Introdução aos Métodos de Monte Carlo Avançados

Adrian Hinojosa Luna
Departamento de Estatística-UFMG

Sumário

Capítulo 1

Introdução

Este manuscrito está baseado nas notas de aula da disciplina, Métodos de Monte Carlo Avançados, disciplina oferecida na Pôs-graduação do Departamento de Estatística. A bibliografia usadas na disciplina:

1. *Advanced Markov chain Monte Carlo methods*, Faming Liang Chuanhai Liu e Raymond Carroll (2010).
2. *Handbook of Markov Chain Monte Carlo.*, Brooks S., et al. (eds.) (2011).

1.1 'Uma historia do MCMC', (Brooks S., 2011).

Vamos a descrever um resumo da história dos métodos MCMC a partir do segundo capitulo do livro *Advanced Markov chain Monte Carlo methods* mencionado acima.

1.1.1 Pré-revolução: 1953 à 1990

Os métodos da cadeia de Markov Monte Carlo (MCMC) existem há quase tanto tempo quanto as técnicas de Monte Carlo, embora seu impacto na estatística não tenha sido sentido até o início dos anos 90. Os algoritmos baseados em Metropolis-Hastings e aqueles relacionados à amostragem de Gibbs, uma vez que cada um deles tem origens radicalmente diferentes, mesmo que sua justificativa matemática, por meio da teoria da cadeias de Markov, seja a mesma. A percepção de que as cadeias de Markov poderiam ser usadas em uma ampla variedade de situações só veio com Gelfand e Smith (1990) Várias razões para isso: falta de desenvolvimento dos computadores (pense nos computadores de 1970!), ou histórico de cadeias de Markov, ou hesitação em confiar na praticidade do método. Assim, foi necessário que pesquisadores visionários como Gelfand e Smith convencessem a comunidade, apoiados por artigos que demonstravam, através de uma série de aplicações, que o método era fácil de entender, fácil de implementar e prático (Gelfand e Smith, 1990). O surgimento rápido do software BUGS (inferência Bayesiana usando amostradores de Gibbs) em 1991, foi outro argumento convincente para adotar, em geral, algoritmos MCMC.

Monte Carlo, MC:

- Métodos Monte Carlo nasceram em Los Alamos, Novo México, durante a Segunda Guerra Mundial, maioria dos físicos que trabalham com física matemática e com a bomba atômica.

- Os métodos de Monte Carlo regulares (somente MC), são geralmente rastreados para Ulam e von Neumann no final da década de 1940. (Eckhardt, 1987) *Stan Ulam, John Von Neumann, and the Monte Carlo method,* descreve esses primeiros desenvolvimentos de Monte Carlo.

- Stanislaw Ulam associa a ideia original a um computo combinatório intratável que ele tentou em 1946: calcular a probabilidade de ganhar no jogo de cartas de paciência.

- Esta ideia foi adotada por John von Neumann para implementação com aplicações diretas à difusão de neutrões, sendo o nome "Monte Carlo" sugerido por Nicholas Metropolis.

- O surgimento do primeiro computador digital, o ENIAC (fevereiro de 1946). O método de Monte Carlo foi criado por von Neumann, que o utilizava em problemas termonucleares e de fissão em 1947. Nesse mesmo ano, Ulam e von Neumann inventaram técnicas de inversão e aceitação-rejeição, para simular amostras de distribuições não uniformes. Note também que, já em 1949, um simpósio sobre Monte Carlo foi apoiado por Rand, o National Bureau of Standards, e o laboratório de Oak Ridge e que Metropolis e Ulam, ver (Metropolis, 1949), publicaram o primeiro artigo sobre o método de Monte Carlo.

Cadeias de Markov e Monte Carlo, MCMC:

- O algoritmo Metropolis data do início dos anos 50, foi desenvolvido pelo mesmo grupo de pesquisadores que produziu o método de Monte Carlo,

- Enquanto os métodos Monte Carlo estavam em uso na época, o MCMC foi trazido para perto da praticidade estatística pelo trabalho de Hastings na década de 1970.

- O que pode ser razoavelmente visto como o primeiro algoritmo MCMC é o que hoje chamamos de algoritmo Metropolis, publicado por Metropolis et al. em 1953, ver (Metropolis, 1949).

Algoritmo Monte Carlo, Metropolis et al.

- O problema foi o cálculo de integrais da forma:

$$\frac{\int F(\theta)\exp\{-E(\theta)/kT\}d\theta}{\int \exp\{-E(\theta)/kT\}d\theta}$$

  onde $\theta \in \mathbb{R}^{2N}$ e N representa o número de partículas.

- A energia $E(\theta)$ destas partículas é:

$$E(\theta) = \frac{1}{2}\sum_i \sum_j V(d_{i,j}).$$

- V é uma *função potencial*, $d_{i,j}$ é a distância entre as partículas i e j.
- A constante normalizadora $Z(\theta)$:

$$Z(\theta) = \int \exp\{-E(\theta)/kT\}d\theta,$$

  só é conhecida em casos simples.

- A *Distribuição de Boltzman*: $\exp\{-E(\theta)/kT\}/Z(\theta)$.
- Usamos a *Lei dos grandes números*, para isso a partir de x_i, y_i procuramos amostrar, independentemente, partículas x_i', y_i':

$$x_i' = x_i + \sigma\varepsilon_{1i}$$
$$y_i' = y_i + \sigma\varepsilon_{2i},$$

  onde $\varepsilon_{1i}, \varepsilon_{2i} \sim U[-1, 1]$,

- Aceitamos x_i', y_i' com probabilidade

$$\min\{1, \exp\{-\Delta E(\theta)/kT\}\}$$

  onde $\Delta E(\theta)$ é a variação da energia $E(\theta)$ entre as duas partículas x_i, y_i e x_i', y_i'.

- Eles mostraram a ergodicidade deste processo (um passeio aleatório).

Hastings (1970)

- Para cadeias de Markov *finitas e reversíveis* e tratando o caso contínuo usando uma analogia de discretização.
- A probabilidade genérica de aceitação para uma mudança do estado i para o estado j é:

$$\alpha_{ij} = \frac{s_{ij}}{1 + \frac{\pi_i}{\pi_j}\frac{q_{ji}}{q_{ij}}},$$

  onde,

  - $s_{ij} = s_{ji} \geq 0$ e tal que $\alpha_{ij} \leq 1$,
  - π_i é a distribuição alvo,
  - q_{ij} é a distribuição proposta.
  - Se $s_{ij} = 1$ então (Barker, 1965).

  - Se $s_{ij} \leq 1$ então (Metropolis, 1949).
  - (Peskum, 1973) mostrou que a escolha ótima é $s_{ij} \leq 1$.

- O *algoritmo* é ergódico, isto é o algoritmo converge a uma única distribuição invariante.

Campos Aleatórios: Amostrador de Gibbs

- O teorema do Hammersley–Clifford (1970), ver (Bremaud, 1999) que providencia uma *condição de 'positividade'* para a representação como campo de Markov dadas as *especificações locais* de tipo Gibbs.

- Besag (1974 e 1986): O uso do teorema de Hammersley–Clifford para construir a distribuição de probabilidade de um *campo de Markov* a partir das *distribuições condicionais*.

- O algoritmo 'Expectation- Maximization, **EM** (Dempster, Laird e Rubin, 1977). e a formulação inicial do **amostrador de Gibbs**: (Gelman e King, 1990) e (Diebolt e Robert, 1994).

- Amostrador de Gibbs: (Geman e Geman, 1984).

1.1.2 Revolução MCMC: Amostrador de Gibbs e Algoritmo de Metropolis - Hastings

- 'Gelfand e Smith escrevem um artigo em 1990, (Gelfand e Smith, 1990): *Sampling based approaches to calculating marginal densities*, que é o genuíno ponto de partida para um uso intensivo de métodos MCMC pela comunidade estatística dominante. Ele provocou novo interesse em métodos Bayesianos, computação estatística, algoritmos e processos estocásticos através do uso de algoritmos computacionais como o amostrador de Gibbs e o algoritmo Metropolis-Hastings'.

- 'Casella e George (1992) escreveram uma introdução elementar ao amostrador de Gibbs que disseminou a técnica para uma comunidade mais ampla, explicando em termos simples por que o algoritmo é válido', ver (Casella e George, 1992).

Amostrador de Gibbs e Algoritmo de Metropolis - Hastings

- Em junho de 1989, em uma oficina Bayesiana em Sherbrooke, Quebec, Smith revelou pela primeira vez as características genéricas do método amostrador de Gibbs: desenvolvimento de amostragem de Gibbs, MCMC, etc.

- A explosão havia começado e, apenas dois anos depois, uma conferência do MCMC na Ohio State University, organizada por Gelfand, Goel e Smith, consistia em três dias inteiros de conversas. Muitas das palestras se tornaram papéis influentes; incluindo (Albert e Chib, 1993), (Gelman e Rubin, 1992), (Geyer, 1992), (Gilks, 1992), (Liu, Wong e Kong, 1994) e (Tierney, 1994).

- Aproximadamente um ano depois, em maio de 1992, houve uma reunião da Royal Statistical Society sobre "O amostrador de Gibbs e outros métodos de Monte Carlo da cadeia de Markov", onde quatro trabalhos foram apresentados seguidos de muita discussão. Os artigos aparecem na primeira edição do Jornal da Royal Statistical Society, Série B, em 1993.

Capítulo 2

Simulação

Simulação de uma variável aleatória consiste em gerar ou amostrar, via algum algoritmo computacional, uma sequencia de observações desta variável. Nesta apostila vamos descrever alguns métodos de simulação mais complexos, usando os métodos MCMC, das ideias presentes nos seguintes algoritmos de simulação mais simples que vamos a apresentar neste capitulo.

Em geral, os métodos de simulação partem da simulação da distribuição uniforme ou, equivalentemente, como a simulação de *números pseudo-aleatórios*. Subsequentemente usamos transformações da uniforme para amostrar de outras variáveis.

2.1 Números Pseudo Aleatórios.

Uma das abordagens mais comuns para gerar números pseudo-aleatórios começa com um valor inicial x_0, chamado de *semente*, e então recursivamente calcula valores sucessivos x_n, x_{n+1}, fazendo,

$$x_{n+1} = a \cdot x_n \text{ modulo } m,$$

onde a e m são inteiros positivos fixos, a notação acima indica que $a \cdot x_n$ é dividido por m e o resto tem o valor de x_{n+1}. Assim, cada x_n pode ser qualquer dos seguintes valores $0, 1, \ldots, m-1$ e a quantidade x_n/m - chamada de um *número pseudo-aleatório* - é tomada como uma aproximação do valor de uma variável aleatória uniforme $U(0,1)$'.
Para um computador de 32 bits as escolhas de $m = 231 - 1$ e $a = 7^5 = 16,807$ resultam em propriedades desejáveis:

1. Para qualquer semente inicial, a sequência resultante tem a "aparência" de ser uma sequência de variáveis aleatórias independentes e com distribuição Uniforme$(0,1)$.

2. Para qualquer semente inicial, o número de variáveis que podem ser geradas antes da finalização é grande.

3. Os valores podem ser calculados eficientemente em um computador.

2.1.1 Números Pseudo Aleatórios para cálculo de integrais

Como foi visto anteriormente, uma das primeiras aplicações de números aleatórios foi no cálculo de integrais, vejamos um exemplo de como é desenvolvido isto. Em geral consideramos $g(x)$ uma função e suponha que nós queremos calcular sua integral θ, onde

$$\theta = \int_0^1 g(x)dx.$$

Para calcular o valor de θ, note que se $U \sim U(0,1)$, então podemos expressar θ como:

$$\theta = \mathbb{E}[g(U)]$$

Se $U_1, \ldots, U_k$ são variáveis aleatórias uniformes, $U(0,1)$, independentes, logo as variáveis aleatórias $g(U_1), \ldots, g(U_k)$ também são independentes, identicamente distribuídas e com média θ.
Pela *lei forte de grandes números* com probabilidade um vale o seguinte resultado:

$$\frac{1}{k}\sum_{i=1}^{k} g(U_i) \to \mathbb{E}[g(U)], \text{ quando } k \to \infty.$$

Observe que se nos quiséssemos computar essa integral com outros limites,

$$\theta = \int_a^b g(x)dx,$$

então, fazendo a substituição $y = (x-a)/(b-a)$, $dy = dx/(b-a)$, temos que

$$\begin{aligned}\theta &= \int_0^1 g(a + [b-a]y)(b-a)dy \\ &= \int_0^1 h(y)dy,\end{aligned}$$

onde $h(y) = (b-a)g(a+[b-a]y)$.
Finalmente, no caso $\theta = \int_0^\infty g(x)dx$, basta fazer $y = 1/(x+1)$.

Exemplo 1 (cálculo de π)**.** *Podemos aproximar o valor de π observando que se $(X,Y) \sim U[-1,1] \times [-1,1]$, i.e. uniforme num quadrado de lado 2 (área=4), e seja C_1 o círculo de raio um, então:*

$$P\left((X,Y) \in C_1\right) = \frac{\textit{area do circulo } C_1}{\textit{area do quadrado}} = \frac{\pi}{4},$$

defina:

$$I_{C_1}(x,y) = \begin{cases} 1 & (x,y) \in C_1, \\ 0 & (x,y) \notin C_1. \end{cases}$$

Então:

$$\mathbb{E}I_{C_1}(X,Y) = P\left((X,Y) \in C_1\right) = \frac{\pi}{4},$$

e aplicando Monte Carlo obtemos a aproximação:

$$\pi \approx 4 \times \left(\frac{1}{k} \sum_{i=1} I_{C_1}(X_i, Y_i) \right).$$

Usando R ,

```
k<-10000 # 10000 observações da uniforme
x<-runif(k, min=-1,max=1)
y<-runif(k, min=-1,max=1)
xy<-list(x=x,y=y)
C1<-0 # C1 conta os pontos dentro do circulo
for (i in 1:k) {
 if(sqrt((xy$x[i])^2+(xy$y[i])^2)<=1)
 {C1=C1+1}
}
Pi<-4*(C1/k)
```

obtemos como resultado $Pi \approx 3.1404$

2.2 Teorema da função inversa

Teorema 1. *Seja U uma variável aleatória $U(0,1)$. Para qualquer função de distribuição contínua F, a variável aleatória X definida por $X = F^{-1}(U)$ tem distribuição F.*
Observe que $F^{-1}(u)$, a 'inversa' de F é definida pelo valor x tal que $F(x) = u$.

Demonstração. Seja F_X a função de distribuição da v.a. X: $F_X(x) = P(X \leq x)$, então:

$$\begin{aligned} F_X(x) &= P(X \leq x) \\ &= P(F^{-1}(U) \leq x) \\ &= P(F(F^{-1}(U)) \leq F(x)) \\ &= P(U \leq F(x)) \\ &= F(x). \end{aligned}$$

□

Exemplo 2. *Se X é uma variável aleatória exponencial com taxa 1, então sua função de distribuição é dada por*

$$F(x) = 1 - e^{-x}.$$

Se fazemos $x = F^{-1}(u)$, então,

$$u = F(x) = 1 - e^{-x},$$

ou

$$1 - u = e^{-x},$$

ou, tomando logaritmos,

$$x = -\log(1 - u).$$

Portanto, podemos gerar um exponencial com o parâmetro 1 gerando um número aleatório U e, em seguida, definindo $X = F^{-1}(U) = -\log(1 - U)$.

2.3 Aceitação-Rejeição, para variáveis discretas

Suponha que tenhamos um método eficiente para simular uma variável aleatória discreta Y com função de massa de probabilidade $\{q_j, j \geq 0\}$.

Podemos usar a simulação da v.a. Y como base para simular a v.a. X função de massa de probabilidade $\{p_j, j \geq 0\}$; primeiro simulando uma variável aleatória Y, tendo função de massa $\{q_j\}$, e então aceitando este valor simulado com uma probabilidade proporcional a p_Y/q_Y.

Especificamente, seja c uma constante tal que $\frac{p_j}{q_j} \leq c$ para todo j, e tal que $p_j \geq 0$.

Método de rejeição ou método de aceitação-rejeição.
Para simular uma variável aleatória X com função de massa $p_j = P(X = j)$ consiste nos seguintes passos,

Passo 1. Simule o valor de Y, tendo função de massa de probabilidade q_j.

Passo 2. Amostre um número aleatório U.

Passo 3. Se $U < p_X/cq_Y$, defina $X = Y$ e pare. Caso contrário, retorne ao passo 1.

Teorema 2. *Algoritmo aceitação-rejeição gera uma variável aleatória X tal que $P(X = j) = p_j, j = 0, \ldots$. Além disso, o número de iterações do algoritmo necessário para obter X é uma variável aleatória geométrica com média c.*

Demonstração. Para começar, vamos determinar a probabilidade de que uma única iteração produza o valor aceito j. Primeiro note que

$$\begin{aligned} P(Y = j, \text{é aceito}) &= P(Y = j)P(aceita|Y = j) \\ &= q_j \frac{cp_j}{q_j} \\ &= \frac{p_j}{c}. \end{aligned}$$

A soma sobre j gera a probabilidade de que uma variável aleatória gerada seja aceita:

$$P(aceito) = \sum_j \frac{p_j}{c} = \frac{1}{c}.$$

Como cada iteração resulta independentemente em um valor aceito com probabilidade $1/c$, vemos que o número de iterações necessárias é geométrica com a média c.

Além disso,

$$\begin{aligned} P(X=j) &= \sum_n P(\text{ ser aceito na iteração } n) \\ &\sum_n (1-1/c)^{n-1}\frac{p_j}{c} \\ &= p_j. \end{aligned}$$

□

Exemplo 3. *Vamos simular, usando o método da aceitação-rejeição,* $n = 1000$ *observações da v.a.* X *que corresponde ao lançamento de* $m = 3$ *moedas com probabilidade de sair cara* $p = 0.2$*. Vamos a usar a simulação da v.a.* Y *que corresponde ao lançamento de três moedas 'honestas', com probabilidade de sair cara* $q = 0.5$*. Ambos experimentos são binomiais.* $b(m,p)$ *e* $b(m,q)$*.*

```
m=3
q=0.5
p=0.2
dist_X=dbinom(0:3,m,p)# distribuicao da massa de X
dist_Y=dbinom(0:3,m,q)# distribuicao da massa de Y
c<-max(dist_X/dist_Y)
n=1000# 10 observações do lançãmento de uma moeda
x=c()
for (i in 1:n) {
  cd=0
  while (cd==0) {
   y<-rbinom(1,m,p)
   u<-runif(1)
   if(u<(dist_X[y+1]/(c*dist_Y[y+1])))
   {x[i]=y
      cd=1}
  }
}
hist(x)
```

obtemos a figura 2.1

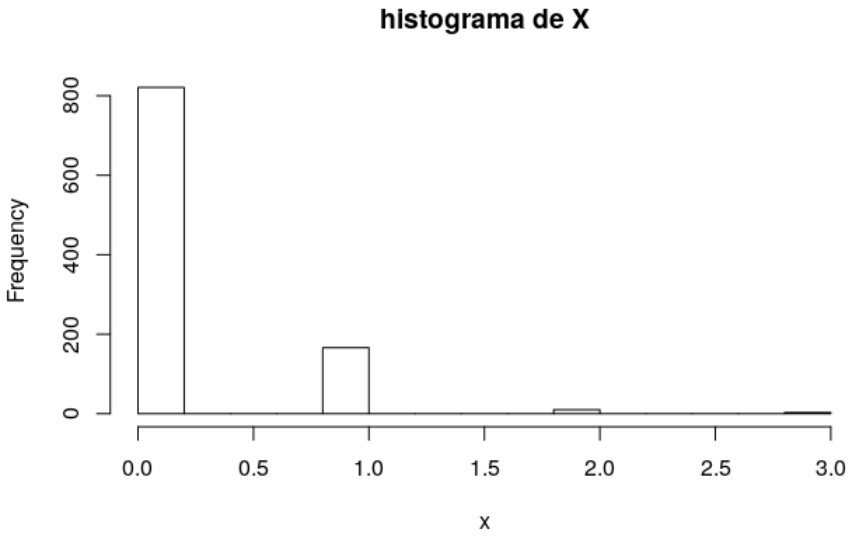

Figura 2.1: Histograma de $n = 1000$ observações da v.a. X com distribuição exata $p = (0.512, 0.384, 0.096, 0.008)$

Capítulo 3

Métodos de Cadeias de Markov e Monte Carlo

3.1 Métodos de Monte Carlo

Um problema desafiador comumente encontrado é avaliar integrais da forma,

$$\mathbb{E}_\nu[h(X)] = \int h(x)\nu(dx) = \int h(x)f(x)dx, \tag{3.1}$$

onde $X \sim \nu$ tem densidade $f(x)$. Por exemplo:

1. Calcular a probabilidade do evento A: $P(X \in A) = \mathbb{E}_\nu I_A(X)$, onde I_A é a função indicadora:

$$I_A(x) = 1 \text{ se } x \in A, \text{ e } I_A(x) = 0 \text{ se } x \notin A.$$

2. Calcular a distribuição marginal de $f_Y(y)$ da distribuição conjunta $f_{(X,Y)}(x, y)$, a representação como integral deste problema é:

$$\mathbb{E}_{f_X}[f_{Y|X}(y)],$$

onde $f_X(x)$ é a distribuição marginal de X e $f_{Y|X}$ é a densidade condicional de Y dado X.

3.2 Aproximação de Monte Carlo

A *aproximação de Monte Carlo* pode ser descrita do seguinte modo, suponha que seja fácil simular uma amostra de tamanho n, denotada por $X_1, \ldots, X_n$, da v.a. X com densidade $f(x)$. Então, para uma função real h, a média amostral de $h(X)$ é

$$\bar{h}_n := \frac{1}{n}\sum_{i=1}^{n} h(X_i),$$

pela lei dos grandes números ela converge quase certamente para $\mathbb{E}_\nu[h(X)]$.

Quando $h(X)$ tem uma variância finita, pelo teorema do limite central, o erro desta aproximação pode ser caracterizado por:

$$\frac{\bar{h}_n - \mathbb{E}_f[h(X)]}{\sqrt{n\text{Var}(h(X))}} \sim N(0,1).$$

O termo de variância $\text{Var}(h(X))$ pode ser aproximado pela variância da amostra:

$$\frac{1}{n-1}\sum_{i=1}^{n}(h(X_i) - \bar{h}_n)^2.$$

3.3 Monte Carlo via amostragem de importância

Quando é difícil extrair amostras de $f(x)$ diretamente, pode-se recorrer à *amostragem por importância*, que é desenvolvida com base na seguinte identidade,

$$\mathbb{E}_f[h(X)] = \int h(x)f(x)dx = \int h(x)\frac{f(x)}{g(x)}g(x)dx = \mathbb{E}_g[h(X)f(X)/g(X)],$$

onde a densidade $g(x)$ é positiva para cada x em que $f(x)$ é positivo. Esta identidade sugere que amostras de funções de densidade diferentes de $f(x)$ também podem ser usadas para aproximar. A teoria padrão de Monte Carlo é aplicada aqui pois:

$$\mathbb{E}_f[h(X)] = \mathbb{E}_g[h(X)f(X)/g(X)] = \mathbb{E}_g[\tilde{h}(X)],$$

onde $\tilde{h}(x) = h(x)f(x)/g(x)$.

O estimador de $\mathbb{E}_f[h(X)]$ se torna:

$$\bar{h} = \frac{1}{m}\sum_{i=1}^{m}\frac{f(x_i)}{g(x_i)}h(x_i) = \frac{1}{m}\sum_{i=1}^{m}\omega_i h(x_i),$$

onde $x_1, \ldots, x_n$ são amostras da densidade $g(x)$. Para cada $i = 1, \ldots, m$, x_i entra com um peso $\omega_i = f(x_i)/g(x_i)$. Por esse motivo, esse método é chamado de *método de amostragem por importância.*

Teorema 3. *A escolha de g que minimiza a variância do estimador por amostragem por importância de* $\mathbb{E}_f[h(X)]$ *é*

$$g^*(x) = \frac{|h(x)|f(x)}{\int |h(x)|f(x)dx}.$$

Observe que a constante normalizadora é, praticamente, a própria quantia a estimar: $\mathbb{E}_f[|h(X)|]$.

Exercício 1. *Mostre o teorema anterior, ver (Christian Robert, 2010).*

Exemplo 4. *Considere* $X \sim$ *exp*(2). *Observe que* $\mathbb{E}X = 1/2$ *e* $VarX = \mathbb{E}X^2 - (\mathbb{E}X)^2 = 1/4$, *logo* $\mathbb{E}X^2 = 1/4 + 1/4 = 1/2$

1. *Ache* $\mathbb{E}X^2$ *usando a aproximação Monte Carlo com* $m = 100, 1000$. *Encontre a variância do estimador em ambos casos.*

```
m=100
#m=1000
x<-rexp(m,rate = 2)
EX2=(1/m)*sum(x)
EX2
Var=(1/(m-1))*sum((x^2-EX2)^2)
Var
sd=sqrt(Var)
sd
```

Obtemos:

n	$\mathbb{E}(X^2)$	$Var(X)$
100	*0.5390*	*0.7282*
1000	*0.5052*	*1.4605*

2. *Ache* $\mathbb{E}X^2$ *usando a aproximação Monte Carlo e amostragem por importância com* $g(x) = e^{-x}, x > 0$ *a densidade da exponencial* $m = 10, 100$. *Encontre a variância do estimador em ambos casos.*
Observe que $w(x) = \frac{f(x)}{g(x)} = \frac{2e^{-2x}}{e^{-x}} = 2e^{-x}$

```
m=100
#m=1000
y<-rexp(m,rate = 1) # y tem densidade exp(1)
# peso w(u)=2exp(-u)
w<-function(u)
{v<-2*exp(-u)
return(v)}
EisX2=(1/m)*sum(y*w(y))
EisX2
var=(1/(m-1))*sum({(y^2-EisX2)*w(y)}^2)
var
sd=sqrt(var)
sd
```

Obtemos:

n	$\mathbb{E}(X^2)$	$Var(X)$
100	*0.4609*	*0.4796*
1000	*0.5082*	*0.4250*

3.4 Monte Carlo Markov Chain, MCMC

3.4.1 Cadeia de Markov

Definição 1. *Uma cadeia de Markov é uma sequência de variáveis aleatórias* $\{X_i : i = 0, 1, 2, \dots\}$ *com a 'propriedade de Markov' que diz que dado o estado presente, os estados futuro e passado são independentes, ou seja, para todos os conjuntos 'mensuráveis' A,*

$$P(X_{t+1} \in A | X_0 = x_0, \dots, X_t = x_t) = P(X_{t+1} \in A | X_t = x_t) \tag{3.2}$$

vale para os tempos $t = 0, 1, \dots$

Escrevemos $P_t(dx)$ para a distribuição marginal de X_t no tempo t. Começando com a distribuição $P_0(dx)$, chamada de distribuição inicial, a cadeia de Markov X_t evolui de acordo com:

$$P_{t+1}(dy) = \int P_t(dx) P_t(x, dy).$$

A distribuição $P_t(x, dy)$ é a *distribuição de probabilidade* para X_{t+1} dado $X_t = x$, chamado de distribuição do *kernel de transição* no tempo t. Na prática, esta é a função de densidade condicional de X_{t+1} dado $X_t = x$.

