# Estatística Bayesiana - Exercícios Computacionais

Eduardo Amaral

26/03/2024

## Exercicio 1

```
library(rstan)
library(pander)

# Dados de falhas
falhas <- c(27, 16, 11, 10, 11, 7, 2, 5, 3, 1, 4, 7, 2, 5, 5, 6, 0, 5, 1, 1, 2, 1, 2, 1, 1)

# Hiperparâmetros conhecidos
b1 <- 1; b2 <- 1; e1 <- 1; e2 <- 1

# Dados para o modelo
data <- list(N = length(falhas), falhas = falhas, b1 = b1, b2 = b2, e1 = e1, e2 = e2)

# Definindo o modelo do problema proposto
modelo <- "
data {
  int<lower=0> N;  // número de períodos de tempo
  int<lower=0> falhas[N];  // número de falhas em cada período de tempo
  real<lower=0> b1;  // hiperparâmetro para lambda_a
  real<lower=0> b2;  // hiperparâmetro para lambda_a
  real<lower=0> e1;  // hiperparâmetro para k1
  real<lower=0> e2;  // hiperparâmetro para k1
}
parameters {
  real<lower=0> lambda_a;  // parâmetro lambda_a
  real<lower=0,upper=1> k1;  // parâmetro k1
}
model {
  lambda_a ~ gamma(b1, b2);  // priori para lambda_a
  k1 ~ beta(e1, e2);  // priori para k1
  for (i in 1:N) {
    falhas[i] ~ poisson(lambda_a * pow(k1, i));  // verossimilhança
  }
}
"
```

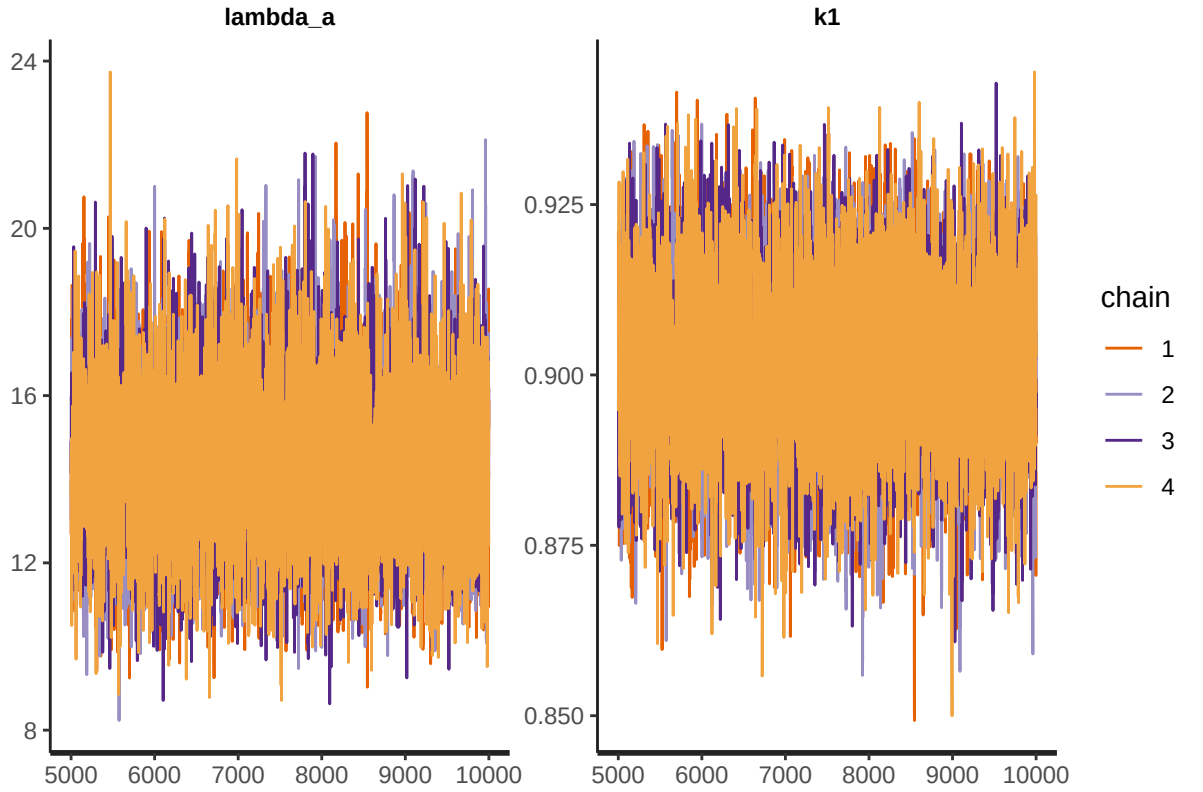
```
# Compilar o modelo
model <- stan_model(model_code=modelo)
```

```r
# Amostragem
fit <- sampling(model, data = data, iter = 10000, chains = 4, seed = 110108)
```

```r
# Resumos a posteriori
print(fit)
```

```
## Inference for Stan model: anon_model.
## 4 chains, each with iter=10000; warmup=5000; thin=1;
## post-warmup draws per chain=5000, total post-warmup draws=20000.
##
##            mean se_mean   sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
## lambda_a  14.38    0.02 1.86  11.01  13.08  14.30  15.58  18.24  6584    1
## k1         0.90    0.00 0.01   0.88   0.89   0.90   0.91   0.93  6922    1
## lp__     121.65    0.01 1.02 118.90 121.26 121.97 122.37 122.63  6574    1
##
## Samples were drawn using NUTS(diag_e) at Sat Mar 16 02:14:44 2024.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

Demonstro no bloco acima os códigos utilizados para a modelagem e amostragem, foi utilizado de 4 cadeias, cada uma com 10000 interações, lembrando que metade dessas interações em cada cadeia é usado como burn, considerando os 4 hiperparâmetros conhecidos iguais a 1. Os parâmetros do modelo são $\lambda_a$ com média 14.38 e desvio padrão 1.86 temos também $k_1$ com média 0.90 e desvio padrão 0.01 . Além disso, lp___ é o logaritmo da densidade de probabilidade a posteriori, que é uma medida do quão bem o modelo se ajusta aos dados. A média de lp___ é 121.65 com um desvio padrão de 1.02. Para todos os parâmetros, $\widehat{R}$ é 1, o que indica uma boa convergência das cadeias.




Ambos os gráficos mostram o comportamento de cada parâmetro ao longo das interações, sendo o da esquerda referente ao $\lambda_a$ e o da direita referente ao $k_1$. Note que cada uma das 4 linhas se referem a uma cadeia de Markov diferente. Podemos identificar que ocorre uma certa sobreposição entre estas, indicando uma convergência da série, com isso temos que os parâmetros foram bem estimados.

Portanto, foi utilizado um modelo Bayesiano para analisar os dados de falhas de um software de tomografia computadorizada. O modelo assume um processo de Poisson homogêneo com função de intensidade $\lambda_i = \lambda_a \sum_{i=1}^{n} k_1^i$. O parâmetro $\lambda_a$ assume uma priori Gama e o $k_1$ Beta, foi realizado uma

As distribuições a priori para os parâmetros $\lambda_a$ e $k_1$ são Gama e Beta, respectivamente. Realizamos a amostragem MCMC em 4 cadeias diferentes, cada uma com 10000 iterações. Os resultados mostram que as cadeias convergiram bem, indicando que as estimativas dos parâmetros são confiáveis. Os gráficos de rastreamento confirmam a convergência das cadeias para os parâmetros $\lambda_a$ e $k_1$. A sobreposição e estabilidade das linhas indicam que as cadeias convergiram, sugerindo que os parâmetros foram bem estimados pelo modelo.

## Exercicio 2

