Metropolis-Hastings

Isabelle Fernandes de Oliveira

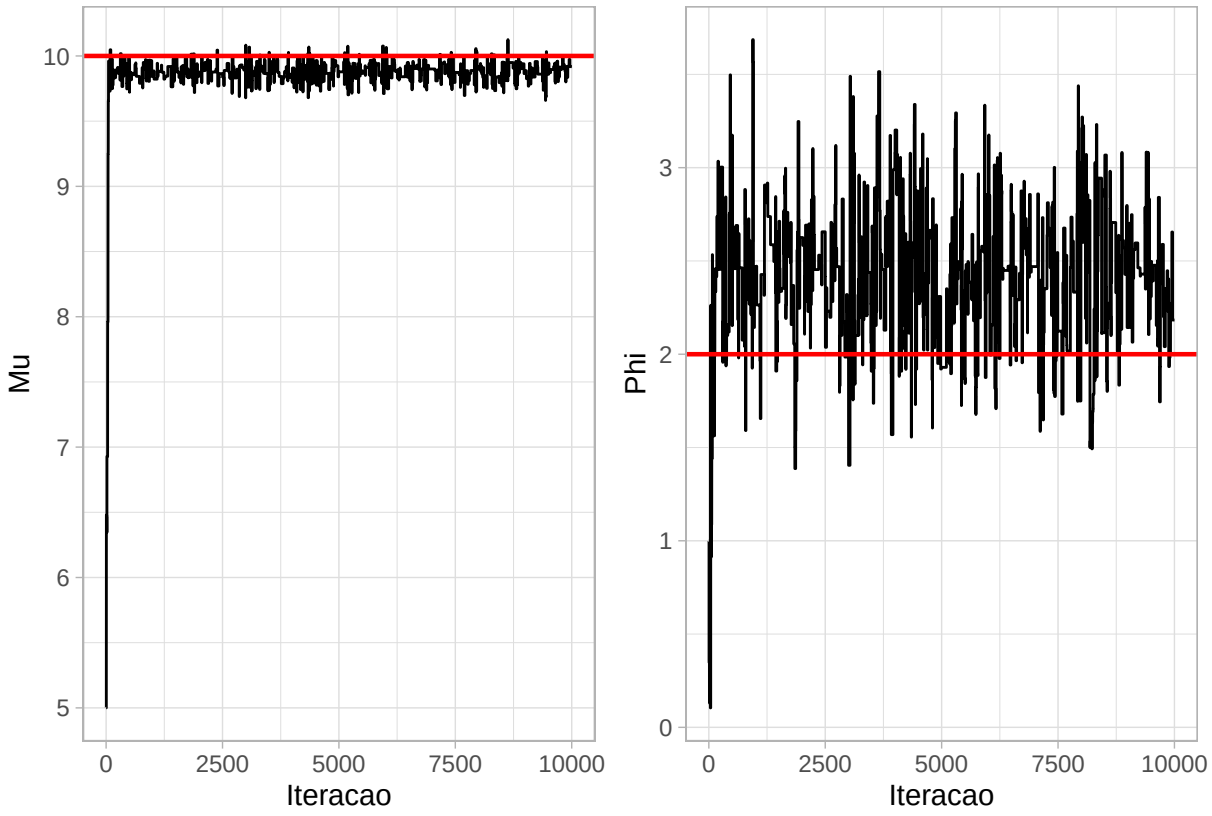
2025-06-09

```
rm(list = ls(all = TRUE))

# gerando dados artificiais.
n = 100
# tamanho amostral.
mu_real = 10
# mu real.
phi_real = 2
# phi real.
y = rnorm(n,mu_real,sqrt(1/phi_real))
# especificacoes a priori: mu ~ N(m,v) e phi ~ Ga(a,b).
m = 5; v = 10; a = 0.1; b = 0.1;

# semente de inicializacao das cadeias MCMC
mu = 5; phi = 1;

# variancia da distribuicao geradora de propostas.
nu = 0.5 # cadidato gerado da N(valor_atual, nu).

# alguns termos presentes no núcleo a posteriori.
as = a + n/2; bs = b + sum(y^2)/2;
N = 10000

# objetos auxiliares para salvar cadeias e taxa de aceitacao.
save_mu = mu; save_phi = phi; ARate = 0

for(j in 1:N){
  mu_c = rnorm(1,mu,sqrt(nu)) # gere o candidado mu.
  phi_c = rnorm(1,phi,sqrt(nu)) # gere o candidado phi.