3.4.2 Cadeia de Markov, Condição de equilíbrio

Uma classe importante de cadeias de Markov comumente usadas no MCMC é a classe de cadeias de Markov *homogêneas no tempo* onde

$$P_t(x, dy) = P(x, dy) \text{ vale para } t = 1, 2, \dots$$

Neste caso, $P_{t+1}(dy) = \int P_t(dx) P(x, dy)$ e $P_t(dx)$ é determinado unicamente pela distribuição inicial $P_0(dx)$ e o núcleo de transição $P(x, dy)$.
Por esta razão, escrevemos $P^n(x, \cdot)$ para a distribuição condicional de X_{t_0+n} dado $X_{t_0} = x$.
A ideia básica do MCMC é criar cadeias de Markov para aproximar $\mathbb{E}_\pi(h(X))$ é construir um núcleo de transição $P(x, dy)$ com $\pi(dx)$ como sua distribuição invariante, isto é, $P(x, dy)$ e $\pi(dx)$ satisfaz a *condição de equilíbrio*:

$$\pi(dy) = \int \pi(dx) P(x, dy).$$

Quando a *distribuição alvo* tem a densidade $f(x)$ e o kernel de transição $P(x, dy)$ tem a densidade condicional $p(y|x)$, esta condição de equilíbrio pode ser escrita como

$$f(y) = \int p(y|x) f(x) dx.$$

Esta condição de equilíbrio pode ser vista como obtida exigindo $P_{t+1}(dx) = P_t(dx) = \pi(dx)$.

Ele diz que se X_t é amostrada do alvo $\pi(x)$ então X_{t+1} também é amostrada, possivelmente dependente de X_t, de $\pi(x)$.
Além disso, para quase qualquer $P_0(dx)$ sob condições moderadas, $P_t(dx)$ converge para $\pi(dx)$.
Se para π-quase todo x, $\lim_{t\to\infty} P(X_t \in A|X_0 = x) = \pi(A)$ vale para todos os conjuntos *mensuráveis* A, $\pi(dx)$ é chamado de *distribuição de equilíbrio da cadeia de Markov*.

3.4.3 Cadeia de Markov, Convergência

Exceto em casos raros em que é satisfatório ter uma amostra pequena da distribuição alvo $f(x)$, a maioria das aplicações o MCMC fornece aproximações às integrais de funções respeito a densidade $f(x)$, que podem ser representadas da forma

$$\mathbb{E}_f(h) = \int h(x)f(x)dx.$$

Assumindo $\mathbb{E}_f(|h|) < \infty$, isso deve ser aproximado por

$$\bar{h}_{m,n} = \frac{1}{n}\sum_{i=1}^{n} h(x_{i+m}),$$

onde X_i denota uma cadeia de Markov simulada e m é um inteiro não negativo denotando o comprimento do que é chamado de período de *burn-in*.

3.4.4 Cadeia de Markov, Notação e Definições

Os conceitos mais importantes são a *irredutibilidade* e a *aperiodicidade*.

- Uma cadeia de Markov com distribuição invariante $\pi(dx)$ é irredutível se para qualquer estado inicial, a cadeia tiver probabilidade positiva de atingir qualquer conjunto ao qual $\pi(dx)$ atribua probabilidade positiva.
- Uma cadeia é periódica se houver partes do espaço de estados que só pode ser visitado em certos *períodos regularmente* espaçados; caso contrário, a cadeia é aperiódica.

OBS: Se uma cadeia tem uma *distribuição invariante* apropriada $\pi(dx)$ e é *irredutível e aperiódica*, então $\pi(dx)$ é a *única* distribuição invariante e é também a distribuição de equilíbrio da cadeia.

3.4.5 Cadeia de Markov, Recorrência

Definição 2. *Seja X_n uma cadeia π-irredutível com distribuição invariante $\pi(\cdot)$ e usando a notação $\{A_n$ i.o.$\}$ que significa que a sequência ocorre infinitamente muitas vezes, isto é, $\sum_i I_{A_i} = \infty$ (com probabilidade um).*

1. *A cadeia é recorrente se, para todo B com $\pi(B) > 0$,*

$$P(X_n \in B \text{ } i.o. | X_0 = x) > 0 \text{ } para \text{ } todo \text{ } x$$

e

$$P(X_n \in B \text{ } i.o. | X_0 = x) = 1 \text{ } para \text{ } \pi - quase \text{ } todo \text{ } x.$$

2. *A cadeia é Harris recorrente se*

$$P(X_n \in B \text{ } i.o. | X_0 = x) = 1 \text{ } para \text{ } todo \text{ } x.$$

3.4.6 Cadeia de Markov, Ergodicidade

Definição 3. *A distância de variação total entre duas medidas em é definida pela norma de variação total de uma medida com sinal λ:*

$$\|\lambda\| = \sup_A \lambda(A) - \inf_A \lambda(A).$$

O tempo de chegada no subconjunto B é a variável aleatória

$$H_B = \inf\{t \geq 0 : X_t \in B\}$$

onde o ínfimo do conjunto vazio é considerado como sendo ∞.

Definição 4. *Ergodicidade.*

a) *Diz-se que uma cadeia de Markov é ergódica se for Harris recorrente positiva e aperiódica.*

b) *Seja H_B o tempo de chegada para o conjunto B. Uma cadeia ergódica com distribuição invariante $\pi(x)$ é dita ergódica de grau 2 se*

$$\int_B \mathbb{E}_x (H_B)^2 \pi(dx) < \infty,$$

vale para todos os B com $\pi(B) > 0$.

c) *Uma cadeia ergódica com distribuição invariante $\pi(x)$ é dita geometricamente ergódica se existir uma função real não-negativa estendida M com $\mathbb{E}(|M(X)|) < \infty$ e uma constante positiva $r < 1$ tal que*

$$\|P^n(x, \cdot) - \pi\| \leq M(x) r^n,$$

para todo x.

d) *Diz-se que a cadeia em (c) é uniformemente ergódica se existir uma constante M e uma constante positiva $r < 1$ tal que*

$$\|P^n(x, \cdot) - \pi\| \leq M r^n.$$

3.4.7 Cadeia de Markov, Convergência

Teorema 4. *Suponha que $P(x, dy)$ seja π-irredutível e π-invariante. Então, $P(x, dy)$ é recorrente positivo e $\pi(dx)$ é a única distribuição invariante de $P(x, dy)$.*
Se $P(x, dy)$ também é aperiódica, então para π-quase todo x,

$$\|P^n(x, \cdot) - \pi\| \to 0,$$

com $\|\cdot\|$ denotando a distância total da variação.
Se $P(x, dy)$ é Harris recorrente, então a convergência ocorre para todo x.

Observe que vale a reciproca: $\|P^n(x, \cdot) - \pi\| \to 0$ implica a ergodicidade da cadeia.

3.4.8 Cadeia de Markov, Convergência das medias

Teorema 5. *Suponha que X_n seja ergódico com distribuição de equilíbrio $f(x)$ e suponha que $h(x)$ tenha valor real e $\mathbb{E}_f(|h(X)|) < \infty$. Então, para qualquer distribuição inicial, $\bar{h}_n \to \mathbb{E}_f(h(X))$ quase certamente.*

Teorema 6. *Suponha que X_n seja ergódico de grau 2 com com distribuição de equilíbrio $f(x)$ e suponha que $h(x)$ seja de valor real e limitado. Então, existe um número real σ_h tal que a distribuição de*

$$\sqrt{n}(\bar{h}_n - \mathbb{E}_f(h(X)),$$

converge fracamente para uma distribuição normal com média 0 e variância σ_h^2 para qualquer distribuição inicial.

Teorema 7. *Suponha que X_n é uniformemente ergódico com distribuição de equilíbrio $f(x)$ e suponha que $h(x)$ tenha valor real e $\mathbb{E}_f(h^2(X)) < \infty$. Então existe um número real σ_h tal que a distribuição de*

$$\sqrt{n}(\bar{h}_n - \mathbb{E}_f(h(X)))$$

converge fracamente para uma distribuição normal com média 0 e variância σ_h^2 para qualquer distribuição inicial.

3.5 Aplicação 1: O estimador de máxima verosimilhança aproximado por Monte Carlo (MCEMV)

Suponha que tenhamos uma família de funções não negativas $\{h_\theta, \theta \in \Theta\}$ que sejam integraveis,

$$c(\theta) = \int h_\theta(u)du < \infty, \tag{3.3}$$

$c(\theta)$ é chamada constante normalizadora, e h_θ são as *densidades não normalizadas.*

Para cada $\theta \in \Theta$ defina a *densidade* f_θ como:

$$f_\theta = \frac{h_\theta(u)}{c(\theta)}. \tag{3.4}$$

Observe que a equação (3.3) implica que $\int f_\theta(u)du = 1$ logo f_θ definida por (3.4) é uma densidade. Suponha que não sabemos como calcular esta constante para todo θ, exceto para um valor ψ.

Considere $\{h_\theta(u) = e^{-\theta u}, \theta \in \Theta = \mathbb{R}^+\}$. Nesse caso temos que

$$c(\theta) = \int h_\theta(u)du = \int_{\mathbb{R}^+} e^{-\theta u}du = 1/\theta.$$

Considere $\psi = 1$ logo $c(\psi) = 1$.

Fixando um parâmetro ψ, e para uma observação x considere ℓ a log-verosimilhança de f_θ (ração da log-verosimilhança):

$$\begin{aligned}\ell(\theta) &= \log \frac{h_\theta(x)}{h_\psi(x)} - \log \frac{c(\theta)}{c(\psi)} \\ &= \log \frac{h_\theta(x)}{h_\psi(x)} - \log \mathbb{E}_\psi \frac{h_\theta(X)}{h_\psi(X)}.\end{aligned} \tag{3.5}$$

Pois

$$\mathbb{E}_\psi \frac{h_\theta(X)}{h_\psi(X)} = \int \frac{h_\theta(u)}{h_\psi(u)} f_\psi(u)du = \int \frac{h_\theta(u)}{h_\psi(u)} \frac{h_\psi(u)}{c(\psi)}du = \frac{c(\theta)}{c(\psi)}.$$

Observe que no caso de uma amostra $\underset{\sim}{x} = (x_1, \ldots, x_m)$, temos

$$\ell(\theta) = \sum_{i=1}^{m} \log \frac{h_\theta(x_i)}{h_\psi(x_i)} - n \log \frac{c(\theta)}{c(\psi)}.$$

Simulando uma amostra $y_1, \ldots, y_n$ de tamanho n de $X \sim f_\psi$ podemos escrever

$$\log \frac{h_\theta(x)}{h_\psi(x)} - \log \frac{c(\theta)}{c(\psi)} = \frac{1}{n} \sum_{j=1}^{n} \frac{h_\theta(y_j)}{h_\psi(y_j)}.$$

Pela lei dos grandes números para n suficientemente grande podemos escrever

$$\mathbb{E}_{n,\psi} \approx \mathbb{E}_\psi \frac{h_\theta(X)}{h_\psi(X)}.$$

Definimos a **log-verosimilhança empírica** $\hat{\ell}_n$ como

$$\hat{\ell}_n(\theta) = \log \frac{h_\theta(x)}{h_\psi(x)} - \log \mathbb{E}_{n,\psi}. \tag{3.6}$$

No caso de observar uma amostra $\underset{\sim}{x} = (x_1, \ldots, x_m)$, temos

$$\hat{\ell}_n(\theta) = \sum_{i=1}^{m} \log \frac{h_\theta(x_i)}{h_\psi(x_i)} - n \log \mathbb{E}_{n,\psi}.$$

Seja $\hat{\theta}_n$ o valor que maximiza a eq. (3.6) então vale que $\lim_{n\to\infty} \hat{\theta}_n = \theta$, onde θ é o máximo da eq. (3.5). Para a prova ver *(Geyer, 1994): "On the Convergence of Monte Carlo Maximum Likelihood Calculations". 56 (1), pp. 261–274.*

Exemplo 5 (MCEMV)**.** *A distribuição normal pode ser escrita como:*

$$f(x) = \frac{e^{-(1/2)(x-\mu_0)^2}}{c}, \quad x \in \mathbb{R},$$

com $c = \sqrt{2\pi}$*. Suponha que sabemos que* $\mu_1 = 0$ *então* $c_1 = \sqrt{2\pi}$*.Vamos usar este parâmetro* $\mu_1 = 0$ *para aproximar via Monte Carlo a log-verosimilhança* $\ell_{MC}(\mu)$ *do problema.Para isso precisamos simular valores de* $\tilde{X} \sim N(\mu_1, 1)$ *para achar* $\ell_{MC}()$*.Suponha que vamos maximizar* $\ell_{MC}(\mu_i)$ *no conjunto* $\mu_i = (i-200)/100 \in [-2, 2], i = 0, \ldots, 400$*. Este parâmetro é nosso candidato para MCEMV.*

Monte Carlo MC *Simule uma amostra de tamanho* $N = 500$ *de observações* $x_1, \ldots, x_N$ *da* $N(\mu_1, 1)$ *e use elas para achar* $\ell_{MC}(\cdot)$*. Usando esta função ache o MCEMV.*

```
h_m <- function(x, m)
  {
  return((exp(-0.5*(x-m)^2) ))
}
mu<-seq(-2,2, by=0.01)
x_obs <- rnorm(100, 1, 1) #mu=1
x_sim <- rnorm(1000, 0, 1) #mu=0

logverMC <-function(u,x,y)
{ for (i in 1:length(x)) {
  A<-log(h_m(x[i],u)/h_m(x[i],0))}
  A<--sum(A)
  for (i in 1:length(y)) {
  B<-(h_m(y[i],u)/h_m(y[i],0))}
  B<-(length(y))*log(mean(B))
  return(A+B)
}
C<-c()
for (i in 1:length(mu)) {
  C[i]<-logverMC(mu[i],x_obs,x_sim)
}
mu_est<-mu[which.max(C)]
 mu_est
[1] 1.18
plot(mu, C, ylab = "MC-Logverossimilhança",
xlab = expression("mu"), main = "MCEMV")
```

Cadeia de Markov Monte Carlo MCMC *Vamos a usar uma cadeia de Markov que tem distribuição invariante uma distribuição normal, assim simulando esta*

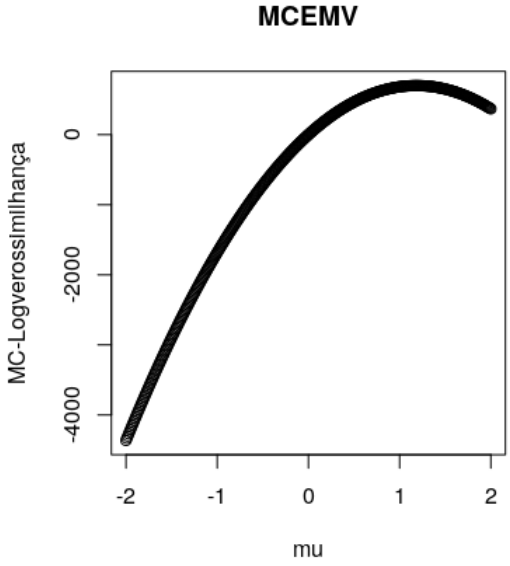

Figura 3.1: MC-logverosimilhança

cadeia por um tempo suficientemente longo obteríamos uma amostra da distribuição normal e com ela aproximamos $\ell_{MC}(\cdot)$.

Sabe-se que uma serie temporal $x_t = \rho x_{t-1} + N(0, \sigma^2)$, *do tipo AR(1), é ergodica e tem como distribuição estacionaria uma normal* $\pi \sim N(0, \frac{\sigma^2}{1-\rho^2})$ *se* $|\rho| < 1$. *Fazendo* $\sigma^2 = 3/4$ *e* $\rho = 1/2$ *temos que* $\pi \sim N(0,1)$. *Este processo de Markov tem como distribuição estacionaria uma distribuição normal* ($N(0,1)$). *Vamos a usar este fato para criar a cadeia de Markov que precisamos. Para isso considere a cadeia de Markov definida por:*

$$x_0 = 0$$

e

$$x_{t+1} = \frac{1}{2}x_t + N(0, 3/4).$$

a) Simule a cadeia até o instante $t = 1000$, *e considerando as observações* $x_{501}, \ldots, x_{1000}$ *e mostre que o q-q plot desta amostra esta próximo da distribuição Normal com parâmetro* $\mu_1 = 0$.

```
   x_s<-rep(0,1000)
x_s[1]<-0 #Valor inicial
for(i in 2:1000){
  x_s[i]<-(1/2)*x_s[i-1]+rnorm(1,0,3/4)

}
x_sim <- x_s[501:1000]
```

```
qqnorm(x_sim); qqline(x_sim, col = 2, xlab="quantis teoricos",
                      ylab="quantis observados")
```

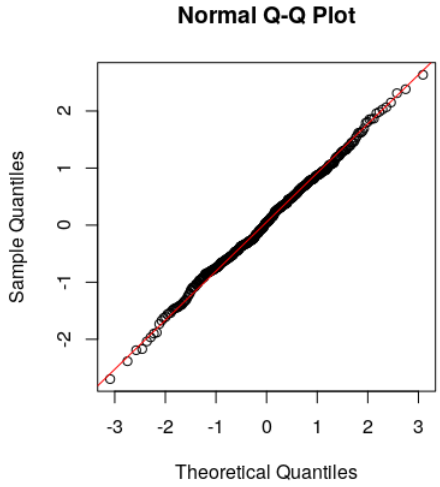

Figura 3.2: Q-Qplot Normal da amostra

b) Use estas $N = 500$ observações para achar $\ell_{MC}(\cdot)$. Usando esta função ache o MCEMV. Usando o codiga da parte anterior

```
C<-c()
for (i in 1:length(mu)) {
  C[i]<-logverMC(mu[i],x_obs,x_sim)
}
mu_est<-mu[which.max(C)]
plot(mu, C, ylab = "MCMC-Logverossimilhança",
xlab = expression("mu"), main = "MCEMV")
mu_est
[1] 1.65
```

3.6 Aplicação 2: Aproximação Bayesiana Computacional (ABC)

Exemplo 6 (**Aproximação Bayesiana Computacional ABC).** *Usando as $n = 100$ observações da v.a. $X \sim N(1, 1)$ e a prior $\theta \sim Unif[0, 2]$ queremos aproximar a posterior $f(x|\theta)$. Observe que não tem uma expressão fechada para esta função. Para isso vamos a usar a Aproximação Bayesiana Computacional ABC, na versão mais simples este algoritmo consiste em:*

1. *Fixamos $\epsilon = 0.1$.*

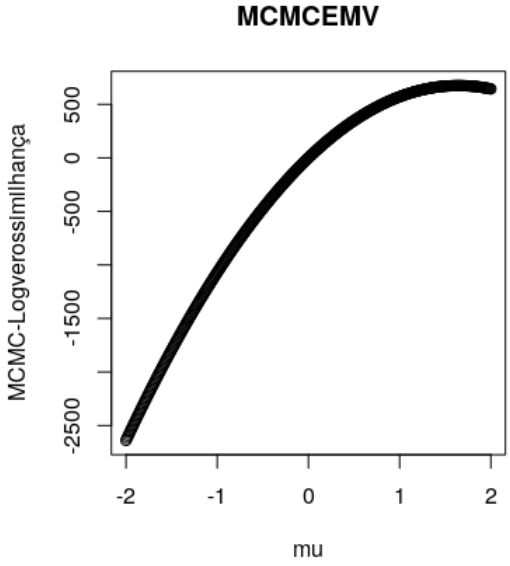

Figura 3.3: MCMC-logverosimilhança

2. *Amostramos* θ_0 *de* $\theta \sim$ *Unif*$[0, 2]$.

3. *Na iteração* t *amostramos um candidato* θ^* *de* $\theta \sim$ *Unif*$[0, 2]$.

4. $(*)$ *Simulamos* $\tilde{x}_1, \ldots, \tilde{x}_{100}$ *da distribuição* $N(0, 1)$

5. *Verificamos se a distância entre as estatísticas sumarias* $\tilde{S}(x_1, \ldots, x_{500}) = \bar{\tilde{x}}$ *e* $S(x_1, \ldots, x_{500}) = \bar{x}$ *é menor que* ϵ*:*

$$d(\tilde{S}, S) = |\tilde{S} - S| < \epsilon$$

se isso acontecer fazemos $\theta_{(t+1)} = \theta^*$*, caso contrario voltamos ao passo (2).*

Use este algoritmo com cada um dos métodos: Monte Carlo MC, Cadeia de Markov Monte Carlo MCMC, só que desta vez vai a usar estes métodos de simulação aplicados ao item (4), com $N(0, 1)$*. small*

```
 ####ABC MC
x_obs<-rnorm(100,1,1)
T=100 # thetas simulados da posterior
#theta_prior~Unif(0,2)
theta_post<-c()

theta_post[1]<-0
epsilon<-0.1
```

```
for (i in 2:T) {
  id<-TRUE
  while(id)
  {theta_prior<-runif(1, min=0, max=2)
  x_sim<-rnorm(100,theta_prior,1)
  if(abs(mean(x_obs)-mean(x_sim))<epsilon)
  {theta_post[i]<- theta_prior
  id<-FALSE
  }}
}
boxplot(theta_post, type = 'l', main = 'Distribuição Posteriori', ylab = 'Amostra de t
theta_max<-mean(theta_post)
theta_max
> theta_max
[1] 0.9365159
```

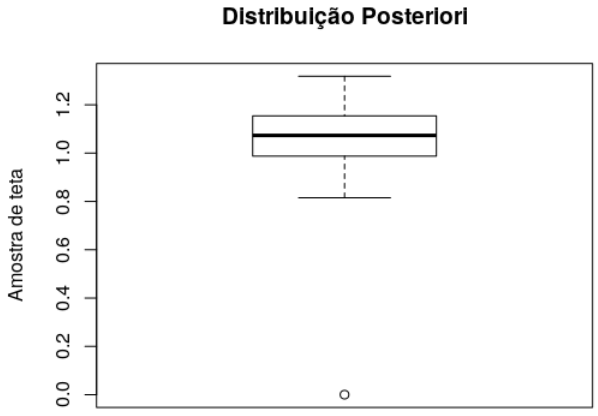

Figura 3.4: Boxplot da Posteriori usando MC para simular, $\theta = 1$.

```
x_obs<-rnorm(100,1,1)
T=100 # thetas simulados da posterior
#theta_prior~Unif(0,2)
theta_post<-c()

theta_post[1]<-0
epsilon<-1
for (i in 2:T) {
  id<-TRUE
  while(id)
```

```
  {theta_prior<-runif(1, min=0, max=2)
  x_s<-rep(0,100)
  x_s[1]<-rnorm(1,0,3/4) #Valor inicial
  for(j in 2:length(x_s)){
    x_s[j]<-(1/2)*x_s[j-1]+rnorm(1,0,3/4)
  }
  x_sim <- x_s
  if(abs(mean(x_obs)-mean(x_sim))<epsilon)
  {theta_post[i]<-theta_prior
  id<-FALSE}
  }
}
boxplot(theta_post, type = 'l', main = 'Distribuição Posteriori', ylab = 'Amostra de teta')
theta_max<-mean(theta_post)
theta_max
[1] 1.101261
```

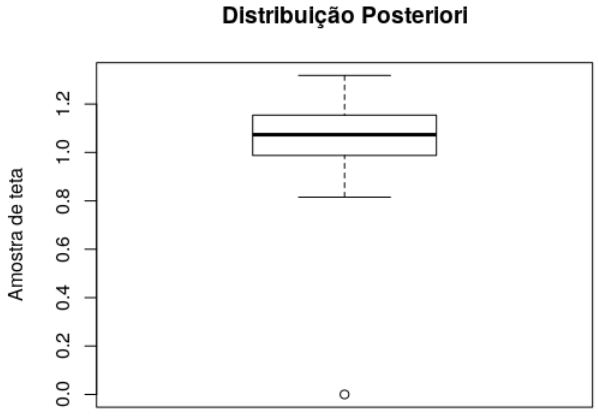

Figura 3.5: Boxplot da Posteriori usando MCMC para simular, $\theta = 1$.

Capítulo 4

Amostrador de Gibbs

Neste capitulo descrevemos um dos mais famosos algoritmos de simulação e o mais usado pela estatística Bayesiana. Vamos descrever ele da seguinte forma. Suponha que queremos amostrar da densidade $f(x), x \in \mathbb{R}^d$. Para generalizar o algoritmo suponha que particionamos o vetor x em K blocos e vamos a escrever $x = (x_1, \ldots, x_K)$, onde $K \leq d$ e $\dim(x_1) + \cdots + \dim(x_K) = d$ com $\dim(x_k)$ representando o dimensão de x_k, este vector tem densidade condicional f_k dada por:

$$f_k(x_k|x_1, \ldots, x_{k-1}, x_{k+1}, \ldots, x_K) \quad k = 1, \ldots, K$$

Sob condições simples, este conjunto *completo de condicionais* $\{f_k\}_{k\in K}$, por sua vez, determina a distribuição alvo $f(x)$; isto de acordo com o teorema de Hammersley-Clifford, ver (Besag, 1974) e (Gelman e Speed, 1993):

Teorema 8 (Hammersley-Clifford)**.** *Se $f(x) > 0$ para todo x, então a distribuição conjunta $f(x)$ é determinada unicamente pelas condicionais completas. Mais precisamente,*

$$f(x) = f(y) \prod_{k=1}^{K} \frac{f_{j_k}(x_{j_k}|x_{j_1}, \ldots, x_{j_{k-1}}, y_{j_{k+1}}, \ldots, y_{j_K})}{f_{j_k}(y_{j_k}|x_{j_1}, \ldots, x_{j_{k-1}}, y_{j_{k+1}}, \ldots, y_{j_K})},$$

para cada permutação $j = (j_k)$ em $\{1, \ldots, n\}$ e todo y.

Algorítmicamente, o *amostrador de Gibbs* é um esquema de amostragem iterativo. Começando com um ponto arbitrário $x^{(0)}$ tal que $f(x^{(0)}) > 0$, cada iteração do amostrador de Gibbs percorre o conjunto completo de condicionais para gerar um número aleatório de cada $f_k(x_k|x_1, \ldots, x_K)$ definindo $x_1, \ldots, x_{k-1}, x_{k+1}, \ldots, x_K$ em seus valores gerados mais recentemente.

O amostrador de Gibbs

Tome $x^{(0)} = (x_1^{(0)}, \ldots, x_K^{(0)})$ com $f(x^{(0)}) > 0$ e iterar para $t = 1, 2, \ldots$

(1) Gere $x_1^{(t)} \sim f_1(x_1|x_2^{(t-1)}, \ldots, x_K^{(t-1)})$.

...

(k) Gere $x_k^{(t)} \sim f_k(x_k|x_1^{(t)}, \ldots, x_{t-1}^{(t)}, x_{t+1}^{(t-1)}, \ldots, x_K^{(t-1)})$.

...

(K) Gere $x_K^{(t)} \sim f_K(x_K|x_1^{(t)}, \ldots, x_{K-1}^{(t)})$.

Sob condições moderadas de regularidade, a distribuição $f^{(t)}(x)$ de $x^{(t)} = (x_1^{(t)}, \ldots, x_K^{(t)})$, converge para $f(x)$.

4.0.1 Exemplo: A distribuição normal multivariada $N_p(\mu, \Sigma)$

A distribuição normal p-dimensional, denotada por $N_p(\mu, \Sigma)$ com o parâmetro consistindo de um vetor de medias $\mu \in \mathbb{R}^p$ e uma matriz de covariância $p \times p$ positivamente definida $\Sigma \in \mathbb{M}_{p\times p}$,o conjunto de todas as matrizes positivas $p \times p$ definidas. A densidade de $N_p(\mu, \Sigma)$ é dada por:

$$f_{\mu,\Sigma)}(x) = \frac{1}{(2\pi)^{n/2}|\Sigma|} e^{-\frac{1}{2}(x-\mu)'\Sigma^{-1}(x-\mu)} \quad x \in \mathbb{R}^p.$$

Seja $X \sim N_p(\mu, \Sigma)$, então $Y = AX$, onde $A \in M_{q\times p}$, é uma q-normal com vetor médio $A\mu$ e matriz de covariância $A\Sigma A'$. Isso significa que o cálculo para distribuições marginais é simples. Tomando A como matrizes de permutação e X é particionado em dois blocos X_1 e X_2, escrevemos

$$\begin{pmatrix} X_1 \\ X_2 \end{pmatrix} \sim N_p \left(\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \begin{pmatrix} \Sigma_{1,1} & \Sigma_{1,2} \\ \Sigma_{2,1} & \Sigma_{2,2} \end{pmatrix} \right).$$

Quando condicionado em X_1, X_2 também é normal

$$X_2|X_1 \sim N_{\dim(X2)}(\mu_2 + \Sigma_{2,1}\Sigma_{1,1}^{-1}(X_1 - \mu), \Sigma_{2,2} - \Sigma_{2,1}\Sigma_{1,1}^{-1}\Sigma_{1,2}).$$

Exemplo 7 (A distribuição normal trivariada)**.** *A Normal trivariada com media* $\mu = (\mu_1, \mu_2, \mu_3)'$ *e covariância;*

$$\Sigma = \begin{pmatrix} 1 & \rho & \rho^2 \\ \rho & 1 & \rho \\ \rho^2 & \rho & 1 \end{pmatrix}.$$

O amostrador de Gibbs de três etapas com a partição de $X = (X_1, X_2, X_3)$ *em* X_1, X_2 *e* X_3 *é então implementado como segue. O amostrador de Gibbs para* $N_3(0, \Sigma)$*: Defina um valor inicial* $x^{(0)} \in \mathbb{R}$ *e itere para* $t = 1, 2, \ldots$

(1) *Gera* $x_1^{(t)} \sim N(\mu_1 + \rho(x_2^{(t-1)} - \mu_2), 1 - \rho^2)$.

