100:n]), 4), round(mean(m2$x[100:n]),
+ 4), round(mean(m3$x[100:n]), 4)))
```

$\sigma_Y$	0.05	2	4
$\hat{\mu}_X$	-0.4872	0.0989	0.0057

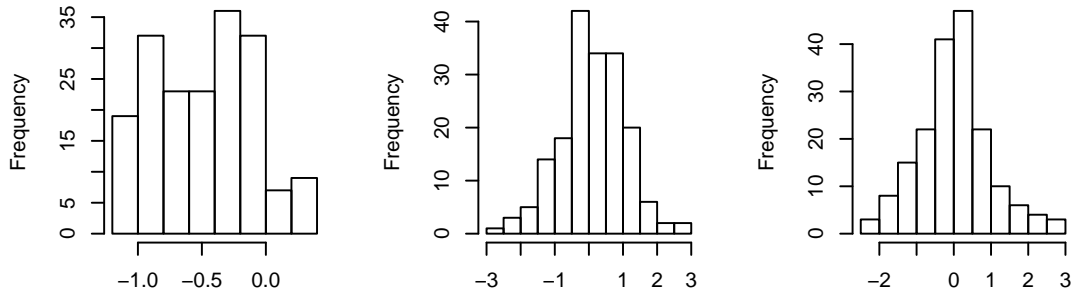


Figura 11: Distribuição da *posteriori* para  $\sigma_1 = 0.05$  (esquerda),  $\sigma_2 = 2$  (centro) e  $\sigma_3 = 4$  (direita).

Como esperado, as estimativas obtidas estão próximas de zero já que  $X \sim N(0; 1)$ .

**Exemplo 8:** Suponha que queiramos simular valores  $X \sim N(0; 1)$  propondo valores  $Y \sim Cauchy(0; \sigma)$ .

- Para a distribuição proposta (Cauchy) selecione experimentalmente o valor de  $\sigma$  que maximiza a taxa de aceitação.
- Para este valor de  $\sigma$  faça os gráficos dos valores simulados da cadeia ao longo das iterações e verifique se há indicação de convergência.
- Repita os itens anteriores com a distribuição proposta  $Cauchy(\theta; \sigma)$ .

A distribuição Cauchy deriva de uma distribuição *t-student* com 1 grau de liberdade. Considerando a distribuição *t-student*:

$$f(t|\mu; \sigma^2) = \frac{\Gamma\left(\frac{n+1}{2}\right) n^{n/2}}{\Gamma\left(\frac{n}{2}\right)} \left[ n + \frac{(t-\mu)^2}{\sigma^2} \right]^{-\frac{(n+1)}{2}} \quad t \in \mathbb{R}$$

então, para  $n = 1$  grau de liberdade, temos:

$$Cauchy(y|\mu; \sigma) = \left( \pi \sigma \left[ 1 + \left( \frac{y-\mu}{\sigma} \right)^2 \right] \right)^{-1}$$

que representa a distribuição de Cauchy com parâmetro de locação  $\mu$  e parâmetro de escala  $\sigma$ .

**Solução item (a):**

- Obtendo a expressão da probabilidade de aceitação/rejeição.

$$\alpha(x; y) = \min \left\{ 1; \frac{\pi(y) q(x|y)}{\pi(x) q(y|x)} \right\}$$

$$\alpha(x; y) = \min \left\{ 1; \frac{\exp\{-0.5 y^2\}}{\exp\{-0.5 x^2\}} \frac{\left( \pi \sigma \left[ 1 + \left( \frac{x-0}{\sigma} \right)^2 \right] \right)^{-1}}{\left( \pi \sigma \left[ 1 + \left( \frac{y-0}{\sigma} \right)^2 \right] \right)^{-1}} \right\}$$

$$\alpha(x; y) = \min \left\{ 1; \exp \left\{ -\frac{1}{2} (y^2 - x^2) \right\} \left( \frac{\sigma^2 + y^2}{\sigma^2 + x^2} \right) \right\}$$

- Definindo a função para executar o algoritmo.

```

> # proposta Cauchy
> metrop = function(n, sigma){
+ x = matrix(NA, nrow=n)
+ x[1] = 0 # escolha arbitraria
+ sucess = 0
+ for (i in 2:n){
+     y = rcauchy(1,0,sigma)
+     prob = min(1, exp(-0.5*(y^2-x[i-1]^2))*(y^2 + sigma^2)/(x[i-1]^2 + sigma^2))
+     u = runif(1)
+     if (u < prob) {
+         x[i] = y
+         sucess = sucess + 1
+     }
+     else x[i] = x[i-1]
+ }
+ return(list(x=x, taxa=round(sucess/n,4)))
+ }

```

Nesta função,  $n$  representa o número de simulações,  $\sigma$  é o parâmetro de escala da “proposta”,  $y$  é simulado da distribuição da “proposta” com média igual zero e  $\sigma$  como parâmetro de escala.

- Simulando 1000 valores com diferentes valores de  $\sigma$  da “proposta  $q()$ ”.

```

> n = 1000
> sigma = c(0.545, 2, 3)
> m1 = metrop(n, sigma[1])
> m2 = metrop(n, sigma[2])
> m3 = metrop(n, sigma[3])
> taxa.r = as.numeric(c(m1$taxa, m2$taxa, m3$taxa))
> cat("\n Taxa de aceitação", taxa.r, "\n")

```

Taxa de aceitação 0.693 0.439 0.32

Atribuindo valores ao vetor  $\sigma$  observamos, após várias e sucessivas tentativas, que o valor máximo ou maior taxa de aceitação, 69.3%, ocorreu para  $\sigma = 0.545$ .

### Solução item (b):

- Construindo gráficos dos valores simulados para  $\sigma_1 = 0.545$ .

