Simulated Annealing (Recozimento Simulado)

UFMG

10/11/2025

Sumário

Recozimento Simulado - Introdução

Também chamado de método meta-heurístico, este tem como intuito resolver problemas de otimização de grande complexidade. Este algoritmo foi introduzido por volta de 1980 por três pesquisadores, sendo eles: Kirkpatrick, Gelatt e Vecchi. O método traz soluções sub-ótimas, ou seja, eficientes e sem grande esforço computacional.
Tem como objetivo encontrar resultados satisfatórios, ainda que não exatos, sendo que muitas vezes estes resultados são inviáveis e/ou difíceis de serem alcançados.
Se enquadra na classe *Markov chain Monte Carlo* (MCMC) e, por ter caráter estocástico alinhado à sua capacidade de escapar de mínimos locais, este é um ótimo candidato para resolução de problemas complexos de otimização.

Processo de Recozimento

O conceito foi introduzido da ideia de recozimento, onde um sólido é levado para um estado de baixa energia após aumentar sua temperatura.
O processo consiste em dua estapas, sendo elas:

- "Derretimento" da sua estrutura ao levá-lo a uma temperatura muito elevada.
- Resfriamento esquematizado buscando atingir um estado sólido de energia mínima.

Uma vez em estado líquido as partículas do material exposto são distribuídas de forma aleatória.
O estado mínimo de energia só é alcançado com uma temperatura inicial que seja suficientemente alta e um tempo de resfriamento que seja logo o suficiente.

Aplicabilidade

O algoritmo **Simulated Annealing** tem aplicabilidade em várias áreas e aqui serão expostas algumas delas:

- Engenharia de software
- *Machine Learning*
- Teoria de filas
- Processos de manufatura
- logística e transporte com otimização

Dist. Weibull Tri-Paramétrica

O estudo será centrado em estimar os parâmetros primeiramente de uma distribuição Weibull com a seguinte função densidade:

$$f(x) = \frac{\beta}{\eta}\left(\frac{x-\gamma}{\eta}\right)^{\beta-1} e^{-\frac{x-\gamma}{\eta}^{\beta}}; \ \beta > 0, \ \eta > \gamma \geq 0. \tag{1}$$

Tenha ciência também que sua acumulada tem o seguinte resultado:

$$F(x) = 1 - e^{-\frac{x-\gamma}{\eta}^{\beta}},$$

sendo β, η, γ os parâmetros de forma, escala e locação, respectivamente.

A distribuição Weibull tri-paramétrica é raramente usada devido à dificuldade de se estimar estes parâmetros. Portanto, o uso de SA será feito para tentar estimar estes parâmetros. Aqui o algoritmo será fundamental para maximizar a função de verossimilhança.

Estimação por EMV

Temos ciência que a função de máxima verossimilhança é descrita da seguinte forma:

$$L(x) = \prod_{i=1}^{n} f_{x_i}(x_i\theta)$$

Com isso, a log-verossimilhança é a seguinte:

$$\text{Ln}(L(x_1, \ldots, x_n, \beta, \eta, \gamma)) = n\text{Ln}\left(\frac{\beta}{\eta}\right) + \sum_{i=1}^{n}\left(-\left(\frac{x_i - \gamma}{\eta}\right)^{\beta} + (\beta - 1)\text{Ln}\left(\frac{x_i - \gamma}{\eta}\right)\right). \tag{2}$$

Para encontrar as estimativas dos parâmetros devemos realizar as derivadas parciais e igualar a zero, da seguinte forma:

$$\frac{\partial}{\partial\theta} Ln(L(x)) = 0$$

Fica evidente que esta é uma função bem difícil de derivar, necessitando derivar gradiente ou mesmo aplicar métodos numéricos, portanto prosseguiremos para a ideia geral do algoritmo SA

Algoritmo

A principal vantagem deste método perante os demais é fato de que este, ao se empregar uma busca aleatória, **não se prende a pontos de mínimos locais**. O algoritmo aceita mudanças que aceitam a diminuição e também o aumento da função objetivo f. O seu aumento é aceito com probabilidade: $p = e^{-\frac{\Delta}{T}}$ sendo θ o aumento e T o parâmetro de controle, analogamente conhecido como temperatura do sistema. A sua implementação é simples:

