Trabalho 02 - MCCD II

Emanuelle, Maria Luiza e Mariana Fleming

2025-11-09

```
library(MASS)
library(mvtnorm)
library(coda)
```

```
## Warning: pacote 'coda' foi compilado no R versão 4.5.2
```

```
library(ggplot2)
library(gridExtra)
set.seed(123)
```

Implementação - AM

```
metropolis_adaptativo <- function(log_funcao_alvo, ponto_inicial, cov_inicial, num_iteracoes) {

  # log_funcao_alvo: A função log-densidade que queremos amostrar (sua f(x)).
  # ponto_inicial: O vetor de onde a cadeia começa (X^0).
  # cov_inicial: Um "chute" inicial para a matriz de covariância (Sigma^0).
  # num_iteracoes: O número total de passos que o algoritmo vai dar (T).

  p <- length(ponto_inicial) # número de parâmetros

  # Matriz para guardar todos os resultados
  amostras <- matrix(NA, nrow = num_iteracoes + 1, ncol = p)
  amostras[1, ] <- ponto_inicial # Guarda o ponto inicial na primeira linha

  # Ponto atual da cadeia
  ponto_atual <- ponto_inicial

  # Parâmetros que serão adaptados
  media_adaptativa <- ponto_inicial
  cov_adaptativa <- cov_inicial

  # Fator de escala
  lambda <- (2.38^2) / p

  # Termo de estabilidade
  epsilon <- 1e-6
```

```
  for (t in 1:num_iteracoes) {

    # Algoritmo MH

    # Passo 1: Gerar um candidato X*
    Sigma_proposta <- lambda * cov_adaptativa + diag(p) * epsilon
    candidato <- mvrnorm(1, ponto_atual, Sigma_proposta)

    # Passo 2: Calcular a chance de aceitar (em log)
    log_r <- log_funcao_alvo(candidato) - log_funcao_alvo(ponto_atual)

    # Passo 3: Aceitar ou Rejeitar
    if (runif(1) < exp(log_r)) {
      ponto_atual <- candidato # Aceita o candidato
    }
    # Se não rejeita, o 'ponto_atual' continua o mesmo

    # Guarda o ponto da iteração
    amostras[t + 1, ] <- ponto_atual

    # Passo Adaptativo:

    # Parâmetro de adaptação (gamma)
    gamma <- 1 / t

    # Atualiza a média
    media_adaptativa <- media_adaptativa + gamma * (ponto_atual - media_adaptativa)

    # Atualiza a covariância
    cov_adaptativa <- cov_adaptativa + gamma * (tcrossprod(ponto_atual - media_adaptativa) - cov_adapta
  }

  return(amostras)
}
```

Implementação - AMwG

```
metropolis_gibbs_adaptativo <- function(log_funcao_alvo, x_inicial, num_iteracoes, batch_size = 50) {

  # log_funcao_alvo: função que devolve o log da densidade alvo f(x)
  # x_inicial: vetor de valores iniciais (X^0)
  # num_iteracoes: número total de iterações (T)
  # batch_size: tamanho do "batch" usado na adaptação (T_b)

  p <- length(x_inicial)  # número de parâmetros
  amostras <- matrix(NA, nrow = num_iteracoes + 1, ncol = p)
  amostras[1, ] <- x_inicial

  # Inicializações
  x_atual <- x_inicial
  sigma <- rep(1, p)            # variâncias iniciais das propostas
```

```r
log_sigma <- log(sigma)       # escala logarítmica
b <- 0                        # contador de batches
aceitacoes_batch <- rep(0, p) # número de aceitações por parâmetro
tentativas_batch <- rep(0, p) # número de tentativas por parâmetro

for (t in 1:num_iteracoes) {

  # Atualiza cada parâmetro condicionalmente aos outros (Gibbs loop)
  for (j in 1:p) {

    # Passo 1: gerar candidato via proposta normal
    x_cand <- x_atual
    x_cand[j] <- rnorm(1, mean = x_atual[j], sd = exp(log_sigma[j]))

    # Passo 2: calcular razão de aceitação (em log)
    log_r <- log_funcao_alvo(x_cand) - log_funcao_alvo(x_atual)

    # Passo 3: regra de aceitação
    if (runif(1) < exp(log_r)) {
      x_atual[j] <- x_cand[j]
      aceitacoes_batch[j] <- aceitacoes_batch[j] + 1
    }
    # Conta a tentativa
    tentativas_batch[j] <- tentativas_batch[j] + 1
  }

  # Guarda a amostra atual
  amostras[t + 1, ] <- x_atual

  # Passo 4: Adaptação em tempos pré-definidos (batches)
  if (t %% batch_size == 0 && t > 0) { # Adicionado t > 0 para evitar b=0
    b <- b + 1

    # Taxa de aceitação por parâmetro no batch anterior
    taxa_aceitacao <- aceitacoes_batch / tentativas_batch
    taxa_aceitacao[is.nan(taxa_aceitacao)] <- 0 # Evita NaN

    # O delta deve diminuir com o tempo. Usamos 'b' (contador de batch)
    delta <- min(0.01, 1 / sqrt(b))

    # Regra adaptativa
    for (j in 1:p) {
      if (taxa_aceitacao[j] > 0.44) {
        log_sigma[j] <- log_sigma[j] + delta
      } else {
        log_sigma[j] <- log_sigma[j] - delta
      }
    }

    # Reinicia contadores para o próximo batch
    aceitacoes_batch <- rep(0, p)
    tentativas_batch <- rep(0, p)
  }
```

```
  }

  return(list(amostras = amostras, log_sigma = log_sigma))
}
```