(2) *Gera* $x_2^{(t)} \sim N\left(\mu_2 + \frac{\rho}{1+\rho}(x_1^{(t)} - \mu_1 + x_3^{(t-1)} - \mu_3), \frac{1-\rho^2}{1+\rho^2}\right)$.

(3) *Gera* $x_3^{(t)} \sim N(\mu_3 + \rho(x_2^{(t)} - \mu_2), 1 - \rho^2)$.

```
 rho<-0
#rho<-0.99
mu1<-10
mu2<-10
mu3<-10
T<-1000 # numero de simulacoes
x<-matrix(data = NA, nrow =T, ncol=3)
x[1,]<-c(1,1,1)
for (t in 2:T) {
  x[t,1]<-rnorm(1,mu1 + rho*(x[t-1,2]-mu2),1-(rho^2))
  x[t,2]<-rnorm(1,mu2 + (rho/(1+rho))*(x[t,1]-mu1+x[t-1,3]-mu3)
  ,(1-rho^2)/(1+rho^2) )
  x[t,3]<-rnorm(1,mu3 + rho*(x[t,2]-mu2),1-(rho^2))
}
par(mfrow=c(3,1))
plot(1:T,x[,1], type = 'l')
abline(h=10, col="red")
plot(1:T,x[,2], type = 'l')
abline(h=10, col="red")
plot(1:T,x[,3], type = 'l')
abline(h=10, col="red")
```

A Figura 4.1 mostra gráficos de trajetória de duas cadeias de Markov geradas por este amostrador de Gibbs para as distribuições normais trivariadas com $\mu = (10, 10, 10)$ e $\rho = 0$ e $0,99$.

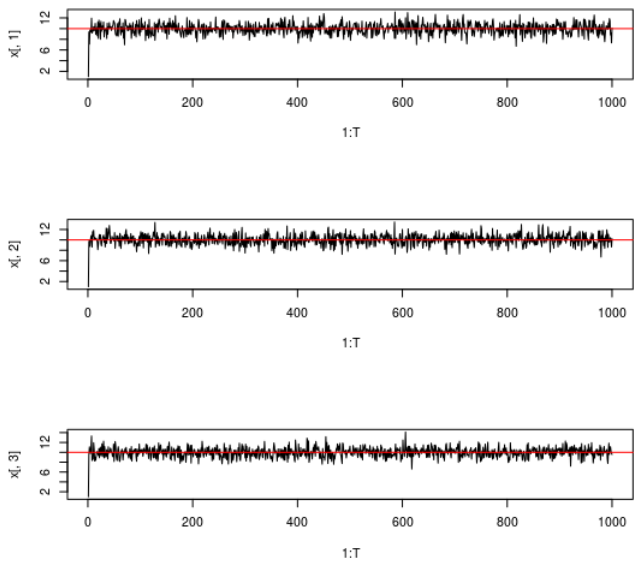

Figura 4.1: Cadeias para (X_1, X_2, x_3) com $\rho = 0$

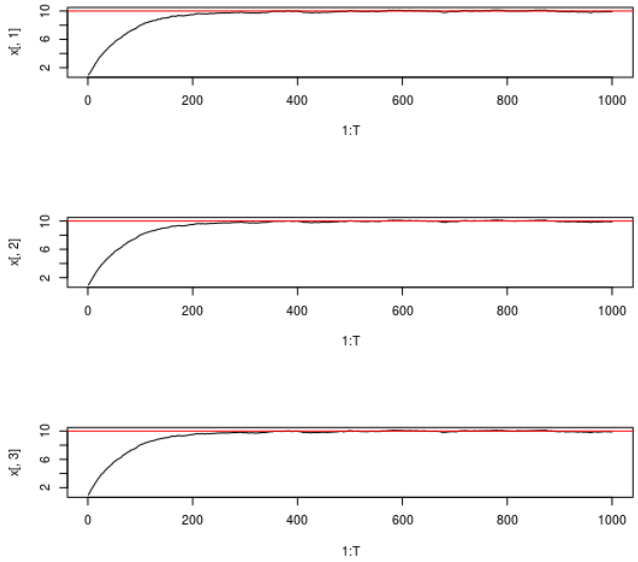

Figura 4.2: Cadeias para (X_1, X_2, x_3) com $\rho = 0.99$

Mostra que o desempenho do amostrador de Gibbs depende da estrutura de dependência entre $X_1, \ldots, X_K$. No caso de $\rho = 0$, o amostrador produz amostras iid da distribuição alvo. No caso de $\rho = 0,99$, a sequência de Gibbs é altamente correlacionada.

4.1 Aumento de dados,'data augmentation' (DA)

Uma das aplicações do algoritmo *DA* é no contexto da análise bayesiana de dados incompletos. Denote por X_{obs} os dados observados e por X_{mis} os dados faltantes. Escreva $X_{\text{com}} = (X_{\text{obs}}, X_{\text{mis}})$ os dados completos com densidade $g(X_{\text{obs}}, X_{\text{mis}}|\theta)$ com o parâmetro $\theta \in \Theta \subset \mathbb{R}^d$. O objetivo é fazer inferência bayesiana com uma distribuição prior $p(\theta)$ para o parâmetro θ. Seja $f(x_{\text{obs}}|\theta)$ o modelo de dados observado, ou seja,

$$f(x_{\text{obs}}|\theta) = g(x_{\text{obs}}, x_{\text{mis}}|\theta)dx_{\text{mis}}.$$

Para inferência Bayesiana sobre usando métodos MCMC, é necessário amostrar a partir de

$$p(\theta|x_{\text{obs}}) \propto f(x_{\text{obs}}|\theta)p(\theta),$$

ou mais geralmente, a distribuição conjunta de θ e x_{mis},

$$p(\theta, x_{\text{mis}}|x_{\text{obs}}) \propto g(x_{\text{obs}}, x_{\text{mis}}|\theta)p(\theta).$$

Seja $h(x_{\text{mis}}|\theta, x_{\text{obs}})$ a distribuição condicional de x_{mis} dada com θ e x_{obs}. Suponha

que tanto $h(x_{\text{mis}}|\theta, x_{\text{obs}})$ como $p(\theta|x_{\text{obs}}, x_{\text{mis}}) \propto g(x_{\text{obs}}, x_{\text{mis}}|\theta)p(\theta)$ sejam fáceis de simular.

O amostrador Gibbs de duas etapas baseado nessas duas condicionais é conhecido como o *algoritmo DA* e pode ser resumido da seguinte forma:
O Algoritmo DA: Um Amostrador Gibbs de Duas Etapas Tome $^{(0)} \in \Theta$ e repita para $t = 1, 2, \ldots$

Passo I Gere $x_{\text{mis}}^{(t)} \sim f_{\text{mis}}(x_{\text{mis}}|\theta^{(t-1)}, x_{\text{obs}})$.

Passo P Gere $\theta^{(t)} \sim p(\theta|x_{\text{obs}}, x_{\text{mis}}^{(t)})$.

Como um amostrador Gibbs de dois passos, o DA cria duas cadeias de Markov intercaladas (marginais) $\{\theta^{(t)} : t = 1, 2, \ldots\}$ e $\{x_{\text{mis}}^{(t)} : t = 1, 2, \ldots\}$.
O DA fornece o caso mais simples do amostrador de Gibbs.

4.1.1 Amostrador de Fatias, 'slice sampler'

Exemplo 8. *Uma alternativa ao método de aceitação e rejeição que pode ser visto como um amostrador do tipo DA.*

Seja $f(x)$ uma função de densidade de probabilidade em $\mathbb{R}^d$. Considere a distribuição uniforme na região $\{(x, u) : 0 \leq u \leq f(x)\} \subset \mathbb{R}^{d+1}$. As condicionais completas consistem em:

$$U|\{X = x\} \sim \textit{Unif}\,(0, f(x)).$$

e

$$X|U = u \sim \textit{Unif}\,(\{x : f(x) \geq u\}).$$

Isso leva a um amostrador de Gibbs de duas etapas e está relacionado ao amostrador de fatia. como exemplo considere simular de $X \sim \exp(2)$ para isso na segunda etapa do amostrador precisamos encontrar a inversa da densidade $f(x) = 2e^{-2x}$ da v.a. X, isto é, $f^{-1}(u) = -\frac{1}{2}\ln(\frac{u}{2})$, logo a região $A = \{x : f(x) \geq u\}$ é equivalente à:

$$A = \{x : x \leq f^{-1}(u)\} = \left\{x : x \leq -\frac{1}{2}\ln(\frac{u}{2})\right\}.$$

```
  T=100
f<-function(u)# densidade da X~exp(2)
{2*exp(-2*u)}
f_inv<-function(u)
{-(1/2)*log(u/2, base = exp(1))}
U<-c()
X<-c()
X[1]<-1
for (t in 2:T) {
U[t]<-runif(1, min = 0, max = f(X[t-1]))
X[t]<-runif(1, min = 0, max = f_inv(U[t]))
}
hist(X, main=NULL, ylab = 'frequencia')
```

```
plot(X, U, type='p', cex = .5)
lines(x<-seq(0.01,10, by=0.01), f(x),type = 'l')
```

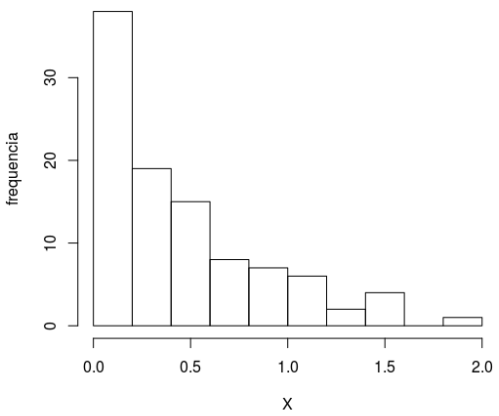

Figura 4.3: Histograma das $T = 100$ simulaçoes da v.a $X \sim \exp(2)$ usando o amostrador de fatias

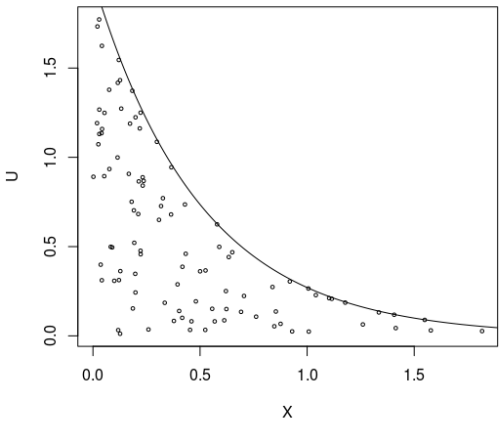

Figura 4.4: Pontos (X_t, U_t) amostrados e a densidade da v.a. X.

Capítulo 5

O algoritmo de Metropolis-Hastings

Considere a distribuição π com densidade $f(x)$ no espaço amostral X.

A idéia básica de criar uma cadeia de Markov com um *núcleo ou kernel de transição* $P(x, dy)$ é ter π como sua distribuição invariante, de modo que:

$$\pi(dy) = \int_X \pi(dx)P(x, dy).$$

A indeterminação do núcleo de transição $P(x, dy)$ permite uma flexibilidade atraente, mas não ajuda muito na construção de $P(x, dy)$ para um $\pi(dx)$ dado.

Para criar nucleos de transição úteis, uma estratégia comumente usada é impor a condição de *reversibilidade*. Diz-seque uma cadeia de Markov com o núcleo de transição $P(x, dy)$ e distribuição invariante π é *reversível* se satisfizer a *condição de equilíbrio detalhado*

$$\int_A \int_B \pi(dx)P(x, dy) = \int_B \int_A \pi(dy)P(y, dx), \forall A, B.$$

Em termos de $f(x)$ e $p(x, y)$, a densidade de $\pi(x)$ e $P(x, dy)$ dado x, a condição de equilíbrio detalhado pode ser escrita como

$$f(x)p(x, y) = f(y)p(y, x).$$

5.1 Amostrador Metropolis

Seja $x = X(t)$ o estado da cadeia de Markov no tempo t. Para construir núcleos de transição satisfatórios, (Metropolis, 1949) consideram uma abordagem em duas etapas:

(i) especificando uma proposta de distribuição, simétrica, com densidade $q(y|x)$, ou seja, $q(y|x) = q(x|y)$ e

(ii) ajustando desenhos a partir de $q(y|x)$ via aceitação-rejeição de tal maneira que a cadeia de Markov resultante seja reversível.

O Amostrador Metropolis

1. Amostre y de $q(y|x_t)$.
2. Calcular a taxa de aceitação

$$\alpha(x_t, y) = \min\left\{1, \frac{f(y)}{f(x_t)}\right\},$$

e defina $x_{t+1} = y$ com a probabilidade $\alpha(x_t, y)$ e $x_{(t+1)} = x_t$ com a probabilidade restante $1 - \alpha(x_t, y)$.

5.2 O Amostrador Metropolis-Hastings

O Amostrador Metropolis-Hastings

(Hastings, 1970) generaliza o amostrador Metropolis, permitindo que as distribuições de propostas sejam assimétricas.
Um algoritmo comumente usado, conhecido como algoritmo M-H, é obtido no amostrador Metropolis, permitindo que $q(y|x)$ na Etapa 1 seja assimétrica e modificando a Etapa 2 para que a condição de equilíbrio detalhado seja válida:

1. Amostre y de $q(y|x_t)$.
2. Calcular a taxa de aceitação

$$\alpha(x_t, y) = \min\left\{1, \frac{f(y)q(x_t|y)}{f(x_t)q(y|x_t)}\right\},$$

e defina $x_{t+1} = y$ com a probabilidade $\alpha(x_t, y)$ e $x_{(t+1)} = x_t$ com a probabilidade restante $1 - \alpha(x_t, y)$.

A taxa de aceitação $\alpha(x_t, y)$ na Etapa 2 do M-H não é determinada exclusivamente pela condição de reversibilidade.Existem alternativas, por exemplo, considere o caso do espaço de estados finito $X \in \{k : k = 1, \ldots, K\}$. Escreva:

- $q_{ij} = q(j|i)$,
- $\pi_i = f(i)$,
- $p_{ij} = p(i, j)$,
- α_{ij} para a probabilidade/razão de aceitação a ser encontrada para todos $i, j \in X$.

Em seguida, para todos os estados $i \neq j$:

$$p_{ij} = q_{ij}\alpha_{ij},$$

com $p_{ii} = 1 - \sum_{j \neq i} p_{ij}$.

Como a p_{ii} não impõe nenhum problema com a condição de reversibilidade, a solução para o problema de encontrar α_{ij} é obtida a partir do sistema de equações:

$$\pi_i q_{ij}\alpha_{ij} = \pi_j q_{ji}\alpha_{ji},$$
$$\text{Sujeito à: } 0 \leq \alpha_{ij} \leq 1.$$

Ou seja, para cada par de i e j $(i \neq j)$:

$$\alpha_{ij} = \frac{c_{ij}}{1 + \frac{\pi_i q_{ij}}{\pi_j q_{ji}}},$$

com constantes positivas simétricas $c_{ij} \leq 1 + \frac{\pi_i q_{ij}}{\pi_j q_{ji}}$,
Observe que $\pi_j q_{ji} = 0$ implica $\alpha_{ij} = 0$.

Daqui resulta que, para a distribuição fixa da proposta $q(y|x)$, a cadeia de Markov resultante com a taxa de mistura mais rápida é a que possui o maior α_{ij} (e α_{ji}) possível:

$$\alpha_{ij} = \begin{cases} 1 & \text{se } \frac{\pi_i q_{ij}}{\pi_j q_{ji}} \geq 1, \\ \frac{\pi_i q_{ij}}{\pi_j q_{ji}} & \text{caso contrario.} \end{cases}$$

ou equivalentemente,

$$\alpha_{ij} = \min\{1, \frac{\pi_i q_{ij}}{\pi_j q_{ji}}\}.$$

Essa probabilidade de aceitação, conhecida como Metropolis (M)-razão, é a usada no algoritmo M-H.
Outra solução específica bem conhecida (Barker, 1965) é dada por

$$\alpha_{ij}^B = \frac{\pi_j q_{ji}}{\pi_i q_{ij} + \pi_j q_{ji}}$$

(Peskum, 1973) fornece um argumento para a otimização da equação anterior em termos de variação mínima das aproximações MCMC às integrais.

No caso geral, a razão M-H $\alpha(x, y)$ é escolhida para que o núcleo de transição resultante obedeça à condição de reversibilidade

$$f(x)q(y|x)\alpha(x, y) = f(y)q(x|y)\alpha(y, x),$$

assumindo que $\alpha(y, x)$ é mensurável em relação à distribuição da proposta $Q(dy|x) = q(y|x)\nu(dy)$.
O núcleo M-H pode ser escrito como, para todos os A,

$$\begin{aligned} P(x, A) &= \int_A Q(dy|x)\alpha(x, y) + Ix \in A \int_X Q(dy|x)(1 - \alpha(x, y)) \\ &= \int_A Q(dy|x)\alpha(x, y) + Ix \in A \left[1 - Q(dy|x)\alpha(x, y)\right], \end{aligned}$$

isso é,

$$P(x, dy) = Q(dy|x)\alpha(x, y) + \delta_x(dy)r(x),$$

onde $\delta_x(dy)$ representa a medida de probabilidade com massa unitária em x e $r(x) = 1 - \int_X Q(dy|x)\alpha(x, y)$, a probabilidade de rejeição (média) no estado x.

5.3 Reversibilidade

A condição de reversibilidade pode ser verificada pelas operações algébricas padrão, para qualquer A, B,

$$\begin{aligned}
\int_B \int_A \pi(dx)P(x, dy) &= \int_B \int_A f(x)q(y|x)\alpha(x, y)\nu(dx)\nu(dy) + \int_{A\cap B} r(x)f(x)\nu(dx) \\
&= \int_B \int_A f(y)q(x|y)\alpha(y, x)\nu(dx)\nu(dy) + \int_{A\cap B} r(x)f(x)\nu(dx) \\
&= \int_A \int_B f(y)q(x|y)\alpha(y, x)\nu(dy)\nu(dx) + \int_{B\cap A} r(y)f(y)\nu(dy) \\
&= \int_A \int_B \pi(dy)P(y, dx).
\end{aligned}$$

Assim, sob a suposição de que o núcleo de transição $P(x, dy)$ determinado por $q(y|x)$ e $\alpha(x, y)$ é irredutível e aperiódico, $\pi(dx)$ é a distribuição de equilíbrio única. A eficiência do algoritmo M-H depende em grande parte da distribuição da proposta.

Exemplo 9. *Vamos a considerar o famoso exemplo do modelo de Ising, considere $n = 10$ partículas com spin ± 1, o espaçõ amostral para elas é:*

$$\Omega = \{r = (r_1, \ldots, r_{10}) \in \{-1, 1\}^{10}\}.$$

A estas particulas está associada uma distribuição de probalilidade:

$$\mathbb{P}(R = r) = \frac{e^{-\frac{1}{T}H(r)}}{Z},$$

onde T é a temperatura, (definimos $\beta = 1/T$) e o potencial H é definida por:

$$H(r) = \sum_i r_i + J \sum_{i,j,i\sim j} r_i r_j,$$

$i \sim j$ significa que i, j são vizinhos. Finalmente, $Z = Z(T, J)$ é uma constante normalizadora. O seguinte codigo em R implementa o algoritmo de Metropolis-Hastings para esta distribuição, $J = 1$, $T = 10$ e $t = 100$ amostras.

```
n<-10 #número de sitios
r0<-rep(1,n)# configuracao original
t<-100 #simulacoes
### potencial H a temperatura T
T=10
H<-function(r,J)
```

```
{Hij<-0
for (i in 1:n-1) {
Hij<-(J/2)*(Hij+r[i]*r[i+1])
}

Hi<-sum(r)
return(-Hij-Hi)
}
deltaH<-function(r1,r2,T,J)
{
beta<-1/T
return(beta*(H(r1,J)-H(r2,J)))
}
ising<-matrix(data=NA, nrow=n, ncol=t)
id=2
ising[,1]<-r0
J<-1
while (id<=t) {
  i<-sample(1:n, size=1)
  rMH<-c()
  rMH<-ising[,id-1]
  rMH[i]<-ising[i,id-1]*sample(c(-1,1),prob=0.5)
  pMH<-min(1,exp(deltaH(rMH,ising[,id-1],T,J)))
  if(runif(1)<pMH)
    {ising[,id]<-rMH
    id<-id+1}
  else
  ising[,id]<-ising[,id-1]
}
```

Os ultimos 5 estados observador

```
> ising[,(t-4):t]
      [,1] [,2] [,3] [,4] [,5]
 [1,]    1    1    1   -1   -1
 [2,]    1   -1   -1   -1    1
 [3,]    1    1   -1   -1   -1
 [4,]    1    1    1    1    1
 [5,]   -1   -1   -1   -1   -1
 [6,]    1    1    1    1    1
 [7,]    1    1    1    1    1
 [8,]   -1   -1   -1   -1   -1
 [9,]   -1   -1   -1   -1   -1
[10,]    1    1    1    1    1
```

definindo a magnetização como a media dos spins de uma comfiguração, magnet= $1/10 \sum_i r_i$ *obtemos uma trajetoria das magnetizações simuladas:*

```
magnet<c()
magnet[1:t]<-apply(ising[,1:t]/n, 2, sum)
```

```
plot(1:t,magnet, type='l')
abline(h=mean(magnet[1:t]),col='red')
```

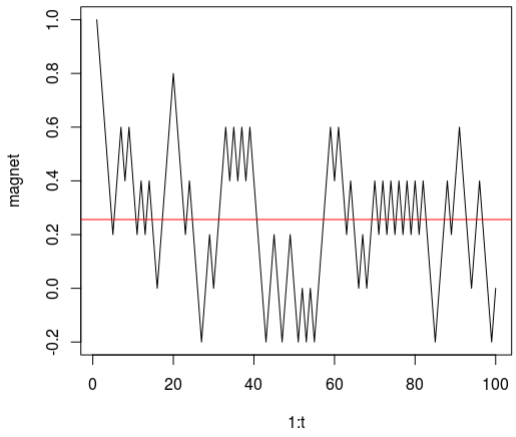

Figura 5.1: Magnetização das $t = 100$ configurações, a linha vermelha é a media de todas as magnetizações

5.4 Amostrador Independente

Para o amostrador independente, temos $q(y|x) = q(y)$; isto é, o estado candidato y é desenhado independentemente do estado atual da cadeia de Markov. A relação M-H se torna a *razão de importância*,

$$r(x, y) = \frac{f(y)q(x)}{f(x)q(y)} = \frac{f(y)/q(y)}{f(x)/q(x)},$$

onde $x = x_t$.
O amostrador independente pode ser visto como uma generalização do algoritmo de aceitação-rejeição.
É fácil ver que a cadeia de independência é irredutível e aperiódica se o suporte de $q(x)$ contiver o suporte de $f(x)$, ou seja,

$$\{x : x \in X, f(x) > 0\} \subseteq \{x : x \in X, q(x) > 0\}.$$

Uma consideração importante ao especificar $q(x)$ é que $q(x)$ deve-se parecer com $f(x)$, ver (Mengersen e Tweedie, 1996) e também (Christian Robert, 2010).

Teorema 9. *A cadeia de independência é uniformemente ergódica se existe uma constante M, tal que*

$$f(x) \leq Mq(x) \quad x \in \{x : f(x) > 0\}.$$

Problemas com este algoritmo incluem o problema da *armadilha local* e a incapacidade de amostrar de distribuições com integrais intratáveis.

5.5 Aumento de dados, (DA)

Vamos a descrever o algoritmo *DA* que aparece no contexto da análise Bayesiana de dados incompletos. Denote por X_{obs} os dados observados e por X_{mis} os dados faltantes. Escreva $X_{\text{com}} = (X_{\text{obs}}, X_{\text{mis}})$ os dados completos com densidade $g(X_{\text{obs}}, X_{\text{mis}}|\theta)$ com o parâmetro $\theta \in \Theta \subset \mathbb{R}^d$. O objetivo é fazer inferência Bayesiana com uma distribuição prior $p(\theta)$ para o parâmetro θ. Seja $f(x_{\text{obs}}|\theta)$ o modelo de dados observado, ou seja,

$$f(x_{\text{obs}}|\theta) = g(x_{\text{obs}}, x_{\text{mis}}|\theta)dx_{\text{mis}}.$$

Para inferência Bayesiana sobre θ usando métodos MCMC, é necessário amostrar a partir de

$$p(\theta|x_{\text{obs}}) \propto f(x_{\text{obs}}|\theta)p(\theta),$$

ou mais geralmente, a distribuição conjunta de θ e x_{mis},

$$p(\theta, x_{\text{mis}}|x_{\text{obs}}) \propto g(x_{\text{obs}}, x_{\text{mis}}|\theta)p(\theta).$$

Seja $h(x_{\text{mis}}|\theta, x_{\text{obs}})$ a distribuição condicional de x_{mis} dada com θ e x_{obs}. Suponha que tanto $h(x_{\text{mis}}|\theta, x_{\text{obs}})$ como $p(\theta|x_{\text{obs}}, x_{\text{mis}}) \propto g(x_{\text{obs}}, x_{\text{mis}}|\theta)p(\theta)$ sejam fáceis de simular.

O amostrador Gibbs de duas etapas baseado nessas duas condicionais é conhecido como o *algoritmo DA* e pode ser resumido da seguinte forma.

O Algoritmo DA: Um Amostrador Gibbs de Duas Etapas

Fixe um $^{(0)} \in \Theta$ e repita para $t = 1, 2, \cdots$:

Passo I Amostre $x_{\text{mis}}^{(t)} \sim f_{\text{mis}}(x_{\text{mis}}|\theta^{(t-1)}, x_{\text{obs}})$.

Passo P Amostre $\theta^{(t)} \sim p(\theta|x_{\text{obs}}, x_{\text{mis}}^{(t)})$.

Como um amostrador Gibbs de dois passos, o DA cria duas cadeias de Markov intercaladas (marginais) $\{\theta^{(t)} : t = 1, 2, \dots\}$ e $\{x_{\text{mis}}^{(t)} : t = 1, 2, \dots\}$.
O DA fornece o caso mais simples do amostrador de Gibbs.

