UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
DEPARTAMENTO DE ESTATÍSTICA
PROCESSOS ESTOCÁSTICOS

Aplicação de Métodos MCMC

Discente: Vitória Sesana

Doscente: Dr. Mauro Campos

Vitória - ES
2025

Sumário

1 Algoritmo Mantel-Hastings

O algoritmo de Metropolis-Hastings foi implementado para obter uma amostra de tamanho $n = 10000$ da distribuição de Rayleigh quando $\sigma = 4$.

Para este cenário, as distribuições exponencial e log-normal foram propostas para o algoritmo.

O código está disponível no anexo.

No gráfico1, percebemos que a amostra via MH usando a distribuição exponencial como distribuição proposta não se adequa bem, principalmente se for comparado quando se utiliza a distribuição Log-Normal. Por conta disso, o resultado do segundo algoritmo será o escolhido para analisar as estimativas.

Figura 1: Gráfico de densidade das distribuições propostas para gerar valores de Rayleigh.

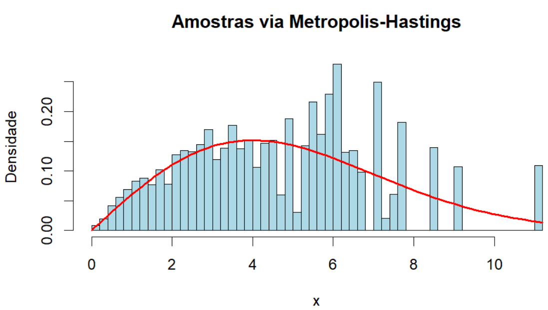


Figura 2: Gráfico 1

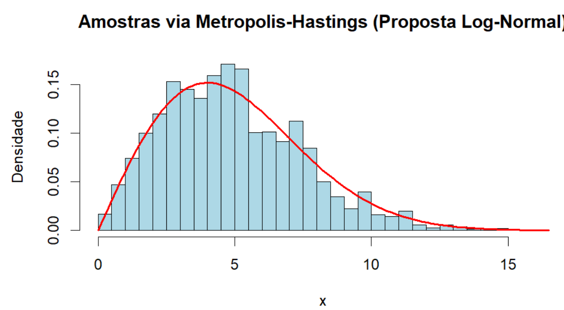


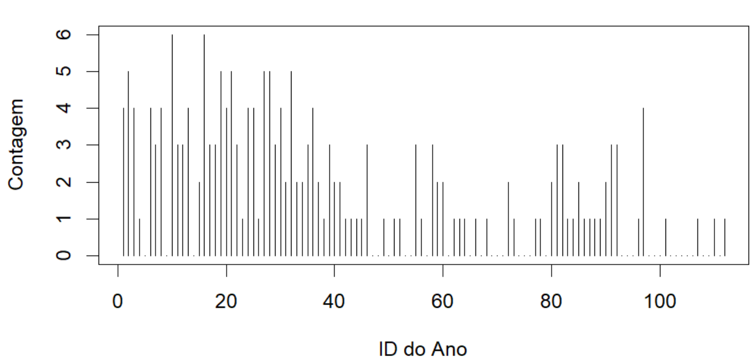
Figura 3: Gráfico 2

Na tabela 1, percebemos que as estimativas não aproximam tanto do valor teórico, apesar de no gráfico apresentado dar a ideia de um bom ajuste.

Tabela 1: Comparação dos valores observados e teóricos para as amostras geradas a partir da distribuição proposta Log-Normal.

	Valor teórico	Média amostral
Média	6.87	4.91
Variância	5.01	6.61

Figura 4: Plot dos dados em contagem.