```

> par(mfrow = c(1, 3))
> plot(m1$x, type = "l", xlab = "iterações", ylab = expression(theta))

```

```

> acf(m1$x, main = "", xlab = "defasagens", ylab = "autocorrelações")
> qqnorm(m1$x[seq(100, n, 5)], pch = "*", main = "", xlab = "Quantil teorico",
+       ylab = "Quantil amostral")
> qqline(m1$x[seq(100, n, 5)], lwd = 1)

```

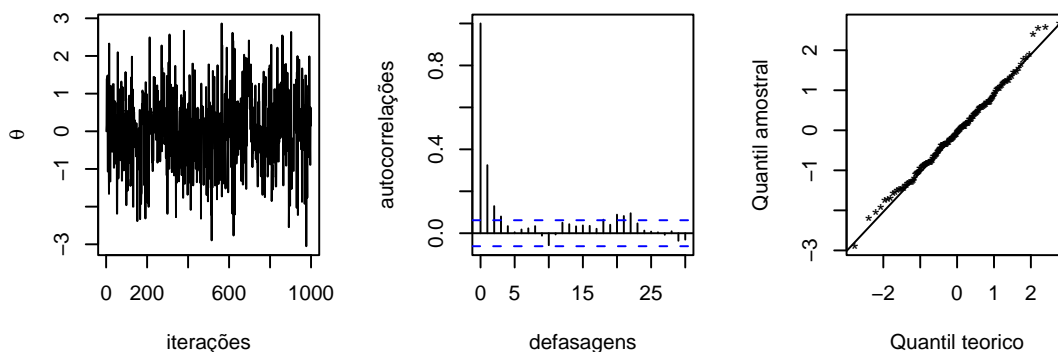


Figura 12: Iterações (esquerda), correlações (centro) de 1000 valores simulados usando o algoritmo de Metropolis-Hastings com  $\sigma = 0.545$  e taxa de aceitação de 69.3% e qqplot (direita) com *burning* dos 100 primeiros valores e *thinning* de 5 em 5.

- Construindo gráficos dos valores simulados para  $\sigma_2 = 2$

```

> par(mfrow = c(1, 3))
> plot(m2$x, type = "l", xlab = "iterações", ylab = expression(theta))
> acf(m2$x, main = "", xlab = "defasagens", ylab = "autocorrelações")
> qqnorm(m2$x[seq(100, n, 5)], pch = "*", main = "", xlab = "Quantil teorico",
+       ylab = "Quantil amostral")
> qqline(m2$x[seq(100, n, 5)], lwd = 1)

```

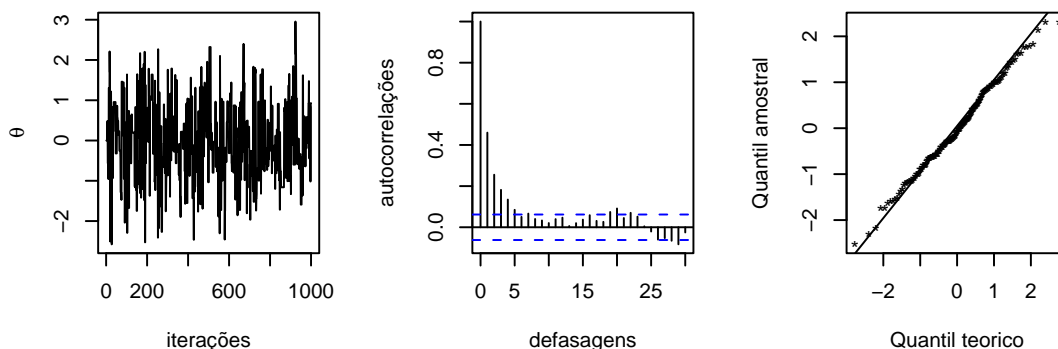


Figura 13: Iterações (esquerda), correlações (centro) de 1000 valores simulados usando o algoritmo de Metropolis-Hastings com  $\sigma = 2$  e taxa de aceitação de 43.9% e qqplot (direita) com *burning* dos 100 primeiros valores e *thinning* de 5 em 5.

- Construindo gráficos dos valores simulados para  $\sigma_3 = 3$

```

> options(width = 80)
> par(mfrow = c(1, 3))
> plot(m3$x, type = "l", xlab = "iterações", ylab = expression(theta))

```

```

> acf(m3$x, main = "", xlab = "defasagens", ylab = "autocorrelações")
> qqnorm(m3$x[seq(100, n, 5)], pch = "*", main = "", xlab = "Quantil teorico",
+       ylab = "Quantil amostral")
> qqline(m3$x[seq(100, n, 5)], lwd = 1)

```

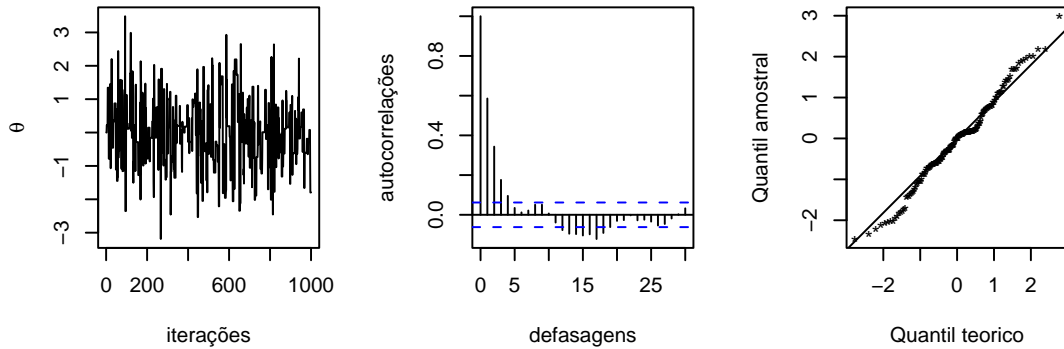


Figura 14: Iterações (esquerda), correlações (centro) de 1000 valores simulados usando o algoritmo de Metropolis-Hastings com  $\sigma = 3$  e taxa de aceitação de 32% e qqplot (direita) com *burning* dos 100 primeiros valores e *thinning* de 5 em 5.

As figuras 12, 13 e 14, à esquerda, mostram estacionariedade sugerindo a convergência das cadeias. Ao centro, os gráficos revelam alta autocorrelação e a normalidade dos dados pode ser verificada pelos gráficos à direita.

- Distribuição da *posteriori* para  $\sigma_1 = 0.545$ , 2 e 3.