Exemplo 10 (Amostrador de Fatias (slice sampler))**.** *Uma alternativa ao método de aceitação e rejeição*

Seja $f(x)$ uma função de densidade de probabilidade em $\mathbb{R}^d$. Considere a distribuição uniforme na região $\{(x, u) : 0 \leq u \leq f(x)\} \subset \mathbb{R}^{d+1}$. As condicionais completas consistem em:

$$U|\{X = x\} \sim \textit{Unif}\,(0, f(x)),$$

e

$$X|U = u \sim \textit{Unif}\,(\{x : f(x) \geq u\}).$$

Isso leva a um sampler de Gibbs de duas etapas e está relacionado ao sampler de fatia

Exemplo 11 (Normal multivariada com dados incompletos)**.** *Considere uma amostra de tamanho n, $Y_1, \ldots, Y_n$, da distribuição normal multivariada p-dimensional $N_p(\mu, \Sigma)$ com vetor médio desconhecido $\mu \in \mathbb{R}^{p \times p}$ (definida positiva) e matriz de covariância Σ . Cada marginal Y_i é totalmente observado ou ausente para cada $i = 1, \ldots, n$. Denotemos Y_{obs} as marginais observadas e Y_{mis} as ausentes de Y_i. Escrevemos a distribuição condicional de Y_{mis} dada Y_{obs} e (μ, Σ) como*

$$Y_{i,mis}|(Y_{i,obs}, \mu, \Sigma) \sim N(\mu^{(i)}_{mis} + \Sigma^{(i)}_{mis,obs}[\Sigma^{(i)}_{obs,obs}]^{-1}(Y_{i,obs} - \mu^{(i)}_{mis}), \\ \Sigma^{(i)}_{mis,mis} - \Sigma^{(i)}_{mis,obs}[\Sigma^{(i)}_{obs,obs}]^{-1}\Sigma^{(i)}_{obs,mis}. \tag{5.1}$$

Suponha que, para a análise Bayesiana, usamos a distribuição prior $p(\mu, \Sigma) \propto |\Sigma| - (q+1)/2$, onde q é um número inteiro conhecido.
Com $q = p$, a priori se torna a priori de Jeffrey para Σ.

Seja $\bar{Y} = n^{-1}\sum_{i=1}^n Y_i$ e seja $S = \sum_{i=1}^n (Y_i - \bar{Y})(Y_i - \bar{Y})$. A distribuição posterior dos dados completos $p(\mu, \Sigma|Y_1, \ldots, Y_n)$ pode ser caracterizada por

$$\Sigma|Y_1, \ldots, Y_n \sim \frac{1}{|\Sigma|^{n+q/2}} \exp\{\frac{1}{2} traço(\Sigma^{-1}S), \} \tag{5.2}$$

isto é, a distribuição inversa de Wishart e

$$\mu|\Sigma, Y_1, \ldots, Y_n \sim N_p(\bar{Y}, \Sigma/n). \tag{5.3}$$

Assim, o algoritmo DA possui os seguintes passos I e P:

Passo I *Para $i = 1, \ldots, n$, amostre $Y_{i,mis}$, (5.1).*

Passo P *Primeiro amostramos Σ de (5.2) dado $Y_1, \ldots, Y_n$ e, em seguida, amostramos μ de (5.3) dado $Y_1, \ldots, Y_n$ e Σ.*

Observamos que o Passo P pode ser dividido em duas sub-etapas, resultando em um amostrador de Gibbs em três etapas:

Etapa 1. *É o mesmo que a etapa I do DA.*

Etapa 2. *Retire μ de sua distribuição condicional dada $Y_1, \ldots, Y_n$ e Σ.*

Etapa 3. *Retire Σ de sua distribuição condicional dada $Y_1, \ldots, Y_n$ e μ.*

Comparado ao algoritmo DA, um amostrador de Gibbs em duas etapas, esse amostrador de Gibbs em três etapas induz mais dependência entre a sequência $\{(\mu(t), \Sigma(t)) : t = 1, 2, \ldots\}$ e, portanto, converge mais lentamente que o DA correspondente.
Em outras palavras, o DA pode ser visto como obtido no amostrador de Gibbs em três etapas, transformando μ e Σ em um único bloco. Essa técnica de agrupamento é chamada de "bloqueio"por (Liu e Rubin, 1994). Deve-se notar também que DA's mais eficientes para dados normais multivariados incompletos podem ser implementados imputando menos dados/informações ausentes. No caso em que o cálculo da distribuição posterior $p(\theta|X_{obs})$ é de interesse, o DA pode ser visto como um algoritmo de amostragem MCMC usando variáveis auxiliares (aleatórias), conforme ilustrado no Exemplo.

Exemplo 12 (Reparametrização, modelo simples de efeitos aleatórios)**.** *Considere uma abordagem Bayesiana para o modelo linear mais simples de efeitos aleatórios com efeito fixo e efeito aleatório Z*

$$Z|\theta \sim N(\theta, v),$$

e

$$Y|(Z,\theta) \sim (\theta + Z, 1), \tag{5.4}$$

com os dados observados $Y \in \mathbb{R}$, $v > 0$ conhecido, e prior $\theta \sim$ Unif$(\mathbb{R})$, isto é, prior plana na reta $\mathbb{R}$, para o efeito fixo θ. O modelo de dados completos para Y e Z possui a distribuição conjunta;

$$N_2\left(\begin{pmatrix}\theta\\0\end{pmatrix}, \begin{pmatrix}1+v & v\\ v & v\end{pmatrix}\right),$$

levando à posterior de (θ, Z)

$$N_2\left(\begin{pmatrix}Y\\0\end{pmatrix}, \begin{pmatrix}1+v & v\\ v & v\end{pmatrix}\right).$$

Neste caso, a distribuição posterior alvo de θ dado Y tem uma solução de forma fechada, $N(Y, 1+v)$, porque o modelo de dados observados é $Y|\theta \sim N(\theta, 1+v)$.
A ideia básica de centralização hierárquica equivale a comparar duas implementações diferentes do algoritmo DA com Z visto como dados ausentes e θ como parâmetro.

Uma implemntação deste algoritmo consiste nas duas etapas a seguir:

Passo I. *Amostre Z a partir de sua distribuição condicional $N(v(\theta - Y)/(1+v), v/(1+v))$, dados Y e θ.*

Passo P *Amostre θ de sua distribuição condicional $N(Y + Z, 1)$, dados Y e Z.*

```
 y<-3 # observacao
v<-4
T<-1000
ztheta<-matrix(data=NA, nrow=2,ncol = T)
ztheta[,1]=c(1,1)
for (t in 2:T) {
  ztheta[1,t]<-rnorm(1,v*(ztheta[2,t-1]-y)/(1+v),v/(1+v))
  ztheta[2,t]<-rnorm(1,y+ztheta[1,t],1)
}
par(mfrow=c(2,1))
plot(1:T,ztheta[1,], type = 'l', ylab = 'Z')
abline(h=mean(ztheta[1,1:T]), col="red")
plot(1:T,ztheta[2,], type = 'l' , ylab = 'theta')
abline(h=mean(ztheta[2,1:T]), col="red")
```

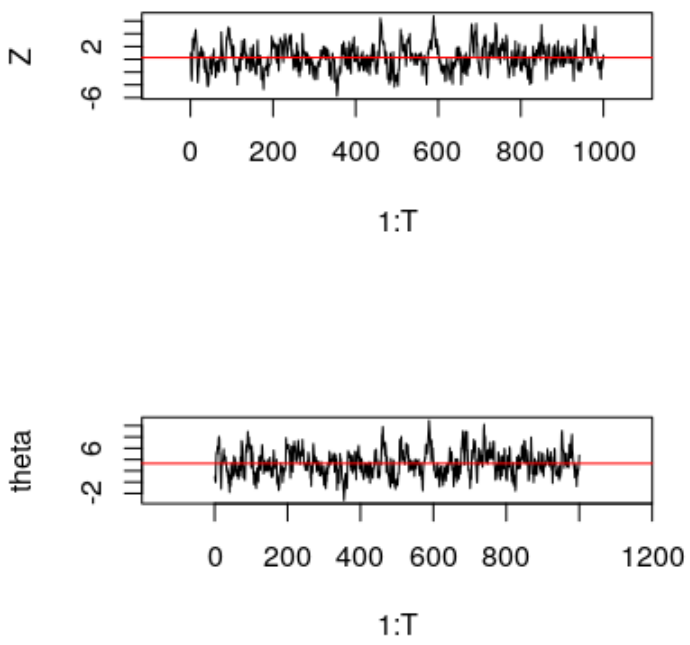

Figura 5.2: DA, modelo simples de efeitos aleatórios com $v = 4$ e autocorrelação $r = v/(1+v) = 0.8$.

A outra implementação do algoritmo é baseada na versão reparametrizada

$$Z|\theta \sim N(\theta, v) \text{ e}$$

$$Y|(Z, \theta) \sim (Z, 1),$$

que possui o mesmo modelo de dados observados. O DA correspondente possui as duas etapas a seguir:

Passo I. *Amostre Z a partir de sua distribuição condicional $N([vY+\theta]/(1+v), v/(1+v))$, dados Y e θ.*

Passo P *Amostre θ de sua distribuição condicional $N(Z, v)$, dados Y e Z.*

```
y<-3 # observacao
v<-4
T<-1000
ztheta<-matrix(data=NA, nrow=2,ncol = T)
ztheta[,1]=c(1,1)
```

```
for (t in 2:T) {
  ztheta[1,t]<-rnorm(1,((v*y)+ztheta[2,t-1])/(1+v),v/(1+v))
  ztheta[2,t]<-rnorm(1,ztheta[1,t],v)
}
par(mfrow=c(2,1))
plot(1:T,ztheta[1,], type = 'l', ylab = 'Z',xlim = c(0,1000))
abline(h=mean(ztheta[1,1:T]), col="red")
plot(1:T,ztheta[2,], type = 'l' , ylab = 'theta', xlim = c(0,1000))
abline(h=mean(ztheta[2,1:T]), col="red")
```

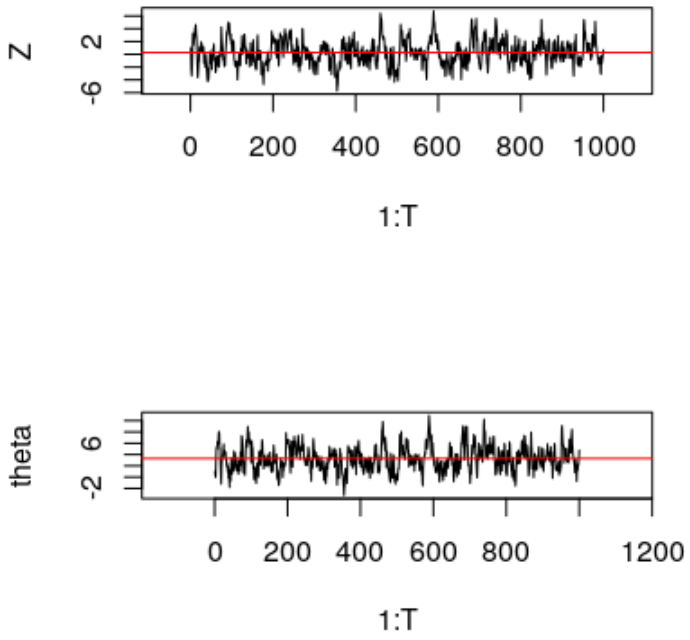

Figura 5.3: DA, modelo simples de efeitos aleatórios com $v = 4$ e autocorrelação $r = 1/(1+v) = 0.2$.

Cada uma das duas implementações de DA induz uma série AR em θ. O primeiro tem o coeficiente de autocorrelação $r = v/(1+v)$; enquanto o segundo tem o coeficiente de autocorrelação $r = 1/(1+v)$. Assim, a taxa de convergência depende do valor de v, comparado com a variação residual unitária de Y condicionada em Z. Para grandes valores de $v, (\gg 1)$, o segundo esquema é mais eficiente que o primeiro. Para pequenos valores de $v(\ll 1)$, o segundo esquema é muito lento e o primeiro é bastante eficiente.

Capítulo 6

Extenções do Amostrador de Metropolis Hastings

6.1 O Amostrador Metropolis-Hastings

(Hastings, 1970) generaliza o amostrador Metropolis, permitindo que as distribuições de propostas sejam assimétricas. Um algoritmo comumente usado, conhecido como algoritmo M-H, é obtido no amostrador Metropolis, permitindo que $q(y|x)$ na Etapa 1 seja assimétrica e modificando a Etapa 2 para que a condição de equilíbrio detalhado seja válida:

1. Amostre y de $q(y|x_t)$.

2. Calcular a taxa de aceitação

$$\alpha(x_t, y) = \min\{1, \frac{f(y)q(x_t|y)}{f(x_t)q(y|x_t)}\},$$

e defina $x_{t+1} = y$ com a probabilidade $\alpha(x_t, y)$ e $x_{(t+1)} = x_t$ com a probabilidade restante $1 - \alpha(x_t, y)$.

6.1.1 O algoritmo Hit-and-Run

O algoritmo *Hit-and-Run*, ((Boneh e Golan, 1979) e (Smith, 1980)), podem ser obtidos separando o processo de criação de um salto proposto no MH em dois subprocessos:

(i) Gere uma direção d a partir de uma distribuição na superfície da esfera unitária $\mathbb{O}$.

(ii) Gere uma distância sinalizada λ ao longo da direção d no espaço restrito $X_{x,d} = \{\lambda : \lambda \in \mathbb{R}, x + \lambda d \in X\}$,

onde $x = X(t)$. Ou seja, o salto proposto é $y = X(t) + \lambda d \in X$. Na estrutura MH, esse algoritmo pode ser resumido da seguinte forma:
Amostre X_0 da distribuição inicial $f_0(X)$ com $f(X_0) > 0$ e itere para $t = 0, 1, 2, \ldots$
Algoritmo Hit-and-Run

1. Amostre $d \sim g(d)(d \in \mathbb{O})$ e $\lambda \sim l(\lambda|d, x)$ sobre $X_{x,d}$ e calcule a probabilidade de aceitação MH $\alpha(x, y)$, onde $x = X(t)$.

2. Gere U a partir de Unif$(0, 1)$ e defina

$$X^{(t+1)} = \begin{cases} x + \lambda d & \text{se } U \leq \alpha(x, y); \\ x, & \text{caso contrario.} \end{cases}$$

6.2 O algoritmo de Langevin(Metropolis-adjusted Langevin algorithm, (MALA))

O *algoritmo de Langevin* está baseado no processo de difusão de Langevin, que é definido pela equação diferencial estocástica

$$dX_t = dB_t + \nabla \log f(X_t)dt,$$

onde B_t é o movimento browniano padrão. Esse processo tem f como a distribuição estacionária. A implementação do algoritmo de difusão envolve uma etapa de discretização que substitui a eq estocástica por uma transição de estilo de passeio aleatório:

$$x^{(t+1)} = x^{(t)} + \frac{1}{2}\sigma^2 \nabla \log f(x^{(t)}) + \sigma\epsilon_t,$$

onde $\epsilon_t \sim N_d(0, I_d)$ e σ é o tamanho do passo da discretização.
No entanto, (Roberts e Tweedie, 1996) mostra que o processo discretizado pode ser *transiente* e não tem mais f como a distribuição estacionária. Para corrigir esse comportamento negativo, (Besag, 1994) sugere moderar a etapa de discretização aplicando a regra MH de aceitação-rejeição; isto é, tratar a discretização como uma proposta convencional e aceitá-la de acordo com a regra MH.

Algoritmo de Langevin

1. Propor um novo estado

$$x^* = x^{(t)} + \frac{1}{2}\sigma^2 \nabla \log f(x^{(t)}) + \sigma\epsilon_t,$$

   onde σ é o parâmetro especificado pelo usuário.

2. Calcule a razão MH:

$$r = \frac{f(x^*)}{f(x^{(t)}} \frac{\exp(-\|x^{(t)} - x^* - \frac{\sigma^2}{2}\nabla \log f(x^*)\|^2/2\sigma^2)}{\exp(-\|x^* - x^{(t)} - \frac{\sigma^2}{2}\nabla \log f(x^*)\|^2/2\sigma^2)}.$$

3. Defina $x^{(t+1)} = x^*$ com probabilidade $\min(1, r)$ e defina $x^{(t+1)} = x^{(t)}$ com a probabilidade restante.

6.3 O algoritmo MH de múltiplas tentativas, (MTM)

Conforme implícito no algoritmo de Langevin, o uso das informações de gradiente para a distribuição de destino pode ser usado para acelerar a convergência da simulação de MH. No entanto, informações de gradiente não estão disponíveis para a maioria das distribuições de destino. Uma maneira de aproximar o gradiente da distribuição alvo é usar amostras de Monte Carlo. Liu et al. (2000) propõem o uso do algoritmo Metropolis (*MTM*) de múltiplas tentativas, que modifica o MH padrão substituindo a proposta única y por um conjunto de propostas k iid $y_1, \ldots, y_k$ de $q(y|x)$ e selecionando um bom candidato (amostragem por importancia) para o qual saltar, de acordo com a regra MH.
Suponha que $q(y|x) > 0$ se e somente se $q(x|y) > 0$. Seja $\lambda(x, y)$ uma função simétrica não negativa em x e y. Suponha que $\lambda(x, y) > 0$ sempre que $q(y|x) > 0$. Defina $w(x, y) = f(x)q(y|x)\lambda(x, y)$. Seja $x = X^{(t)}$.
O algoritmo MH de múltiplas tentativas *MTM*

1. Amostre k iid candidatos $y_1, \ldots, y_k$ de $q(y|x)$ e calcule $w_i = w(y_i, x)$ para $i = 1, \ldots, k$.

2. Selecione $y = y_j$ de $\{y_1, \ldots, y_k\}$ com probabilidade proporcional a w_i, $i = 1, \ldots, k$, amostre $x_1^*, ..., x_{k-1}^*$ de $q(\cdot|y)$, defina $x_k^* = x$ e calcule $w_i^* = w(x_i^*, y)$ para $i = 1, \ldots, k$.

3. Aceite y com probabilidade

$$a_m = \min\{1, \frac{w_1 + \cdots + w_k}{w_1^* + \cdots + w_k^*}\}$$

e rejeite-o (ou defina $X^{(t+1)} = x$) com probabilidade $1 - a_m$.

Quando $q(x|y)$ é simétrico, por exemplo, pode-se escolher

$$\lambda(x, y) = 1/q(y|x)$$

e depois $w(x, y) = f(x)$.

Exercício 2. *Use os algoritmos MALA e MTM para simular uma amostra de tamanho $N = 100$ da v.a. exponencial com taxa $\lambda = 2$.*

Capítulo 7

Métodos MCMC da variável auxiliar

As amostras de $f(x)$ podem então ser obtidas a partir das realizações $(x_1, u_1), \ldots, (x_N, u_N)$, de (X, U) por marginalização ou condicionamento.

Os métodos MCMC da variável auxiliar Aumento da distribuição da proposta O método de aumento da distribuição da proposta pode ser implementado da seguinte maneira:

1. Especifique uma distribuição proposta $T(x', u|x)$ e sua versão reversível $T(x, u|x')$, de modo que $\int T(x', u|x)du = T(x'|x)$ e $\int T(x, u|x')du = T(x|x')$.

2. Amostre a candidata x' da proposta $T(x', u|x)$ e aceite-a com probabilidade $\min\{1, r(x, x', u)\}$, onde

$$r(x, x', u) = \frac{f(x')T(x, u|x')}{f(x)T(x', u|x)}.$$

   Repita esta etapa para gerar realizações $x_1, \ldots, x_N$, que será distribuído aproximadamente como $f(x)$ para N grande.

A validade desse método pode ser mostrada da seguinte maneira. Seja

$$K(x'|x) = \int s(x, x', u)du + I_{(x=x')}[1 - \int\int s(x, x^*, u)dudx^*,$$

o *kernel* de transição integrado de x para x', onde $s(x, x', u) = T(x', u|x)r(x, x', u)$ e $I_{(\cdot)}$ é a função do indicadora.
Então

$$f(x)\int s(x, x', u)du = \int \min\{f(x')T(x, u|x'), f(x)T(x', u|x)\}du,$$

que é simétrico sobre x e x'.
Isso implica que $f(x')K(x'|x) = f(x)K(x|x')$; isto é, a condição de equilíbrio detalhada é satisfeita para a transição $x \to x'$.

7.1 Recozimento simulado (Simulated Annealing)

Suponha que se pretenda encontrar o *mínimo global* de uma função objetivo $H(x)$, que também é chamada *função de energia*.
Ao aumentar para o sistema uma variável auxiliar, a *temperatura* T, minimizar $H(x)$, é equivalente à amostragem da distribuição de Boltzmann $f(x,T) \propto \exp(-H(x)/T)$ com um valor muito pequeno de T (perto de 0).
Quando T está próximo de 0, a maior parte da massa da distribuição $f(x,T)$ concentra-se nos minimizadores globais de $H(x)$.
Nesse sentido, dizemos que a amostragem é mais básico que otimização. Para amostrar com sucesso de $f(x,T)$ com um valor muito pequeno de T, (Kirkpatrick, Gelatt e Vecchi, 1983) sugeriram a simulação a partir de uma sequência de distribuições de Boltzmann, $f(x,T_1),\ldots,f(x,T_m)$, de maneira seqüencial, em que as temperaturas formam uma escada decrescente $T_1 > T_2 > \cdots > T_m$ com $T_m \approx 0$ e T_1 sendo razoavelmente grande, de modo que a maioria dos movimentos de MH subindo nesse nível possa ser aceita.
A simulação em níveis de alta temperatura visa fornecer uma boa amostra inicial, esperançosamente um ponto na bacia de atração do mínimo global de $H(x)$, para a simulação em níveis de baixa temperatura.

7.2 Revenimento simulado (Simulated Tempering)

Suponha que queremos amostrar da distribuição $f(x) \propto \exp(H(x))$, $x \in \mathcal{X}$. O temperamento simulado (Marinari e Parisi, 1992) aumenta a distribuição alvo para $f(x,T) \propto exp(-H(x)/T)$ incluindo uma variável auxiliar T, chamada *temperatura*, que tem valores num conjunto finito pré especificado pelo usuário.

A têmpera simulada é obtida a partir do *recozimento simulado*, tratando a temperatura T como uma variável aleatória auxiliar a ser simulada em conjunto com x:

- A têmpera simulada atualiza (x,T) usando o amostrador de Gibbs; isto é, atualizar x e T iterativamente.
- A temperatura mais baixa é definida como 1, pois o objetivo é amostrar a partir de $f(x)$.

Suponha que T use valores $T_1 > T_2 > \cdots > T_m$, onde $T_m = 1$ é chamada de temperatura alvo. Seja $f(x,T_i) = \exp(-H(x)/T_i)/Z_i$ a distribuição experimental definida no nível de temperatura T_i, onde Z_i é a constante normalizadora da distribuição.
Se q_{ij} denota a probabilidade proposta de transição do nível T_i para T_j. Normalmente, define-se $q_{i,i+1} = q_{i,i-1} = q_{i,i} = 1/3$ para $1 < i < m$, $q_{1,2} = 1/3$, $q_{m,m-1} = 1/3$, $q_{1,1} = 2/3$ e $q_{m,m} = 2/3$. Começando com $i_0 = 1$ e uma amostra inicial $x_0 \in \mathcal{X}$,

Revenimento Simulado

1. Amostre um número aleatório $U \sim \text{Unif}[0,1]$ e determine o valor de j de acordo com a matriz de transição da proposta $(q_{(ij)})$.

2. Se $j = i_t$, seja $i_{t+1} = i_t$ e x_{t+1} amostrado de um *kernel* MH $K_{i_t}(x, y)$ que admite $f(x, T_{i_t})$ como a distribuição invariante.

3. Se $j \neq i_t$, $x_{t+1} = x_t$ e aceite a proposta com probabilidade

$$\min\{1, \frac{\hat{Z}_j}{Z_{i_t}} \exp(-H(x)(\frac{1}{T_j} - \frac{1}{T_{i_t}}))\frac{q_{j,i_t}}{q_{i_t,j}}\}$$

onde $\hat{Z}_i$ denota uma estimativa de Z_i.
Se for aceito, defina $i_{t+1} = j$. Caso contrário, defina $i_{t+1} = i_t$.

7.3 Amostrador de Fatias (slice sampler)

Uma alternativa ao método de aceitação e rejeição. Seja $f(x)$ uma função de densidade de probabilidade em $\mathbb{R}^d$. Considere a distribuição uniforme na região $\{(x, u) : 0 \leq u \leq f(x)\} \subset \mathbb{R}^{d+1}$. As condicionais completas consistem em:

$$U|\{X = x\} \sim \text{Unif } (0, f(x)),$$

e

$$X|U = u \sim \text{Unif } (\{x : f(x) \geq u\}).$$

Isso leva a um sampler de Gibbs de duas etapas e está relacionado ao sampler de fatia

7.4 Algoritmo de Møller

Modelos espaciais, por exemplo, o modelo autologístico, o modelo de Potts e o modelo autonormal (Besag, 1974), têm sido utilizados na modelagem de muitos problemas científicos. Exemplos incluem análise de imagens (Hurn, Husby e Rue, 2003), mapeamento de doenças (Green e Richardson, 2002), análise genética geográfica (François, Ancelet e Guillot, 2006), entre outros. Um grande problema com esses modelos é que a constante de normalização é intratável. O problema pode ser descrito da seguinte maneira. Suponha que tenhamos um conjunto de dados X gerado a partir de um modelo estatístico com a função de probabilidade

$$f(x|\theta) = \frac{1}{Z(\theta)} \exp\{-U(x, \theta)\},$$

onde θ é o parâmetro e $Z(\theta)$ é a constante de normalização que depende de θ e não está disponível na forma fechada. Seja $f(\theta)$ denotado a densidade prior de θ. A distribuição posterior de θ dado x é então dada por

$$f(x|\theta) = \frac{1}{Z(\theta)} \exp\{-U(x, \theta)\} f(\theta),$$

O algoritmo MH não pode ser aplicado diretamente para simular $f(x|\theta)$, porque a probabilidade de aceitação envolveria uma computação razão intratável $Z(\theta)/Z(\theta')$, em que θ indica o valor proposto.

Um passo significativo feito por (Møller et al., 2006) propõe aumentar a distribuição $f(\theta|x)$ por uma variável auxiliar, de modo que a razão constante de normalização $Z(\theta)/Z(\theta')$ possa ser cancelada. O algoritmo Møller pode ser descrito da seguinte maneira.

Seja y a variável auxiliar, que compartilha o mesmo espaço de estado com x.

Seja $f(\theta, y|x) = f(x|\theta)f(\theta)f(y|\theta, x)$, denotar a distribuição conjunta de θ e y condicional em x, onde $f(y|\theta, x)$ é a distribuição da variável auxiliar y. Para simular essa função usando o algoritmo MH, pode-se usar a distribuição da proposta $q(\theta', y'|\theta, y) = q(\theta'|\theta, y)q(y'|\theta')$, que corresponde à mudança usual no vetor de parâmetros $\theta \to \theta'$, seguida por uma etapa de *amostragem exata* y' de $q(\cdot|\theta')$. Se $q(y'|\theta')$ for definido como $f(y'|\theta)$, a razão MH poderá ser escrita como

$$r(\theta, y, \theta', y'|x) = \frac{f(x|\theta')f(\theta')f(y'|\theta', x)q(\theta|\theta', y')f(y|\theta)}{f(x|\theta)f(\theta)f(y|\theta, x)q(\theta'|\theta, y)f(y'|\theta')},$$

onde a constante normalizadora desconhecida $Z(\theta)$ pode ser cancelada. Para facilitar a computação, (Møller et al., 2006) sugerem ainda definir as distribuições da proposta $q(\theta'|\theta, y) = q(\theta'|\theta)$ e $q(\theta|\theta', y') = q(\theta|\theta')$ e as distribuições auxiliares

$$f(y|\theta, x) = f(y|\hat{\theta}) \quad f(y'|\theta', x) = f(y'|\hat{\theta}),$$

onde $\hat{\theta}$ denota uma estimativa de θ, por exemplo, que pode ser obtida por pseudo-verosimilhança.