Funções de retornos

```
gera_graficos_comparativos <- function(res_am, res_amwg, valor_real, Sigma,
                                       burnin_frac = 0.5, titulo_base = "Comparação AM vs AMwG",
                                       nome_cenario = NULL) {
  # res_am, res_amwg: listas com elemento $amostras (matrix)
  # valor_real: vetor de valores verdadeiros (length p)
  # Sigma: matriz de covariância alvo
  # nome_cenario: string ("Cenário 1", "Cenário 2" ou "Cenário 3")

  require(mvtnorm)

  # --- FUNÇÃO AUXILIAR PARA PLOT ---
  plot_por_metodo <- function(amostras, metodo_nome) {
    burnin <- floor(nrow(amostras) * burnin_frac)
    pos <- amostras[(burnin + 1):nrow(amostras), , drop = FALSE]
    p <- ncol(pos)

    op <- par(no.readonly = TRUE)
    on.exit(par(op), add = TRUE)
    par(mfrow = c(2, 2), mar = c(4.2, 4, 2.5, 1), oma = c(0, 0, 3, 0))

    for (j in 1:p) {
      # --- 1  TRAJETÓRIA ---
      plot(
        pos[, j],
        type = "l",
        main = paste("Trajetória da var", j),
        xlab = "Iterações",
        ylab = "",
        col = "blue"
      )
      abline(h = valor_real[j], col = "red", lty = 2, lwd = 2)

      # --- 2  DENSIDADE ESTIMADA ---
      dens_est <- density(pos[, j])
      x_vals <- seq(min(dens_est$x), max(dens_est$x), length.out = 400)

      # --- 3  DENSIDADE TEÓRICA DEPENDENTE DO CENÁRIO ---
      if (is.null(nome_cenario) || grepl("1", nome_cenario, ignore.case = TRUE)) {
        # Cenário 1: Normal padrão
        dens_teorica <- dnorm(x_vals, mean = valor_real[j], sd = sqrt(Sigma[j, j]))
        dens_label <- "N(0,1)"

      } else if (grepl("2", nome_cenario, ignore.case = TRUE)) {
        # Cenário 2: Mistura bimodal
```

```
        dens_teorica <- 0.5 * dnorm(x_vals, mean = -3, sd = 1) +
                        0.5 * dnorm(x_vals, mean =  3, sd = 1)
        dens_label <- "0.5N(-3,1)+0.5N(3,1)"

      } else if (grepl("3", nome_cenario, ignore.case = TRUE)) {
        # Cenário 3: Regressão - sem densidade teórica fechada
        dens_teorica <- NULL
        dens_label <- NULL
      } else {
        dens_teorica <- NULL
        dens_label <- NULL
      }

      # --- 4  PLOTAGEM ---
      ylim_max <- max(dens_est$y, if (!is.null(dens_teorica)) max(dens_teorica) else 0)
      plot(
        dens_est,
        main = paste("Densidade da var", j),
        xlab = "Valor",
        ylab = "Densidade",
        ylim = c(0, ylim_max * 1.05)
      )

      if (!is.null(dens_teorica)) {
        lines(x_vals, dens_teorica, col = "red", lwd = 2, lty = 2)
      }
  }

    mtext(paste(titulo_base, "-", metodo_nome),
          outer = TRUE, line = 1, cex = 1.2, font = 2)
  }

  # --- Executa para AM ---
  plot_por_metodo(res_am$amostras, "AM")
  if (interactive()) readline(prompt = "Pressione [enter] para ver os gráficos do AMwG...")

  # --- Executa para AMwG ---
  plot_por_metodo(res_amwg$amostras, "AMwG")
}
```

Cenário 1 — Alvo Normal Multivariado Correlacionado

Objetivo: Comparar a adaptação global (AM) vs local (AMwG) em um caso controlado e de fácil visualização.

Definição do problema:

$$\theta \sim N_p(0, \Sigma), \quad \Sigma_{ij} = 0.9^{|i-j|}$$

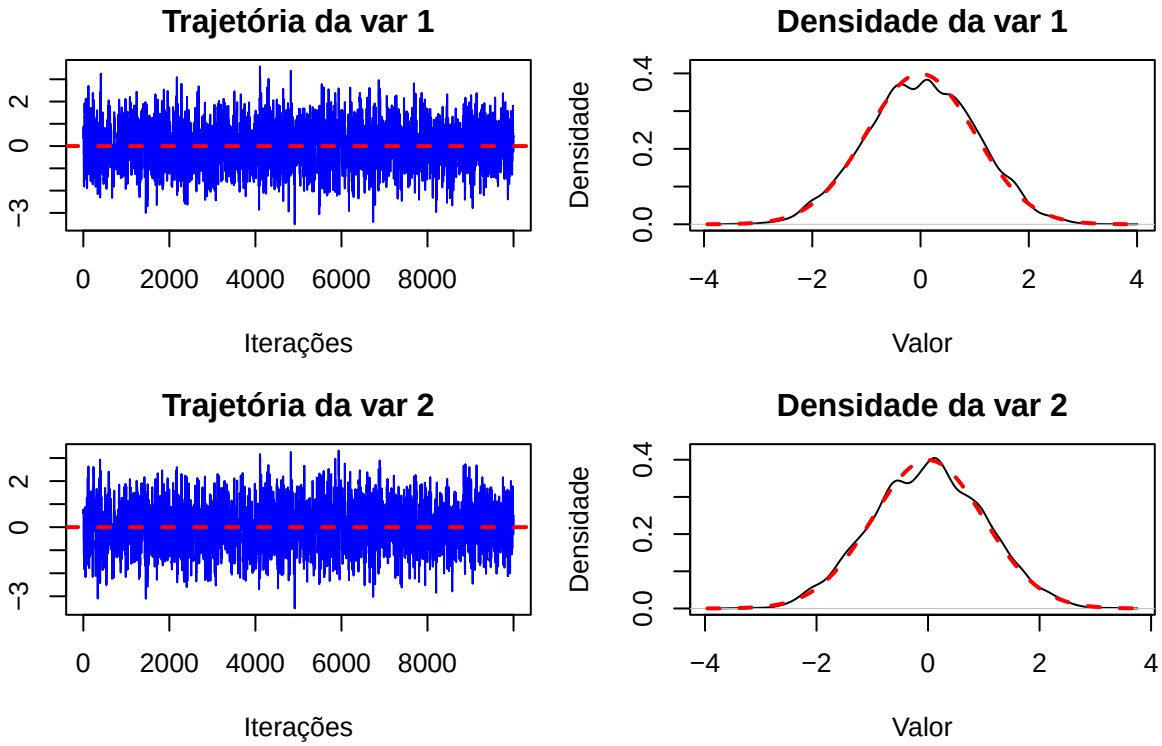
Configuração:

- AM: adapta a covariância completa da proposta.
- AMwG: atualiza cada coordenada condicionalmente, adaptando o desvio padrão de cada proposta.

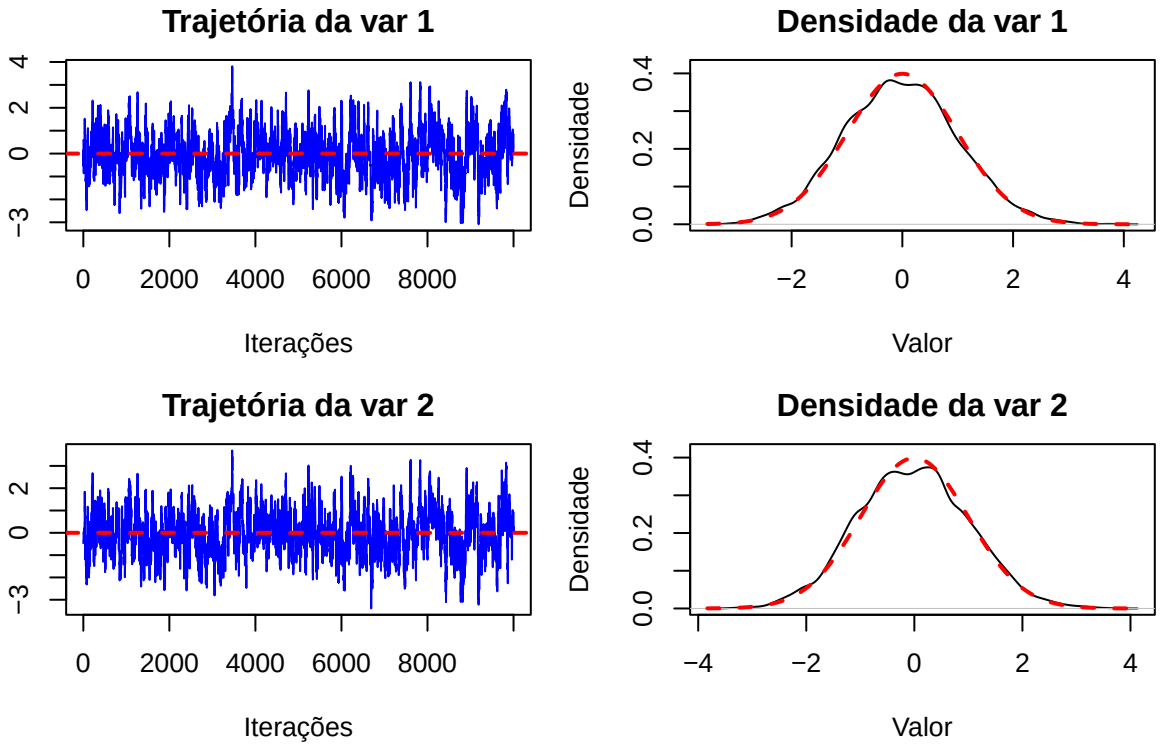
```r
# Dimensão e parâmetros verdadeiros (p = 2)
p <- 2
valor_real <- rep(0, p)

# Matriz de covariância com correlação 0.9^{|i-j|}
rho <- 0.9
Sigma <- outer(1:p, 1:p, function(i, j) rho^abs(i - j))

# Função log-alvo: densidade normal multivariada
log_alvo <- function(theta) {
  dmvnorm(theta, mean = valor_real, sigma = Sigma, log = TRUE)
}

# Condições iniciais e número de iterações
x0 <- rep(2, p)
num_iter <- 20000
cov_inicial <- diag(p) * 0.01
```

```
##
## Anexando pacote: 'dplyr'

## O seguinte objeto é mascarado por 'package:gridExtra':
##
##     combine

## O seguinte objeto é mascarado por 'package:MASS':
##
##     select

## Os seguintes objetos são mascarados por 'package:stats':
##
##     filter, lag

## Os seguintes objetos são mascarados por 'package:base':
##
##     intersect, setdiff, setequal, union

##          Método Parâmetro Valor_Verdadeiro   Média    Viés   RMSE     ESS
## var1...1     AM         1               0  0.0199  0.0199 0.9973 1504.67
## var2...2     AM         2               0  0.0010  0.0010 1.0011 1436.65
## var1...3   AMwG         1               0 -0.0260 -0.0260 1.0187  239.12
## var2...4   AMwG         2               0 -0.0208 -0.0208 1.0260  240.49
##          Taxa_Aceitação
## var1...1          0.357
## var2...2          0.357
## var1...3          0.451
## var2...4          0.451
```