```

> par(mfrow = c(1, 3))
> hist(m1$x[seq(100, n, 5)], main = "", xlab = "")
> hist(m2$x[seq(100, n, 5)], main = "", xlab = "")
> hist(m3$x[seq(100, n, 5)], main = "", xlab = "")

```

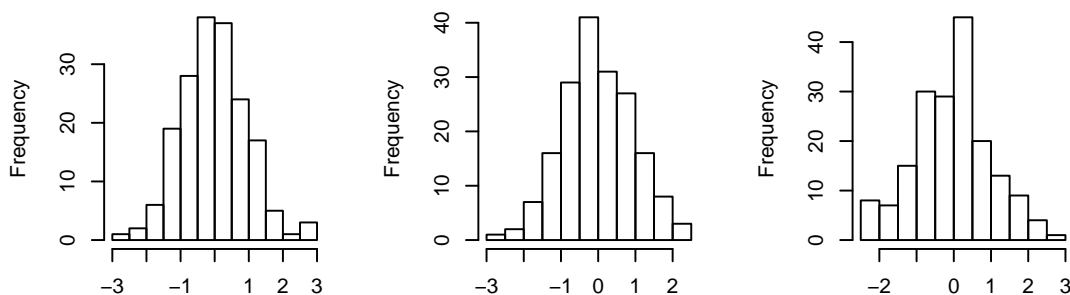


Figura 15: Distribuição da *posteriori* para  $\sigma = 0.545$  (esquerda),  $\sigma_2 = 2$  (centro) e  $\sigma_3 = 3$  (direita).

- Determinando as estimativas da *posteriori*.

As estimativas de  $\theta$  em relação a cada variância atribuída a “proposta” com “burning” das 100 primeiras observações resulta em:

```

> theta.hat = as.numeric(c(round(mean(m1$x[100:n]), 4), round(mean(m2$x[100:n]),
+ 4), round(mean(m3$x[100:n]), 4)))

```

$\sigma_Y$	0.545	2	3
$\hat{\mu}_X$	0.048	-0.0763	-0.0396

Novamente, sendo  $X \sim N(0; 1)$ , as estimativas se aproximaram de zero.

- Densidade de uma distribuição *Cauchy* com parâmetros de dispersão iguais a  $\sigma = 0.545$ ,  $\sigma = 2$  e  $\sigma = 3$ .