```r
# Dados a serem utilizados
t1 <- c(26, 63, 19, 66, 40, 49, 8, 69, 39, 82, 72, 66, 25, 41, 16, 18, 22, 42, 36)
t2 <- c(20, 18, 19, 85, 40, 49, 8, 71, 39, 48, 72, 62, 9, 3, 75, 18, 14, 42, 52)

# Definindo a função de verossimilhança
loglike <- function(par) {
  if (any(par <= 0)) return(Inf)
  lambda1 <- par[1]
  lambda2 <- par[2]
  a1 <- par[3]
  b1 <- par[4]
  a2 <- par[5]
  b2 <- par[6]

  # Calculamos a log-verossimilhança para as distribuições exponenciais e gama
  ll <- sum(dexp(t1, rate=lambda1, log=TRUE)) + sum(dexp(t2, rate=lambda2, log=TRUE)) +
    dgamma(lambda1, shape=a1, rate=b1, log=TRUE) + dgamma(lambda2, shape=a2, rate=b2, log=TRUE)

  return(-ll)
}

# Maximizando a verossimilhança
start <- c(1, 1, 1, 1, 1, 1) # Valores iniciais para os parâmetros
result <- optim(start, loglike, method="BFGS") # Usamos o método BFGS para otimização

df <- data.frame(result$par); names(df) <- 'Estimativas'
pander(df)
```

| Estimativas |
|---|
| 0.02342 |
| 0.02482 |
| 3.168 |
| 104.1 |
| 2.842 |
| 96.29 |

```r
# Verificando a convergência
result$convergence
```

```
## [1] 0
```

Portanto, podemos concluir que o algoritmo convergiu.

## Exercicio 3

```
# Dados
y <- c(0.10, 0.65, 0.30, 0.30, 0.28, 0.78, 0.28, 0.45)
X1 <- c(0.08, 0.17, 0.08, 0.30, 0.05, 0.18, 0.09, 0.45)
X2 <- c(0.40, 0.40, 0.38, 0.50, 0.52, 0.32, 0.45, 0.65)
X3 <- c(0.75, 1.02, 1.09, 1.35, 1.20, 2.20, 2.95, 2.50)
dados <- list(N = length(y), y = y, X1 = X1, X2 = X2, X3 = X3)