Alvo Normal Multivariado Correlacionado − AM



Alvo Normal Multivariado Correlacionado − AMwG



Cenário 2 — Mistura Bimodal Multivariada

Objetivo: Testar a robustez e o mixing de cada método em superfícies multimodais.

Definição do problema:

$$\pi(\theta) = 0.5, N((-3,-3), I) + 0.5, N((3,3), I)$$

```
p <- 2
x0 <- c(2, 2)
num_iter <- 50000
cov_inicial <- diag(p) * 0.01

log_alvo_mistura <- function(x) {
  mu1 <- c(-3, -3)
  mu2 <- c(3, 3)
  sigma <- diag(2)
  logdens <- log(0.5 * dmvnorm(x, mean = mu1, sigma = sigma) +
                   0.5 * dmvnorm(x, mean = mu2, sigma = sigma))
  return(logdens)
}

conta_trocas_modos <- function(amostras) {
  mu1 <- c(-3, -3)
  mu2 <- c(3, 3)
  d1 <- apply(amostras, 1, function(x) dmvnorm(x, mean = mu1, sigma = diag(2)))
  d2 <- apply(amostras, 1, function(x) dmvnorm(x, mean = mu2, sigma = diag(2)))
  modos <- ifelse(d1 > d2, 1, 2)
  trocas <- sum(diff(modos) != 0)
  prop1 <- mean(modos == 1)
  prop2 <- mean(modos == 2)
  c(n_trocas = trocas, prop_modo1 = prop1, prop_modo2 = prop2)
}

res_am <- list(amostras = metropolis_adaptativo(log_alvo_mistura, x0, cov_inicial, num_iter))
res_amwg <- metropolis_gibbs_adaptativo(log_alvo_mistura, x0, num_iter, batch_size = 50)

diag_am <- conta_trocas_modos(res_am$amostras[(floor(nrow(res_am$amostras)/2)+1):nrow(res_am$amostras),
diag_amwg <- conta_trocas_modos(res_amwg$amostras[(floor(nrow(res_amwg$amostras)/2)+1):nrow(res_amwg$am
print(diag_am); print(diag_amwg)
```

```
##    n_trocas  prop_modo1  prop_modo2
## 1395.000000    0.525859    0.474141
```