2 Amostrador de gibbs

2.1 Questão 1: Função de Verossimilhança

Seja y_i o número de acidentes no ano i, para $i = 1, \dots, 112$. Suponha que haja um ponto de mudança $k \in \{1, \dots, 112\}$ tal que:

$$\begin{cases} y_i \sim \text{Poisson}(\lambda), & \text{para } i = 1, \dots, k \\ y_i \sim \text{Poisson}(\nu), & \text{para } i = k+1, \dots, 112 \end{cases}$$

A função de verossimilhança conjunta dos dados, condicional a λ, ν e k, é dada por:

$$\mathcal{L}(\lambda, \nu, k \mid y) = \prod_{i=1}^{k} \frac{\lambda^{y_i} e^{-\lambda}}{y_i!} \cdot \prod_{i=k+1}^{n} \frac{\nu^{y_i} e^{-\nu}}{y_i!}$$

Desprezando os termos constantes em relação aos parâmetros, temos a verossimilhança proporcional a:

$$\mathcal{L}(\lambda, \nu, k \mid y) \propto \lambda^{\sum_{i=1}^{k} y_i} e^{-k\lambda} \cdot \nu^{\sum_{i=k+1}^{n} y_i} e^{-(n-k)\nu}$$

2.2 Questão 2: Distribuição a Posteriori

Assumindo as seguintes distribuições a priori independentes:

$$\lambda \sim \text{Gamma}(\alpha, \beta), \quad \nu \sim \text{Gamma}(\gamma, \delta), \quad k \sim \text{Uniforme Discreta}\{1, 2, \dots, n\}$$

A distribuição a posteriori conjunta dos parâmetros é proporcional ao produto da verossimilhança pelas distribuições a priori:

$$p(\lambda, \nu, k \mid y) \propto \mathcal{L}(\lambda, \nu, k \mid y) \cdot p(\lambda) \cdot p(\nu) \cdot p(k)$$

Substituindo as expressões:

$$p(\lambda, \nu, k \mid y) \propto \lambda^{\sum_{i=1}^{k} y_i + \alpha - 1} e^{-(k+\beta)\lambda} \cdot \nu^{\sum_{i=k+1}^{n} y_i + \gamma - 1} e^{-((n-k)+\delta)\nu}$$

2.3 Questão 3: Distribuições Condicionais Completas

As distribuições completas condicionais, necessárias para a implementação de um amostrador de Gibbs, são:

a) Para λ, condicional a k e y:

$$\lambda \mid \nu, k, y \sim \text{Gamma}\left(\alpha + \sum_{i=1}^{k} y_i,\ \beta + k\right)$$

b) Para ν, condicional a k e y:

$$\nu \mid \lambda, k, y \sim \text{Gamma}\left(\gamma + \sum_{i=k+1}^{n} y_i,\ \delta + (n-k)\right)$$

c) Para k, condicional a λ, ν e y (discreta):

$$p(k \mid \lambda, \nu, y) \propto \lambda^{\sum_{i=1}^{k} y_i} e^{-k\lambda} \cdot \nu^{\sum_{i=k+1}^{n} y_i} e^{-(n-k)\nu}$$

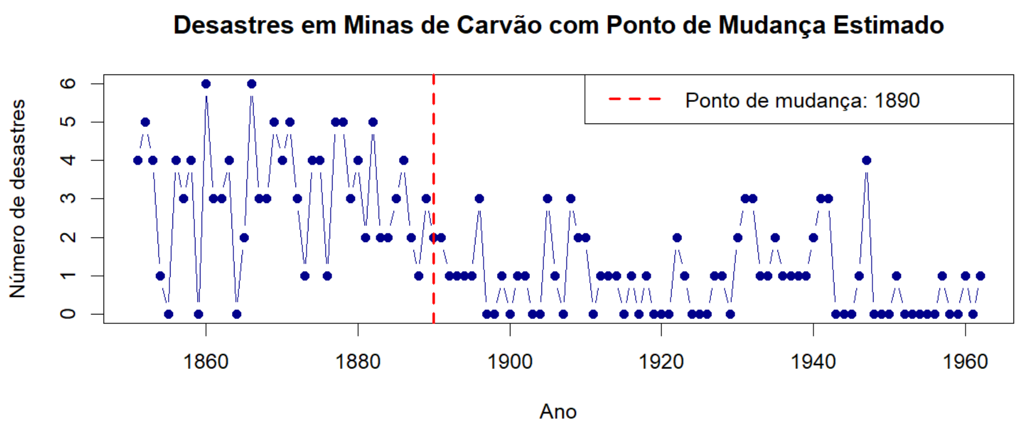
para $k = 1, 2, \dots, n-1$.