  # se phi_c < 0 ou phi_c = 0, rejeite mu_c e phi_c e siga para a iteracao j+1

  if(phi_c > 0){

    # --- CÁLCULOS PARA O ESTADO CANDIDATO (c) ---
    vs_c <- 1 / (n * phi_c + 1 / v)
    ms_c <- vs_c* (phi_c * sum(y) + m / v)

    logR1_1 <- -((mu_c - ms_c)^2 / (2 * vs_c)) +
      (ms_c^2 / (2 * vs_c)) +
      (as - 1) * log(phi_c) -
      (phi_c * bs)
```

```
    # --- CÁLCULOS PARA O ESTADO ATUAL ---
    vs <- 1 / (n * phi + 1 / v)
    ms <- vs * (phi * sum(y) + m / v)

    logR1_2 <- -((mu - ms)^2 / (2 * vs)) +
      (ms^2 / (2 * vs)) +
      (as - 1) * log(phi) -
      (phi * bs)

    # A razão de aceitação
    logR1 = logR1_1 - logR1_2

    # calcule a razao R2 (escala log). Razao entre as funcoes geradoras
    dp = sqrt(nu)
    logR2_1 = dnorm(mu, mu_c, dp, log=TRUE) + dnorm(phi, phi_c, dp, log=TRUE)
    logR2_2 = dnorm(mu_c, mu, dp, log=TRUE) + dnorm(phi_c, phi, dp, log=TRUE)
    logR2 = logR2_1 - logR2_2

    alpha = exp(min(0,(logR1+logR2))) # prob. de aceitacao
    # teste de aceitacao/rejeicao.
    if(runif(1,0,1) < alpha){ mu = mu_c; phi = phi_c; ARate = ARate + 1 }
  }
  # salvando cadeia.
  save_mu = c(save_mu, mu)
  save_phi = c(save_phi, phi)
}

df_mu <- data.frame(Mu = save_mu, Iteracao = 1:(N+1))
df_phi <- data.frame(Phi = save_phi, Iteracao = 1:(N+1))


grafico1 <- ggplot(df_mu, aes(x = Iteracao, y = Mu)) +
  geom_line() +
  geom_hline(yintercept = mu_real, color = "red",
             linetype = "solid", size = 0.8) +
  theme_light()
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```
grafico2 <- ggplot(df_phi, aes(x = Iteracao, y = Phi)) +
  geom_line() +
  geom_hline(yintercept = phi_real, color = "red",
             linetype = "solid", size = 0.8) +
  theme_light()

grafico1 + grafico2
```



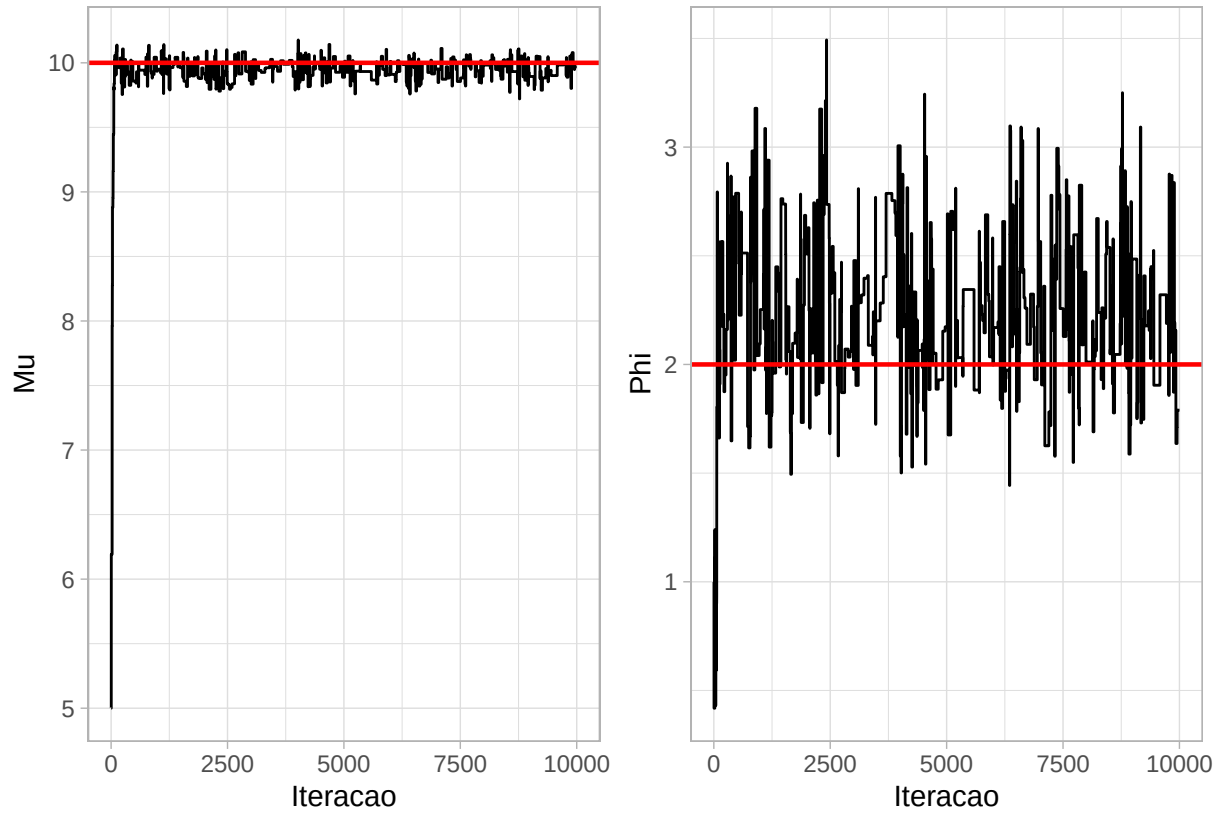