- Uma representação de soluções possíveis,
- Um gerador de mudanças aleatórias nas soluções,
- Um meio de avaliar as funções do problema, e
- Um esquema de resfriamento (annealing schedule) – uma temperatura inicial e regras para diminuí-la à medida que a busca progride.

```
simulated_annealing <- function(f, S0, T0, L, alpha, T_min) {
   T <- T0; S <- S0
   S_star <- S0; f_S <- f(S)
   f_S_star <- f_S
   while (T > T_min) {
       for (n in 1:L) {
           S_n <- gerar_vizinho(S)
           f_S_n <- f(S_n); Delta <- f_S_n - f_S
           if (Delta <= 0) {
               S <- S_n; f_S <- f_S_n
           } else {
               p <- exp(-Delta / T)
               if (runif(1) < p) {
                   S <- S_n; f_S <- f_S_n
               }}}
       if (f_S < f_S_star) {
           S_star <- S; f_S_star <- f_S
       }
       T <- T * alpha
   }
   return(list(best_solution = S_star, best_cost = f_S_star))}
```

Por ser um algoritmo meta-heurístico, urge a importância de o mesmo ter a possibilidade de aceitar uma solução pior, evitando que o algoritmo caia em um máximo local e permaneça por lá. Claramente a probabilidade de aceitar uma pior solução diminui na mesma medida que a temperatura cai em cada um dos outros ciclos. A performance do algoritmo portanto é definida dentro de parâmetros de controle, sendo eles:

- A temperatura inicial (T_0) deve ser grande o suficiente para que na primeira iteração do algoritmo,a probabilidade de aceitação de um cenário ruim seja, pelo menos 80%;
- É utilizada a função de redução, sendo esta geométrica comprrendida por: $T_i = CT_{i-1}$, onde $C < 1$, constante e se encontra usualmente entre 0.75 e 0.95;
- O tamanho de cada nível de temperatura determina o número de soluções geradas em uma certa temperatura T;
- O critério de parada define quando o sistema encontrou o nível de eneergia desejado. Em suma, isso define:
  - O totald e números de soluções geradas.
  - A temperatuira a cada nível de energia desejado.
  - A razão de aceitação (entre o número de soluções aceitadas e geradas).

Algoritmo SA para Maximização da Verossimilhança

Estimativa de β, η, γ

Passo	Ação / Descrição
1	**Obter Amostra** aleatória (tamanho grande o suficiente).
2	**Determinar Parâmetros de Controle** (T_0, T_f, C, I).
3	**Gerar Solução Inicial** aleatória (a, b, c).
4	**Computar Verossimilhança Inicial** (L) na solução (a, b, c).
	Loop Externo (Resfriamento)
5	**Enquanto** $T > T_0$: $T \leftarrow CT$
	Loop Interno (Equilíbrio)
6	**Para** $i = 1$ a I:
6.1	**Gerar Vizinho:** Gerar a_1, b_1, c_1 vizinhos de a, b, c.
6.2	**Calcular Verossimilhança Vizinha** (L_0) em (a_1, b_1, c_1).
6.3	**Avaliar Parâmetros (Critério de Metropolis):**
6.3.1	**Se** $L_0 > L$ (Melhora): $a \leftarrow a_1, b \leftarrow b_1, c \leftarrow c_1,$ e $L \leftarrow L_0$
6.3.2	**Senão** (Piora): Gerar $u \sim \text{Uni}(0, 1)$.

6.3.2.1	**Se** $u < e^{\frac{(F_0 - F)}{T}}$ (Aceitação Probabilística): $a \leftarrow a_1, b \leftarrow b_1, c \leftarrow c_1$
(Fim do Loop Interno)	
(Fim do Loop Externo)	
7	**Resultado:** Imprimir a, b, c e L.
Nota: a, b, c são estimativas de β, η, γ. F é uma função de custo/energia (ex: $-\ln(L)$).	