2.4 Estimativas

Tabela 2: Estimativas pontuais e intervalares de λ, νe k.

	Estimativa pontual	2.5%	97.5%
λ	3.06	2.52	3.63
ν	0.92	0.71	1.16
k	40.00	36.00	46.00

Figura 5: Gráfico de ponto de mundaça estimado.



3 Anexo I

3.1 Código Algoritmo MH com distribuição proposta Exponencial

```
# aplica  o rayleight ---------------------------------------------------

# distribui  o alvo = rayleight
# distribui  o proposta = exponencial

# Fun  o densidade da distribui  o Rayleigh com sigma = 4
rayleigh_target <- function(x, sigma = 4) {
  ifelse(x < 0, 0, (x / sigma^2) * exp(-x^2 / (2 * sigma^2)))
}

# Fun  o densidade da Exponencial(1)
exp_proposal <- function(x, lambda = 1) {
  ifelse(x < 0, 0, lambda * exp(-lambda * x))
}

# Algoritmo de Metropolis-Hastings
set.seed(123)  # para reprodutibilidade
n_samples <- 10000
samples <- numeric(n_samples)
samples[1] <- rexp(1)  # ponto inicial

for (i in 2:n_samples) {
  current_x <- samples[i - 1]
  proposed_x <- rexp(1)  # proposta a partir de Exponencial(1)

  # C lculo da raz o de aceita  o
  A <- (rayleigh_target(proposed_x) / rayleigh_target(current_x)) *
    (exp_proposal(current_x) / exp_proposal(proposed_x))

  # Aceita ou rejeita
  if (runif(1) < A) {
    samples[i] <- proposed_x
  } else {
    samples[i] <- current_x
  }
}

# Visualiza  o
hist(samples, breaks = 50, probability = TRUE, col = "lightblue",
     main = "Amostras via Metropolis-Hastings",
     xlab = "x", ylab = "Densidade")
curve((x / 16) * exp(-x^2 / (2 * 16)), add = TRUE, col = "red", lwd = 2)  # Densidade Rayleigh(
    =4)


# # comparando -------------------------------------------------------------
#
# library(VGAM)
#
# # Fun  o densidade da distribui  o Rayleigh com sigma = 4
# rayleigh_target <- function(x, sigma = 4) {
#   ifelse(x < 0, 0, (x / sigma^2) * exp(-x^2 / (2 * sigma^2)))
# }
#
# # Fun  o densidade da Exponencial(1)
# exp_proposal <- function(x, lambda = 1) {
#   ifelse(x < 0, 0, lambda * exp(-lambda * x))
# }
#
# # Algoritmo de Metropolis-Hastings
# set.seed(123)
# n_samples <- 10000
# samples <- numeric(n_samples)
# samples[1] <- rexp(1)
#
# for (i in 2:n_samples) {
#   current_x <- samples[i - 1]
#   proposed_x <- rexp(1)
#
```

```r
#   A <- (rayleigh_target(proposed_x) / rayleigh_target(current_x)) *
#     (exp_proposal(current_x) / exp_proposal(proposed_x))
#
#   if (runif(1) < A) {
#     samples[i] <- proposed_x
#   } else {
#     samples[i] <- current_x
#   }
# }
#
# # Gr fico: histograma + densidade te rica com dRayleigh
# hist(samples, breaks = 50, probability = TRUE, col = "lightblue",
#      main = "Amostras Metropolis-Hastings vs Rayleigh( =4)",
#      xlab = "x", xlim = c(0, max(samples)))
#
# # Curva da densidade Rayleigh( =4) com dRayleigh do pacote VGAM
# curve(drayleigh(x, scale = 4), add = TRUE, col = "red", lwd = 2)
#
# legend("topright", legend = c("Densidade te rica Rayleigh( =4)"),
#        col = c("red"), lwd = 2)
#
# samples
#
# mean(samples)
# sd(samples)^2
#
#
# # gamma -------------------------------------------------------------------
#
# # comparando --------------------------------------------------------------
#
# library(VGAM)
#
# # Fun  o densidade da distribui  o Rayleigh com sigma = 4
# rayleigh_target <- function(x, sigma = 4) {
#   ifelse(x < 0, 0, (x / sigma^2) * exp(-x^2 / (2 * sigma^2)))
# }
#
# # Fun  o densidade da Exponencial(1)
# exp_proposal <- function(x, lambda = 1) {
#   ifelse(x < 0, 0, lambda * exp(-lambda * x))
# }
#
# # Algoritmo de Metropolis-Hastings
# set.seed(123)
# n_samples <- 10000
# samples <- rpois(n_samples, lambda = 6)
# samples[1] <- rexp(1)
#
# for (i in 2:n_samples) {
#   current_x <- samples[i - 1]
#   proposed_x <- rexp(1)
#
#   A <- (rayleigh_target(proposed_x) / rayleigh_target(current_x)) *
#     (exp_proposal(current_x) / exp_proposal(proposed_x))
#
#   if (runif(1) < A) {
#     samples[i] <- proposed_x
#   } else {
#     samples[i] <- current_x
#   }
# }
#
# # Gr fico: histograma + densidade te rica com dRayleigh
# hist(samples, breaks = 50, probability = TRUE, col = "lightblue",
#      main = "Amostras Metropolis-Hastings vs Rayleigh( =4)",
#      xlab = "x", xlim = c(0, max(samples)))
#
# # Curva da densidade Rayleigh( =4) com dRayleigh do pacote VGAM
# curve(drayleigh(x, scale = 4), add = TRUE, col = "red", lwd = 2)
#
# legend("topright", legend = c("Densidade te rica Rayleigh( =4)"),
#        col = c("red"), lwd = 2)
```

```
#
# samples
#
# mean(samples)
# sd(samples)^2
#
```

