Implementação Bayesiana via Stan da distribuição normal com ϕ e μ desconhecidos

Aluna: Isabelle Fernandes de Oliveira - Curso Estatística

Estatística Bayesiana - 1º Semestre de 2025

Exercício

$Y_i \sim \mathcal{N}(\mu, \frac{1}{\phi})$

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))
```

Especificação a priori

$\mu \sim \mathcal{N}(m, v)$ $\phi \sim \text{Ga}(a, b)$

```
m = 5 # media de mu
v = 10 # variacia de mu
a = 0.1 # media phi = 1
b = 0.1 # variancia phi = 10
```

Transmitindo informações para o Stan

```
data = list(n = n, y = y, m = m, v = v, a = a, b = b)
```

```
# Lista requisitando que mu_real e phi_real sejam salvos.
pars = c("mu_real", "phi_real")
```

```
# Lista de sementes de inicialização
init = list()
init[[1]] = list(mu_real=5, phi_real=1)
init[[2]] = list(mu_real=-5, phi_real=5)
```

```r
iter = 2000 #Total de iterações (incluindo burn-in).
warmup = 1000 # Numero de iterações do burn-in.
chains= 2 # Numero de cadeias do MCMC.
```

```stan
// Bloco de declaração de dados

data{
  int<lower=1> n;
  vector[n] y;
  real m;
  real<lower=0> v;
  real<lower=0> a;
  real<lower=0> b;
}

 // Bloco de declaração de parâmetros

parameters{
  real mu_real;
  real<lower=0> phi_real;
}

// Bloco de parâmetros transformados

transformed parameters{
  real sigma2;
  sigma2 = 1/phi_real;
}

// Bloco do modelo.

model{
  // Verossimilhança
  for(i in 1:n){y[i] ~ normal(mu_real[i], sqrt(1/phi_real));}

  // Priori 1:  mu_real ~ N(m,v)
  mu_real ~ normal(m,v);

  // Priori 2: phi ~ Gama(a, b)
  phi ~ gamma(a,b);

}
```

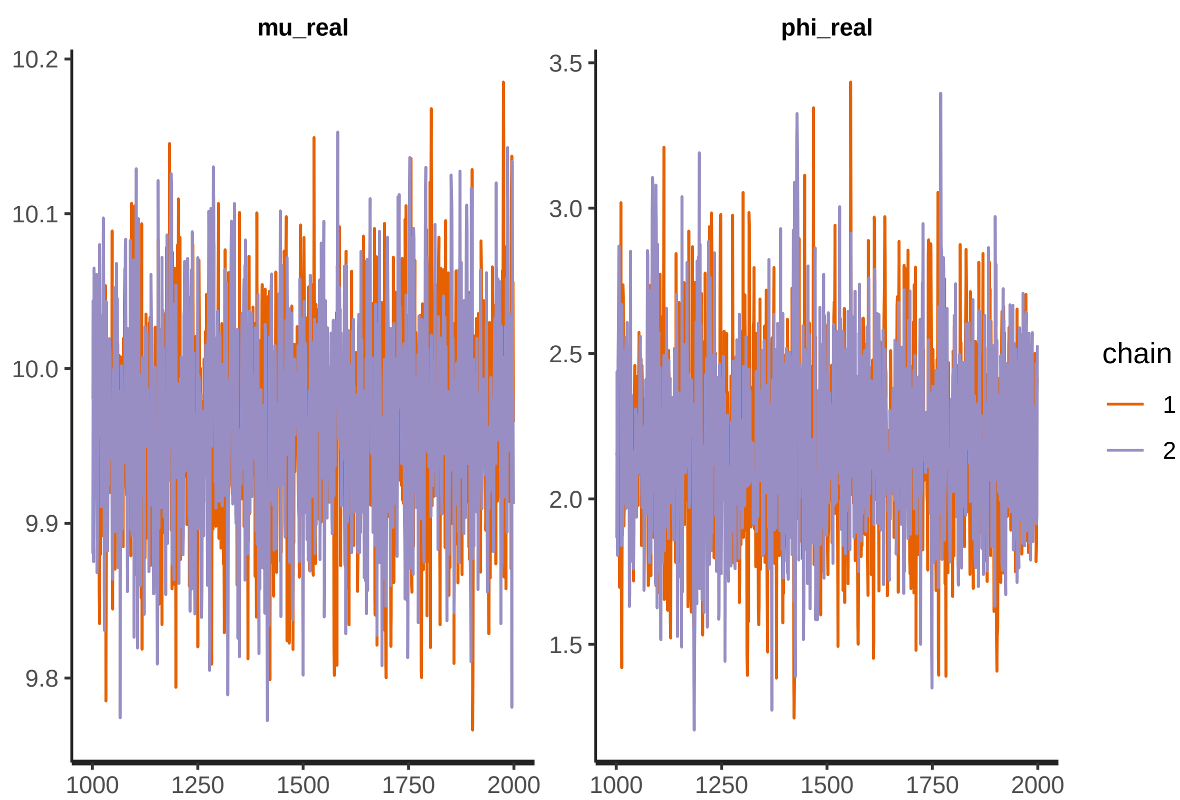
```r
output = stan(file = "StanNormal.stan", data = data, iter = iter,
              warmup = warmup, chains = chains, pars = pars,
              init = init, verbose = FALSE)
```

Explorando os resultados