```
cat("Taxa de aceitacao: ", sum(ARate)/N)
```

```
## Taxa de aceitacao:  0.0659
```

```
rm(list = ls(all = TRUE))

# gerando dados artificiais.
n = 100
# tamanho amostral.
mu_real = 10
# mu real.
phi_real = 2
# phi real.
y = rnorm(n,mu_real,sqrt(1/phi_real))
# especificacoes a priori: mu ~ N(m,v) e phi ~ Ga(a,b).
m = 5; v = 10; a = 0.1; b = 0.1;

# semente de inicializacao das cadeias MCMC
mu = 5; phi = 1;

# variancia da distribuicao geradora de propostas.
nu = 1 # cadidato gerado da N(valor_atual, nu).

# alguns termos presentes no núcleo a posteriori.
as = a + n/2; bs = b + sum(y^2)/2;
```

```r
N = 10000

# objetos auxiliares para salvar cadeias e taxa de aceitacao.
save_mu = mu; save_phi = phi; ARate = 0

for(j in 1:N){
  mu_c = rnorm(1,mu,sqrt(nu)) # gere o candidado mu.
  phi_c = rnorm(1,phi,sqrt(nu)) # gere o candidado phi.

  # se phi_c < 0 ou phi_c = 0, rejeite mu_c e phi_c e siga para a iteracao j+1

  if(phi_c > 0){

    # --- CÁLCULOS PARA O ESTADO CANDIDATO (c) ---
    vs_c <- 1 / (n * phi_c + 1 / v)
    ms_c <- vs_c* (phi_c * sum(y) + m / v)

    logR1_1 <- -((mu_c - ms_c)^2 / (2 * vs_c)) +
      (ms_c^2 / (2 * vs_c)) +
      (as - 1) * log(phi_c) -
      (phi_c * bs)

    # --- CÁLCULOS PARA O ESTADO ATUAL ---
    vs <- 1 / (n * phi + 1 / v)
    ms <- vs * (phi * sum(y) + m / v)

    logR1_2 <- -((mu - ms)^2 / (2 * vs)) +
      (ms^2 / (2 * vs)) +
      (as - 1) * log(phi) -
      (phi * bs)

    # A razão de aceitação
    logR1 = logR1_1 - logR1_2

    # calcule a razao R2 (escala log). Razao entre as funcoes geradoras
    dp = sqrt(nu)
    logR2_1 = dnorm(mu, mu_c, dp, log=TRUE) + dnorm(phi, phi_c, dp, log=TRUE)
    logR2_2 = dnorm(mu_c, mu, dp, log=TRUE) + dnorm(phi_c, phi, dp, log=TRUE)
    logR2 = logR2_1 - logR2_2