3.2 Código Algoritmo MH com distribuição proposta Log-normal

```
# -----------------------------------------------------
# Aplica  o Rayleigh com proposta Log-Normal
# -----------------------------------------------------

# Distribui  o alvo = Rayleigh
# Distribui  o proposta = Log-Normal

# Fun  o densidade da distribui  o Rayleigh com sigma = 4
rayleigh_target <- function(x, sigma = 4) {
  ifelse(x < 0, 0, (x / sigma^2) * exp(-x^2 / (2 * sigma^2)))
}

# Fun  o densidade da Log-Normal
lognorm_proposal <- function(x, meanlog = 0, sdlog = 1) {
  dlnorm(x, meanlog = meanlog, sdlog = sdlog)
}

# Par metros da proposta Log-Normal
meanlog <- 0
sdlog <- 1

# Algoritmo de Metropolis-Hastings
set.seed(123)  # para reprodutibilidade
n_samples <- 10000
samples <- numeric(n_samples)
samples[1] <- rlnorm(1, meanlog = meanlog, sdlog = sdlog)  # ponto inicial

for (i in 2:n_samples) {
  current_x <- samples[i - 1]
  proposed_x <- rlnorm(1, meanlog = meanlog, sdlog = sdlog)  # proposta a partir de Log-Normal

  # C lculo da raz o de aceita  o
  A <- (rayleigh_target(proposed_x) / rayleigh_target(current_x)) *
    (lognorm_proposal(current_x, meanlog, sdlog) / lognorm_proposal(proposed_x, meanlog, sdlog))

  # Aceita ou rejeita
  if (runif(1) < A) {
    samples[i] <- proposed_x
  } else {
    samples[i] <- current_x
  }
}

# Visualiza  o
hist(samples, breaks = 50, probability = TRUE, col = "lightblue",
     main = "Amostras via Metropolis-Hastings (Proposta Log-Normal)",
     xlab = "x",
     ylab = "Densidade")
curve((x / 16) * exp(-x^2 / (2 * 16)), add = TRUE, col = "red", lwd = 2)  # Densidade Rayleigh(  =


media_obs <- mean(samples)
var_obs <- sd(samples)^2

# valor teorico

media_teorica <- ((4-pi)/2)*(4^2)
var_teorica <- 4*sqrt(pi/ 2)

tab1 <-
  cbind(
      media = c(media_teorica, media_obs),
      var = c(var_teorica, var_obs)
      ) %>% t() %>% round(4) %>%
```

```r
  as.data.frame()

colnames(tab1) <- c("Valor te rico", "M dia amostral")

tab1 %>%
  xtable::xtable()
```

```r
y <- data$Contagem
n <- length(y)

# ----------------------------
# 2. Hiperpar metros
# ----------------------------
alpha <- 1
beta <- 1
gamma <- 1
delta <- 1

n_iter <- 10000

# ----------------------------
# 3. Inicializa  o
# ----------------------------
lambda_samples <- numeric(n_iter)
nu_samples <- numeric(n_iter)