```
> par(mfrow = c(1, 1))
> x <- seq(-4, 4, l = 501)
> C1 <- dcauchy(x, 0, sigma[1])
> C2 <- dcauchy(x, 0, sigma[2])
> C3 <- dcauchy(x, 0, sigma[3])
> plot(x, C1, ty = "l")
> lines(x, C2, lty = 2)
> lines(x, C3, lty = 3)
```

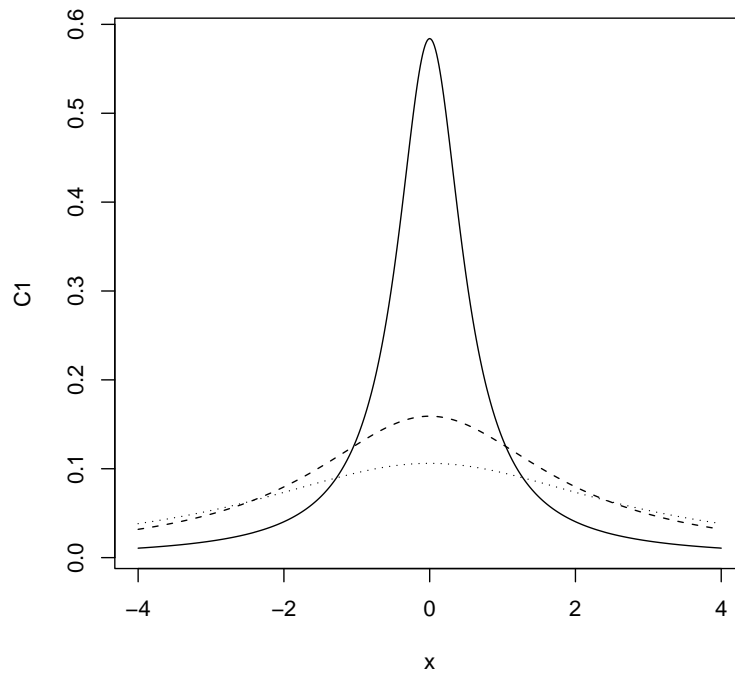


Figura 16: Densidade de uma distribuição *Cauchy* com parâmetros de dispersão iguais a  $\sigma = 0.545$  (linha contínua),  $\sigma = 2$  (linha pontilhada maior) e  $\sigma = 3$  (linha pontilhada menor)

## 5.2 Algoritmo Amostrador de Gibbs

No método amostrador de Gibbs, as transações de estado são feitas de acordo com as distribuições condicionais completas, dadas por:

$$\pi(\theta_i | \boldsymbol{\theta}_{-i}) = \frac{\pi(\boldsymbol{\theta})}{\int \pi(\boldsymbol{\theta}) d\theta_i} \quad (4)$$

onde  $\boldsymbol{\theta}_{-i} = \mathbb{I}_{-i}\boldsymbol{\theta}$ , sendo  $\boldsymbol{\theta} = (\theta_1, \theta_2, \dots, \theta_p)'$ ,  $i = 1, 2, \dots, p$ , ou seja:

$$\boldsymbol{\theta}_{-i} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{i-1} \\ \theta_{i+1} \\ \vdots \\ \theta_p \end{bmatrix}_{(p-1) \times 1} = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & 1 \end{bmatrix}_{(p-1) \times p} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \theta_p \end{bmatrix}_{p \times 1}$$

A matriz  $\mathbb{I}_{-i}$  corresponde a uma matriz identidade excluída a  $i$ -ésima linha. Em outras palavras, podemos dizer que a distribuição condicional completa é a distribuição da  $i$ -ésima componente de  $\boldsymbol{\theta}$  condicionada em todas as outras componentes. A equação 4 também pode ser escrita como:  $\pi(\boldsymbol{\theta}) = \pi(\theta_i | \boldsymbol{\theta}_{-i}) \pi_{\boldsymbol{\theta}_{-i}}(\boldsymbol{\theta}_{-i})$ .

Queremos gerar uma amostra de  $\pi(\boldsymbol{\theta})$  onde a transição se dá da seguinte forma:

- especificamos valores para um vetor inicial  $\boldsymbol{\theta}^{(0)}$ ;
- calcula-se a probabilidade de  $\theta_1$  condicionada aos  $\theta_r$ 's restantes com os valores especificados inicialmente;
- nas transições de  $\theta_{r-1}$  para  $\theta_r$ ,  $r = 2, 3, \dots, p$ , calcula-se, para  $\theta_r$ , a probabilidade  $\pi(\theta_r | \boldsymbol{\theta}_{-r})$  substituindo em  $\boldsymbol{\theta}_{-r}$  o valor de  $\theta_{r-1}$  calculado no passo anterior.

Segundo Paulino, Turkman e Murteira (2003)<sup>3</sup> este procedimento é o adotado pelo algoritmo de Metropolis-Hastings com  $q(\boldsymbol{\theta}; \boldsymbol{\theta}')$  e  $q(\boldsymbol{\theta}'; \boldsymbol{\theta})$  positivas se, e somente se,  $\boldsymbol{\theta}$  e  $\boldsymbol{\theta}'$  diferirem no máximo em uma coordenada.

Se as distribuições condicionais completas forem conhecidas, então o algoritmo será desenvolvido como:

1. Inicie o contador de iterações em  $t = 0$  e defina o número  $n$  de iterações;
2. Especifique valores iniciais para  $\boldsymbol{\theta}^{(0)}$ ;
3. Avance  $t$  fazendo  $t = t + 1$  e obtenha  $\boldsymbol{\theta}^{(t)}$  a partir de  $\boldsymbol{\theta}^{(t-1)}$  por geração sucessiva de valores como:

$$\begin{aligned} \theta_1^{(t)} &\sim \pi(\theta_1 | \theta_2^{(t-1)}, \theta_3^{(t-1)}, \theta_4^{(t-1)}, \dots, \theta_p^{(t-1)}) \\ \theta_2^{(t)} &\sim \pi(\theta_2 | \theta_1^{(t)}, \theta_3^{(t-1)}, \theta_4^{(t-1)}, \dots, \theta_p^{(t-1)}) \\ \theta_3^{(t)} &\sim \pi(\theta_3 | \theta_1^{(t)}, \theta_2^{(t)}, \theta_4^{(t-1)}, \dots, \theta_p^{(t-1)}) \\ &\vdots \\ \theta_p^{(t)} &\sim \pi(\theta_p | \theta_1^{(t)}, \theta_2^{(t)}, \theta_3^{(t)}, \dots, \theta_{p-1}^{(t)}) \end{aligned}$$

4. Se  $t = n$ , encerrar;
5. Voltar a etapa 3.

**Exemplo 9:** Em um processo de contagem no qual foram observados  $Y_1, Y_2, \dots, Y_m, Y_{m+1}, \dots, Y_n$ , suspeita-se que houve um ponto de mudança  $m$  tal que:

$$\begin{aligned} Y_i &\sim \text{Poisson}(\lambda), & i = 1, \dots, m \\ Y_i &\sim \text{Poisson}(\phi), & i = (m + 1), \dots, n. \end{aligned}$$

<sup>3</sup>Paulino, Carlos Daniel; Turkman, M. Antónia Amaral; Murteira, Bento. *Estatística Bayesiana*. Lisboa: Fundação Calouste Gulbenkian, 2003. ISBN 972-31-1043-1

Assumindo-se as distribuições *a priori* independentes  $\lambda \sim \text{Gamma}(a; b)$ ,  $\phi \sim \text{Gamma}(c; d)$  e  $p(m) = 1/n$ , estimar o ponto de mudança  $m$  e os parâmetros  $\lambda$  e  $\phi$  (CARLIN, *et all*)<sup>4</sup>.

### Solução

- Calculando a *posteriori*.

$$p(\lambda; \phi; m | \mathbf{y}) \propto \left( \prod_{i=1}^m e^{-\lambda y_i} \right) \left( \prod_{i=1}^m e^{-\phi \phi^{y_i}} \right) \lambda^{a-1} e^{-b\lambda} \phi^{c-1} e^{-d\phi} \left( \frac{1}{n} \right)$$

$$p(\lambda; \phi; m | \mathbf{y}) \propto \lambda^{a+t_1-1} e^{-(b+m)\lambda} \phi^{c+t_2-1} e^{-(d+n-m)\phi} \left( \frac{1}{n} \right)$$

onde  $t_1 = \sum_{i=1}^m y_i$  e  $t_2 = \sum_{i=m+1}^n y_i$ .

- Calculando as distribuições condicionais completas.

(a) Para o parâmetro  $\lambda$ :

$$p(\lambda | \phi; m; \mathbf{y}) = \frac{p(\lambda; \phi; m; \mathbf{y})}{p(\phi; m; \mathbf{y})} = \frac{p(\mathbf{y} | \lambda; \phi; m) p(\lambda; \phi; m)}{p(\phi; m; \mathbf{y})}$$

$$p(\lambda | \phi; m; \mathbf{y}) = \frac{p(\mathbf{y} | \lambda; \phi; m) p(\lambda) p(\phi) p(m)}{p(\phi; m; \mathbf{y})}$$

Como  $p(m)$ ,  $p(\phi)$  e  $p(\phi; m; \mathbf{y})$  não envolvem o parâmetro  $\lambda$  de interesse, então:

$$p(\lambda | \phi; m; \mathbf{y}) \propto p(\mathbf{y} | \lambda; \phi; m) p(\lambda)$$

$$p(\lambda | \phi; m; \mathbf{y}) \propto \left( \prod_{i=1}^m e^{-\lambda y_i} \right) \left( \prod_{i=1}^m e^{-\phi \phi^{y_i}} \right) \lambda^{a-1} e^{-b\lambda}$$

Novamente  $\prod_{i=1}^m e^{-\phi \phi^{y_i}}$  não envolve  $\lambda$ , então:

$$p(\lambda | \phi; m; \mathbf{y}) \propto \left( \prod_{i=1}^m e^{-\lambda y_i} \right) \lambda^{a-1} e^{-b\lambda}$$

$$p(\lambda | \phi; m; \mathbf{y}) \propto e^{-m\lambda} \lambda^{t_1} \lambda^{a-1} e^{-b\lambda}$$

$$p(\lambda | \phi; m; \mathbf{y}) \propto \lambda^{a+t_1-1} e^{-(m+b)\lambda} \sim \text{Gamma}(a + t_1; b + m)$$

(b) Para o parâmetro  $\phi$ :

Com um desenvolvimento análogo ao item (a) temos:

$$p(\phi | \lambda; m; \mathbf{y}) \propto \phi^{c+t_2-1} e^{-(d+n-m)\phi} \sim \text{Gamma}(c + t_2; d + n - m)$$

(c) Para o parâmetro  $m$ :

Com um desenvolvimento análogo ao item (a) temos:

$$p(m | \lambda; \phi; \mathbf{y}) \propto \lambda^{t_1} e^{-m\lambda} \phi^{t_2} e^{-(n-m)\phi}, \quad m = 1, \dots, n$$

- Escrevendo a função de implementação do algoritmo.

```
> rm(list=ls(all=TRUE))
> Gibbs=function(a,b,c,d,y,n,nburn){
+ # Amostrador de Gibbs para dados Poisson com
+ # mudanca de regime. Prioris Gamma para as medias.
+ #
+ # n: numero de simulacoes
+ # nburn: numero de amostras de aquecimento
+ # a,b,c,d: parametros das prioris Gamma
+ # N: numero de observacoes de Y
+ #
```

<sup>4</sup>... localizar ...