Código implementado

```
loglik_weibull3 <- function(params, x) {
  b <- params[1] # beta (forma)
  g <- params[2] # eta (escala)
  c <- params[3] # gamma (locação)

  if (b <= 0 || g <= 0 || any(x <= c)) {
    return(-Inf)
  }

  z <- (x - c) / g
  # Fórmula Log-Verossimilhança (Eq 7 do artigo)
  ll <- length(x) * (log(b) - log(g)) + sum((b - 1) * log(z) - (z^b))
  return(ll)
}
```

```
propose_neighbor <- function(curr, x, sds = c(0.1, 0.1, 0.1)) {
 attempt <- 0
 min_x <- min(x)
 repeat {
   attempt <- attempt + 1
   b1 <- curr[1] + rnorm(1, mean = 0, sd = sds[1])
   g1 <- curr[2] + rnorm(1, mean = 0, sd = sds[2])
   c1 <- curr[3] + rnorm(1, mean = 0, sd = sds[3])
   if (b1 <= 0) {
     b1 <- abs(b1) + 1e-6
   }
   if (g1 <= 0) {
     g1 <- abs(g1) + 1e-6
   }
   if (c1 < min_x - 1e-8) {
     return(c(b1, g1, c1))
   }
   if (attempt > 50) {
     sds[3] <- sds[3] * 1.5
   }
   if (attempt > 1000) {
     return(c(b1, g1, min_x - 1e-6))}}}
```

```
    sa_weibull3 <- function(
  x,
  init = NULL,
  T0 = 100,
  Tf = 0.001,
  cooling_rate = 0.99,
  L = 5,
  sds = c(0.1, 0.1, 0.1)
) {
  n <- length(x)
  # Lógica de Inicialização (Simplificada)
  if (is.null(init)) {
    c0 <- min(x) - 0.1 * sd(x)
    if (c0 >= min(x)) {
      c0 <- min(x) - 1e-3
    }
    y <- x - c0
    y[y <= 0] <- min(y[y > 0]) * 0.1
    y_sorted <- sort(y)
    prob_rank <- (1:n) / (n + 1)
    ln_y <- log(y_sorted)
    ln_ln <- log(-log(1 - prob_rank))
```

```
  fit <- try(lm(ln_ln ~ ln_y), silent = TRUE)
  if (inherits(fit, "try-error")) {
    init <- c(1.5, sd(x), min(x) - 0.1)
  } else {
    b0 <- fit$coefficients[2]
    g0 <- exp(-fit$coefficients[1] / b0)
    init <- c(b0, g0, c0)
    if (
      !is.finite(b0) || !is.finite(g0) || !is.finite(c0) || b0 <= 0 || g0
    ) {
      init <- c(1.5, sd(x), min(x) - 0.1)
    }
  }
}
curr <- init
curr_ll <- loglik_weibull3(curr, x)
best <- curr
best_ll <- curr_ll
T <- T0
history <- data.frame(eval = 1, ll = curr_ll)
eval_count <- 1

# Loop SA
```

```
while (T > Tf) {
  for (i in 1:L) {
    prop <- propose_neighbor(curr, x, sds = sds)
    prop_ll <- loglik_weibull3(prop, x)
    if (prop_ll > curr_ll) {
      curr <- prop
      curr_ll <- prop_ll
    } else {
      delta_ll <- prop_ll - curr_ll

      if (runif(1) < exp(delta_ll / T)) {
        curr <- prop
        curr_ll <- prop_ll
      }
    }
    if (curr_ll > best_ll) {
      best <- curr
      best_ll <- curr_ll
    }
    eval_count <- eval_count + 1
    history <- rbind(history, data.frame(eval = eval_count, ll = curr_ll)
  }
  T <- cooling_rate * T}
```

```
  return(list(
    best_params = best,
    best_ll = best_ll,
    history = history,
    evals = eval_count
  ))
}
```