    alpha = exp(min(0,(logR1+logR2))) # prob. de aceitacao
    # teste de aceitacao/rejeicao.
    if(runif(1,0,1) < alpha){ mu = mu_c; phi = phi_c; ARate = ARate + 1 }
  }
  # salvando cadeia.
  save_mu = c(save_mu, mu)
  save_phi = c(save_phi, phi)
}

df_mu <- data.frame(Mu = save_mu, Iteracao = 1:(N+1))
df_phi <- data.frame(Phi = save_phi, Iteracao = 1:(N+1))
```

```
grafico1 <- ggplot(df_mu, aes(x = Iteracao, y = Mu)) +
  geom_line() +
  geom_hline(yintercept = mu_real, color = "red",
             linetype = "solid", size = 0.8) +
  theme_light()

grafico2 <- ggplot(df_phi, aes(x = Iteracao, y = Phi)) +
  geom_line() +
  geom_hline(yintercept = phi_real, color = "red",
             linetype = "solid", size = 0.8) +
  theme_light()

grafico1 + grafico2
```



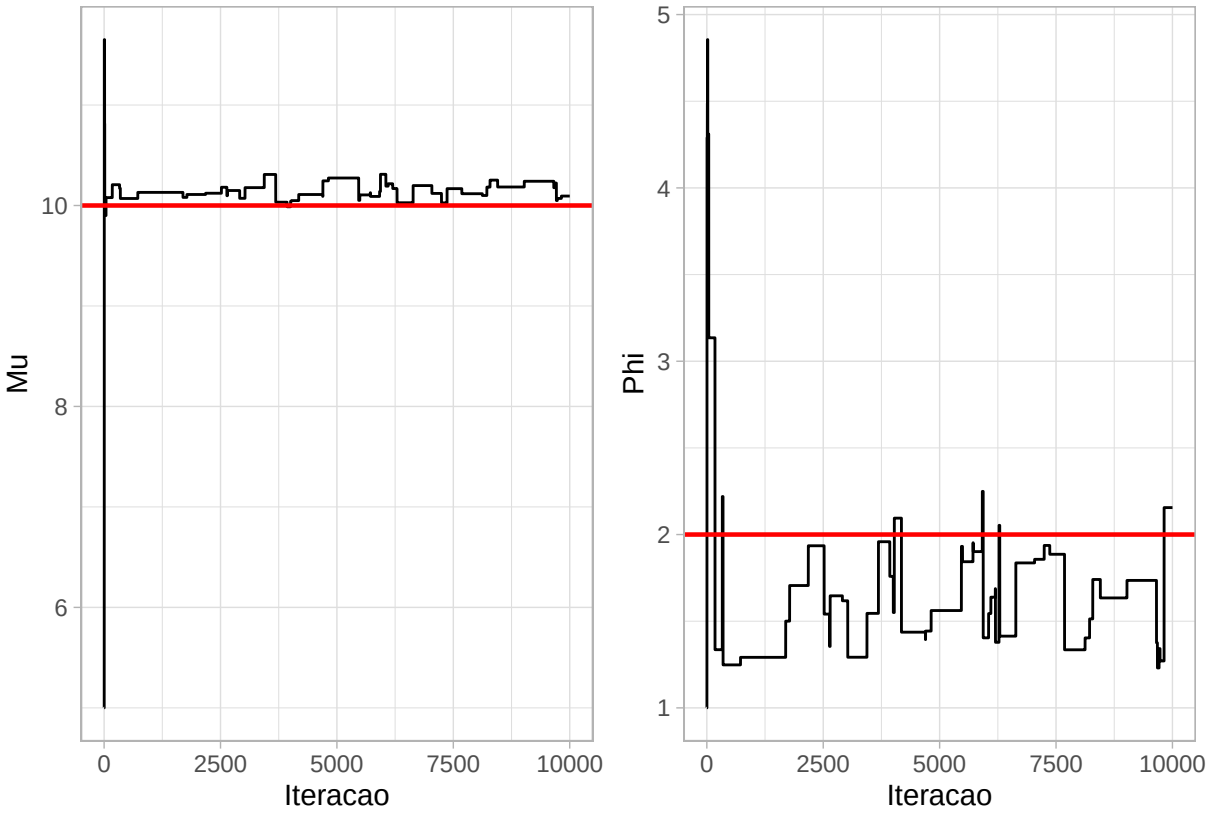

```
cat("Taxa de aceitacao: ", sum(ARate)/N)
```

```
## Taxa de aceitacao:  0.0426
```

```
rm(list = ls(all = TRUE))