# Modelo
modelo <- "
data {
  int<lower=0> N;
  vector[N] y;
  vector[N] X1;
  vector[N] X2;
  vector[N] X3;
}
parameters {
  real alpha;
  real beta1;
  real beta2;
  real beta3;
  real<lower=0> sigma;
}
model {
  y ~ normal(alpha + beta1 * X1 + beta2 * X2 + beta3 * X3, sigma);
}
"

# Compilar o modelo
mod <- stan_model(model_code = modelo)

# Amostragem
fit <- sampling(mod, data = dados, iter = 10000, chains = 4,
                seed = 110108,
                control = list(adapt_delta = 0.95, max_treedepth = 15))
```

```
## Warning: There were 2 divergent transitions after warmup. See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

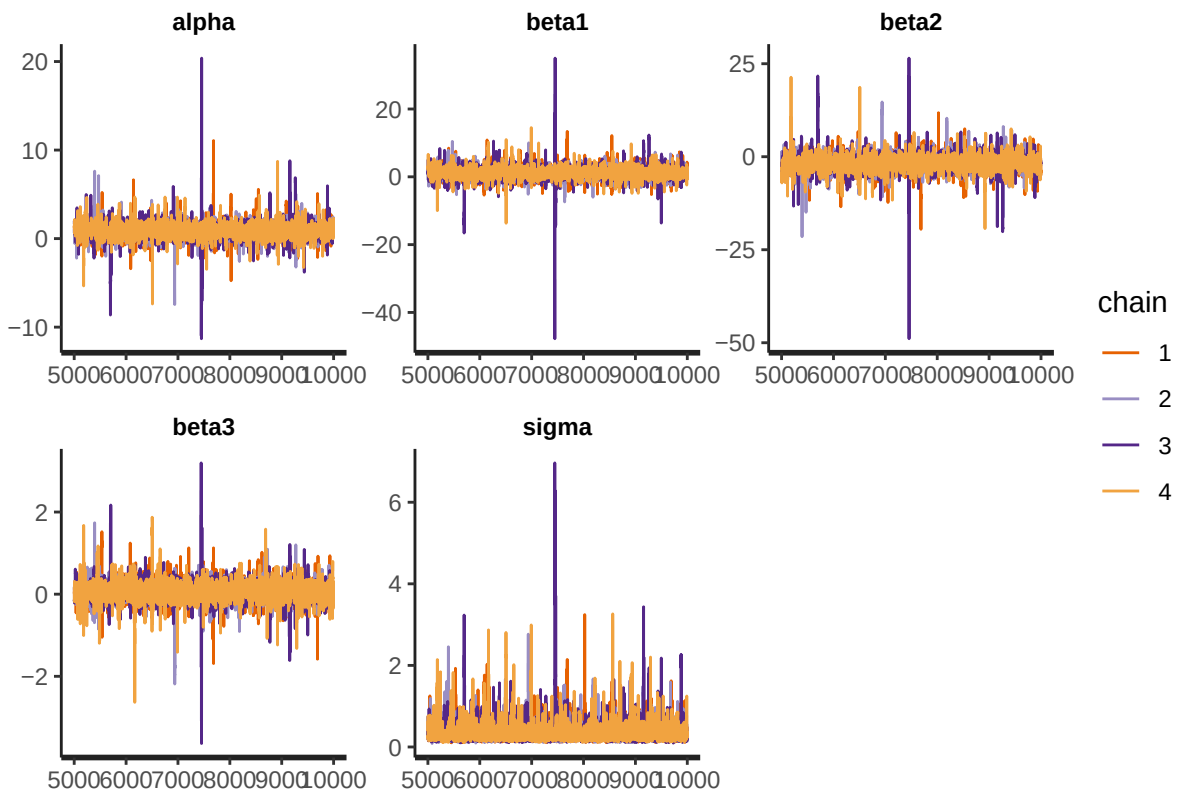
```r
# Resumos a posteriori
print(fit, probs = c(0.025, 0.975))
```

```
## Inference for Stan model: anon_model.
## 4 chains, each with iter=10000; warmup=5000; thin=1;
## post-warmup draws per chain=5000, total post-warmup draws=20000.
##
##        mean se_mean   sd  2.5% 97.5% n_eff Rhat
## alpha  0.87    0.01 0.80 -0.53  2.37  5088    1
## beta1  1.21    0.02 1.58 -1.51  4.09  6307    1
## beta2 -1.76    0.03 1.95 -5.47  1.76  4553    1
## beta3  0.07    0.00 0.21 -0.33  0.46  7317    1
## sigma  0.34    0.01 0.25  0.14  0.91  1914    1
## lp__   5.14    0.06 2.83 -1.87  8.80  2028    1
##
## Samples were drawn using NUTS(diag_e) at Sat Mar 16 02:16:02 2024.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

Temos então que o modelo sera da forma:

$$y_i = 0.87 + 1.21X_{1i} - 1.76X_{2i} + 0.07X_{3i} + \epsilon_i, \epsilon \sim N(0, 0.34^2)$$

```
traceplot(fit)
```




Podemos observar que as cadeias apresentam de forma geral uma convergência,