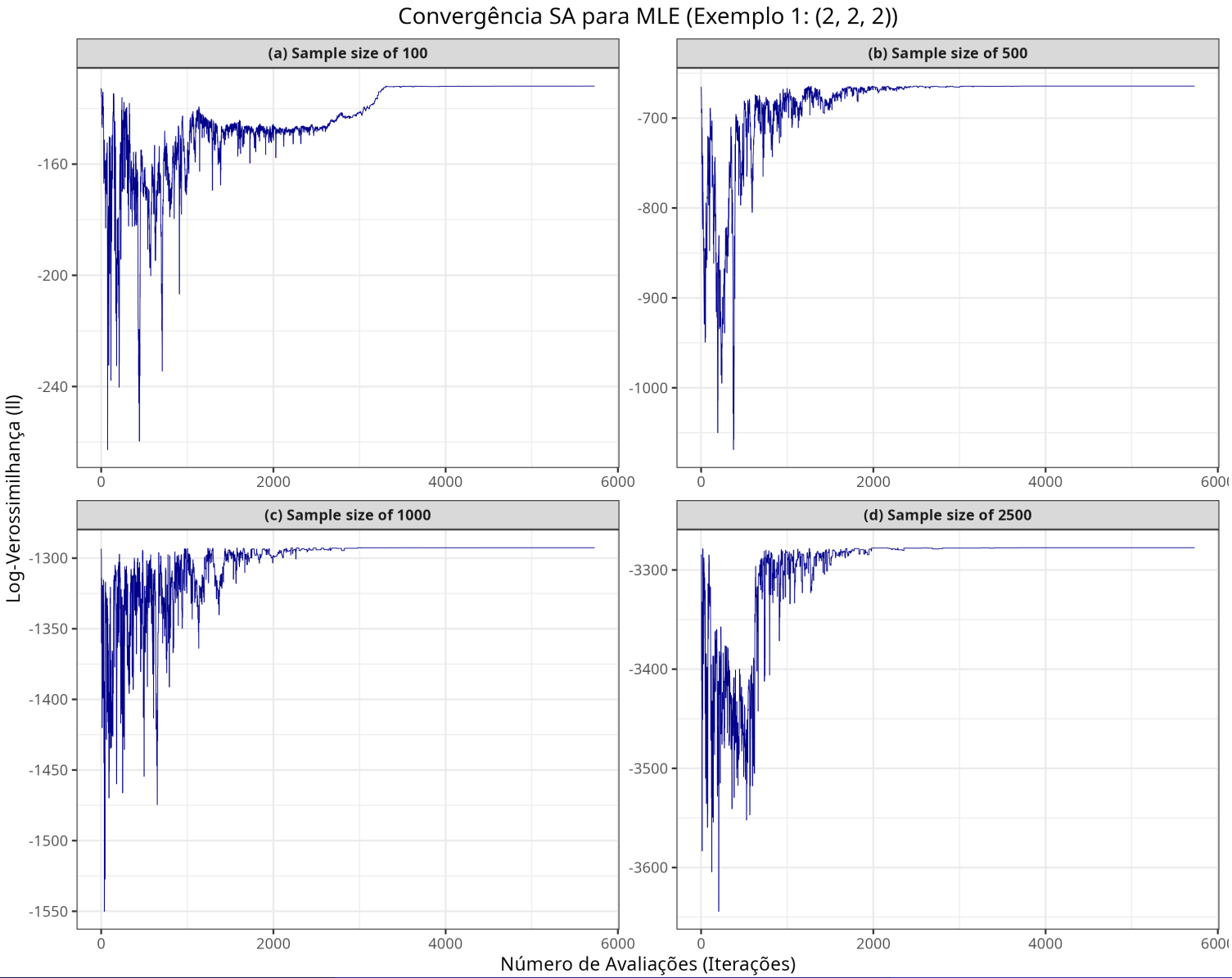
Resultados da Estimação por Recozimento Simulado

Weibull ($\beta = 2,\ \eta = 2,\ \gamma = 2$)

Weibull parameters			**(2, 2, 2)**	
Measure	**2500**	**1000**	**500**	**100**
Estimated parameters (β, η, γ)	(2.0298, 2.0338, 1.9995)	(1.9511, 1.9051, 2.0354)	(1.8329, 1.9306, 2.0718)	(1.8392, 1.6363, 2.2518)
Likelihood function at the real values	-3234.8000	-1305.5000	-618.3805	-137.2584
Likelihood function at the estimated value	-3235.4000	-1301.8000	-615.3697	-134.6563
Run time (s)	100.6719	96.2500	89.6094	72.2188

Resultados da Estimação por Recozimento Simulado

Weibull ($\beta = 2,\ \eta = 2,\ \gamma = 2$)