```r
# Sumario global do objeto stan fit.
print(output, pars = c("mu_real","phi_real"))
```

```
## Inference for Stan model: anon_model.
## 2 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=2000.
##
##          mean se_mean   sd 2.5%  25%  50%   75% 97.5% n_eff Rhat
## mu_real  9.97    0.00 0.07 9.83 9.92 9.97 10.01 10.10  1492    1
## phi_real 2.20    0.01 0.32 1.60 1.97 2.18  2.42  2.88  1434    1
##
## Samples were drawn using NUTS(diag_e) at Mon Jun 16 12:47:14 2025.
## 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).
```

```r
rstan::traceplot(output, pars = c("mu_real","phi_real"))
```




```r
samp = extract(output)
```

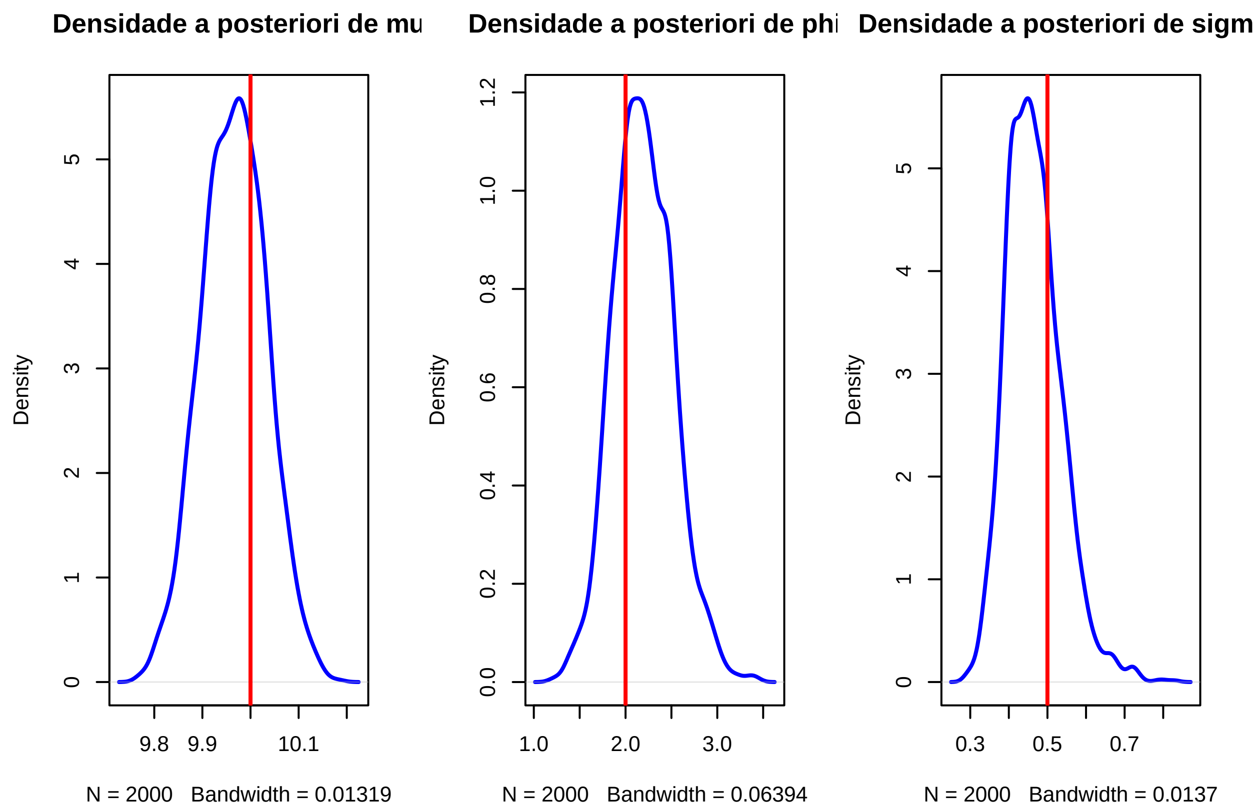
```r
par(mfrow = c(1,3))