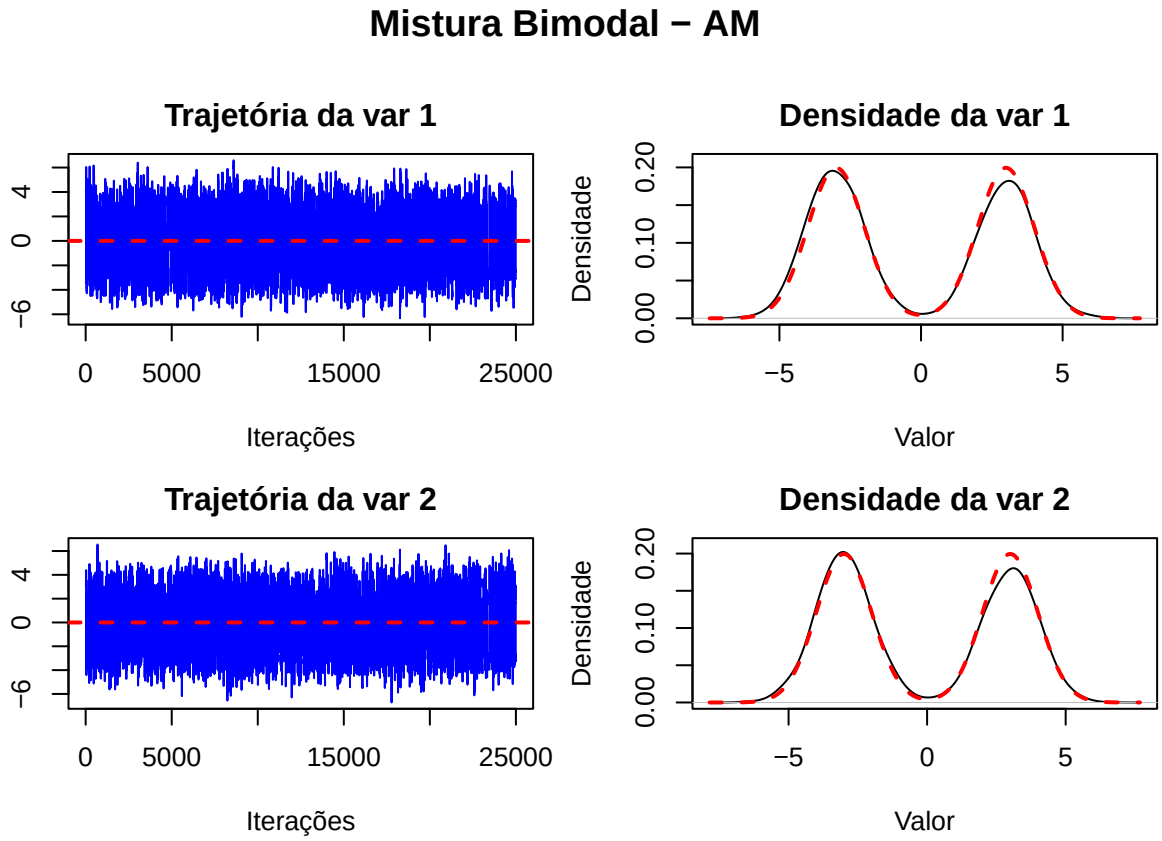
```
##   n_trocas prop_modo1 prop_modo2
## 12.0000000  0.4565417  0.5434583
```

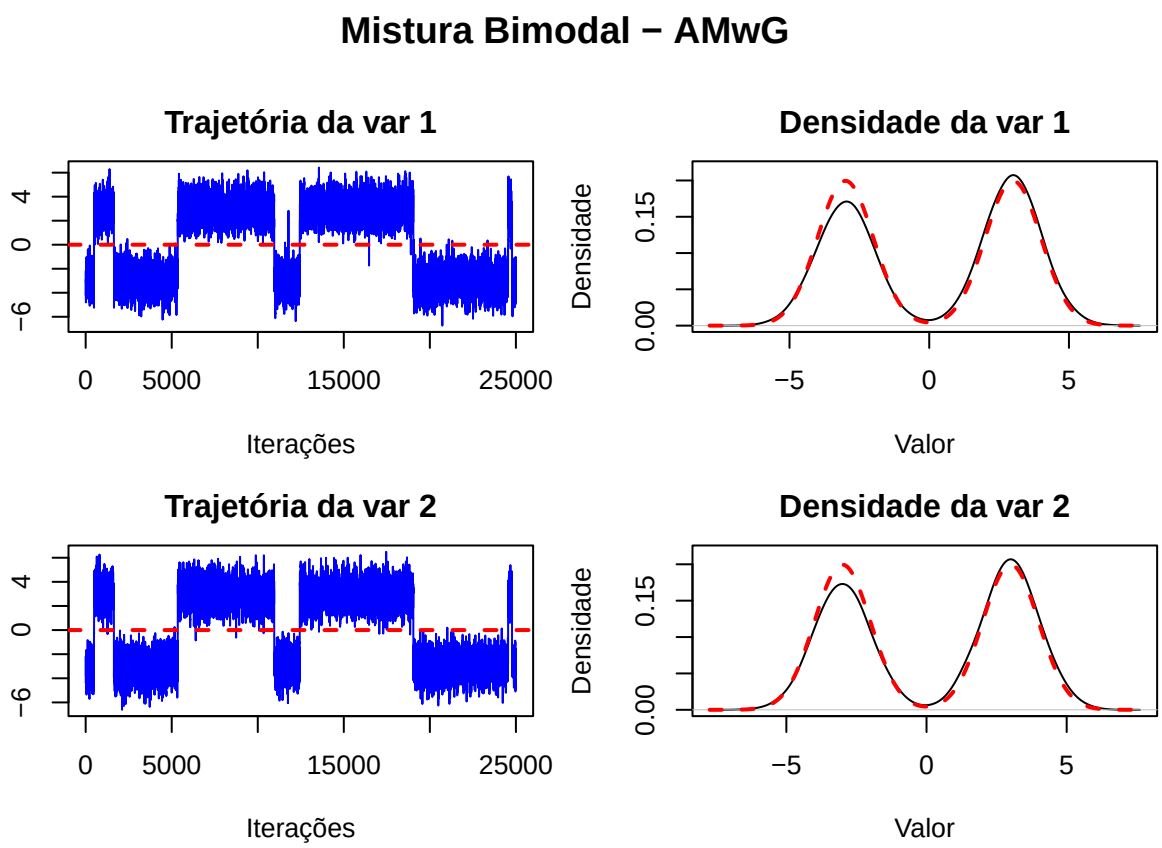
```
# Use suas funções de resumo (para mistura, passe valor_real = c(NA, NA) ou c(0,0) apenas para compatib
tabela_comp_mistura <- gera_tabela_comparativa(res_am, res_amwg, valor_real = c(0,0))
print(tabela_comp_mistura)
```

```
##          Método Parâmetro Valor_Verdadeiro   Média    Viés   RMSE     ESS
```

```
## var1...1     AM         1                0 -0.1732 -0.1732 3.1904 1379.86
## var2...2     AM         2                0 -0.1564 -0.1564 3.1711 1414.15
## var1...3   AMwG         1                0  0.2579  0.2579 3.1541   14.62
## var2...4   AMwG         2                0  0.2590  0.2590 3.1587   13.54
##          Taxa_Aceitação
## var1...1          0.181
## var2...2          0.181
## var1...3          0.465
## var2...4          0.465
```

```
gera_graficos_comparativos(res_am, res_amwg, valor_real = c(0,0), Sigma = diag(2), burnin_frac = 0.5,
                           titulo_base = "Mistura Bimodal", nome_cenario = "Cenário 2")
```