k_samples <- numeric(n_iter)

lambda <- 1
nu <- 1
k <- floor(n / 2)

# ----------------------------
# 4. Gibbs Sampling
# ----------------------------
for (t in 1:n_iter) {
  # Atualizar lambda
  shape_lambda <- alpha + sum(y[1:k])
  rate_lambda <- beta + k
  lambda <- rgamma(1, shape_lambda, rate_lambda)

  # Atualizar nu
  shape_nu <- gamma + sum(y[(k+1):n])
  rate_nu <- delta + (n - k)
  nu <- rgamma(1, shape_nu, rate_nu)

  # Atualizar k
  log_probs <- numeric(n - 1)
  for (ki in 1:(n - 1)) {
    sum1 <- sum(y[1:ki])
    sum2 <- sum(y[(ki + 1):n])
    log_probs[ki] <- sum1 * log(lambda) - ki * lambda +
      sum2 * log(nu) - (n - ki) * nu
  }

  probs <- exp(log_probs - max(log_probs)) # estabiliza  o
  probs <- probs / sum(probs)
  k <- sample(1:(n - 1), 1, prob = probs)

  # Armazenar
  lambda_samples[t] <- lambda
  nu_samples[t] <- nu
  k_samples[t] <- k
}


# analises -----------------------------------------------------------------
# ------------------------------------------------------


# Estimativas pontuais
lambda_est <- mean(lambda_samples)
nu_est <- mean(nu_samples)
k_est <- round(mean(k_samples))
```

```r

cat("Estimativas pontuais:\n")
cat(sprintf("   (antes): %.2f\n", lambda_est))
cat(sprintf("   (depois): %.2f\n", nu_est))
cat(sprintf("Ponto de mudan a estimado (k): %d (ano %d)\n", k_est, 1850 + k_est))

# Intervalos de credibilidade
ci_lambda <- quantile(lambda_samples, c(0.025, 0.975))
ci_nu <- quantile(nu_samples, c(0.025, 0.975))
ci_k <- quantile(k_samples, c(0.025, 0.975))

cat("\nIntervalos de credibilidade 95%:\n")
cat(sprintf("  : (%.2f, %.2f)\n", ci_lambda[1], ci_lambda[2]))
cat(sprintf("  : (%.2f, %.2f)\n", ci_nu[1], ci_nu[2]))
cat(sprintf("k: (%d, %d)     anos (%d, %d)\n",
            ci_k[1], ci_k[2], 1850 + ci_k[1], 1850 + ci_k[2]))


plot(data$Ano, y, type = "b", pch = 16, col = "darkblue",
     xlab = "Ano", ylab = "N mero de desastres",
     main = "Desastres em Minas de Carv o com Ponto de Mudan a Estimado")
abline(v = 1850 + k_est, col = "red", lwd = 2, lty = 2)
legend("topright", legend = paste("Ponto de mudan a:", 1850 + k_est),
       col = "red", lty = 2, lwd = 2)

estimativas <- rbind(lambda_est, nu_est, k_est) %>%
  round(2)

ics <-
  rbind(t(ci_lambda), t(ci_nu), t(ci_k)) %>%
  round(2)

cbind(estimativas, ics) %>%
  xtable::xtable()
```