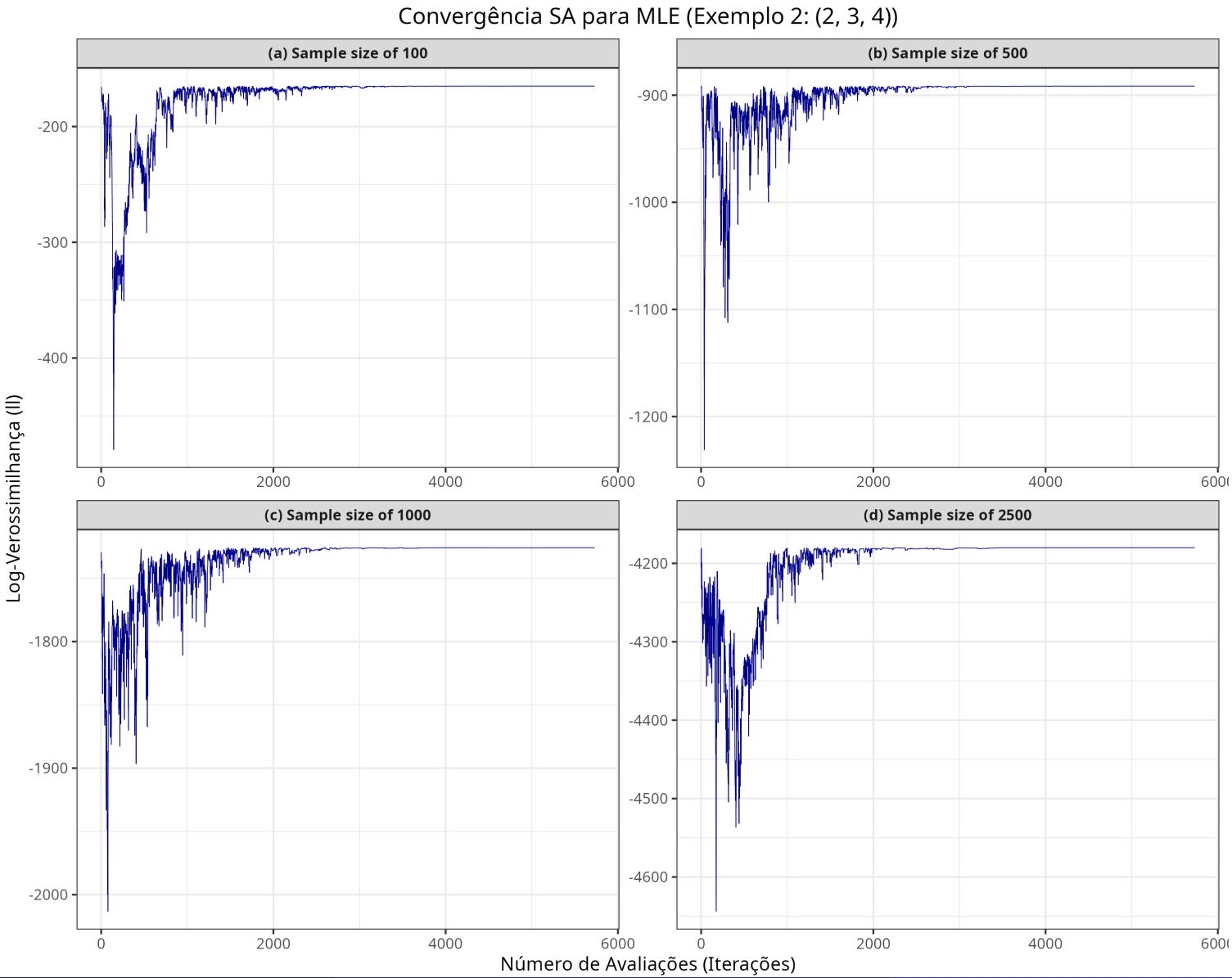
Resultados da Estimação por Recozimento Simulado

Weibull ($\beta = 2,\ \eta = 2,\ \gamma = 2$)

Weibull parameters			**(2, 2, 2)**	
Measure	**2500**	**1000**	**500**	**100**
Estimated parameters (β, η, γ)	(2.0298, 2.0338, 1.9995)	(1.9511, 1.9051, 2.0354)	(1.8329, 1.9306, 2.0718)	(1.8392, 1.6363, 2.2518)
Likelihood function at the real values	-3234.8000	-1305.5000	-618.3805	-137.2584
Likelihood function at the estimated value	-3235.4000	-1301.8000	-615.3697	-134.6563
Run time (s)	100.6719	96.2500	89.6094	72.2188

Resultados da Estimação por Recozimento Simulado

Weibull ($\beta = 2,\ \eta = 2,\ \gamma = 2$)



Resultados da Estimação por Recozimento Simulado

Weibull ($\beta = 2,\ \eta = 2,\ \gamma = 2$)

Weibull parameters			**(2, 2, 2)**	
Measure	**2500**	**1000**	**500**	**100**
Estimated parameters (β, η, γ)	(2.0298, 2.0338, 1.9995)	(1.9511, 1.9051, 2.0354)	(1.8329, 1.9306, 2.0718)	(1.8392, 1.6363, 2.2518)
Likelihood function at the real values	-3234.8000	-1305.5000	-618.3805	-137.2584
Likelihood function at the estimated value	-3235.4000	-1301.8000	-615.3697	-134.6563
Run time (s)	100.6719	96.2500	89.6094	72.2188

Resultados da Estimação por Recozimento Simulado

Weibull ($\beta = 2,\ \eta = 2,\ \gamma = 2$)