Cenário 3 — Modelo Bayesiano Multivariado

Objetivo: Testar AM vs AMwG em um modelo com parâmetros correlacionados e interpretação prática.
Modelo:

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \varepsilon_i, \quad \varepsilon_i \sim N(0, \sigma^2)$$

Prioris:

$$\beta \sim N(0, 100I), \quad \sigma^2 \sim \text{Inv-Gamma}(1, 1)$$

Dimensão: (p = 3).

Configuração:

- Rodar AM e AMwG sobre a mesma posterior.
- Avaliar convergência e eficiência das cadeias para $(\beta_0, \beta_1, \beta_2, \sigma^2)$.

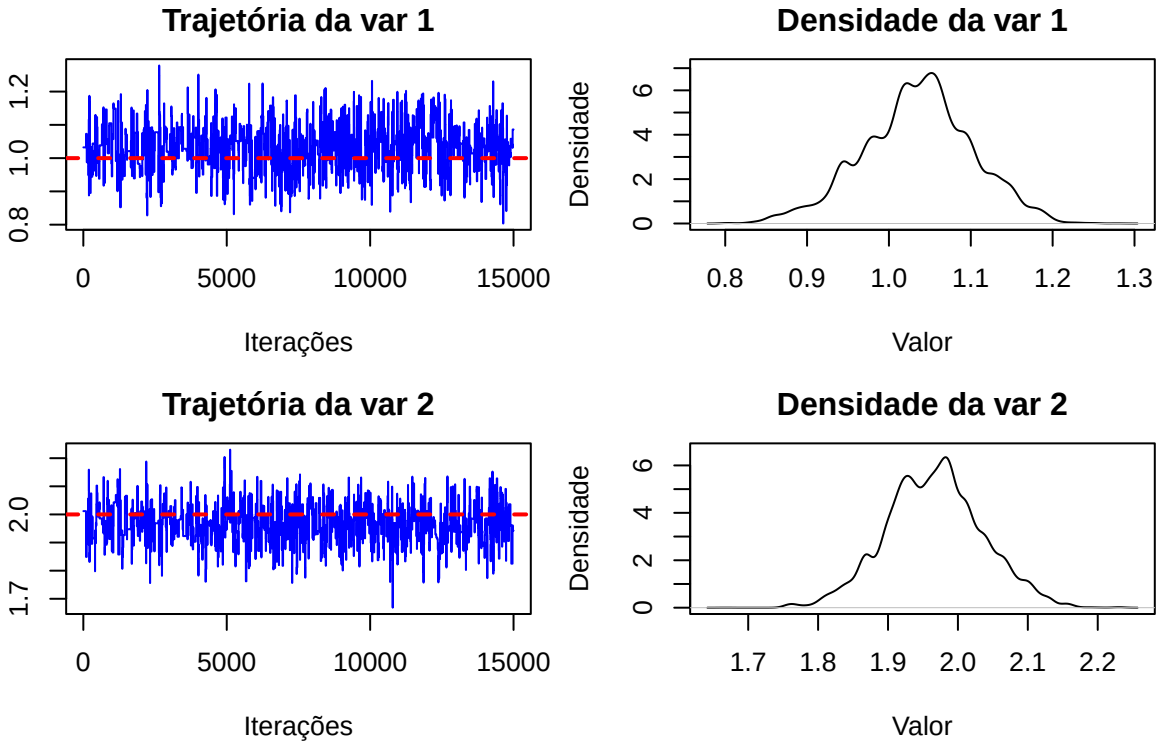
```
simula_dados_reg <- function(n = 200, beta = c(1, 2, -1), sigma = 1, seed = 123) {
  set.seed(seed)
  p <- length(beta)
  X <- cbind(1, matrix(rnorm(n * (p - 1)), n, p - 1))  # Inclui intercepto
  y <- X %*% beta + rnorm(n, 0, sigma)
  list(X = X, y = y, beta = beta, sigma = sigma)
```