```

+ N=length(y)
+ lambda=matrix(0,nrow=n)
+ phi=matrix(0,nrow=n)
+ m=matrix(0,nrow=n)
+ #
+ # especificacao dos valores iniciais dos parametros
+ lambda[1]=1
+ phi[1]=1
+ m[1]=10
+ #
+ # loop 1
+ for (i in 2:n) {
+   t1=sum(y[1:m[i-1]])
+   t2=0
+   # testa condicao 1
+   if (m[i-1]<N){
+     t2=sum(y[(m[i-1]+1):N])
+     } # fim do teste da condicao 1
+   # gerando 1 valor de lambda da gamma(a+t1;b+m) que
+   # corresponde a distribuicao condicional completa da
+   # posteriori de lambda dado m, phi e y
+   #
+   set.seed(121)
+   lambda[i]=rgamma(1,(a+t1),(b+m[i-1]))
+   # gerando 1 valor de phi da gamma(c+t2;d+N-m) que
+   # corresponde a distribuicao condicional completa da
+   # posteriori de phi dado m, phi e y
+   #
+   set.seed(121)
+   phi[i]=rgamma(1,(c+t2),(d+N-m[i-1]))
+   prob=NULL
+   # loop 2
+   for (j in 1:N) {
+     t1=sum(y[1:j])
+     t2=0
+     # testa condicao 2
+     if (j<N){
+       t2=sum(y[(j+1):N])
+     } # fim do teste da condicao 2
+     # calculando a distribuicao condicional completa da
+     # posteriori de m dado lambda, phi e y
+     aux=(lambda[i]^t1)*exp(-j*lambda[i])*(phi[i]^t2)*exp(-(N-j)*phi[i])
+     # a linha seguinte é a posteriori condicional
+     # completa de m mas nao eh f.d.p.
+     prob=c(prob,aux)
+     } # fim do loop 2
+   soma=sum(prob)
+   # torna m uma distribuicao de probabilidades (f.d.p)
+   probm=prob/soma
+   # amostra um valor de m de um conjunto que varia
+   # de 1 a N, com probabilidade 'probm'
+   m[i]=sample(x=(1:N), size=1, prob=probm)
+   } # fim do loop1
+ # print(round(table(m[nburn+1:n])/(n-nburn),3))
+ return(theta=list(lambda=lambda,phi=phi,m=m))
+ } # fim da funcao Gibbs

```

- Especificando dados de entrada da função Gibbs.

```

> a=0.1
> b=0.1
> c=0.1
> d=0.1
> y=c(rpois(23,2),rpois(17,5)) # simulando o vetor de observações
> n=1000 # numero de iteracoes
> nburn=100

```

- Passando valores para a função.

```
> g = Gibbs(a, b, c, d, y, n, nburn)
```

- Determinando a distribuição de probabilidade dos 1000 valores simulados de  $m$  (ponte de mudança), descontados os 100 primeiros valores de *Burn in* simulados.

```

> #print(round(table(g$m[nburn+1:n])/(n-nburn),3))
> # extraindo os resultados de m 'queimando' os \Sexpr{nburn} primeiros
> foo1=prop.table(table(g$m[nburn+1:n]))
> # convertendo em um data.frame
> foo2=data.frame(m=as.integer(names(foo1)),Freq=as.numeric(foo1))

> library(xtable)
> xtable(foo2, caption = "Tabela de frequências relativas dos valores de corte simulados",
+       "tab:one")

```

	m	Freq
1	14	0.00
2	15	0.00
3	16	0.00
4	17	0.00
5	19	0.01
6	20	0.01
7	21	0.03
8	22	0.47
9	23	0.20
10	24	0.19
11	25	0.03
12	26	0.05
13	27	0.01
14	30	0.00

Tabela 1: Tabela de frequências relativas dos valores de corte simulados

A tabela 1 indica que o valor de  $m=22$  aparece com uma frequência de 46.78% dos resultados, indicando ser o valor do ponto de mudança com maior probabilidade.

- Gerando gráficos das cadeias simuladas, após um período de “aquecimento” (*Burn in*) para o parâmetro  $m$ .

```

> par(mar = c(3.5, 3.5, 0.5, 0.5), mgp = c(2, 0.8, 0), mfrow = c(1,
+   3))
> range = ((nburn + 1):n)
> plot(g$m[range], type = "l", ylab = "m", lwd = 0.5, xlab = "Iteracao")
> acf(g$m[range], main = "", xlab = "defasagem", ylab = "autocorrelações")
> plot(foo1, ylab = "Frequencia relativa", xlab = "Ponto de mudança")

```

- Gerando gráficos das cadeias simuladas, após um período de “aquecimento” (*Burn in*) para o parâmetro  $\lambda$ .

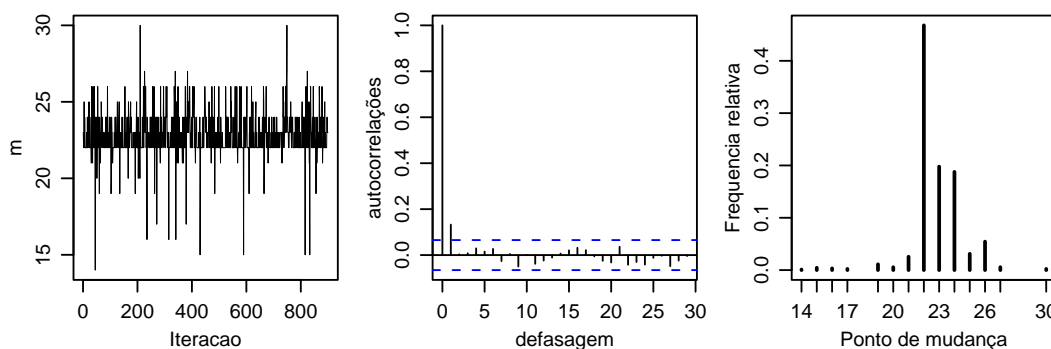


Figura 17: Iterações, autocorrelações e distribuição de frequências relativa do ponto de mudança do processo de Poisson.