Algoritmo Moøller

1. Gere θ' a partir da distribuição da proposta $q(\theta'|\theta_t)$.
2. Gere uma amostra exata y' da distribuição $f(y|\theta')$.
3. Aceite (θ', y') com probabilidade $\min(1, r)$, onde

$$r = \frac{f(x|\theta')f(\theta')f(y'|\hat{\theta})q(\theta_t|\theta')f(y|\theta_t)}{f(x|\theta_t)f(\theta_t)f(y|\hat{\theta})q(\theta'|\theta_t)f(y'|\theta')}.$$

Se for aceito, defina $(\theta_{t+1}, y_{t+1}) = (\theta', y')$.
Caso contrário, defina $(\theta_{t+1}, y_{t+1}) = (\theta_t, y_t)$.

7.5 O algoritmo de troca (Exchange Algorithm), (Murray, Ghahramani e MacKay, 2012)

Oobjetivo de amostrar da distribuição posterior $f(\theta|x)$ quando pode ter o problema de ter de avaliar a constante normalizadora. O algoritmo de troca é motivado pelo

algoritmo de têmpera paralela (Geyer, 1991; Hukushima e Nemoto, 1996) e pode ser descrito da seguinte forma. Considere a distribuição aumentada

$$f(y_1, \ldots, y_m, \theta|x) = \pi(\theta) f(x|\theta) \prod_{j=1}^{m} f(y_j|\theta_j),$$

onde θ_i são fixos, e $y_1, \ldots, y_m$ são variáveis auxiliares independentes com o mesmo espaço de estado que x e a distribuição conjunta $\prod_{j=1}^{m} f(y_j|\theta_j)$.

Suponha que uma mudança para θ seja proposta com probabilidade $q(\theta_i|\theta)$. Para garantir que $y_i = x$, trocamos as configurações de x e y_i. A relação MH resultante para a mudança é

$$\begin{aligned} r(\theta, \theta_i, y_i|x) &= \frac{\pi(\theta_i) f(x|\theta_i) f(y_i|\theta) \prod_{j \neq i} f(y_j|\theta_j) q(\theta|\theta_i)}{\pi(\theta) f(x|\theta) f(y_i|\theta_i) \prod_{j \neq i} f(y_j|\theta_j) q(\theta_i|\theta)} \\ &= \frac{\pi(\theta_i) f(x|\theta_i) f(y_i|\theta) q(\theta|\theta_i)}{\pi(\theta) f(x|\theta) f(y_i|\theta_i) q(\theta_i|\theta)}. \end{aligned}$$

Algoritmo de troca

1. Propor $\theta' \sim q(\theta'|\theta, x)$.
2. Gere uma variável auxiliar $y \sim f(y|\theta')$ usando um amostrador exato.
3. Aceite θ' com probabilidade $\min\{1, r(\theta, \theta', y|x)\}$, em que

$$r(\theta, \theta', y|x) = \frac{\pi(\theta') f(x|\theta') f(y|\theta) q(\theta|\theta')}{\pi(\theta) f(x|\theta) f(y|\theta') q(\theta'|\theta)}$$

Como uma mudança de troca entre (θ, x) e (θ', y) está envolvida, o algoritmo é chamado de algoritmo de troca. Geralmente, melhora o desempenho do algoritmo Moøller, pois elimina a necessidade de estimar o parâmetro antes do início da amostragem.
(Murray, Ghahramani e MacKay, 2012) relatam que o algoritmo de troca tende a ter uma maior probabilidade de aceitação para as amostras exatas que o algoritmo Møller.
O algoritmo de troca também pode ser visto como um algoritmo MCMC de variável auxiliar, com a distribuição da proposta sendo aumentada, para a qual a distribuição da proposta pode ser escrita como

$$T(\theta \to (\theta', y)) = q(\theta'|\theta) f(y|\theta'), \quad T(\theta' \to (\theta, y)) = q(\theta|\theta') f(y|\theta).$$

Isso simplesmente valida o algoritmo, seguindo os argumentos para as cadeias de Markov variáveis auxiliares feitas anteriormente.

7.6 O amostrador MH duplo

Embora o *algoritmo do Møller* e o *algoritmo de troca* funcionem para o modelo Ising, eles não podem ser aplicados a muitos outros modelos para os quais a amostragem exata não está disponível.

Além disso, mesmo para o modelo de Ising, a amostragem exata pode ser muito cara quando a temperatura está próxima ou abaixo do ponto crítico.

Para superar essa dificuldade, (Faming Liang, 2010) propõe o algoritmo MH duplo, que evita a exigência de amostragem exata, sendo as variáveis auxiliares geradas usando kernels MH e, portanto, pode ser aplicado a muitos modelos estatísticos para os quais a amostragem exata não está disponível ou está disponível ou é muito caro.

Suponha que queremos simular uma amostra y de $f(y|\theta')$. Se a amostra é gerada através de m atualizações MH começando com o estado atual x, a probabilidade de transição, $P_{\theta'}^{(m)}(y|x)$, é

$$P_{\theta'}^{(m)}(y|x) = K_{\theta'}(x \to x_1) \dots K_{\theta'}(x_{m-1} \to y),$$

onde $K(\cdot, \cdot)$ é o kernel de transição MH. Então:

$$\begin{aligned}
\frac{P_{\theta'}^{(m)}(x|y)}{P_{\theta'}^{(m)}(y|x)} &= \frac{K_{\theta'}(y \to x_{m-1}) \cdots K\theta'(x_1 \to x)}{K_{\theta'}(x \to x_1) \cdots K_{\theta'}(x_{m-1} \to y)} \\
&= \frac{f(x|\theta')}{f(y|\theta')} \frac{f(y|\theta')}{f(x|\theta')} \frac{K_{\theta'}(y \to x_{m-1}) \cdots K_{\theta'}(x_1 \to x)}{K_{\theta'}(x \to x_1) \cdots K_{\theta'}(x_{m-1} \to y)} \\
&= \frac{f(x|\theta')}{f(y|\theta')}
\end{aligned}$$

onde a última igualdade segue do *equilíbrio detalhado:*

$$f(x|\theta')K_{\theta'}(x \to x_1) \cdots K_{\theta'}(x_{m-1} \to y) = f(y|\theta')K_{\theta'}(y \to x_{m-1}) \cdots K_{\theta'}(x_1 \to x).$$

Voltando ao problema de simular a partir da distribuição posterior. A relação MH pode ser reexpressa como

$$r(\theta, \theta', y|x) = \frac{\pi(\theta')q(\theta|\theta')f(y|\theta)P_{\theta'}^{(m)}(x|y)}{\pi(\theta)q(\theta'|\theta)f(x|\theta)P_{\theta'}^{(m)}(y|x)}$$

É fácil ver que, se escolher $q(\theta'|\theta)$ como um núcleo de transição MH que satisfaça a condição de equilíbrio detalhada, então $\pi(\theta')q(\theta|\theta') = \pi(\theta)q(\theta'|\theta)$ e a atualização do *Exchange* é reduzida a uma atualização MH simples para a qual $f(x|\theta)$ funciona como a distribuição alvo e $P_{\theta}^{(m)}(y|x)$ funciona como a distribuição da proposta.

1. Simule uma nova amostra θ' de $\pi(\theta)$ usando o algoritmo MH começando com θ_t.

2. Gere uma variável auxiliar $y \sim P_{\theta'}^{(m)}(y|x)$ e aceite-a com probabilidade

$$\min\{1, r(\theta_t, \theta, y|x)\},$$

em que:

$$r(\theta_t, \theta, y|x) = \frac{f(y|\theta_t)P_{\theta'}^{(m)}(x|y)}{f(x|\theta_t)P_{\theta'}^{(m)}(y|x)} = \frac{f(y|\theta_t)f(x|\theta')}{f(x|\theta_t)f(y|\theta')}.$$

3. Defina $\theta_{t+1} = \theta'$ se a variável auxiliar for aceita na etapa (2) e defina $\theta_{t+1} = \theta_t$ caso contrário.

Capítulo 8

Métodos MCMC baseados na população

Matematicamente, o *MCMC de base populacional* pode ser descrito da seguinte maneira. Para simular a partir de uma distribuição alvo $f(x)$, simula-se um sistema aumentado com a distribuição invariante

$$f(x_1, \ldots, x_N) = \prod_{i=1}^{N} f_i(x_i),$$

onde $(x_1, \ldots, x_N) \in \mathcal{X}^N$, N é chamado de *tamanho da população*, $f(x) \equiv f_i(x)$ para pelo menos um $i \in \{1, 2, \ldots, N\}$ e aqueles diferentes de $f(x)$ são chamados de distribuições de teste em termos de amostragem de importância.

8.1 Amostragem de direção adaptativa *(ADS)*

A *amostragem por direção adaptativa (ADS)* (Gilks, Roberts e George, 1994) é um método MCMC de base populacional, no qual cada distribuição $f_i(x)$ é idêntica à distribuição alvo e, a cada iteração, uma amostra é selecionada aleatoriamente população atual e tem *uma atualização ao longo de uma linha que passa por outra amostra selecionada aleatoriamente* do restante conjunto da população atual. Uma forma importante do *ADS* é o *algoritmo de sinuca*.

A cada iteração, o algoritmo de sinuca construe uma *população de amostras*.

Seja $x^{(t)} = (x_1^{(t)}, \ldots, x_N^{(t)})$ denota a população obtida na iteração t, onde $x_i^{(t)}$ é chamado de *indivíduo da população*.

8.1.1 O algoritmo de sinuca

1. Selecione um indivíduo, $x_c^{(t)}$, aleatoriamente na população atual $x^{(t)}$. O $x_c^{(t)}$ é chamado de *ponto corrente*.

2. Selecione outro indivíduo, $x_a^{(t)}$, no conjunto restante da população atual (ou seja, $\{x_i^{(t)} : i \neq c\}$) e forme uma *direção*: $e_t = x_c^{(t)} - x_a^{(t)}$.
   O indivíduo $x_a^{(t)}$ é chamado de ponto de ancoragem.

3. Defina $y_c = x_a^{(t)} + r_t e_t$, onde r_t é um escalar amostrado da densidade

$$f(r) \propto |r|^{d-1} f(x_a^{(t)} + re_t),$$

   onde d é a dimensão de x, e o fator $|r|^{d-1}$ é derivado de um transformação jacobiana (Roberts e Gilks, 1994).

4. Forme a nova população $x^{(t+1)}$ substituindo $x_c^{(t)}$ por y_c e deixando todos os outros indivíduos inalterados (isto é, defina $x_i^{(t+1)} = x_i^{(t)}$ para $i \neq c$).

Para mostrar que o amostrador é adequado, precisamos mostrar que no equilíbrio a nova amostra y_c é independente do $x_i^{(t)}$ para $i = a$ e é distribuída como $f(x)$. Esse fato decorre diretamente do lema a seguir, uma versão generalizada do Lema de (Roberts e Gilks, 1994),

Lema 1. *Suponha que $x \sim \pi(x)$ e y seja qualquer ponto fixo em um espaço $d-$dimensional. Seja $e = x - y$.*

Se r é extraído da distribuição $f(r) \propto |r|^{d-1}\pi(y + re)$, então $x' = y + re$ segue a distribuição $\pi(x)$.
Se y é gerado a partir de uma distribuição independente de x, então x' é independente de y.

Demonstração. Assumimos que y é a origem e, em seguida, $e = x$. Se r é amostrado de $f(r) \propto |r|^{d-1}\pi(rx)$, então para qualquer função mensurável $h(x)$,

$$\mathbb{E}\{h(x')\} = \mathbb{E}[\mathbb{E}\{h(x')|x\}] = \int\int h(rx)\frac{|r|^{d-1}\pi(rx)}{\int |r'|^{d-1}\pi(r'x)dr'}\pi(x)drdx$$

Seja $g(x) = \int |r'|^{d-1}\pi(r'x)dr'$. Então $g(x)$ tem a propriedade de que $g(sx) = |s|^{-d}g(x)$. Seja $z = rx$, então

$$\begin{aligned}
\mathbb{E}\{h(x')\} &= \int\int h(z)\pi(z)|r|^{-1}\pi(r^{-1}z)/g(r^{-1}z)drdz \\
&= \int h(z)\pi(z)/g(z)\int |r|^{-d-1}\pi(r^{-1}z)drdz \\
&= \int h(z)\pi(z)dz = \mathbb{E}_\pi\{h(z)\}.
\end{aligned}$$

Assim, a amostra x' segue a distribuição $\pi(x)$. □

8.1.2 Gradiente conjugado Monte Carlo CGMC

Seja $x^{(t)} = (x_1^{(t)}, \ldots, x_N^{(t)})$ denotar a população atual de amostras. Uma iteração do amostrador *CGMC* consiste nas seguintes etapas.

Gradiente Conjugado Monte Carlo

1. Selecione um indivíduo, $x_c^{(t)}$, aleatoriamente na população atual $x^{(t)}$.

2. Selecione outro indivíduo, digamos $x_a^{(t)}$, aleatoriamente, do conjunto restante da população (ou seja, $\{x_i^{(t)} : i \neq c\}$).

   Começando com $x_a^{(t)}$, realize uma pesquisa determinística, usando o *método de gradiente conjugado* ou o *método de descida mais profunda*, para encontrar um modo local de *f(x)*. Denote o modo local por $z_a^{(t)}$, chamado ponto de ancoragem.

3. Defina $y_c^{(t)} = z_a^{(t)} + r_t e_t$, onde $e_t = x_c^{(t)} - z_a^{(t)}$ e r_t é um escalar amostrado da densidade
$$f(r) \propto |r|^{d-1} f(z_a^{(t)} + r_t e_t),$$
   onde d é a dimensão de x, e o fator $|r|^{d-1}$ é derivado da transformação jacobiana.

4. Forme a nova população $x^{(t+1)}$ substituindo $x_c^{(t)}$ por $y_c^{(t)}$ e deixando outros indivíduos inalterados (isto é, defina $x_i^{(t+1)} = x_i^{(t)}$ para $i \neq c$).

8.1.3 Algoritmo da amostra Metropolis-Hastings SMH

Na amostragem de direção adaptativa e no gradiente conjugado Monte Carlo, ao atualizar a população, primeiro se seleciona um indivíduo da população e depois atualiza o indivíduo selecionado usando o procedimento padrão de Metropolis-Hastings.

Se o estado candidato é de alta qualidade em relação a toda a população, certamente queremos mantê-lo na população.

No entanto, a aceitação do estado candidato depende da qualidade do indivíduo selecionado para atualização. Para melhorar a taxa de aceitação de candidatos de alta qualidade e para melhorar o conjunto $\{x_i^{(t)} : i = 1, \ldots, n\}$ como uma amostra do tamanho n de $f(x)$, (Lewandowski e Liu, 2008) propuseram o algoritmo de amostragem Metropolis-Hastings *SMH*

Algoritmo MH de amostra, SMH

1. Amostre um candidato $x_0^{(t)}$ de uma distribuição de proposta $g(x)$ em $\mathcal{X}$ e calcule a probabilidade de aceitação:
$$\alpha_0^{(t)} = \frac{\sum_{i=1}^{n} \frac{g(x_i^{(t)})}{f(x_i^{(t)})}}{\sum_{i=0}^{n} \frac{g(x_i^{(t)})}{f(x_i^{(t)})} - \min_{0 \leq k \leq n} \frac{g(x_k^{(t)})}{f(x_k^{(t)})}}.$$

2. Amostre $U \sim U(0,1)$ e defina

$$\begin{aligned}\mathcal{S}_{t+1} &= \left\{x_1^{(t+1)}, \ldots, x_n^{(t+1)}\right\} \\ &= \begin{cases}\mathcal{S}_t, & \text{se } U > \alpha_0^{(t)} \\ \left\{x_1^{(t)}, \ldots, x_{i-1}^{(t)}, x_0^{(t)}, x_{i+1}^{(t)}, \ldots, x_n^{(t)}\right\}, & \text{se } U \leq \alpha_0^{(t)},\end{cases}\end{aligned}$$

onde i é escolhido de $(1, \ldots, n)$ com os pesos de probabilidade

$$\left(\frac{g(x_1^{(t)})}{f(x_1^{(t)})}, \ldots, \frac{g(x_n^{(t)})}{f(x_n^{(t)})}\right)$$

Assim, $x^{(t+1)}$ e $x^{(t)}$ diferem em um elemento no máximo.

É fácil perceber que, no caso de $N = 1$, o *SMH* se reduz ao MH tradicional com propostas de independência.

O mérito do *SMH* é que, para aceitar um estado candidato, ele o compara com toda a população, em vez de um único indivíduo selecionado aleatoriamente da população atual.

(Lewandowski e Liu, 2008) mostram que o *SMH* convergirá para a distribuição alvo $\prod_{i=1}^{n} f(x_i)$ para $\{x_1, \ldots, x_n\}$ e pode ser mais eficiente que o MH tradicional e a amostragem de direção adaptativa.

8.1.4 Têmpera Paralela

O algoritmo de *têmpera paralela* (Geyer, 1991) é também conhecido como *intercâmbio Monte Carlo*. Na têmpera paralela, cada cadeia é equipada com uma distribuição invariante ligada por uma variável auxiliar, a chamada temperatura inversa.
Seja $f(x) \propto \exp(-H(x))$, $x \in \mathcal{X}$ a distribuição de interesse, onde $H(x)$ é chamado função de energia em termos de física.
Na estatística bayesiana, $H(x)$ corresponde à distribuição log-posterior negativa dos parâmetros.
A têmpera paralela simula em paralelo uma sequência de distribuições

$$f_i(x) \propto \exp(-H(x)/T_i), \quad i = 1, \ldots, n$$

onde T_i é a temperatura associada à distribuição $f_i(x)$. As temperaturas formam uma escada $T_1 > T_2 > \cdots > T_{n-1} > T_n \equiv 1$, então $f_n(x) \equiv f(x)$ corresponde à distribuição de destino.
A idéia subjacente a esse algoritmo pode ser explicada da seguinte maneira: O aumento da temperatura achata o cenário energético da distribuição e, assim, facilita a passagem de MH do espaço amostral, as amostras de alta densidade geradas nos níveis de alta temperatura podem ser transmitidas ao nível de temperatura alvo através de as operações de câmbio (descritas abaixo), e isso, por sua vez, melhora a convergência da cadeia alvo de Markov.

Seja $x^{(t)} = (x_1^{(t)}, \ldots, X_N^{(t)})$ denotar a população atual de amostras.
Uma iteração da têmpera paralela consiste nas seguintes etapas.

1. *Etapa MH paralela*: atualize cada $x_i^{(t)}$ para $x_i^{(t+1)}$ usando o algoritmo MH.

2. Etapa de troca de estado: tente trocar $x_i^{(t+1)}$ com seus vizinhos: Defina $j = i-1$ ou $i+1$ de acordo com as probabilidades $q_e(i,j)$, onde $q_e(i,i+1) = q_e(i,i-1) = 0,5$ para $1 < i < N$ e $q_e(1,2) = q_e(N,N-1) = 1$, e aceite a troca com probabilidade

$$\min\left\{1, \exp\left(\left[H(x_i^{(t+1)}) - H(x_j^{(t+1)})\right]\left[\frac{1}{T_i} - \frac{1}{T_j}\right]\right)\right\}.$$

Na prática, para ter uma taxa de aceitação razoável da troca proposta, as temperaturas precisam ser escolhidas com cuidado.
Quanto à têmpera simulada, recomendamos o seguinte método para definir a escala de temperatura de maneira seqüencial:
Começa com a temperatura mais alta T_1 e define a próxima temperatura mais baixa, de modo que

$$\mathrm{Var}_i(H(x))\delta^2 = O(1),$$

onde $\delta = 1/T_{i+1} - 1/T_i$, e a variância $\mathrm{Var}_i(\cdot)$ é obtida em relação a $f_i(x)$ e pode ser estimada através de uma execução preliminar.
A condição requer essencialmente que as distribuições nos níveis de temperatura vizinhos tenham uma sobreposição considerável.

8.2 Monte Carlo evolucionário (MCE)

O *algoritmo genético* (Holland, 1992) foi aplicado com sucesso a muitos problemas de otimização, como o problema do vendedor ambulante, dobragem de proteínas e aprendizado de máquina entre outros. Motivados pelo algoritmo genético, (Liang e Wong, 2000) propõem o *algoritmo evolutivo de Monte Carlo (EMC)*, que incorpora as características mais atraentes do algoritmo genético na estrutura da cadeia de Markov Monte Carlo.
O *MCE* funciona de maneira semelhante à *têmpera paralela*: uma população de cadeias de Markov é simulada em paralelo com cada corrente com uma temperatura diferente.
A diferença entre os dois algoritmos é que o *MCE* inclui um operador genético, a saber, o operador crossover em sua simulação. Os resultados numéricos indicam que o operador de crossover melhora a convergência da simulação e que a EMC pode superar a têmpera paralela em quase todos os cenários.

O *operador de crossover* pode ser usado na *MCE*. Suponha que a distribuição alvo seja,

$$f(x) \propto \exp\{-H(x)\}, x \in \mathcal{X} \subseteq \mathbb{R}^d,$$

onde a dimensão $d > 1$ e $H(x)$ é chamada de *função de ajuste* nos algoritmos genéticos. Seja $\{x = \{x_1, \ldots, x_N\}$ denota uma população de tamanho N com x_i da distribuição com densidade

$$f_i(x) \propto \exp\{-H(x)/T_i\}.$$

Nos algoritmos genéticos, x_i é chamado de *cromossomo ou indivíduo*, cada elemento de x_i é chamado de gene e uma realização do elemento é chamada de *genótipo*.
Como na têmpera paralela, as temperaturas formam uma escada decrescente $T_1 > T_2 > \cdots > T_N \equiv 1$, com $f_N(x)$ a distribuição alvo.

Monte Carlo evolucionário no espaço com código binário

Algoritmo *MCE* quando x_i é codificado por um vetor binário.
Seja $x = \{x_1, \ldots, x_N\}$ denota a *população* atual, onde $x_i = (\beta_{i,1}, \ldots, \beta_{i,d})$ é um vetor binário d-dimensional e $\beta_{i,j} \in \{0, 1\}$. Os operadores usados no algoritmo genético *MCE* são:

1. Mutação,
2. Cruzamento,
3. Troca.

8.2.1 Mutação

Na *mutação*, um *cromossomo*, x_k, é primeiro aleatoriamente selecionado da população atual x, depois é alterado para um novo cromossomo y_k, revertendo o valor $(0 \leftrightarrow 1)$ de alguns genótipos que também são selecionados aleatoriamente.

Uma nova população é formada como $y = \{x_1, \ldots, x_{k-1}, y_k, x_{k+1}, \ldots, x_N\}$ e é aceita com probabilidade $\min(1, r_m)$ de acordo com a regra Metropolis, Onde

$$r_m = \frac{f(y)T(x|y)}{f(x)T(y|x)} = \exp\left\{-\frac{H(y_k) - H(x_k)}{T_k}\right\}\frac{T(x|y)}{T(y|x)}.$$

e $T(\cdot|\cdot)$ denota a probabilidade de transição entre populações. Se a proposta for aceita, substitua a população atual x por y; caso contrário, mantenha x como a população atual.

Além das mutações de 1 e 2 pontos, também é possível usar a mutação uniforme na qual cada genótipo de x_k é mutado com uma probabilidade diferente de zero.
Todos esses operadores são simétricos, ou seja, $T(x|y) = T(y|x)$.

8.2.2 Crossover

Um par de cromossomos, x_i e x_j $(i = j)$, é selecionado da população atual x de acordo com algum procedimento de seleção, por exemplo, uma seleção de roleta ou uma seleção aleatória. Sem perda de generalidade, assumimos que $H(x_i) \geq H(x_j)$. Dois *descendentes* são gerados de acordo com algum *operador de cruzamento*, os descendentes com um *valor menor de ajuste* são denotados por y_j e o outro por y_i.

Uma nova população é formada como $y = \{x_1, x_{i-1}, y_i, x_{i+1}, x_{j-1}, y_j, x_{j+1}, x_N\}$.
De acordo com a regra do algoritmo Metropolis, a nova população é aceita com probabilidade $\min(1, r_c)$,

$$r_c = \frac{f(y)T(x|y)}{f(x)T(y|x)} = \exp\left\{-\frac{H(y_i) - H(x_i)}{T_i} - \frac{H(y_i) - H(x_i)}{T_j}\right\}\frac{T(x|y)}{T(y|x)},$$

onde:

1. $T(y|x) = P(x_i, x_j|x)P(y_i, y_j|x_i, x_j)$,
2. $P(x_i, x_j|x)$ é a probabilidade de seleção de (x_i, x_j) da população x,
3. $P(y_i, y_j|x_i, x_j)$ denota a probabilidade geradora de (y_i, y_j) a partir dos cromossomos parentais (x_i, x_j).

(Liang e Wong, 2000) recomendaram o seguinte procedimento para a seleção de cromossomos parentais:

1. Selecione o primeiro cromossomo x_i de acordo com um *procedimento de roleta* com probabilidade

   $$p(x_i) = \frac{\exp(-H(x_i)/T_s)}{\sum_{i=1}^{N}\exp(-H(x_i)/T_s)},$$

   onde T_s é chamado de temperatura de seleção e não é necessariamente o mesmo que T_i.
   Intuitivamente, se T_s for baixo, um cromossomo de alta qualidade provavelmente será selecionado na população atual para acasalar-se com outros.

2. Selecione o segundo cromossomo x_j aleatoriamente do restante da população.

Então, a probabilidade de seleção de (x_i, x_j) é

$$P((x_i, x_j)|x) = \frac{1}{(N-1)Z(x)}\left\{\exp\left(-\frac{H(x_i)}{T_s}\right) + \exp\left(-\frac{H(x_j)}{T_s}\right)\right\},$$

onde $Z(x) = \sum_{i=1}^{N}\exp(-H(x_i)/T_s)$.

A probabilidade $P((y_i, y_j)|y)$ pode ser calculada similarmente.

Existem muitos operadores de crossover possíveis que podem deixar a distribuição conjunta invariável. O operador de cruzamento de 1 ponto é talvez o mais simples.

Dados os dois cromossomos parentais, x_i e x_j, o operador de crossover de 1 ponto prossegue da seguinte maneira.

Primeiro, um ponto de cruzamento inteiro k é desenhado aleatoriamente do conjunto $\{1, 2, \ldots, d\}$; em seguida, x_i e x_j são construídos trocando os genótipos à direita do ponto de cruzamento entre os dois cromossomos parentais:

$$\begin{matrix}(x_{i,1}, x_{i,d}) \\ (x_{j,1}, x_{j,d})\end{matrix} \Rightarrow \begin{matrix}(x_{i,1}, x_{j,k}, x_{j,k+1}, x_{j,d}) \\ (x_{j,1}, x_{j,k}, x_{i,k+1}, x_{i,d})\end{matrix}$$

Se houver k ($k > 1$) pontos de cruzamento, ele é chamado de cruzamento de k-pontos.

8.2.3 Troca

Esta operação é a mesma usada na *têmpera paralela*. Dada a população atual x e a escada de temperatura t, $(x,t) = (x_1, T_1, \ldots, x_N, T_N)$, tenta-se fazer uma troca entre x_i e x_j sem alterar os t's.
A nova população é aceita com probabilidade $\min(1, r_e)$,

$$r_e = \frac{f(x')T(x|x')}{f(x)T(x'|x)} = \exp\left\{(H(x_i) - H(x_j))\left(\frac{1}{T_i} - \frac{1}{T_j}\right)\right\}$$

Normalmente, a troca é realizada apenas em níveis de temperatura vizinhos.

8.2.4 Algoritmo

Com base nos operadores descritos acima, o algoritmo pode ser resumido da seguinte forma.
Dada uma população inicial $x = \{x_1, \ldots, x_N\}$ e uma escada de temperatura $t = \{T_1, T_2, \ldots, T_N\}$, o *MCE* itera entre as duas etapas:

1. Aplique o operador de *mutação* ou *crossover* à população com probabilidade q_m e $1 - q_m$, respectivamente.
   O q_m é chamado de taxa de mutação.

2. Tente trocar x_i com x_j por N pares (i, j), com i sendo amostrado uniformemente em $\{1, \ldots, N\}$ e $j = i \pm 1$ com probabilidade $q_e(i, j)$, onde $q_e(i, i+1) = q_e(i, i-1) = 0,5$ e $q_e(1, 2) = q_e(N, N-1) = 1$.