# gerando dados artificiais.
n = 100
# tamanho amostral.
mu_real = 10
```

```r
# mu real.
phi_real = 2
# phi real.
y = rnorm(n,mu_real,sqrt(1/phi_real))
# especificacoes a priori: mu ~ N(m,v) e phi ~ Ga(a,b).
m = 5; v = 10; a = 0.1; b = 0.1;

# semente de inicializacao das cadeias MCMC
mu = 5; phi = 1;

# variancia da distribuicao geradora de propostas.
nu = 10 # cadidato gerado da N(valor_atual, nu).

# alguns termos presentes no núcleo a posteriori.
as = a + n/2; bs = b + sum(y^2)/2;
N = 10000

# objetos auxiliares para salvar cadeias e taxa de aceitacao.
save_mu = mu; save_phi = phi; ARate = 0

for(j in 1:N){
  mu_c = rnorm(1,mu,sqrt(nu)) # gere o candidado mu.
  phi_c = rnorm(1,phi,sqrt(nu)) # gere o candidado phi.

  # se phi_c < 0 ou phi_c = 0, rejeite mu_c e phi_c e siga para a iteracao j+1

  if(phi_c > 0){

    # --- CÁLCULOS PARA O ESTADO CANDIDATO (c) ---
    vs_c <- 1 / (n * phi_c + 1 / v)
    ms_c <- vs_c* (phi_c * sum(y) + m / v)

    logR1_1 <- -((mu_c - ms_c)^2 / (2 * vs_c)) +
      (ms_c^2 / (2 * vs_c)) +
      (as - 1) * log(phi_c) -
      (phi_c * bs)

    # --- CÁLCULOS PARA O ESTADO ATUAL ---
    vs <- 1 / (n * phi + 1 / v)
    ms <- vs * (phi * sum(y) + m / v)

    logR1_2 <- -((mu - ms)^2 / (2 * vs)) +
      (ms^2 / (2 * vs)) +
      (as - 1) * log(phi) -
      (phi * bs)

    # A razão de aceitação
    logR1 = logR1_1 - logR1_2

    # calcule a razao R2 (escala log). Razao entre as funcoes geradoras
    dp = sqrt(nu)
    logR2_1 = dnorm(mu, mu_c, dp, log=TRUE) + dnorm(phi, phi_c, dp, log=TRUE)
    logR2_2 = dnorm(mu_c, mu, dp, log=TRUE) + dnorm(phi_c, phi, dp, log=TRUE)
```

```
    logR2 = logR2_1 - logR2_2

    alpha = exp(min(0,(logR1+logR2))) # prob. de aceitacao
    # teste de aceitacao/rejeicao.
    if(runif(1,0,1) < alpha){ mu = mu_c; phi = phi_c; ARate = ARate + 1 }
  }
  # salvando cadeia.
  save_mu = c(save_mu, mu)
  save_phi = c(save_phi, phi)
}

df_mu <- data.frame(Mu = save_mu, Iteracao = 1:(N+1))
df_phi <- data.frame(Phi = save_phi, Iteracao = 1:(N+1))


grafico1 <- ggplot(df_mu, aes(x = Iteracao, y = Mu)) +
  geom_line() +
  geom_hline(yintercept = mu_real, color = "red",
             linetype = "solid", size = 0.8) +
  theme_light()

grafico2 <- ggplot(df_phi, aes(x = Iteracao, y = Phi)) +
  geom_line() +
  geom_hline(yintercept = phi_real, color = "red",
             linetype = "solid", size = 0.8) +
  theme_light()

grafico1 + grafico2
```




```
cat("Taxa de aceitacao: ", sum(ARate)/N)
```

```
## Taxa de aceitacao:  0.0054
```

Conclusão: O parâmetro ν controla o tamanho do salto da proposta: valores grandes como ν=10 geram baixa aceitação e "congelam" a cadeia.