```
> par(mar = c(3.5, 3.5, 0.5, 0.5), mgp = c(2, 0.8, 0), mfrow = c(1,
+ 3))
> range = ((nburn + 1):n)
> plot(g$lambda[range], type = "l", ylab = expression(lambda),
+ lwd = 0.5, xlab = "Iteracao")
> acf(g$lambda[range], main = "", xlab = "defasagem", ylab = "autocorrelações")
> plot(density(g$lambda[range]), cex = 0.7, lwd = 0.5, main = "",
+ xlab = expression(lambda))
```

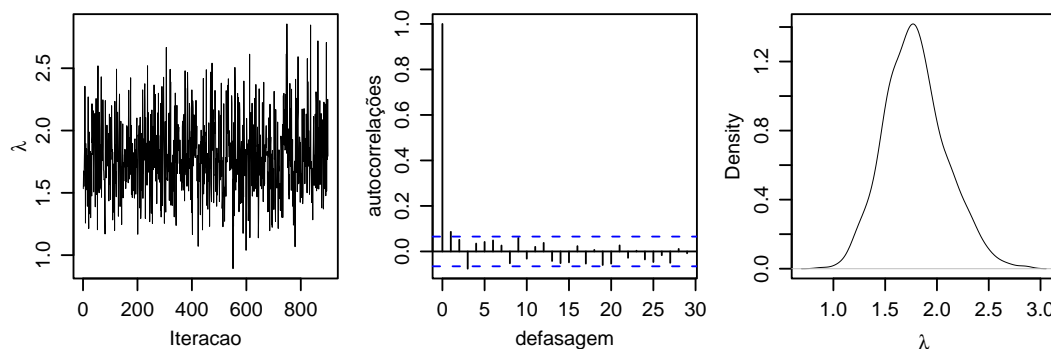


Figura 18: Iterações, autocorrelações e densidade (distribuição de probabilidades) do parâmetro  $\lambda$ .

- Gerando gráficos das cadeias simuladas, após um período de “aquecimento” (*Burn in*) para o parâmetro  $\phi$ .

```
> par(mar = c(3.5, 3.5, 0.5, 0.5), mgp = c(2, 0.8, 0), mfrow = c(1,
+ 3))
> range = ((nburn + 1):n)
> plot(g$phi[range], type = "l", lwd = 0.5, ylab = expression(phi),
+ xlab = "Iteracao")
> acf(g$phi[range], main = "", xlab = "defasagem", ylab = "autocorrelações")
> plot(density(g$phi[range]), cex = 0.7, lwd = 0.5, main = "",
+ xlab = expression(phi))
```

- Comparando  $\phi$  com  $\lambda$  pela diferença entre eles.

```
> par(mar = c(3.5, 3.5, 0.5, 0.5), mgp = c(2, 0.8, 0), mfrow = c(1,
+ 3))
> dif = g$phi[range] - g$lambda[range]
> par(mfrow = c(1, 3))
> plot(dif, type = "l", ylab = expression(phi - lambda), xlab = "Iteracao")
```

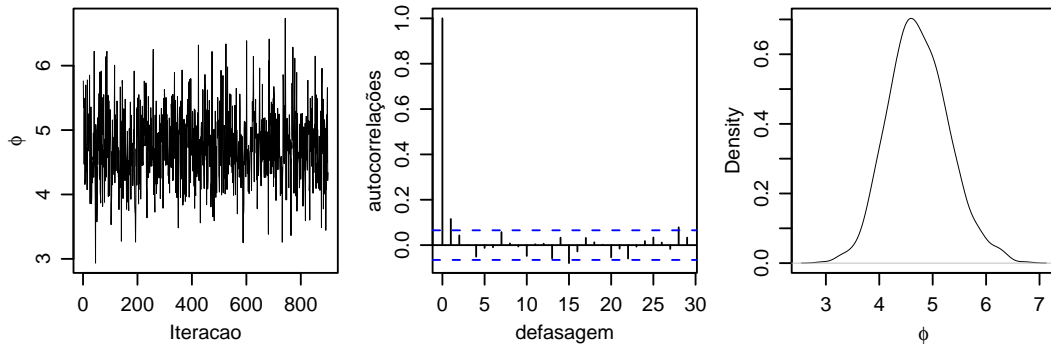


Figura 19: Iterações, autocorrelações e densidade (distribuição de probabilidades) do parâmetro  $\phi$ .

```
> acf(dif, main = "")
> plot(density(dif), main = "", lwd = 0.5, xlab = expression(phi -
+   lambda))
> rug(dif)
```

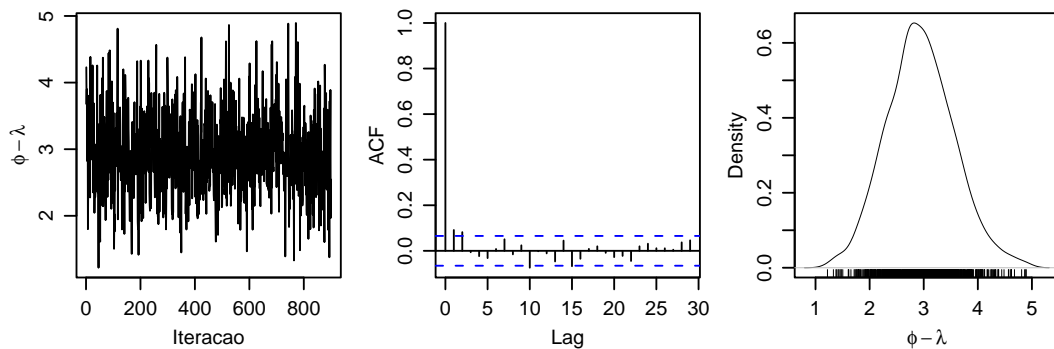


Figura 20: Iterações, autocorrelações e densidade da diferença dos parâmetros  $\phi$  e  $\lambda$ .

- Estimação das contagens médias dos processos de Poisson condicionado nos valores de  $m = 22$  com maior probabilidade a posteriori.