Capítulo 9

Ponderação Dinâmica

O algoritmo Metropolis-Hastings possui um requisito rigoroso para a condição de equilíbrio detalhada. Para atravessar uma barreira de energia, o tempo de espera esperado é aproximadamente exponencial à diferença de energia. Portanto, o algoritmo sofre de um dilema de tempo de espera: esperar para sempre em um mínimo profundo de energia local ou ter uma distribuição de equilíbrio incorreta, em simulações de um sistema complexo para o qual o cenário energético é robusto.

(Wong e Liang, 1997) propuseram uma saída do dilema do tempo de espera, que pode ser descrito de maneira vaga da seguinte forma: Se necessário, o sistema pode fazer uma transição contra uma barreira de probabilidade íngreme sem um tempo de espera proporcionalmente longo. Para explicar o viés introduzido por meio disso, um peso de importância é calculado e registrado junto com os valores amostrados. Essa regra de transição não satisfaz mais a condição detalhada de equilíbrio, mas satisfaz o que é chamado de invariância em relação aos pesos de importância (IWIW).

No equilíbrio, as aproximações de Monte Carlo às integrais são obtidas pela média ponderada pela importância dos valores amostrados, e não pela média simples, como no algoritmo Metropolis-Hastings.

9.1 O princípio da invariância IWIW

Na *ponderação dinâmica*, o estado da cadeia de Markov é aumentado por um peso de importância para (x, w), onde o peso w carrega as informações das amostras anteriores e pode ajudar o sistema a escapar das armadilhas locais.
Seja (x_t, w_t) o estado atual da cadeia de Markov, uma transição dinâmica de ponderação envolve as seguintes etapas:

1. Amostre y de uma função de proposta $T(x_t, y)$.

2. Calcule a taxa dinâmica:

$$r_d = w_t \frac{f(y)T(y, x_t)}{f(x_t)T(x_t, y)}.$$

3. Seja θ_t um número não negativo, que pode ser definido como uma função de (x_t, w_t).
   Com probabilidade $a = r_d/(\theta_t + r_d)$, defina $x_{t+1} = y$ e $w_{t+1} = r_d/a$; caso contrário, defina $x_{t+1} = x_t$ e $w_{t+1} = w_t/(1-a)$.

Essa transição é chamada de *movimento do tipo R* em (Wong e Liang, 1997). Não satisfaz a condição de *balanço detalhado*, mas é invariante em relação ao *peso da importância (IWIW)*; isto é, se

$$\int w_t g(x_t, w_t) dw_t \propto f(x_t),$$

é valida, após de um passo de transição,

$$\int w_{t+1} g(x_{t+1}, w_{t+1}) dw_{t+1} \propto f(x_{t+1}),$$

também vale, onde $g(x, w)$ denota a densidade conjunta de (x, w).

Exemplo 13. *Considere a função $U(x) = (x-b)^2$, vamos a usar o algoritmo DW para simular valores que fiquem perto do mínimo b. para isso consideramos uma densidade proporcional à $f(x) = \exp(-U(x))$.*

```
  b=2
U<-function(u,b)
(u-b)^2
f<-function(u)
{
  exp(-U(u,b))
}
x <- seq(0,4,0.1)
plot(x,f(x),type = 'l')
```

No algoritmo DW anterior vamos a considerar $\theta = 0.01$ e $\theta = 1$ como a proposta $X_{t+1} \sim N(X_t, \tau), \tau = 0.2$ e $(w_0, X_0) = (c = 1, 0)$.

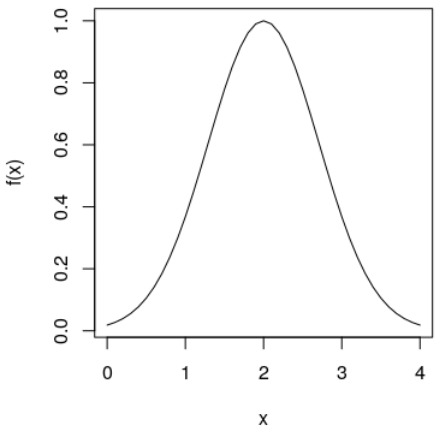

Figura 9.1: A função $f(x)$ com $b = 2$.

```
c<-1
T<-100
wx<-matrix(data = NA,nrow=2,ncol=T)
thetat<-0.01
tau<-0.2
wx[,1]<-c(c,0)
for (t in 2:T) {
  x<-rnorm(1,wx[2,t-1],tau)
  rd<-wx[1,t-1]*((f(x)*dnorm(x,wx[2,t-1],tau))
  /(f(wx[2,t-1])*dnorm(wx[2,t-1],x,tau)))
  a<-(rd/(thetat+rd))
  if(runif(1)<a)
  {
    wx[,t]<-c(rd/a,x)
  }
  else
  {
    wx[,t]<-c(wx[1,t-1]/(1-a),wx[2,t-1])
  }
}
x<-1:T
par(mfrow=c(2,1))
plot(x,wx[1,x], xlab = 't',
ylab = 'Pesos w_t',type='l'
,main =bquote(paste("theta=",.(deparse(thetat)))))
plot(x,wx[2,x],xlab = 't',ylab = ' x_t',type='l')
abline(h=2, col="red")
par(mfrow=c(1,1))
```

```
#hist(wx[2,x],xlab = 'Xt',
main =bquote(paste("Histograma de Xt com theta=",.(deparse(thetat)))))
boxplot(wx[2,x],xlab = 'Xt',
main =bquote(paste("Boxplot de Xt com theta=",.(deparse(thetat)))))
```

Obtemos (w_t, X_t) *para* $\theta = 1$,

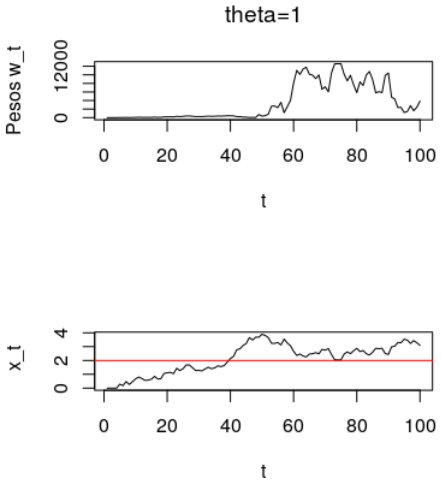

Figura 9.2: As trajetórias com $\theta = 1$

o resultado para X_t

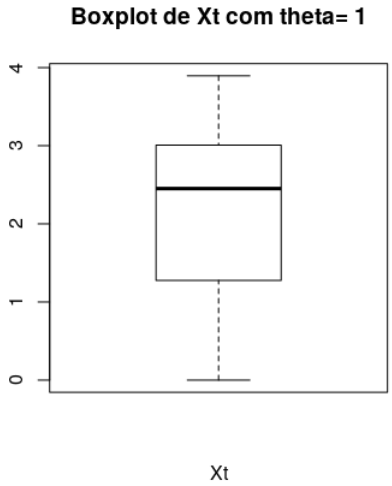

Figura 9.3: Boxplot para X_t com $\theta = 1$

Compare com (w_t, X_t) *para* $\theta = 0,01$*, e o resultado para* X_t

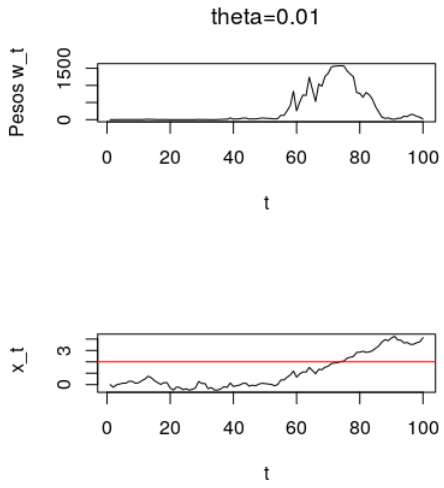

Figura 9.4: As trajetórias com $\theta = 0,01$

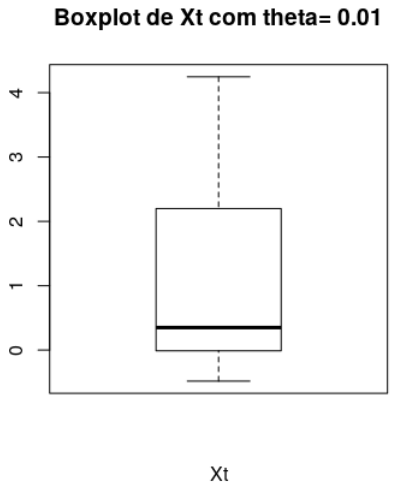

Figura 9.5: Boxplot para X_t com $\theta = 0,01$

A propriedade *IWIW* pode ser mostrada da seguinte maneira. Seja $x = x_t$,

$w = w_t$, $x' = x_{t+1}$ e $w' = w_{t+1}$. Então,

$$\begin{aligned}
&\int_0^\infty w' g(x, w') dw' \\
&\quad = \int_{\mathcal{X}} \int_0^\infty [\theta_t + r_d(x, x', w)] g(x, w) T(x, x') \frac{r_d(x, x', w)}{\theta_t + r_d(x, x', w)} dw dx \\
&\quad + \int_{\mathcal{X}} \int_0^\infty \frac{w[\theta_t + r_d(x', z, w)]}{\theta_t} g(x', w) T(x', z) \frac{\theta_t}{[\theta_t + r_d(x', z, w)]} dw dz \\
&\quad = \int_{\mathcal{X}} \int_0^\infty w g(x, w) \frac{f(x') T(x', x)}{f(x)} dw dx + \int_{\mathcal{X}} \int_0^\infty w g(x', w) T(x', z) dw dz \\
&\quad \propto f(x) \int_{\mathcal{X}} T(x', x) dx + f(x') = 2 f(x).
\end{aligned}$$

Portanto, dada uma sequência de amostras de ponderação dinâmica $(x_1, w_1), (x_2, w_2), \ldots, (x_n, w_n)$, a média ponderada de uma função $h(x)$:

$$\hat{\mu} = \sum_{i=1}^{n} w_i h(x_i) / \sum_{i=1}^{n} w_i,$$

convergirá para $\mathbb{E}_f h(x)$, a expectativa de $h(x)$ em relação à distribuição de destino $f(x)$.

O mérito da ponderação dinâmica é o seguinte: Se uma tentativa for rejeitada, a ponderação dinâmica será auto-ajustada para um valor maior dividindo a probabilidade de rejeição dessa tentativa, tornando uma probabilidade total menor de rejeição na próxima tentativa.

O uso de pesos de importância fornece um meio de ponderação dinâmica para fazer transições que não são permitidas pela regra MH padrão e, portanto, pode atravessar o cenário energético do sistema mais livremente. Mas essa vantagem tem um preço: os pesos de importância têm uma expectativa infinita, e a estimativa é de alta variabilidade e converge para os valores reais muito lentamente, aparentemente a uma taxa de $\log(n)$ (Liu, Liang e Wong, 2001).

Em resumo, o tempo de espera infinito no processo MH padrão agora se manifesta como uma quantidade infinita de peso no processo de ponderação dinâmica.

Para obter uma estimativa estável de $\mu = \mathbb{E}_f h(x)$, as operações de estratificação e ajuste dos pesos de importância geralmente são realizadas antes do cálculo da estimativa ponderada de μ.

Primeiro, as amostras são estratificadas de acordo com o valor da função $h(x)$. Os estratos são aproximadamente do mesmo tamanho e dentro de cada estrato a variação de $h(x)$ é pequena. Os pesos mais altos de $k\%$ (geralmente $k = 1$) de cada estrato são então truncados para o percentil $(100 - k)\%$ dos pesos dentro desse estrato.

Essas operações induzem um viés desprezível na estimativa ponderada, mas podem reduzir substancialmente sua variação. Observe que os pesos aparados dependem da função de interesse.

A transição usual de MH pode ser considerada como um tipo especial de transição de IWIW:

Se aplicarmos uma transição de MH para x e deixar w inalterado, o resultado será satisfatório para IWIW. Isso pode ser demonstrado do seguinte modo:

$$\begin{aligned}\int w' g(x', w') dw' &= \int w g(x, w) K(x \to x') dw dx \\ &\propto \int f(x) K(x \to x') dx = \int f(x') K(x' \to x) dx \\ &= f(x'),\end{aligned}$$

onde $K(\cdot \to \cdot)$ denota um núcleo de transição MH com $f(x)$ como sua distribuição invariante.
Portanto, as distribuições ponderadas corretamente permanecerão quando as transições dinâmicas de ponderação e as transições MH forem alternadas na mesma execução da cadeia de Markov.
Essa observação leva diretamente ao algoritmo de ponderação dinâmica de têmpera (Liang e Wong, 1999).

9.2 Algoritmo de ponderação dinâmica de têmpera

O revenido simulado (Marinari e Parisi, 1992) geralmente sofre com dificuldades na transição entre diferentes níveis de temperatura.
Para aliviar essa dificuldade, é preciso empregar muitos níveis de temperatura no sistema aumentado, que afetam adversamente a eficiência do algoritmo.
Alternativamente, essa dificuldade pode ser aliviada pela regra de ponderação dinâmica, conforme prescrita por (Liang e Wong, 1999).
O algoritmo de ponderação dinâmica de têmpera (*TDW*) (Liang e Wong, 1999) é essencialmente o mesmo que o algoritmo de têmpera simulado, exceto que um peso dinâmico está agora associado à configuração (x, i) e a regra de ponderação dinâmica é usada para guiar o transições entre níveis de temperatura adjacentes.

Seja $f_i(x)$ a distribuição do teste no nível $i, i = 1, \ldots, N$. Seja $0 < \alpha < 1$ especificado antecipadamente e seja (x_t, i_t, w_t) o estado atual da cadeia de Markov. Uma iteração do algoritmo *TDW* consiste nas seguintes etapas:

Algoritmo de ponderação dinâmica de têmpera

1. Amostre U da distribuição uniforme $U[0, 1]$.
2. Se $U \leq \alpha$, defina $i_{t+1} = i_t$ e $w_{t+1} = w_t$ e simule x_{t+1} de $f_{i_t}(x)$ por meio de uma ou várias atualizações de MH.
3. Se $U > \alpha$, defina $x_{t+1} = x_t$ e proponha uma transição de nível, $i_t \to i'$, usando uma proposta da função de transição $q(i_t, i')$. Realize uma *transição dinâmica de ponderação* para a atualização (i_t, w_t):
    - Calcule a taxa de ponderação dinâmica

$$r_d = w_t \frac{c_i f_{i'}(x_t) q(i', i_t)}{c_{i'} f_i(x_t) q(i_t, i')},$$

onde c_i denota a constante pseudo-normalizante de $f_i(x)$.

- Aceite a transição com probabilidade $a = r_d/(\theta_t + r_d)$, onde θ_t pode ser escolhido como uma função de (i_t, w_t). Se for aceito, defina $i_{t+1} = i'$ e $w_{t+1} = r_d/a$; caso contrário, defina $i_{t+1} = i_t$ e $w_{t+1} = w_t/(1-a)$.

Exemplo 14. *Vamos a considerar a função $U(x) = (x-b)^2$ do exemplo anterior, com $b = 2$ e três níveis $i \in \{1, 2, 3\}$com temperaturas $T_1 = 10, 5$ e 1. As transições entre estas temperaturas são propostas pela cadeia $\{i_n\}$ pela matriz q:*

$$q = \begin{pmatrix} 0.5 & 0.5 & 0 \\ 1/3 & 1/3 & 1/3 \\ 0 & 0.5 & 0.5 \end{pmatrix}$$

```
 ####
b<-2
U<-function(u,b)
(u-b)^2
### i= 1,2,3 niveis
#Ti temperaturas
T1<-10
T2<-5
T3<-1
##  fi~e^(-U/Ti) para cada nivel
fi<-function(u,i)
{
  f1<-if(i==1){exp(-U(u,b)/T1)}else 1
  f2<-if(i==2){exp(-U(u,b)/T2)}else 1
  f3<-if(i==3){exp(-U(u,b)/T3)}else 1
  return(f1*f2*f3)
}
integrando1<-function(u){fi(u,1)}
integrando2<-function(u){fi(u,2)}
integrando3<-function(u){fi(u,3)}
c<-c()
c1<-integrate(integrando1, -Inf, Inf)
c[1]<-c1$value
c2<-integrate(integrando2, -Inf, Inf)
c[2]<-c2$value
c3<-integrate(integrando3, -Inf, Inf)
c[3]<-c3$value
#grafico
require(ggplot2)
x <- seq(-10,10,0.1)
y1 <- fi(x,1)
```

```
y2 <- fi(x,2)
y3<- fi(x,3)
df <- data.frame(x,y1,y2,y3)
ggplot(df, aes(x)) +
  geom_line(aes(y=y1), colour="red") +
  geom_line(aes(y=y2), colour="green") +
  geom_line(aes(y=y3), colour="blue")+
  theme_bw() +
  theme(axis.line = element_line(colour = "black"),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.border = element_blank(),
        panel.background = element_blank())
#dev.off()
cste<-1
T<-1000
wxi<-matrix(data = NA,nrow=3,ncol=T)
#thetat<-0.01
thetat=1
alpha<-0.5
tau<-0.2
wxi[,1]<-c(cste,0,1)
q<-matrix(c(0.5,0.5,0,1/3,1/3,1/3,0,0.5,0.5),nrow=3, byrow = TRUE)
for (t in 2:T) {
  u<-runif(1)
  if(u<alpha){
    wxi[3,t]<-wxi[3,t-1]
    wxi[1,t]<-wxi[3,t-1]
    id=1
    mh_ite=5
    while(id<=mh_ite)
    {x<-rnorm(1,wxi[2,t-1],tau)
    rd<-((fi(x,wxi[3,t])*dnorm(x,wxi[2,t-1],tau))
    /(fi(wxi[2,t-1],wxi[3,t])*dnorm(wxi[2,t-1],x,tau)))
    if(runif(1)<rd)
    {wxi[2,t]<-x}
    else{wxi[2,t]<-wxi[2,t-1]}
    id<-id+1
    }
  }
  else
  { wxi[2,t]<-wxi[2,t-1]
    i<-sample(1:3,size=1,prob = q[wxi[3,t-1],])
    rd<-wxi[1,t-1]*(c[wxi[3,t-1]]*fi(wxi[2,t],i)
    *q[i,wxi[3,t-1]])/(c[i]*fi(wxi[2,t],wxi[3,t-1])
    *q[wxi[3,t-1],i])
    a<-rd/(thetat+rd)
    if(runif(1)<a)
    {wxi[3,t]<-i
```

```
      wxi[1,t]<-rd/a}
    else
    {wxi[3,t]<-wxi[3,t-1]
     wxi[1,t]<-wxi[1,t-1]/(1-a)}
  }

}
x<-1:T
par(mfrow=c(2,1))
plot(x,wxi[1,x], xlab = 't',ylab = 'Pesos w_t',
type='l',main =bquote(paste("theta=",.(deparse(thetat)))))
plot(x,wxi[2,x],xlab = 't',ylab = ' x_t',type='l')
abline(h=2, col="red")
par(mfrow=c(1,1))
#hist(wx[2,x],xlab = 'Xt', main = bquote(paste(
"Histograma de Xt com theta=", .(deparse(thetat)))))
boxplot(wxi[2,x],xlab = 'Xt', main = bquote(paste(
"Boxplot de Xt com theta=", .(deparse(thetat)))))
```

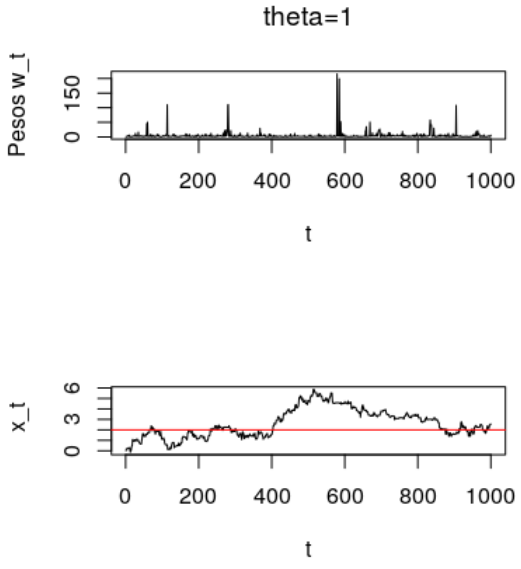

Figura 9.6: Trajetória da cadeia (w_t, x_t) com $\theta = 1$

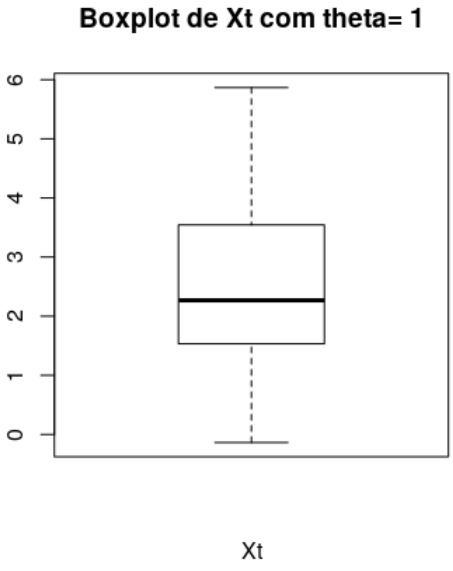

Figura 9.7: Boxplot de x_t com $\theta = 1$

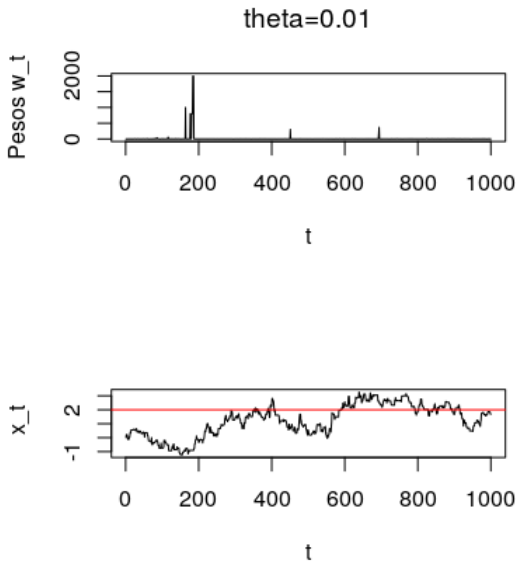

Figura 9.8: Trajetória da cadeia (w_t, x_t) com $\theta = 0.01$

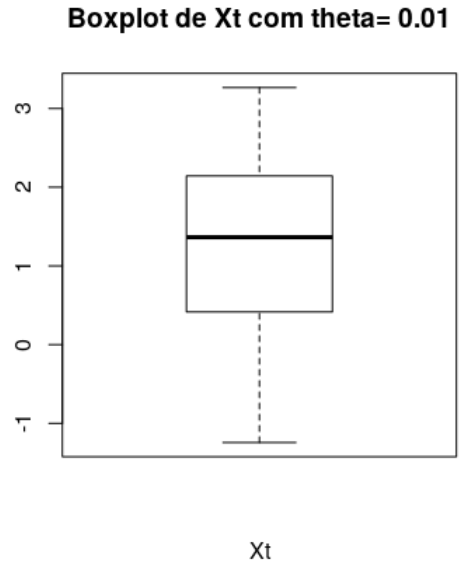

Figura 9.9: Boxplot de x_t com $\theta = 0.01$

9.2.1 Exemplo: Simulação do Modelo de Ising em temperatura subcrítica.

Exemplo 15. *Considere um modelo Ising* $2D$ *com a densidade de Boltzmann*

$$f(x) = \frac{1}{Z(K)} \exp\left\{ K \sum_{i \sim j} x_i x_j \right\},$$

onde os spins $x_i = \pm 1$, $i \sim j$ *indicam os vizinhos mais próximos na rede,* $Z(K)$ *é a função de partição e* K *é a temperatura inversa.*
Quando a temperatura está no ponto crítico ou abaixo dele ($K = 0,4407$*), o modelo possui dois estados magnetizados opostos, separados por uma barreira de energia muito íngreme.*
Devido à sua simetria, o modelo de Ising é mais passível de análise teórica. No entanto, para um algoritmo de amostragem que não depende da simetria do modelo, como têmpera simulada, paralela e ponderação dinâmica, isso é muito difícil, especialmente quando o tamanho do modelo é grande.
Na literatura, o modelo de Ising há muito tempo serve como exemplo de referência para testar a eficiência de novos algoritmos de Monte Carlo.

(Liang e Wong, 1999) realizaram simulações TDW nas redes de tamanho 322, 642 e 1282 usando níveis de temperatura 6, 11 e 21 (com os valores de K igualmente espaçados entre 0,4 e 0,5), respectivamente. No mesmo nível de temperatura, o amostrador Gibbs (Geman e Geman, 1984) é usado para gerar novas configurações; enquanto isso, os pesos foram mantidos inalterados. A regra de ponderação dinâmica

é usada apenas para controlar transições entre níveis. Após cada varredura das atualizações de Gibbs, propõe-se aleatoriamente mudar para um nível de temperatura adjacente com igual probabilidade. O parâmetro θ_t é definido como 1 se $w_t < 10^6$ e 0 caso contrário. Cinco execuções independentes foram realizadas para o modelo.
Em cada execução, a simulação continua até que 10.000 configurações sejam obtidas no nível final de temperatura.
Para o modelo de tamanho 1282, o número médio de varreduras em cada execução é 776 547.
Para o modelo de Ising, uma quantidade de interesse é a magnetização espontânea, definida por $M = \sum_i x_i/d^2$, em que d é o tamanho linear da rede.

A figura representa a magnetização espontânea obtida no nível $K = 0,5$ em uma execução para a rede 1282. Claramente, o algoritmo TDW conseguiu atravessar a barreira muito íngreme que separa os dois estados fundamentais e o sistema é capaz de atravessar livremente entre os dois poços de energia.

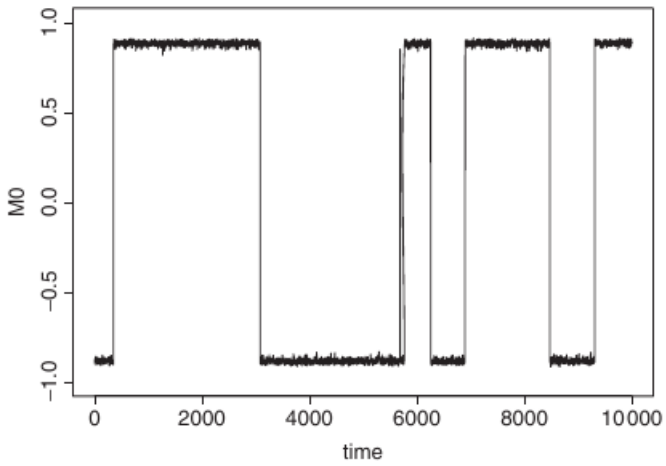

Figura 9.10: Magnetização espontânea contra iterações para uma rede de tamanho1282 no nível $K = 0,5$ (Liang e Wong, 1999).

A figura representa a expectativa de $|M|$, a magnetização espontânea absoluta,e para as redes de tamanho $K = 322$, 642 e 1282. A curva suave é o resultado da rede infinita (Onsager, 1949). O valor esperado de $|M|$ foi calculado pelo método de truncamento estratificado: Para cada execução, as 500 amostras iniciais foram descartadas, e as amostras restantes foram estratificadas em 10 estratos, de acordo com o valor de $|M|$ com os percentis ($j \times 10$) como limites dos estratos, $j = 1, 2, \ldots, 9$, e os pesos mais altos de 1% de cada estrato foram truncados para o percentil 99% dos pesos desse estrato.