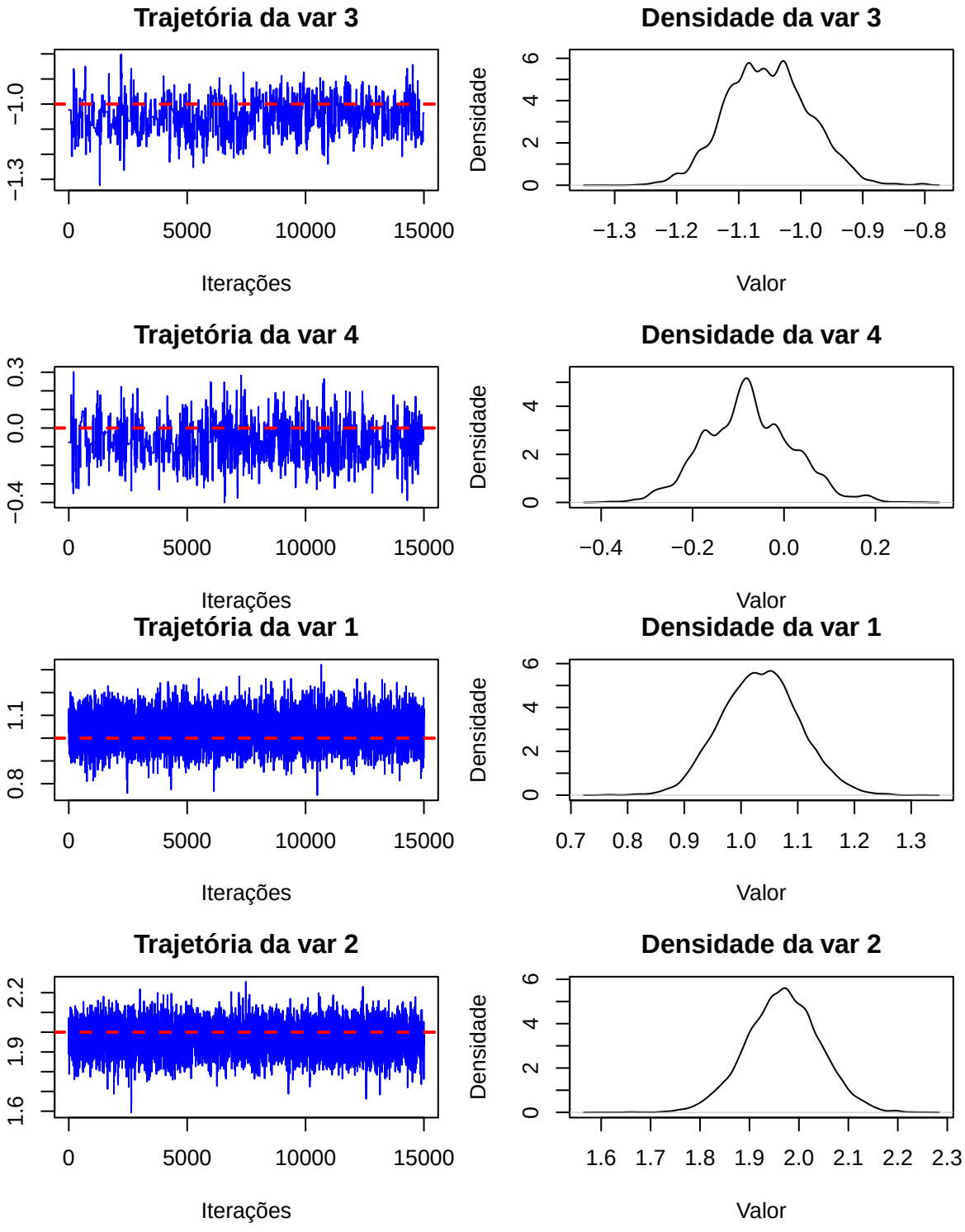
```r
}

log_posterior_reg <- function(params, X, y) {
  beta <- params[1:3]
  sigma2 <- exp(params[4])  # trabalhamos com log(sigma^2)

  # Log-verossimilhança
  mu <- X %*% beta
  log_veross <- sum(dnorm(y, mu, sqrt(sigma2), log = TRUE))

  # Priors:
  # beta ~ N(0, 100I)
  log_prior_beta <- sum(dnorm(beta, 0, 10, log = TRUE))

  # sigma^2 ~ InvGamma(1,1)
  log_prior_sigma <- -2 * log(sigma2) - 1 / sigma2

  return(log_veross + log_prior_beta + log_prior_sigma)
}


dados <- simula_dados_reg(n = 200, beta = c(1,2,-1), sigma = 1, seed = 123)
X <- dados$X
y <- dados$y
p_beta <- ncol(X)
x0_reg <- c(rep(0, p_beta), log(1))
num_iter <- 30000
res_am_reg <- list(amostras = metropolis_adaptativo(
  log_funcao_alvo = function(th) log_posterior_reg(th, X, y),
  ponto_inicial = x0_reg,
  cov_inicial = diag(length(x0_reg)) * 0.01,
  num_iteracoes = num_iter
))
res_amwg_reg <- metropolis_gibbs_adaptativo(
  log_funcao_alvo = function(th) log_posterior_reg(th, X, y),
  x_inicial = x0_reg,
  num_iteracoes = num_iter,
  batch_size = 50
)
valor_real_reg <- c(dados$beta, log(dados$sigma^2))
tabela_reg <- gera_tabela_comparativa(res_am_reg, res_amwg_reg, valor_real_reg)
print(tabela_reg)
```