```
> xp = matrix(0, nrow = n - nburn, ncol = 3)
> xp[, 1] = g$lambda[range]
> xp[, 2] = g$phi[range]
> xp[, 3] = g$m[range]
> lambda.22 = xp[, 1][xp[, 3] == max.m]
> phi.22 = xp[, 2][xp[, 3] == max.m]
> par(mfrow = c(1, 2))
> plot(lambda.22, type = "l", ylab = "", xlab = "Iteração")
> plot(phi.22, type = "l", ylab = "", xlab = "Iteração")
```

**Exemplo 10:** Suponha que  $Y_1, \dots, Y_n \sim N(\mu; \sigma^2)$  com  $\mu$  e  $\sigma$  desconhecidos. Defina  $\tau = \sigma^{-2}$  e obtenha a distribuição condicional completa dos parâmetros, a partir de 50 observações de  $Y$  simuladas.

- Obtendo a função de verossimilhança.
- Especificando as prioris independentes:
  - (a)  $\mu \sim N(0; s^2)$ ;
  - (b)  $\tau \sim \text{Gamma}(a; b)$ .

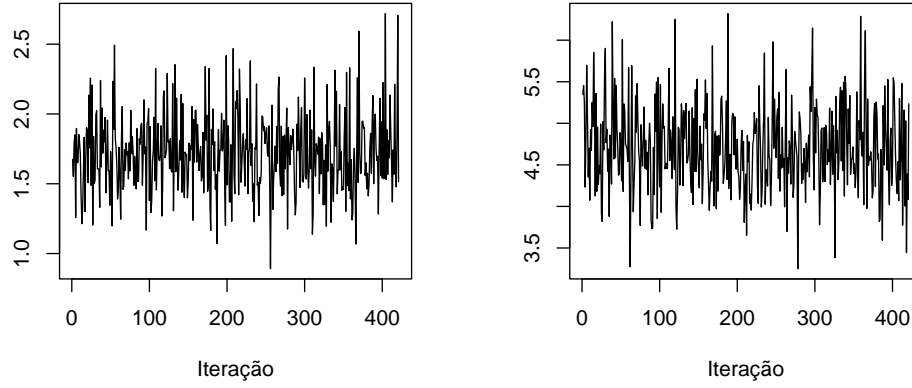


Figura 21: À esquerda, densidade de  $(\lambda|m = 22)$  e a direita, densidade de  $(\phi|m = 22)$ .

- Obtendo a *posteriori* conjunta.

$$\begin{aligned}
 p(\mu; \tau|\mathbf{y}) &\propto p(\mathbf{y}|\mu; \tau)p(\mu)p(\tau) \\
 &\propto \tau^{n/2} \exp\left\{-\frac{\tau}{2} \sum_{t=1}^n (y_t - \mu)^2\right\} \exp\left\{-\frac{\mu^2}{2s^2}\right\} \tau^{a-1} e^{-b\tau}
 \end{aligned}$$

- Calculando as condicionais completas.

**Solução para  $(\mu|\mathbf{y}; \tau)$**

$$\begin{aligned}
 p(\mu|\mathbf{y}; \tau) &\propto p(\mu; \mathbf{y}; \tau) \\
 &\propto p(\mathbf{y}|\mu; \tau) p(\mu; \tau) \\
 &\propto p(\mathbf{y}|\mu; \tau) p(\mu) p(\tau) \\
 &\propto p(\mathbf{y}|\mu; \tau) p(\mu) \\
 &\propto \tau^{n/2} \exp\left\{-\frac{\tau}{2} \sum_{t=1}^n (y_t - \mu)^2\right\} \exp\left\{-\frac{\mu^2}{2s^2}\right\} \\
 &\propto \exp\left\{-\frac{\tau}{2} \sum_{t=1}^n (y_t^2 - 2\mu y_t + \mu^2)\right\} \exp\left\{-\frac{\mu^2}{2s^2}\right\} \\
 &\propto \exp\left\{-\frac{\tau}{2} \left(\sum_{t=1}^n y_t^2 - 2n\mu\bar{y} + n\mu^2\right)\right\} \exp\left\{-\frac{\mu^2}{2s^2}\right\} \\
 &\propto \exp\left\{n\tau\mu\bar{y} - \frac{n\tau\mu^2}{2} - \frac{\mu^2}{2s^2}\right\} \\
 &\propto \exp\left\{-\frac{1}{2} \left[-2n\tau\mu\bar{y} + \left(n\tau + \frac{1}{s^2}\right) \mu^2\right]\right\}
 \end{aligned}$$

fazendo  $C^{-1} = n\tau + \frac{1}{s^2}$  e  $m = C\bar{y}n\tau \Rightarrow \bar{y} = \frac{m}{Cn\tau}$ . Assim:

$$\begin{aligned}
 p(\mu|\mathbf{y}; \tau) &\propto \exp\left\{-\frac{1}{2} \left(-2n\tau\mu \frac{m}{Cn\tau} + \frac{\mu^2}{C}\right)\right\} \\
 &\propto \exp\left\{-\frac{1}{2C} (-2m\mu + \mu^2 + m^2 - m^2)\right\} \\
 &\propto \exp\left\{-\frac{1}{2C} (\mu - m)^2\right\}
 \end{aligned}$$

Logo,  $(\mu|\mathbf{y}; \tau) \sim N(m; C)$

**Solução para  $(\mu|\mathbf{y}; \tau)$**

$$\begin{aligned}
 p(\tau|\mathbf{y}; \mu) &\propto p(\mu; \mathbf{y}; \tau) \\
 &\propto p(\mathbf{y}|\mu; \tau) p(\mu; \tau) \\
 &\propto p(\mathbf{y}|\mu; \tau) p(\mu) p(\tau) \\
 &\propto p(\mathbf{y}|\mu; \tau) p(\tau) \\
 &\propto \tau^{n/2} \exp\left\{-\frac{\tau}{2} \sum_{t=1}^n (y_t - \mu)^2\right\} \tau^{a-1} e^{-b\tau} \\
 &\propto \tau^{(a+n/2)-1} \exp\left\{-\tau \left(b + \frac{1}{2} \sum_{t=1}^n (y_t - \mu)^2\right)\right\}
 \end{aligned}$$

Logo,  $(\tau|\mathbf{y}; \mu) \sim \text{Gamma}\left(a + \frac{n}{2}; b + \frac{1}{2} \sum_{t=1}^n (y_t - \mu)^2\right)$

- Escrevendo a função de implementação do algoritmo.