{plot(density(samp$mu_real),cex.lab =1, cex.axis= 1,lwd = 2,
      main = "Densidade a posteriori de mu",col = "blue")
  abline(v = mu_real,lwd = 2,col ="red")}

{plot(density(samp$phi_real),cex.lab =1, cex.axis= 1, lwd = 2,
      main= "Densidade a posteriori de phi",col = "blue")
```

```r
  abline(v = phi_real, lwd = 2,col ="red" )}

{plot(density(1/samp$phi_real),cex.lab =1, cex.axis= 1, lwd = 2,
      main= "Densidade a posteriori de sigma2",col = "blue")
  abline(v = 1/phi_real, lwd = 2,col ="red" )}
```



O gráfico acima mostra a distribuição estimada para os parâmetros μ,ϕ e $\sigma 2$. Para os três parâmetros, o valor real, demarcado pela linha vertical vermelha, está contido na distribuição. Importante dizer que para a média, os valores reais estão próximo da média das respectivas distribuições, indicando que o valor estimado pontual está bem aproximado do valor real.

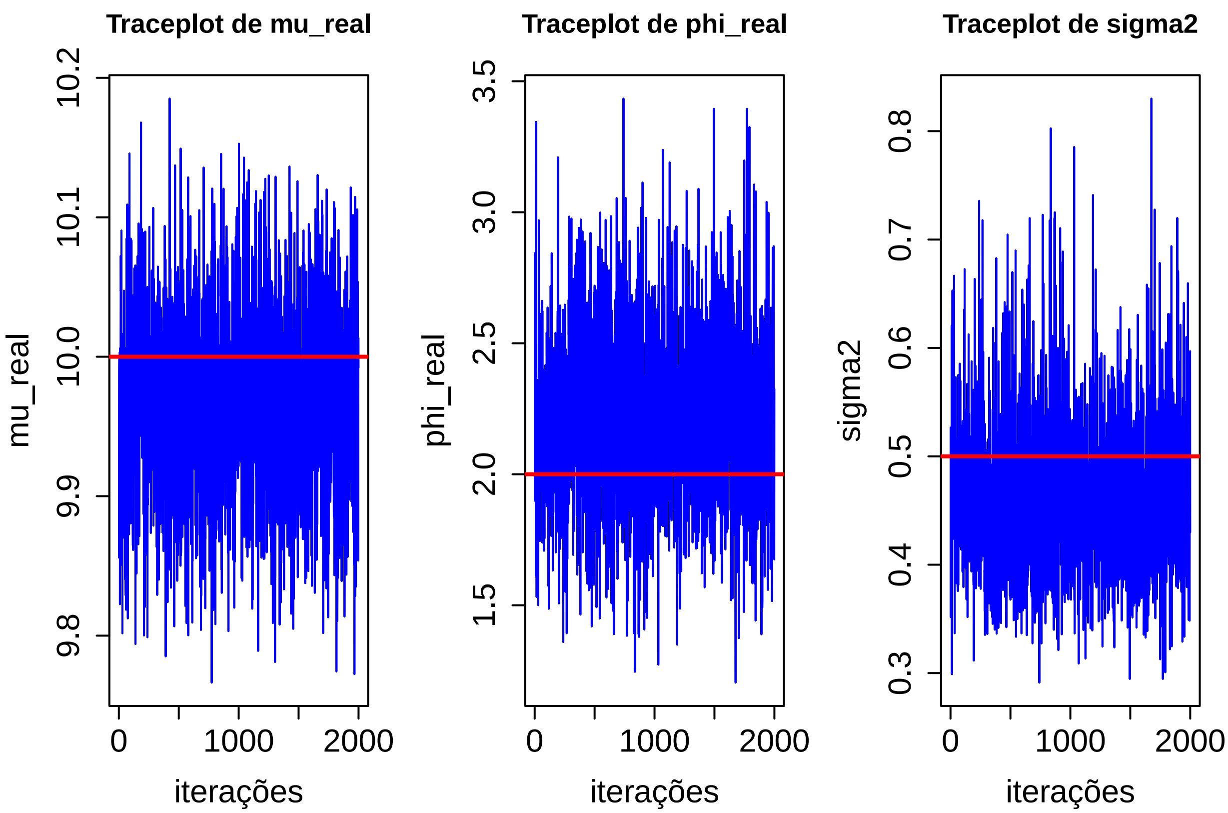
```r
par(mfrow =c(1,3))