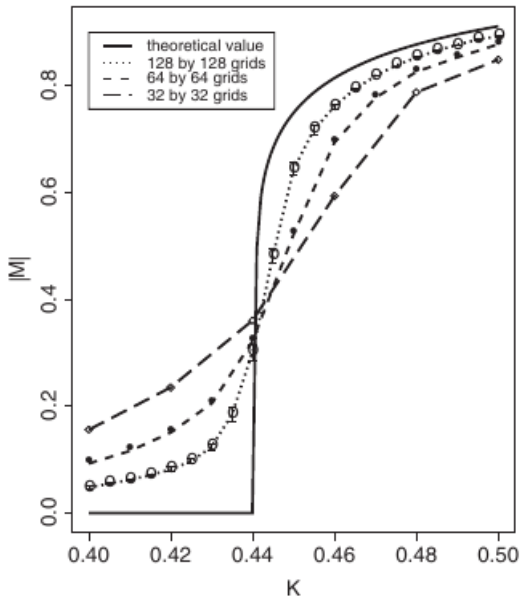

Figura 9.11: Expectativa de $|M|$ contra a temperatura inversa K. Os pontos são médias acima de 5 corridas independentes. A curva suave é da rede infinita (Liang e Wong, 1999).

9.2.2 Ponderação dinâmica na otimização

Exemplo 16. *Usamos o problema do vendedor ambulante (TSP) (Reinelt, 1994) para ilustrar o uso da ponderação dinâmica na otimização. Seja n o número de cidades em uma área e d_{ij} denote a distância entre a cidade i e a cidade j. O TSP deve encontrar uma permutação x das cidades para que o tamanho do passeio*

$$H(x) = \sum_{i=1}^{n-1} d_{x(i),x(i+1)} + d_{x(n),x_(1)},$$

é minimizado. Sabe-se que o TSP é um problema completo NP. Para aplicar a ponderação dinâmica ao TSP, (Wong e Liang, 1997) primeiro criaram uma ordem de construção sequencial das cidades.
Seja V o conjunto de cidades, seja A o conjunto de cidades que foram ordenadas e $A^c = V \setminus A$ o conjunto de cidades ainda não ordenadas. Em seguida, as cidades podem ser ordenadas da seguinte maneira:

- *Selecione aleatoriamente uma cidade de V.*
- *Repita as etapas (1) e (2) até que A^c esteja vazia:*
  1. *defina $k = \arg\max_{i \in A^c} \min_{j \in A} d_{ij}$;*
  2. *defina $A = A \cup \{k\}$ e $A^c = A^c \setminus \{k\}$.*

Este procedimento garante que cada vez que a cidade adicionada em A seja a que tenha a separação máxima do conjunto de cidades já ordenadas.

Uma sequência de TSPs de complexidade crescente foi considerada.
No nível mais baixo, o TSP inclui apenas as primeiras m_0 (por exemplo, 10 ou 15) cidades na ordem de construção. No próximo nível mais baixo, eles adicionaram um bloco das próximas m (por exemplo, $m = 5$) cidades para obter um TSP um pouco maior. O tamanho do bloco m depende do número de níveis que se deseja empregar. Assim, uma escada de complexidade composta por TSPs pode ser construída em sub-conjuntos cada vez maiores das cidades. Normalmente, são usados de 15 a 25 níveis e, geralmente, o tamanho s do TSP no nível mais alto de complexidade ainda é muito menor que o tamanho n do TSP original.
(Wong e Liang, 1997) sugerem escolher s entre $0,2n$ a $0,4n$, pois esse problema de s-cidade pode ser uma boa aproximação ao problema de n-cidade, no sentido de que um bom passeio para o primeiro pode ser um esboço de boas excursões completas para o último.

Dada a complexidade das cidades, a pesquisa mínima de passeios consiste em duas etapas. Primeiro, os passeios pela cidade são amostrados usando ponderação dinâmica da distribuição Boltzmann

$$f(x) \propto \exp(-H(x)),$$

onde $H(x)$ é o hamiltoniano anterior.

Para cada passeio pela cidade gerado por ponderação dinâmica, o algoritmo branch and bound é usado para inserir as cidades restantes, uma a uma de acordo com a ordem de construção, pesquisando uma série de pequenos espaços de caminho (15 cidades).
Na etapa de ponderação dinâmica, para adicionar uma cidade a um passeio, os 15 (digamos) vizinhos mais próximos do passeio são considerados e um novo sub-passeio por essas 16 cidades é amostrado.
Para adicionar um bloco de m cidades ao passeio, as cidades são adicionadas uma a uma, conforme descrito, e uma decisão de aceitação/rejeição é tomada depois que todas as m cidades são adicionadas com o peso da importância sendo atualizado de acordo.
A exclusão de cidades de um passeio segue um procedimento semelhante.

Capítulo 10

Aproximação Estocástica Monte Carlo

Considere amostragem de uma distribuição com a função densidade/massa

$$f(x) = \frac{1}{Z(\tau)} \exp\{-H(x)/\tau\}, x \in \S,$$

onde τ é chamado de temperatura e $H(x)$ a função de energia em termos de física. Algoritmos MCMC convencionais, como o algoritmo Metropolis-Hastings, amostram cada configuração $x \in \S$ com uma probabilidade proporcional ao seu *peso Boltzmann*

$$w_b(x) = \exp(-H(x)/\tau).$$

Na física estatística, esse tipo de simulação é chamado de simulação de *conjunto/ensemble canônico (CE)*, que produz a distribuição de probabilidade em forma de sino da energia $U = H(x)$:

$$\mathbb{P}_{\mathrm{CE}}(u) = \frac{1}{Z(\tau)} g(u) \exp(u/\tau),$$

onde $g(u)$ é chamado de densidade de estados (ou densidade espectral). As atualizações locais têm probabilidade $\exp(\triangle\ u/\tau)$ para o sistema atravessar uma barreira energética de $\triangle\ u$. Portanto, em baixas temperaturas, o amostrador tende a ficar preso em um dos mínimos de energia locais, tornando a simulação ineficaz.

10.1 Monte Carlo multicanônico

O *Monte Carlo multicanônico* (Berg e Neuhaus, 1992) procura coletar amostras em um conjunto onde a cada configuração com energia $u = H(x)$ é atribuida um peso

$$w_m(u) \propto \frac{1}{g(u)} = e^{-S(u)},$$

onde $S(u) = \log(g(u))$ é chamado de *entropia microcanônica*. Uma simulação com esta função de peso produzirá uma distribuição uniforme de energia:

$$P_m(u) \propto g(u)w_m(u) = \text{ constante},$$

e leva a uma *caminhada aleatória livre no espaço de energia*.

Isso permite que o amostrador escape de quaisquer barreiras de energia e explore quaisquer regiões do espaço da amostra, mesmo para aqueles com pequenos $g(u)$'s. As amostras geradas formarão um histograma plano no espaço de energia, portanto, o algoritmo é chamado de *algoritmo Monte Carlo de histograma plano* (Liang, 2006).

O procedimento inicializa a estimativa de $g(u)$ através de uma simulação de Monte Carlo com uma distribuição alvo *temperada* $f_T(x) \propto \exp(-H(x)/T)$.

Por simplicidade, suponha que a função de energia U tenha valores em um conjunto finito $\{u_1, \ldots, u_m\}$.
Seja $x_1, \ldots, x_N$ a amostra MCMC extraída de $f_T(x)$ e seja $N_T(i) = \sharp\{x_j : H(x_j) = u_i\}$ o número de amostras com energia u_i. Como,

$$N_T(i)/N \approx \frac{1}{Z(T)} g(u_i)e^{-u_i/T}, i = 1, \ldots, m$$

então a densidade espectral pode ser estimada por

$$\hat{g}(u_i) = \frac{N_T(i)e^{u_i}/T}{\sum_{j=1}^{m} N_T(i)e^{u_j}/T}, i = 1, \ldots, m.$$

Na prática, a temperatura T deve ser suficientemente alta para que cada valor da energia possa ser visitado com uma frequência razoavelmente grande. Dada a estimativa inicial da densidade espectral, o algoritmo multicanônico itera entre as duas etapas a seguir:

1. Execute um amostrador MCMC (o algoritmo MH), suficientemente longo de acordo com a função de ponderação atual

   $$w_m^{(t)}(x) \propto \frac{1}{\hat{g}_t(H(x))},$$

   onde t indexa os estágios da simulação.

2. Atualize a estimativa da densidade espectral por:

   $$\log(\hat{g}_{t+1}(u_i)) = c + \log(\hat{g}_t(u_i)) + \log(\hat{\pi}_t(i) - \alpha_i), i = 1, \ldots, m,$$

   onde a constante c é introduzida para garantir que $\log(\hat{g}_{t+1})$ é uma estimativa do $\log(g)$, e $\hat{\pi}_t(i)$ é a frequência de amostragem relativa da energia u_i no estágio t, e $\alpha_1, \ldots, \alpha_m$ são pequenas constantes positivas que servem como 'valores anteriores' para suavizar a estimativa $\hat{g}$.

Como o número de iterações realizadas na etapa 1 é suficientemente grande, é razoável supor que a simulação tenha atingido o equilíbrio e, portanto,

$$\hat{\pi}_t(i) \propto \frac{g(u_i)}{\hat{g}_t(u_i)}, i = 1, \ldots, m.$$

Substituindo isto acima, então $\log(\hat{g}_{t+1}(u_i)) = c + \log(g(u_i))$,$i = 1, \ldots, m$ o que implica a validade do algoritmo para estimar $g(u)$ (até uma constante multicanônica).
Por outro lado a independência de $\hat{g}_{t+1}(u)$ em relação à estimativa anterior $\hat{g}_t(u)$ implica que a estimativa da densidade espectral só pode alcançar precisão limitada, que é determinada pelo comprimento da simulação realizada na Etapa 1.
Após certo estágio, aumentar o número de estágios não melhorará a precisão da estimativa da densidade espectral.

Dado um conjunto de amostras multicanônicas $x_t^{(1)}, \ldots, x_t^{(N_t)}$ gerado no estágio t, a quantidade $\mathbb{E}_f\rho(x)$ (a expectativa de $\rho(x)$ em relação à distribuição alvo $f(x)$) pode ser estimada usando a técnica de *reponderação* por

$$\hat{\mathbb{E}}_f^{(t)}\rho(x) = \frac{\sum_{i=1}^{N_t} \hat{g}_t(H(x_t^{(i)}))\rho(x_t^{(i)})}{\sum_{i=1}^{N_t} \hat{g}_t(H(x_t^{(i)}))},$$

que é consistente como o estimador de amostragem de importância convencional. De um modo mais geral, $\mathbb{E}_f(x)$ pode ser estimado, com base nas amostras geradas em múltiplos estágios, por

$$\tilde{\mathbb{E}}_f(x) = \sum_{k=t_0+1}^{t} \lambda_k \hat{\mathbb{E}}_f^{(k)}(x),$$

onde t_0 indica o número de estágios de queima e λ_k's estão sujeitos à restrição $\sum_{k=t_0+1}^{t} \lambda_k = 1$ e pode ser escolhido para minimizar a variação total do estimador.

10.2 O algoritmo Wang-Landau, (WL)

Assim como o *Monte Carlo multicanônico*, o *algoritmo de Wang-Landau*, (Wang e Landau, 2001), procura coletar amostras em um conjunto onde cada configuração com energia u recebe um peso

$$w_m(u) \propto \frac{1}{g(u)},$$

onde $g(u)$ é a densidade espectral. A diferença entre os dois algoritmos está nos procedimentos de aprendizado para a densidade espectral. O algoritmo Wang-Landau pode ser considerado uma implementação inovadora do Monte Carlo multicanônico, mas vai além disso. No Monte Carlo multicanônico, o amostrador tende a ser bloqueado pela borda da área já visitada e leva muito tempo para atravessar uma área devido às características gerais da *caminhada aleatória*. O algoritmo *WL* consegue

remover esses problemas penalizando oportunamente a mudança e a permanência na energia que foi visitada muitas vezes.

Suponha que o espaço de amostra $\mathcal{X}$ seja finito e a função de energia $H(x)$ assuma valores em um conjunto finito $\{u_1, \ldots, u_m\}$. A simulação do algoritmo Wang-Landau consiste em várias etapas.
No primeiro estágio, ele começa com uma configuração inicial de $\hat{g}(u_1), \ldots, \hat{g}(u_m)$, digamos $\hat{g}(u_1) =, \ldots, = \hat{g}(u_m) = 1$ e uma amostra aleatória x_0 extraída de $\mathcal{X}$ e itera entre as seguintes etapas,

Algoritmo Wang-Landau

1. Simule uma amostra x com uma única atualização do Metropolis que admita a distribuição invariânte:
$$f(x) \propto 1/g(H(x)).$$

2. Defina
$$\hat{g}(u_i) \leftarrow \hat{g}(u_i)\delta^{I(H(x)=u_i)},$$
para $i = 1, \ldots, m$, onde δ é um fator de modificação maior que 1 e $I(\cdot)$ é a função indicadora.

O algoritmo intera até que um *histograma plano* seja produzido no espaço de energia. Um histograma é geralmente considerado plano se a frequência de amostragem de cada interface do usuário não for inferior a 80% da frequência média de amostragem. Quando essa condição é satisfeita, as estimativas $\hat{g}(u_i)$ e a amostra atual x são transferidas para o estágio seguinte como valores iniciais, o *fator de modificação* é reduzido para um valor menor de acordo com um esquema especificado, digamos, $\delta \leftarrow \sqrt{\delta}$, e o coletor de amostras é retomado.
A próxima etapa da simulação é iniciada, continuando até que o novo histograma esteja nivelado novamente. O processo é repetido até que δ esteja muito próximo de 1, digamos, $\log(\delta) < 10^{-8}$.

(Liang, 2005) generalizou o algoritmo Wang-Landau para sistemas contínuos. A generalização é principalmente em três aspectos, a saber, o espaço amostral, a função de trabalho e o esquema de atualização de estimativas.
Suponha que o espaço amostral $\mathcal{X}$ seja contínuo e tenha sido particionado de acordo com uma parametrização escolhida de x, digamos, a função energética $H(x)$, em m sub-regiões disjuntas: $E_1 = \{x : H(x) \leq u_1\}$, $E_2 = \{x : u_1 < H(x) \leq u_2\}$, $\ldots$, $E_{m-1} = \{x : u_{m-2} < H(x) \leq u_{m-1}\}$ e $E_m = \{x : H(x) > u_{m-1}\}$, em que $\infty < u_1 < u_2 < \cdots < u_{m-1} < \infty$ são especificados pelo usuário. Seja $\psi(x)$ uma função não negativa definida no espaço de amostra com $0 < \int_{\mathcal{X}} \psi(x)dx < \infty$, que também é chamada de função de trabalho do algoritmo Wang-Landau generalizado. Na prática, geralmente se define $\psi(x) = \exp(-H(x)/\tau)$. Seja $g = (g^{(1)}, \ldots, g^{(m)})$ o peso associado às sub-regiões.

Uma iteração do algoritmo generalizado de Wang-Landau consiste nas seguintes etapas,

Algoritmo generalizado de Wang-Landau

1. Simule uma amostra x por um número, denotado por k, de etapas MH que admite
$$\hat{f}(x) \propto \sum_{i=1}^{m} \frac{\psi(x)}{\hat{g}^{(i)}} I(x \in E_i),$$
como a distribuição invariante.

2. Seja $\hat{g}^{(k)} \leftarrow \hat{g}^{(k)} + \delta\rho^{k-J(x)}\hat{g}(J(x))$ para $k = J(x), \ldots, m$, onde $J(x)$ é o índice da sub-região à qual x pertence e $\rho > 0$ é um parâmetro que controla a frequência de amostragem para cada uma das sub-regiões.

A medida que o número de estágios se torna grande, ver (Liang, 2005)

$$\hat{g}^{(1)} \approx g^{(1)} = c\int_{E_1} \psi(x)dx,$$
$$\hat{g}^{(i)} \approx g^{(i)} = c\int_{E_i} \psi(x)dx + \rho g^{(i-1)}, i = 2, \ldots, m$$

onde c é uma constante desconhecida; e a frequência de visitas a cada sub-região será aproximadamente proporcional a $\int_{E_i} \psi(x)dx/g^{(i)}$ para $i = 1, \ldots, m$.

A generalização da densidade espectral para a integral $\int_{E_i} \psi(x)dx$ é de grande interesse para estatísticos, o que leva a uma ampla gama de aplicações do algoritmo em estatística, como a seleção de modelos e alguns outros problemas computacionais bayesianos, como discutido em (Liang, 2005).

Exemplo 17 (pacote PAWL do R). `library(PAWL)`

```
### Exemplo com tres estados
# distribution alvo:
targetpdf <- c(0.5, 0.4, 0.1)
# log probabilidade do alvo:
parameters <- list(logpi = log(targetpdf))
logdensity <- function(x, parameters){
  parameters$logpi[x]
}
# proposta
transitionmatrix <- t(matrix(c(0.5, 0.5, 0.0,
                               0.3, 0.5, 0.2,
                               0.0, 0.6, 0.4), ncol = 3))

proposalparam <- list(transitionmatrix =
transitionmatrix, card = 3)
# function that generates proposal:
rproposal <- function(states, proposalparam){
  for (index in 1:length(states)){
    states[index] <- sample(x = 1:proposalparam$card,
                            size = 1, prob =
 proposalparam$transitionmatrix[states[index],])
  }
```

```
  return(list(states = states))
}

# funcao para calcular a densidade da proposta
dproposal <- function(states, ys, proposalparam){
  for (index in 1:(length(states))){
    states[index] <- log(transitionmatrix[states[index], ys[index]])
  }
  return(states)
}

proposalinstance <- proposal(rproposal = rproposal,
                             dproposal = dproposal,
                             proposalparam = proposalparam)

# funcao para os pontos iniciais do  MCMC :
rinit <- function(size) return(rep(1, size))
# define the target
discretetarget <- target(name = "discrete toy example",
dimension = 1, type = "discrete",
                         rinit = rinit,
                         logdensity = logdensity,
                         parameters = parameters)
# especificar parametros do Metropolis-Hastings:
mhparameters <- tuningparameters(nchains = 1,
niterations = 10000, storeall = TRUE)
# Rprof(tmp <- tempfile())
amhresults <- adaptiveMH(discretetarget,
mhparameters, proposalinstance)
# Rprof()
# print(summaryRprof(tmp))
# unlink(tmp)
chains <- ConvertResults(amhresults)

cat("AMH: probabilidades alvo:", targetpdf, "\n")
amhcount <- tabulate(chains$X1, nbins = 3)
cat("AMH: frequencias obtidas:", amhcount / sum(amhcount), "\n")

# particionamos para que 1 e 2 estejam no bloco 1, e estado 3 no bloco 2
getPos <- function(points, logdensity) 2 - (points <= 2)
# mais parametros:
positionbinning <- binning(position = getPos,
                           name = "position",
                           bins = c(1, 2),
                           desiredfreq = c(0.8, 0.2),
                           useLearningRate = FALSE)
pawlresults <- pawl(discretetarget,
binning = positionbinning, AP = mhparameters, proposalinstance)
pawlchains <- ConvertResults(pawlresults)
```

```
cat("PAWL: freq. desejadas:", positionbinning@desiredfreq, "\n")
pawlcount <- tabulate(getPos(pawlchains$X1, pawlchains$logdens), nbins = 2)
cat("PAWL: freq obtidas:", pawlcount / sum(pawlcount), "\n")
# graficando log theta:
plot(pawlresults$logtheta[,1],type='l')
plot(pawlresults$logtheta[,2],type='l')
counts <- tabulate(pawlchains$X1, nbins = 3)
counts <- counts[1:2]
counts <- counts / sum(counts)
cat("PAWL: proporcoes  no bloco 1:", counts, "\n")
cat("PAWL: comparar a:", targetpdf[1:2] / sum(targetpdf[1:2]), "\n")
```

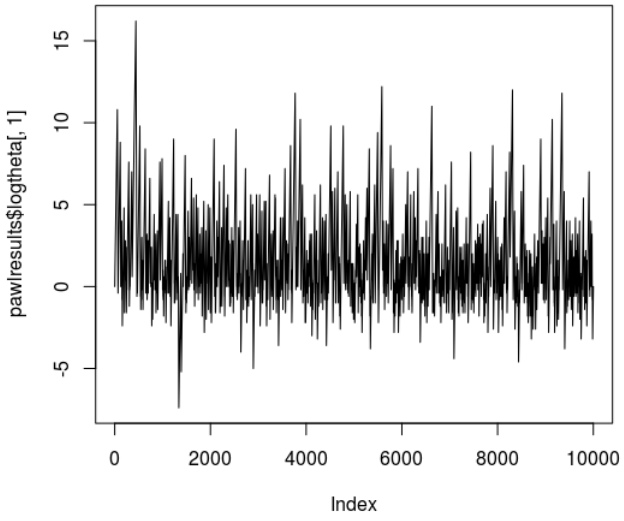

Figura 10.1: Cadeia $\log \theta_1$

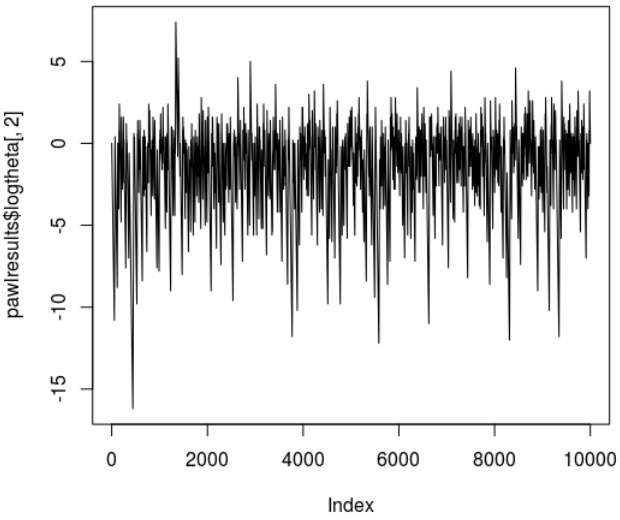

Figura 10.2: Cadeia $\log \theta_2$

10.3 Aproximação Estocástica Monte Carlo *SAMC*

Considere o problema de amostragem da distribuição$f(x) = \exp(-H(x)/\tau)/Z(\tau)$. Suponha que, como no algoritmo generalizado de Wang-Landau (Liang, 2005), o espaço amostral $\mathcal{X}$ tenha sido particionado de acordo com uma função $\lambda(x)$ em m sub-regiões, $E_1 = \{x : \lambda(x) \leq u_1\}, E_2 = \{x : u_1 < \lambda(x) \leq u_2\}, \ldots, E_{m-1} = \{x : u_{m-2} < \lambda(x) \leq u_{m-1}\}, E_m = \{x : \lambda(x) \geq u_{m-1}\}$, onde $-\infty < u_1 < \cdots < u_{m-1} < \infty$. Aqui $\lambda(\cdot)$ pode ser qualquer função de x, como um componente de x, a função de energia $H(x)$, etc. Seja $\psi(x)$ uma função não negativa com $0 < \int_{\mathcal{X}} \psi(x)dx < \infty$, chamada de *função de trabalho do SAMC*. Na prática se define $\psi(x) = \exp(-H(x)/\tau)$. Seja $g_i = \int_{E_i} \psi(x)dx$ para $i = 1, \ldots, m$ e $g = (g_1, \ldots, g_m)$. A sub-região E_i é chamada de sub-região vazia se $g_i = 0$. Uma especificação inadequada dos pontos de corte u_i's pode resultar em algumas sub-regiões vazias. Tecnicamente, o SAMC permite a existência de sub-regiões vazias em simulações. Para apresentar claramente a ideia, assumimos temporariamente que todas as sub-regiões não são vazias; isto é, assumindo $g_i > 0$ para todos $i = 1, \ldots, m$. O *SAMC* procura coletar amostras da distribuição

$$f_g(x) \propto \sum_{i=1}^{m} \frac{\pi_i \psi(x)}{g_i} I(x \in E_i),$$

onde π_i's são valores de frequência pré-especificados, de modo que $\pi_i > 0$ para todos i e $\sum_{i=1}^{m} \pi_i = 1$.

É fácil ver, se $g_1, \ldots, g_m$, a amostragem de $f_g(x)$ resultará em uma *caminhada*

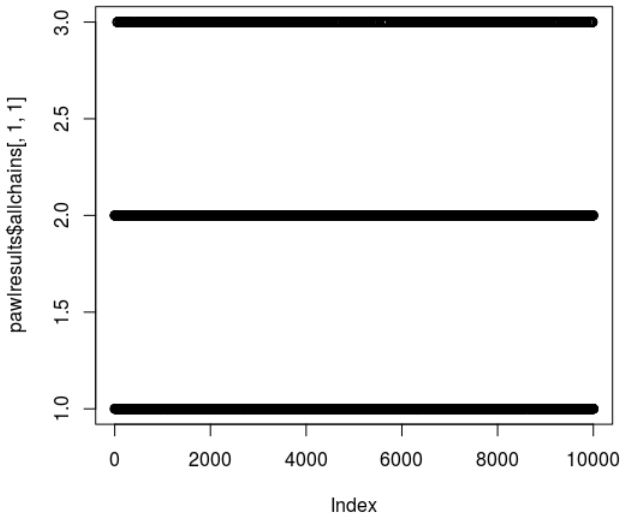

Figura 10.3: Cadeia dos estados $x_t = 1, 2, 3$

aleatória no espaço das sub-regiões, com cada sub-região sendo visitada com uma frequência proporcional a π_i. A distribuição $\pi = (\pi_1, \ldots, \pi_m)$ é denominada *distribuição de amostragem desejada das sub-regiões*. Como g é desconhecido, (Liang, Liu e Carroll, 2007) estimam-o sob a estrutura do algoritmo de aproximação estocástica (Robbins e Monro, 1951), o que garante consistência na estimativa de g. Seja $\theta_t^{(i)}$ a estimativa de $\log(g_i/\pi_i)$ obtida na iteração t, e seja $\theta_t = (\theta_t^{(1)}, \ldots, \theta_t^{(m)})$. Como m é finito, a partição $E_1, \ldots, E_m$ é fixo e $0 < \int_{\mathcal{X}} \psi(x) < \infty$, deve existir um número $\epsilon > 0$ tal que

$$\epsilon < \min_i \log(\frac{g_i}{\pi_i}) < \max_i \log(\frac{g_i}{\pi_i}) < \frac{1}{\epsilon},$$

o que implica que θ_t pode ser restrito à obtenção de valores em um conjunto compacto. A partir de agora, este conjunto compacto será indicado por Θ. Na prática, Θ pode ser definido para um conjunto enorme, digamos, $\Theta = [-10^{100}, 10^{100}]^m$. Por uma questão prática, isso é equivalente ao conjunto $\Theta = \mathbb{R}^m$. Caso contrário, se se assumir $\Theta = \mathbb{R}^m$, uma versão variável do truncamento desse algoritmo poderá ser considerada, como em Liang (2009e).

Seja $\{\gamma_t\}$ a sequência do *fator de ganho*, que satisfaz a condição (A1):

(A1) A sequência $\{\gamma_t\}$ é positiva e não aumenta, e satisfaz as condições:

(i) $\bar{\lim}_{t\to\infty} |\gamma_t^{-1} - \gamma_{t+1}^{-1}| < \infty$

(ii) $\sum_{t=1}^{\infty} \gamma_t = \infty$

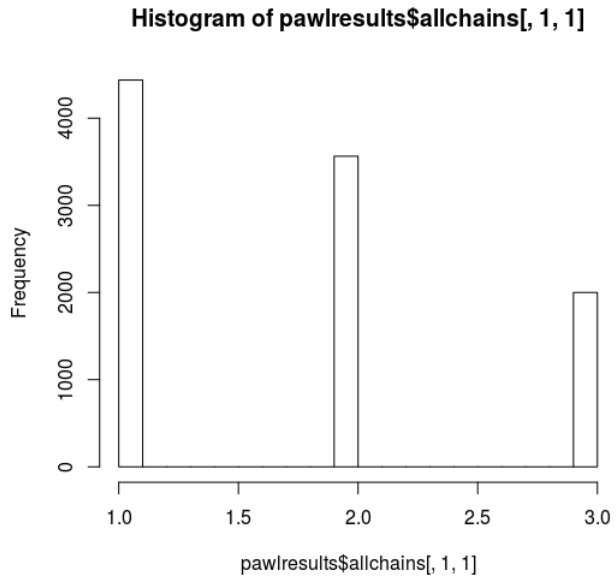

Figura 10.4: Frequencia dos estados $f_{x_t=i}, i = 1, 2, 3$

(iii) $\sum_{t=1}^{\infty} \gamma_t^{\eta} < \infty$ para alguns $\eta \in (1, 2]$.