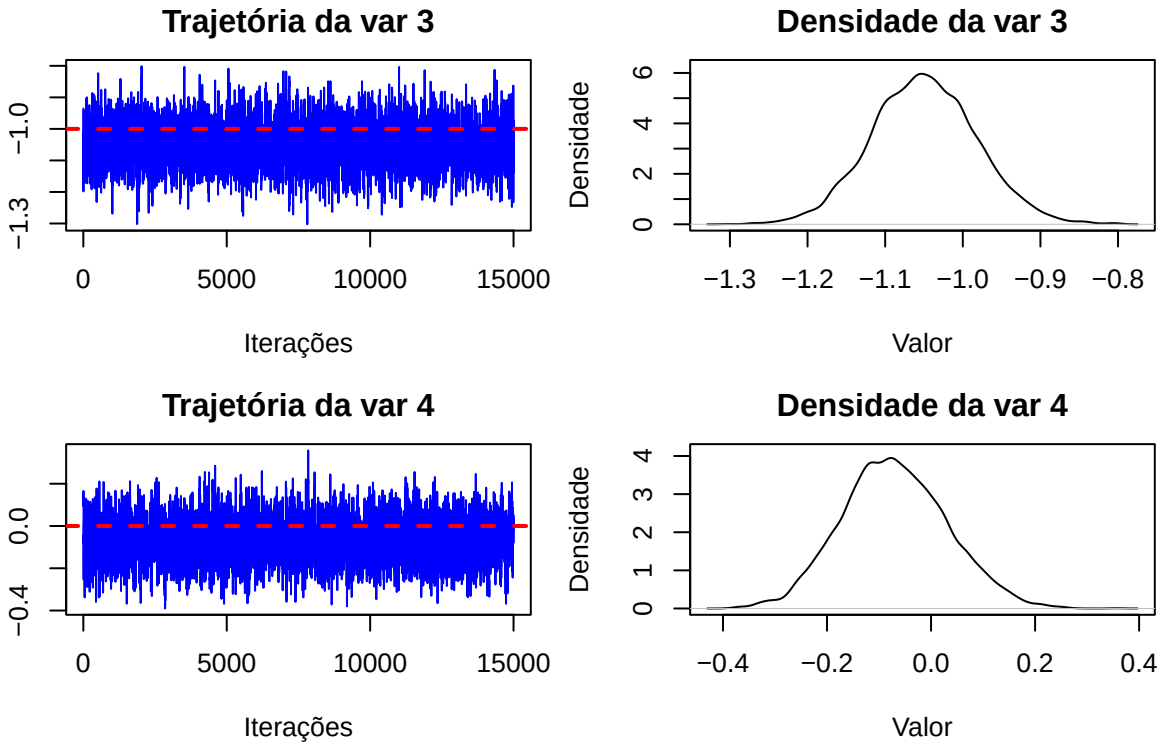
```
##          Método Parâmetro Valor_Verdadeiro   Média    Viés   RMSE     ESS
## var1...1     AM         1                1  1.0351  0.0351 0.0755  524.62
## var2...2     AM         2                2  1.9664 -0.0336 0.0765  488.86
## var3...3     AM         3               -1 -1.0528 -0.0528 0.0850  416.09
## var4...4     AM         4                0 -0.0773 -0.0773 0.1240  656.91
## var1...5   AMwG         1                1  1.0340  0.0340 0.0765 3253.84
## var2...6   AMwG         2                2  1.9679 -0.0321 0.0806 3152.59
## var3...7   AMwG         3               -1 -1.0499 -0.0499 0.0842 3385.20
## var4...8   AMwG         4                0 -0.0700 -0.0700 0.1226 3140.40
##          Taxa_Aceitação
```

```
## var1...1          0.060
## var2...2          0.060
## var3...3          0.060
## var4...4          0.060
## var1...5          0.385
## var2...6          0.385
## var3...7          0.385
## var4...8          0.385
```

```
gera_graficos_comparativos(res_am_reg, res_amwg_reg, valor_real_reg, Sigma =
                             diag(length(valor_real_reg)), burnin_frac = 0.5,
                           titulo_base = "Regressão Bayesiana (posterior)",
                           nome_cenario = "Cenário 3")
```




Regressão Bayesiana (posterior) − AM
Trajetória da var 3
Densidade da var 3
Trajetória da var 4
Densidade da var 4
Trajetória da var 1
Densidade da var 1
Trajetória da var 2
Densidade da var 2
Iterações
Valor
Densidade


Regressão Bayesiana (posterior) − AMwG
Trajetória da var 3
−1.3
−1.0
0
5000
10000
15000
Iterações
Densidade da var 3
Densidade
0
2
4
6
−1.3
−1.2
−1.1
−1.0
−0.9
−0.8
Valor
Trajetória da var 4
−0.4
0.0
0
5000
10000
15000
Iterações
Densidade da var 4
Densidade
0
1
2
3
4
−0.4
−0.2
0.0
0.2
0.4
Valor