{plot(samp$mu_real, type ="l", cex.lab= 1.5, cex.axis = 1.5,
      xlab ="iterações",ylab = "mu_real",
      main ="Traceplot de mu_real", col= "blue")
  abline(h= mu_real,lwd = 2,col ="red" )}

{plot(samp$phi_real, type = "l", cex.lab =1.5, cex.axis = 1.5,
      xlab ="iterações",ylab = "phi_real",
      main ="Traceplot de phi_real", col = "blue")
  abline(h= phi_real, lwd = 2,col ="red" )}

{plot(1/samp$phi_real, type = "l", cex.lab =1.5, cex.axis = 1.5,
      xlab ="iterações",ylab = "sigma2",
      main ="Traceplot de sigma2", col = "blue")
```

```
  abline(h= 1/phi_real, lwd = 2,col ="red" )}
```




```
aux = cbind(samp$mu_real, samp$phi_real, 1/samp$phi_real)
me = apply(aux, 2, mean) # media
md = apply(aux, 2, median) # mediana
sd = apply(aux, 2, sd) # desvio padrao
parametro = c("mu_real", "phi_real", "sigma2")
`valor real` = c(mu_real, phi_real, 1/phi_real)
aux = as.mcmc(aux)
hpd = HPDinterval(aux)
tab = bind_cols(parametro, `valor real`, me, md, sd, hpd[,"lower"],hpd[,"upper"])
```

```
## New names:
## * `` -> `...1`
## * `` -> `...2`
## * `` -> `...3`
## * `` -> `...4`
## * `` -> `...5`
## * `` -> `...6`
## * `` -> `...7`
```

```
colnames(tab) = c("parametro","valor real", "mean", "median","s.d.", "HPD_inf", "HPD_sup")
knitr::kable(tab, digits = 4, caption = "Tabela de Resumo da Inferência Bayesiana")
```

Table 1: Tabela de Resumo da Inferência Bayesiana

parametro	valor real	mean	median	s.d.	HPD_inf	HPD_sup
mu_real	10.0	9.9658	9.9665	0.0670	9.8332	10.0938
phi_real	2.0	2.1952	2.1790	0.3249	1.6093	2.8777
sigma2	0.5	0.4659	0.4589	0.0714	0.3352	0.5988

Com base nos resultados, o modelo apresentou um desempenho satisfatório na estimação dos parâmetros. Para o primeiro parâmetro, o ajuste foi bem-sucedido, pois o valor real (10.0) está contido no intervalo de credibilidade de 95% HPD. Para o segundo parâmetro, o ajuste foi bem-sucedido também, pois o valor real (2.0) está contido no intervalo de credibilidade de 95% HPD. O mesmo vale para o parâmetro $\sigma 2$.