Na prática,

$$\gamma_t \frac{t_0}{\max\{t_0, t_\xi\}} = t =, t = 0, 1, 2, \ldots,$$

para alguns valores pré-especificados $t_0 > 1$ e $1/2 < \xi \leq 1$. Um grande valor de t_0 permitirá que o amostrador alcance todas as sub-regiões muito rapidamente, mesmo para um sistema grande. Seja $J(x)$ o índice da sub-região à qual a amostra x pertence.

O *SAMC* começa com uma amostra aleatória x_0 gerada no espaço $\mathcal{X}$ e uma estimativa inicial $\theta_0 = (\theta_0^{(1)}, \ldots, \theta_0^{(m)}) = (0, \ldots, 0)$ e depois itera entre as etapas a seguir:

O algoritmo SAMC

(a) Amostragem Simule uma amostra x_{t+1} por uma única atualização MH que admita a seguinte distribuição como distribuição invariante:

$$f_{\theta_t}(x) \propto \sum_{i=1} \frac{\psi(x)}{\exp(\theta_t^{(i)})} I(x \in E_i).$$

(a.1) Gere y no espaço amostral $\mathcal{X}$ de acordo com uma distribuição $q(x_t, y)$.

(a.2) Calcule a ração

$$r = e^{\theta_t^{(J(x_t))} - \theta_t^{(J(y))}} \frac{\psi(y) q(y, x_t)}{\psi(x_t) q(x_t, y)}.$$

(a.3) Aceite a proposta com probabilidade $\min(1, r)$. Se for aceito, defina $x_{t+1} = y$; caso contrário, defina $x_{t+1} = x_t$.

(b) Atualização dos pesos Para $i = 1, \ldots, m$ faça

$$\theta_{t+1/2}^{(i)} = \theta_t^{(i)} + \gamma_{t+1}\left(I_{\{x_{t+1}\in E_i\}} - \pi_i\right).$$

Se $\theta_{t+1/2} \in \Theta$, defina $\theta_{t+1} = \theta_{t+1/2}$; caso contrário, defina $\theta_{t+1} = \theta_{t+1/2} + c^*$, onde $c^* = (c^*, \ldots, c^*)$ pode ser qualquer vetor constante que satisfaça a condição $\theta_{t+1/2} + c^* \in \Theta$.

Exemplo 18 (pacote SAMC do R)**.** *Vamos a amostrar da distribuição multimodal:*

$$f(x) \propto \exp\{H(x)\},$$

onde $x = (x_1, x_2) \in [1.1, 1.1]^2$ *e*

$$\begin{aligned} H(x) = &-\{x_1 \sin(20x_2) + x_2 \sin(20x_1)\}^2 \cosh\{\sin(10x_1)x_1\} \\ &- \{x_1 \cos(10x_2) - x_2 \sin(10x_1)\}^2 \cosh\{\cos(20x_2)x_2\}. \end{aligned}$$

```
 library(SAMCpack)

##passo 1: Defina  log-densidade negativa
func_r = function(x){
  x1 = x[1]; x2 = x[2];
  val1 = (-(x1*sin(20*x2)+x2*sin(20*x1))^2)*cosh(sin(10*x1)*x1);
  val2 = (-(x1*cos(10*x2)-x2*sin(10*x1))^2)*cosh(cos(20*x2)*x2);
  return(val1+val2);
}


## paso 2 : Defina os parametros do algorimo
myoption =list()
myoption$partition = c(-Inf,seq(from=-8,to=0,length.out=41))
myoption$tau= 1.0
myoption$domain= c(-1.1,1.1)
myoption$vecpi= as.vector(rep(1/41,41))
myoption$niter= 200000
myoption$stepsize= c(0.25, 10)
## Step 3 : rode o codigo
res = SAMC(2,func_r,options=myoption)
## Step 4 : Visualize
select = seq(from=101,to=myoption$niter,by=100)
# 100 burn-in, 1/100 thinning
par(mfrow=c(1,2))
plot(res$samples[select,1], res$samples[select,2],
xlab='x',ylab='y',main='Amostras')
barplot(as.vector(res$frequency/sum(res$frequency)),
        main="Frequencia das visitas por particao da  energia",
        names.arg=myoption$partition[-1], xlab="energia")
```

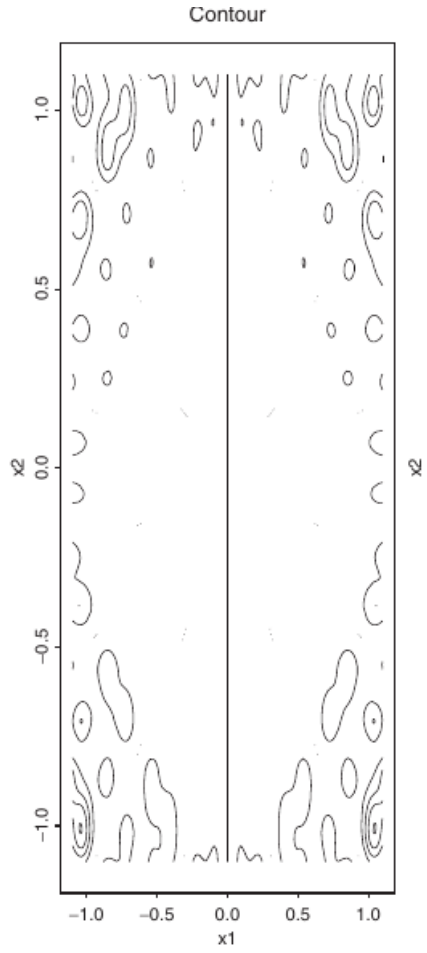

Figura 10.5: Contorno da função $H(x)$

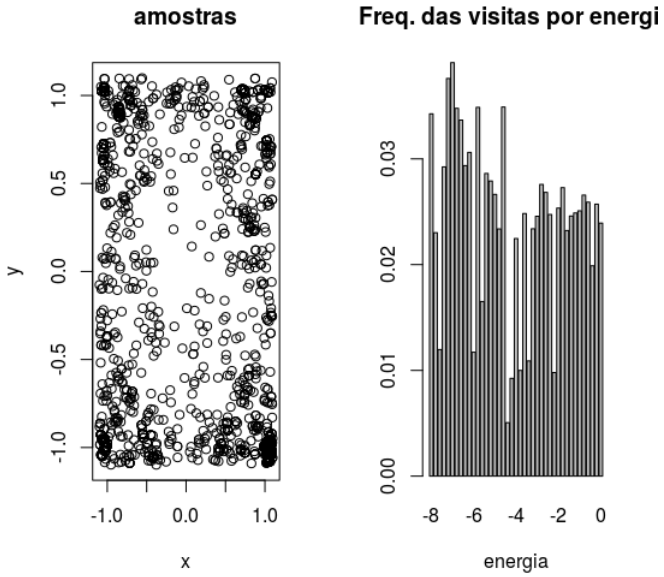

Figura 10.6: Simulação da função H usando o algoritmo $SAMC$

Capítulo 11

Hamiltonian Monte Carlo HMC

Os métodos de amostragem Hamiltoniano Monte Carlo (HMC) (Duane et al., 1987)e (Neal, 2011) fornecem um poderoso algoritmo de amostragem de Monte Carlo em cadeia de Markov (MCMC).
Os métodos definem uma função hamiltoniana em termos da distribuição alvo a partir da qual desejamos amostras - a energia potencial - e um termo de energia cinética parametrizado por um conjunto de variáveis auxiliares de *momento*.
Com base em atualizações simples para as variáveis de momento, simula-se a partir de um sistema dinâmico hamiltoniano que permite propostas de estados distantes.
A distribuição de destino é invariável sob essa dinâmica; na prática, é necessária uma discretização do sistema de tempo contínuo, necessitando de uma correção de Metropolis-Hastings (MH), embora ainda com alta probabilidade de aceitação. Com base nas propriedades atraentes do HMC em termos de exploração rápida do espaço de estados, os métodos do HMC cresceram em popularidade recentemente.

Suponha que desejemos amostrar a partir da **distribuição posterior** de θ dado um conjunto de observações independentes $x \in \mathcal{X}$:

$$p(\theta|x) \propto \exp(-U(\theta)),$$

onde a **função potencial de energia** U é dada por

$$U(\theta) = -\sum_{x \in \mathcal{X}} \log p(x|\theta) - \log(\theta).$$

O **Monte Carlo Hamiltoniano (Híbrido)** (**HMC**) fornece um método para propor amostras de θ em uma estrutura Metropolis-Hastings (MH) que explora com eficiência o espaço de estados em comparação com o padrão de caminhos aleatórios. Essas propostas são geradas a partir de um sistema hamiltoniano baseado na introdução de um conjunto de **variáveis de momento auxiliares**, r. Ou seja, para amostrar a partir de $p(\theta|\mathcal{X})$, o HMC considera gerar amostras a partir da distribuição conjunta de (θ, r) definida por

$$\pi(\theta, r) \propto \exp\left(-U(\theta) - \frac{1}{2} r^T M^{-1} r\right).$$

Se simplesmente descartarmos as r amostras resultantes, as amostras θ terão distribuição marginal $p(\theta|\mathcal{X})$. Aqui, M é uma matriz de **massa** e, juntamente com r, define um **termo de energia cinética**. M geralmente é definido como a matriz de identidade I, mas pode ser usado para pré-condicionar o amostrador quando tivermos mais informações sobre a distribuição de destino. A função **Hamiltoniana** é definida por

$$H(\theta, r) = U(\theta) + \frac{1}{2} r^T M^{-1} r.$$

H mede a energia total de um sistema físico com variáveis de posição θ e variáveis de momento r.

Para propor amostras, o HMC simula a dinâmica hamiltoniana

$$d\theta = M^{-1} r dt$$
$$dr = \bigtriangledown U(\theta) dt.$$

Para se ter uma idéia concreta, uma analogia comum em 2D é a seguinte (Neal, 2011). *Imagine um disco de hóquei deslizando sobre uma superfície de gelo sem atrito de altura variável. O termo energia potencial é baseado na altura da superfície na posição atual do disco, θ, enquanto a energia cinética é baseada no momento do disco, r e sua massa, M. Se a superfície for plana ($\bigtriangledown U(\theta) = 0, \forall \theta$), o disco se move a uma velocidade constante.*
Para declives positivos ($\bigtriangledown U(\theta) > 0$), a energia cinética diminui à medida que a energia potencial aumenta até que a energia cinética seja 0 ($r = 0$).
O disco desliza de volta descendo a colina, aumentando sua energia cinética e diminuindo a energia potencial.
Lembre-se de que no **HMC**, as variáveis de posição são de interesse direto, enquanto as variáveis de momento são construções artificiais (variáveis auxiliares).

Em qualquer intervalo de tempo s, a dinâmica hamiltoniana define uma dinamica do estado no tempo t para o estado no tempo $t + s$.
É importante ressaltar que esse mapeamento é reversível, o que é importante para mostrar que a dinâmica deixa π invariável.
Da mesma forma, a dinâmica preserva a energia total, H, portanto, as propostas são sempre aceitas.
Na prática, porém, nós normalmente não pode simular exatamente do sistema contínuo anterior e, em vez disso, considerar um sistema discretizado.
Uma abordagem comum é o método **leapfrog**, descrito no algoritmo. Devido às imprecisões introduzidas através da discretização, uma etapa de MH deve ser implementada (ou seja, a taxa de aceitação não é mais 1). No entanto, as taxas de aceitação ainda tendem a ser altas, mesmo para propostas que podem estar muito longe de seu último estado.

Algoritmo 1: Monte Carlo Hamiltoniano

Inicio: Posição inicial $\theta^{(1)}$ e tamanho dos passos ϵ.

Passo general: Para $t = 1, 2, \ldots$ faça,

1. Reamostrar o momento r:
   - $r^{(t)} \sim N(0, M)$
   - $(\theta_0, r_0) = (\theta^{(t)}, r^{(t)})$
2. Simule a discretização da *dinâmica hamiltoniana*:
   - $r_0 \leftarrow r_0 - \frac{\epsilon}{2} \nabla U(\theta_0)$
3. para $i = 1, \ldots m$ fazer:
   - $\theta_i \leftarrow \theta_{i-1} + \epsilon M^{-1} r_{i-1}$
   - $r_i \leftarrow r_{i-1} - \epsilon \nabla U(\theta_i)$
4. $r_m \leftarrow r_m - \frac{\epsilon}{2} \nabla U(\theta_m)$
5. $(\hat{\theta}, \hat{r}) = (\theta_m, r_m)$

M-H:
- $u \sim$ Uniforme$[0, 1]$
- $\rho = e^{H(\hat{\theta}, \hat{r}) - H(\theta^{(t)}, r^{(t)})}$
- Se $u < \min(1, \rho)$, então $\theta^{(t+1)} = \hat{\theta}$.

Exemplo 19. *Vamos amostrar da função* $U(u) = (u-2)^2$ *usando o a algoritmo HMC, para isso vamos a usar a função* $f(x) = \exp(-U(x))$*,*

```
  #minimizacao hmc
U<-function(u)
{a<-2
U<-(u-a)^2
return(U)}
grad_U<-function(u)
{a<-2
  gU<-2*(u-a)
}
f<-function(u)
{
  exp(-U(u))
}
x <- seq(0,4,0.1)
plot(x,f(x),type = 'l')


##parametros iniciais
# epsilon= passo
# current_q= posisison incial
# L=número de passos

HMC<-function(U, grad_U, epsilon, L, current_q)
{
```

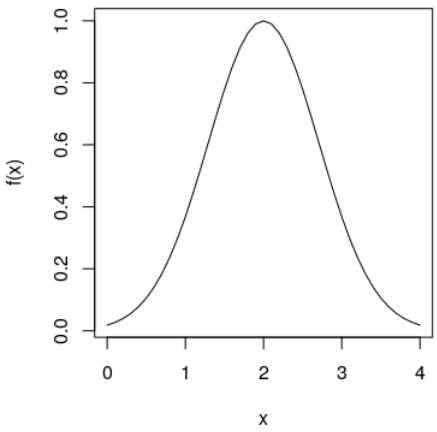

Figura 11.1: A função $f(x) = e^{-U(x)}$

```
  q <-current_q
  p <- rnorm(length(q),0,1)
  current_p <- p
  p <- p - epsilon*grad_U(q) / 2
  for (i in 1:L)
  {
    q <- q + epsilon * p
    if (i!=L) p <- p - epsilon * grad_U(q)
  }
  p  <- p - epsilon * grad_U(q) / 2
  p  <- -p
  current_U<-U(current_q)
  current_K <-sum(current_p^2) / 2
  proposed_U<- U(q)
  proposed_K<-sum(p^2) / 2
  if (runif(1) < exp(current_U-proposed_U+current_K-proposed_K))
  {
    return (q)
  }
  else
  {
    return (current_q) # reject
  }
}
##parametros iniciais
# epsilon= passo
# current_q= posisison incial
# L=número de passos
```

```
epsilon<-0.25
current_q<-0
L<-25
result<-c()
cq<-current_q
m<-1000
for (iter in 1:m) {
  result[iter]<-HMC(U, grad_U, epsilon, L, cq)
  cq<-result[iter]
}
result
par(mfrow=c(1,1))
boxplot(result)
plot(1:m,result,type = 'l')
```

As trajetorias simuladas,

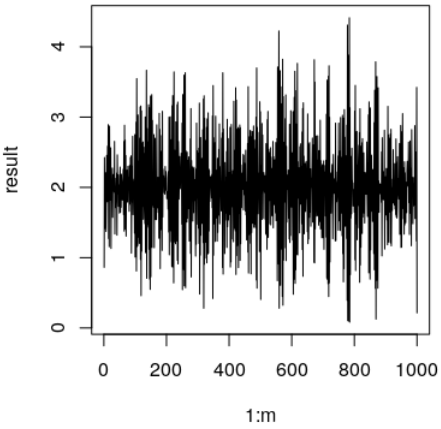

Figura 11.2: Trajetorias da cadeia HMC.

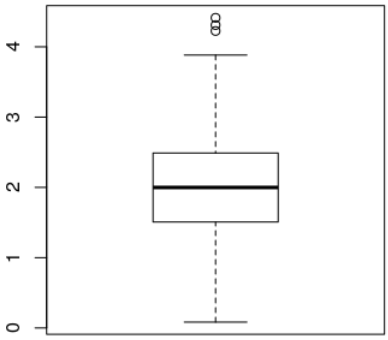

Figura 11.3: Boxplot dos resultados HMC.

,

Bibliografia

Albert, James H e Siddhartha Chib (1993). "Bayesian analysis of binary and polychotomous response data". Em: *Journal of the American statistical Association* 88.422, pp. 669–679.

Barker, AA (1965). "Monte Carlo Calculations of the Radial Distribution Functions for a Proton?Electron Plasma". Em: *Australian Journal of Physics* 18.2, p. 119. DOI: 10.1071/ph650119. URL: https://doi.org/10.1071%2Fph650119.

Berg, Bernd A e Thomas Neuhaus (1992). "Multicanonical ensemble: A new approach to simulate first-order phase transitions". Em: *Physical Review Letters* 68.1, p. 9.

Besag, JE (1994). "Comments on "Representations of knowledge in complex systems" by U. Grenander and MI Miller". Em: *J. Roy. Statist. Soc. Ser. B* 56, pp. 591–592.

Besag, Julian (1974). "Spatial interaction and the statistical analysis of lattice systems". Em: *Journal of the Royal Statistical Society: Series B (Methodological)* 36.2, pp. 192–225.

Boneh, Arnon e A Golan (1979). "Constraints' redundancy and feasible region boundedness by random feasible point generator (RFPG)". Em: *Third European congress on operations research (EURO III), Amsterdam.*

Bremaud, Pierre (1999). *Markov chains: Gibbs fields, Monte Carlo simulation, and queues*. Springer.

Brooks S., et al. (eds.) (2011). *Handbook of Markov Chain Monte Carlo*. Chapman & Hall/CRC handbooks of modern statistical methods. Taylor & Francis.

Casella, George e EI George (1992). "An introduction to Gibbs sampling". Em: *The American Statistician* 46.46, pp. 167–174.

Christian Robert, George Casella (auth.) (2010). *Introducing Monte Carlo Methods with R*. 1ª ed. Use R. Springer-Verlag New York.

David P. Landau, Kurt Binder (2014). *A Guide to Monte Carlo Simulations in Statistical Physics*. 4ª ed. Cambridge University Press.

Dempster, A. P., N. M. Laird e D. B. Rubin (1977). "Maximum Likelihood from Incomplete Data Via theEMAlgorithm". Em: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1, pp. 1–22.

Diebolt, Jean e Christian P. Robert (1994). "Estimation of Finite Mixture Distributions through Bayesian Sampling". Em: *Journal of the Royal Statistical Society Series B (Methodological)* 56 (2), pp. 363–375. ISSN: 0035-9246.

Duane, Simon et al. (1987). "Hybrid monte carlo". Em: *Physics letters B* 195.2, pp. 216–222.

Eckhardt, R. (1987). "Stan Ulam, John von Neumann, and the Monte Carlo method." Em: *Los Alamos Science, Special Issue*. URL: http://library.lanl.gov.cgi--bin/getfile?15--13.pdf.

Faming Liang Chuanhai Liu, Raymond Carroll (2010). *Advanced Markov chain Monte Carlo methods*. Wiley Series in Computational Statistics. Wiley.

François, Olivier, Sophie Ancelet e Gilles Guillot (2006). "Bayesian clustering using hidden Markov random fields in spatial population genetics". Em: *Genetics* 174.2, pp. 805–816.

Gelfand, Alan E. e Adrian F. M. Smith (1990). "Sampling-Based Approaches to Calculating Marginal Densities". Em: *Journal of the American Statistical Association* 85 (410), pp. 398–409. ISSN: 0162-1459,1537-274X. DOI: 10.2307/2289776.

Gelman, Andrew e Gary King (1990). "Estimating the Electoral Consequences of Legislative Redistricting". Em: *Journal of the American Statistical Association* 85.410, pp. 274–282.

Gelman, Andrew, Donald B Rubin et al. (1992). "Inference from iterative simulation using multiple sequences". Em: *Statistical science* 7.4, pp. 457–472.

Gelman, Andrew e TP Speed (1993). "Characterizing a joint probability distribution by conditionals". Em: *Journal of the Royal Statistical Society: Series B (Methodological)* 55.1, pp. 185–188.

Geman, Stuart e Donald Geman (1984). "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images". Em: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-6.6, pp. 721–741.

Geyer, Charles J (1991). "Markov chain Monte Carlo maximum likelihood". Em:

Geyer, Charles J (1992). "Practical markov chain monte carlo". Em: *Statistical science*, pp. 473–483.

Geyer, Charles J. (1994). "On the Convergence of Monte Carlo Maximum Likelihood Calculations". Em: *Journal of the Royal Statistical Society Series B (Methodological)* 56 (1), pp. 261–274. ISSN: 0035-9246. DOI: 10.2307/2346044.

Gilks, Walter R (1992). "Derivative-free adaptive rejection sampling for Gibbs sampling". Em: *Bayesian statistics* 4.2, pp. 641–649.

Gilks, Walter R, Gareth O Roberts e Edward I George (1994). "Adaptive direction sampling". Em: *Journal of the Royal Statistical Society: Series D (The Statistician)* 43.1, pp. 179–189.

Green, Peter J e Sylvia Richardson (2002). "Hidden Markov models and disease mapping". Em: *Journal of the American statistical association* 97.460, pp. 1055–1070.

Hastings, W. K. (1970). "Monte Carlo sampling methods using Markov chains and their applications". Em: *Biometrika* 57.1, pp. 97–109. DOI: 10.1093/biomet/57.1.97. URL: https://doi.org/10.1093%2Fbiomet%2F57.1.97.

Holland, John Henry et al. (1992). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press.

Hukushima, Koji e Koji Nemoto (1996). "Exchange Monte Carlo method and application to spin glass simulations". Em: *Journal of the Physical Society of Japan* 65.6, pp. 1604–1608.

Hurn, Merrilee A, Oddvar K Husby e Håvard Rue (2003). "A tutorial on image analysis". Em: *Spatial statistics and computational methods*. Springer, pp. 87–141.

Kirkpatrick, Scott, C Daniel Gelatt e Mario P Vecchi (1983). "Optimization by simulated annealing". Em: *science* 220.4598, pp. 671–680.

Lewandowski, A. e C. Liu (2008). "The sample Metropolis-Hastings algorithm." Em:

Liang, Faming (2005). "A generalized Wang–Landau algorithm for Monte Carlo computation". Em: *Journal of the American Statistical Association* 100.472, pp. 1311–1327.

— (2006). "A theory on flat histogram Monte Carlo algorithms". Em: *Journal of statistical physics* 122.3, pp. 511–529.

Liang, Faming, Chuanhai Liu e Raymond J Carroll (2007). "Stochastic approximation in Monte Carlo computation". Em: *Journal of the American Statistical Association* 102.477, pp. 305–320.

Liang, Faming e Wing Hung Wong (1999). "Dynamic weighting in simulations of spin systems". Em: *Physics Letters A* 252.5, pp. 257–262.

— (2000). "Evolutionary Monte Carlo: Applications to C p model sampling and change point problem". Em: *Statistica sinica*, pp. 317–342.

Liu, Chuanhai e Donald B Rubin (1994). "The ECME algorithm: a simple extension of EM and ECM with faster monotone convergence". Em: *Biometrika* 81.4, pp. 633–648.

Liu, Jun S, Faming Liang e Wing Hung Wong (2001). "A theory for dynamic weighting in Monte Carlo computation". Em: *Journal of the American Statistical Association* 96.454, pp. 561–573.

Liu, Jun S, Wing Hung Wong e Augustine Kong (1994). "Covariance structure of the Gibbs sampler with applications to the comparisons of estimators and augmentation schemes". Em: *Biometrika* 81.1, pp. 27–40.

Marinari, Enzo e Giorgio Parisi (1992). "Simulated tempering: a new Monte Carlo scheme". Em: *EPL (Europhysics Letters)* 19.6, p. 451.

Mengersen, Kerrie L, Richard L Tweedie et al. (1996). "Rates of convergence of the Hastings and Metropolis algorithms". Em: *The annals of Statistics* 24.1, pp. 101–121.

Metropolis Nicholas; Ulam, S. (1949). "The Monte Carlo Method". Em: *Journal of the American Statistical Association* 44 (247), pp. 335–341.

Møller, Jesper et al. (2006). "An efficient Markov chain Monte Carlo method for distributions with intractable normalising constants". Em: *Biometrika* 93.2, pp. 451–458.

Murray, Iain, Zoubin Ghahramani e David MacKay (2012). "MCMC for doubly-intractable distributions". Em: *arXiv preprint arXiv:1206.6848*.

Neal, Radford M et al. (2011). "MCMC using Hamiltonian dynamics". Em: *Handbook of markov chain monte carlo* 2.11, p. 2.

Onsager, Lars (1949). "Statistical hydrodynamics". Em: *Il Nuovo Cimento (1943-1954)* 6, pp. 279–287.

Peskum, P. H. (1973). "Optimum Monte-Carlo sampling using Markov chains". Em: *Biometrika* 60.3, pp. 607–612.

Reinelt, Gerhard (1994). *The traveling salesman: computational solutions for TSP applications*. Springer-Verlag.

Robbins, Herbert e Sutton Monro (1951). "A stochastic approximation method". Em: *The annals of mathematical statistics*, pp. 400–407.

Robert, Christian e George Casella (2011). "A Short History of MCMC". Em: *Chapman & Hall/CRC Handbooks of Modern Statistical Methods*. Chapman e Hall/CRC. DOI: 10.1201/b10905-3.

Roberts, Gareth O e Richard L Tweedie (1996). "Geometric convergence and central limit theorems for multidimensional Hastings and Metropolis algorithms". Em: *Biometrika* 83.1, pp. 95–110.

Roberts, GO e WR Gilks (1994). "Convergence of adaptive direction sampling". Em: *Journal of multivariate analysis* 49.2, pp. 287–298.

Ross, Sheldon M. (2012). *Simulation*. Academic Press.

Sean Meyn Richard L. Tweedie, Peter W. Glynn (2009). *Markov chains and stochastic stability*. 2ª ed. Cambridge Mathematical Library. Cambridge University Press.

Sisson S. A.; Fan, Y.; Tanaka M. M. (2007). "Sequential Monte Carlo without likelihoods". Em: *Proceedings of the National Academy of Sciences* 104 (6), pp. 1760–1765. DOI: 10.1073/pnas.0607208104.

Smith, RL (1980). "A monte carlo procedure for the random generation of feasible solutions to mathematical programming problems". Em: *Bulletin of the TIMS/ORSA Joint National Meeting*. Vol. 101.

Søren Asmussen, Peter W. Glynn (2010). *Stochastic Simulation Algorithms and Analysis*. 1st Edition. Stochastic Modelling and Applied Probability. Springer.

Tierney, Luke (1994). "Markov chains for exploring posterior distributions". Em: *the Annals of Statistics*, pp. 1701–1728.

Wang, Fugao e David P Landau (2001). "Efficient, multiple-range random walk algorithm to calculate the density of states". Em: *Physical review letters* 86.10, p. 2050.

Wong, Wing Hung e Faming Liang (1997). "Dynamic weighting in Monte Carlo and optimization". Em: *Proceedings of the National Academy of Sciences* 94.26, pp. 14220–14224.