```

> Gibbs=function(a,b,s2,y,n,nburn){
+ # Amostrador de Gibbs para dados Normais N(mu;sigma2) com
+ # prioris independentes mu ~ Normal(0,s2) e tau ~ gamma(a;b)
+ # tau=inv(sigma2)
+ # n: numero de simulacoes (iteracoes)
+ # nburn: numero de amostras de aquecimento
+ # a,b: parametros da priori Gamma
+ # N: numero de observacoes de Y
+ # s2: Variância das observacoes de Y
+
+ N=length(y)
+ mu.post=numeric(n)
+ tau.post=numeric(n)
+ y.soma=sum(y)
+ y.bar=mean(y)
+ y.2=sum(y^2)
+
+ tau.priori = rgamma(1,a,b)
+ tau.post[1] = tau.priori
+
+ mu.priori = rnorm(1,0,s2)
+ mu.post[1]= mu.priori
+
+ C = 1/(N*tau.post[1] + 1/s2)
+ m = C*y.bar*N*tau.post[1]
+
+ for (i in 2:n) {
+   mu.post[i] = rnorm(1,m,sqrt(C))
+   tau.post[i] = rgamma(1, a + N/2, b + 0.5*(y.2 - 2*y.soma*mu.post[i]+N*mu.post[i]^2))
+   C = 1/(N*tau.post[i] + 1/s2)
+   m = C*y.bar*N*tau.post[i]
+ } #final do loop
+ return(list(mu.post=mu.post,tau.post=tau.post))
+ } # final da funcao

```

Notar que esse algoritmo gera  $\mu^{(t)}|\tau^{(t)}; \mathbf{y}$  e  $\tau^{(t)}|\mu^{(t)}; \mathbf{y}$  onde somente  $\mu(1)$  e  $\tau(1)$  são obtidos das respectivas *priori*. A função implementada toma o primeiro valor da *posteriori* de cada parâmetro como a simulação de um valor a partir de sua respectiva *priori*.

- Especificando os dados de entrada.

```
> # geracao de uma amostra de tamanho 50 de uma distribuição normal de média zero e variância
> y = rnorm(50)
> n = 5000 # numero de iteracoes
> nburn=100 # 'queima' as 100 primeiras simulacoes (aquecimento)
> range=(nburn+1):n # faixa de valores aproveitados
> a=0.1
> b=0.1
> s2=1
```

- Executando a função.

```
> g = Gibbs(a, b, s2, y, n, nburn)
```

- Apresentação gráfica das posterioris completas.

```
> par(mar = c(3.5, 3.5, 0.5, 0.5), mgp = c(2, 0.8, 0), mfrow = c(3,
+ 2))
> plot(g$mu.post[range], type = "l", ylab = expression(mu), xlab = "Iteração")
> plot(g$tau.post[range], type = "l", ylab = expression(tau), xlab = "Iteração")
> acf(g$mu.post[range], main = "", xlab = "Iterações")
> acf(g$tau.post[range], main = "", xlab = "Iterações")
> means = cumsum(g$mu.post[range])/(1:(n - nburn))
> plot(means, type = "l", ylab = expression(bar(mu)), xlab = "Iterações")
> means = cumsum(g$tau.post[range])/(1:(n - nburn))
> plot(means, type = "l", ylab = expression(bar(tau)), xlab = "Iterações")

> linha = c("mu", "tau")
> coluna = c("media", "SD")
> data = round(c(mean(g$mu[range]), sd(g$mu[range]), mean(g$tau[range]),
+ sd(g$tau[range])), 4)
> result = matrix(data, nrow = 2, byrow = T)
> dimnames(result) <- list(linha, coluna)
> xtable(as.data.frame(result), caption = "Estatísticas descritivas",
+ "tab:two")
```

	media	SD
mu	0.13	0.14
tau	1.04	0.21

Tabela 2: Estatísticas descritivas

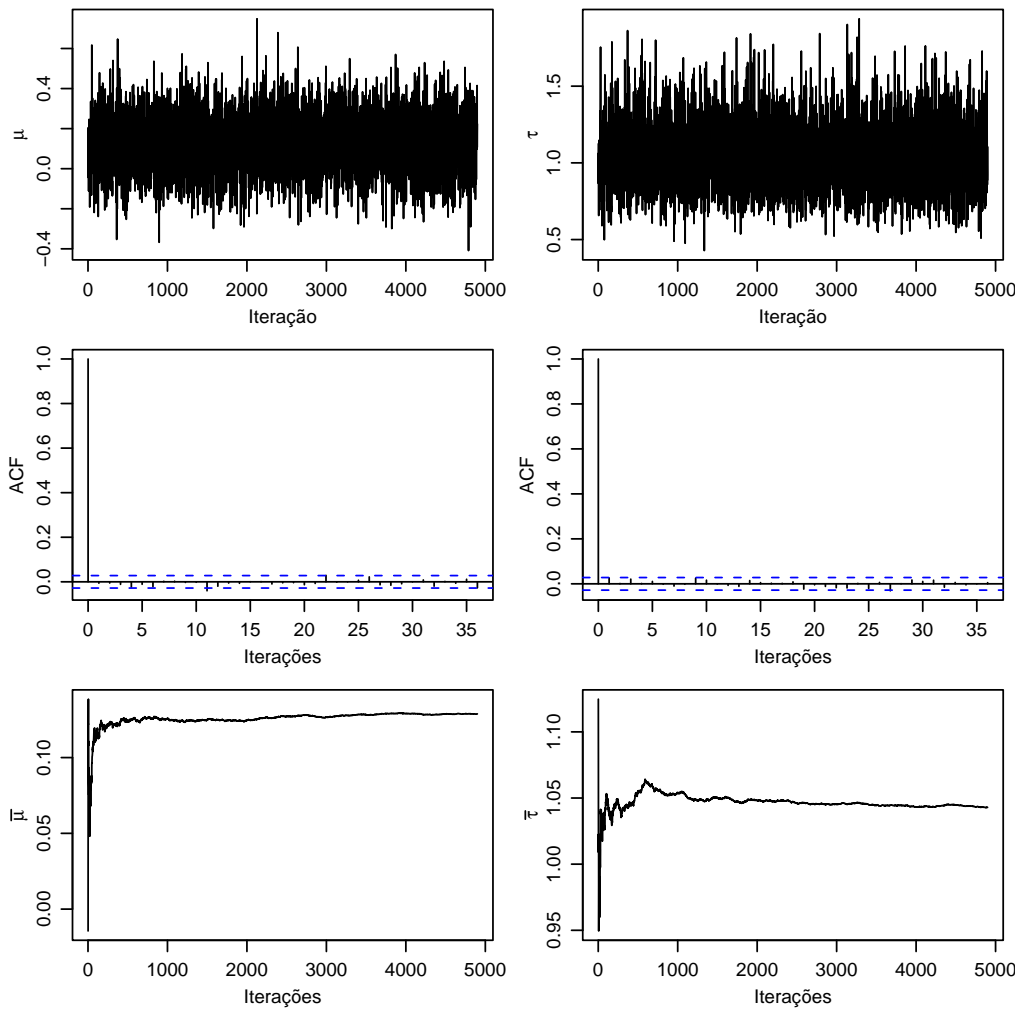


Figura 22: Iteração, autocorrelação e distribuição da média do parâmetro  $\mu$  (esquerda) e do parâmetro  $\tau$  (direita)