

CE-227: Inferência Bayesiana – 1ª Avaliação Semanal (28/02/2014)

GRR: _____ **Nome:** _____ **Turma:** _____

1. Seja x_1, \dots, x_n uma a.a. de uma distribuição normal de média μ conhecida e variância σ^2 desconhecida. Considere a priori $[\sigma^2] \propto 1/\sigma^2$.

- (a) Obtenha a expressão da verossimilhança do modelo.
- (b) Obtenha a expressão da distribuição a posteriori.
- (c) É possível identificar a posteriori do modelo como alguma distribuição conhecida?
- (d) Considere que foi tomada a amostra dada pelos valores a seguir, e que $\mu = 10$. Obtenha a expressão da posteriori.

12,1 ; 8,7 ; 11,3 ; 9,2 ; 10,5 ; 9,7 ; 11,6

- (e) Indique (com comandos do R ou de alguma outra forma) como resumos pontuais e intervalares desta distribuição a posteriori poderiam ser obtidos para fins de inferência

Probabilidades

$$\mu_X = E[X] = \sum_i x_i P(X = x_i)$$

$$\sigma_X^2 = Var[X] = \sum_i (x_i - \mu_x)^2 P(X = x_i)$$

$$\mu_X = E[X] = \int x f_X(x) dx$$

$$\sigma_X^2 = Var[X] = \int (x - \mu_x)^2 f_X(x)$$

$$Y = g(X) \Rightarrow$$

$$f_Y(y) = f_X(g^{-1}(y)) \left| \frac{d}{dy} g^{-1}(y) \right|$$

Distribuições de Probabilidade

Distribuição	Probab./Densidade	Domínio	E[X]	Var[X]
$X \sim U(k)$	$\frac{1}{k}$	$x = 1, 2, \dots, k$	$\frac{\min(X)+\max(x)}{2}$	$\sqrt{\frac{(\max(X)-\min(X)+1)^2-1}{12}}$
$X \sim B(n, p)$	$\binom{n}{x} p^x (1-p)^{n-x}$	$x = 0, 1, 2, \dots, n$	np	$np(1-p)$
$X \sim HG(N, K, n)$	$\frac{\binom{K}{x} \binom{N-K}{n-x}}{\binom{N}{n}}$	$x = 0, 1, \dots, \min(K, n)$	np	$np(1-p) \frac{N-n}{N-1}$
$X \sim P(\lambda)$	$\frac{e^{-\lambda} \lambda^x}{x!}$	$x = 0, 1, 2, \dots$	λ	λ
$X \sim G(p)$	$(1-p)^x p$	$x = 0, 1, 2, \dots$	$\frac{1-p}{p}$	$\frac{(1-p)}{p^2}$
$X \sim BN(r, p)$	$\binom{x+r-1}{r-1} (1-p)^x p^r$	$x = 0, 1, 2, \dots$	$\frac{r(1-p)}{p}$	$\frac{r(1-p)}{p^2}$
$X \sim U[a, b]$	$\frac{1}{b-a}$	$a \leq x \leq b$	$\frac{a+b}{2}$	$\frac{(b-a)^2}{12}$
$X \sim N(\mu, \sigma^2)$	$\frac{1}{\sqrt{2\pi\sigma^2}} \exp\{-\frac{1}{2\sigma^2}(x-\mu)^2\}$	$x \in (-\infty, \infty)$	μ	σ^2
$X \sim \text{HalfN}(0, \sigma^2)$	$\frac{\sqrt{2}}{\sigma\sqrt{\pi}} \exp\{-\frac{x^2}{2\sigma^2}\}$	$x \in (0, \infty)$	0	σ^2
$X \sim \text{LN}(\mu, \sigma^2)$	$\frac{1}{x\sqrt{2\pi\sigma^2}} \exp\{-\frac{1}{2\sigma^2}(\ln(x)-\mu)^2\}$	$x \in (-\infty, \infty)$	$\exp\{\mu + \sigma^2/2\}$	$\exp\{2\mu + \sigma^2\}(\exp\{\sigma^2\} - 1)$
$X \sim \text{Exp}(\lambda)$	$\lambda \exp(-\lambda x)$	$x \geq 0$	$\frac{1}{\lambda}$	$\frac{1}{\lambda^2}$
$X \sim \text{Erlang}(\lambda, r)$	$\lambda^r x^{r-1} \exp(-\lambda x) / (r-1)!$	$x \geq 0 ; r = 1, 2, \dots$	$\frac{r}{\lambda}$	$\frac{r}{\lambda^2}$
$X \sim \text{Ga}(\alpha, \beta)$	$\frac{1}{\Gamma(\alpha)\beta^\alpha} x^{\alpha-1} \exp\{-x/\beta\}$	$x \geq 0$	$\alpha\beta$	$\alpha\beta^2$
$X \sim \text{IGa}(\alpha, \beta)$	$\frac{1}{\Gamma(\alpha)\beta^\alpha} x^{-\alpha-1} \exp\{-1/(x\beta)\}$	$x \geq 0$	$1/(\beta(\alpha-1))$	
$X \sim \text{Weibull}(\alpha, \beta)$	$\frac{\alpha}{\beta} (x/\beta)^{\alpha-1} \exp\{-(x/\beta)^\alpha\}$	$x \geq 0$	$\beta \Gamma(1 + \frac{1}{\alpha})$	$\beta^2 [\Gamma(1 + \frac{2}{\alpha}) - \Gamma^2(1 + \frac{1}{\alpha})]$
$X \sim \text{Beta}(\alpha, \beta)$	$\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1} (1-x)^{\beta-1}$	$0 \leq x \leq 1$	$\frac{\alpha}{\alpha+\beta}$	$\frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$

$$Z = (X - \mu)/\sigma$$

$$\Gamma(\alpha) = \int_0^\infty x^{\alpha-1} \exp\{-x\} dx$$

$$\Gamma(\alpha) = (\alpha - 1)!$$

$$\Gamma(\alpha + 1) = \alpha \Gamma(\alpha)$$

Solução:

1. > y <- c(12.1,8.7,11.3,9.2,10.5,9.7,11.6)

(a) As funções de verossimilhança ($L(\theta)$), log-verossimilhança ($l(\theta)$), escore ($U(\theta)$) e hessiana ($H(\theta)$) são dadas por:

$$\begin{aligned}L(\sigma^2; y) &= \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(y_i - \mu)^2\right\} \\&= (2\pi)^{-n/2}(\sigma^2)^{-n/2} \exp\left\{-\frac{1}{2\sigma^2} \sum_{i=1}^n (y_i - \mu)^2\right\} \\l(\sigma^2; y) &= \log\{L(\sigma^2; y)\} = (-n/2)[\log(2\pi) + \log(\sigma^2) + \frac{\sum_{i=1}^n (y_i - \mu)^2}{n}(\sigma^2)^{-1}] \\U(\sigma^2) &= \frac{dl(\sigma^2; y)}{d\sigma^2} = (-n/2)[(\sigma^2)^{-1} - \frac{\sum_{i=1}^n (y_i - \mu)^2}{n}(\sigma^2)^{-2}] \\H(\sigma^2) &= \frac{d^2l(\sigma^2; y)}{d(\sigma^2)^2} = (-n/2)[-(\sigma^2)^{-2} + \frac{\sum_{i=1}^n (y_i - \mu)^2}{n}(\sigma^2)^{-3}]\end{aligned}$$

O estimador máxima verossimilhança $\hat{\sigma}^2$ obtido fazendo $U(\sigma^2) = 0$, o valor da informação observada ao redor de $\hat{\sigma}^2$, $I_o(\hat{\sigma}^2) = -H(\hat{\sigma}^2)$ e a variância do estimador $\text{Var}(\hat{\sigma}^2) = I^{-1}(\hat{\sigma}^2)$ são obtidos analiticamente neste caso, com expressões

$$\begin{aligned}\hat{\sigma}^2 &= \frac{\sum_{i=1}^n (y_i - \mu)^2}{n}, \\I_o(\hat{\sigma}^2) &= \frac{n}{2(\hat{\sigma}^2)^2}, \\ \text{Var}(\hat{\sigma}^2) &= \frac{2(\hat{\sigma}^2)^2}{n},\end{aligned}$$

que para os dados disponíveis produzem os seguintes resultados.

```
> n <- length(y)
> (MLE <- sum((y-10)^2)/n)
[1] 1.619
> H.MLE <- - n/(2*MLE^2) ; Io.MLE <- -H.MLE ; Var.MLE = 1/Io.MLE ; se.MLE <- sqrt(Var.MLE)
> c(H.MLE, Io.MLE, Var.MLE, se.MLE)
[1] -1.3360 1.3360 0.7485 0.8652
```

As funções $l(\sigma^2)$, $U(\sigma^2)$ e $I_o(\sigma^2)$ podem ser definidas como se seguem¹.

```
> llfun <- function(par, mu, dados, log=TRUE){
+   res <- -(length(dados)/2)*log(par) - sum((dados-mu)^2)/(2*par)
+   if(!log) res <- exp(res)
+   return(res)
+ }
> Ufun <- function(par, mu, dados){
+   return( -(length(dados)/2) * ((1/par) - sum((dados-mu)^2)/(n*par^2)))
+ }
> Hfun <- function(par, mu, dados){
+   return( -(length(dados)/2) * (-1/par^2) + 2*sum((dados-mu)^2)/(n*par^3))
+ }
```

Verificando os valores obtidos.

```
> round(c(llfun(MLE, dados=y, mu=10), Ufun(MLE, dados=y, mu=10), Hfun(MLE, dados=y, mu=10)), dig=5)
[1] -5.185 0.000 -1.336
```

A aproximação quadrática da função de verossimilhança é dada por

$$\tilde{l}(\sigma^2; y) = l(\hat{\sigma}^2) + (1/2)(\sigma^2 - \hat{\sigma}^2)^2 H(\hat{\sigma}^2)$$

e definida no código a seguir.

```
> llapprox <- function(par, dados, mu){
+   est <- sum((dados - mu)^2)/length(dados)
+   llfun(est, dados=dados, mu=mu) + 0.5 * (par - est)^2 * Hfun(est, dados=dados, mu=mu)
+ }
```

```

> curve(llfun(x, mu=10, dados=y, log=TRUE), from=0.55, to=8.5,
+       xlab=expression(sigma^2), ylab=expression(l(sigma^2)))
> abline(v=c(MLE), lty=3)
> #
> curve(llapprox(x, dados=y, mu=10), from=0.55, to=8.5, add=T, col=2, lt=2)
> legend("topright", c("verossimilhança", "aproximação quadrática"), lty=1:2, col=1:2)

```

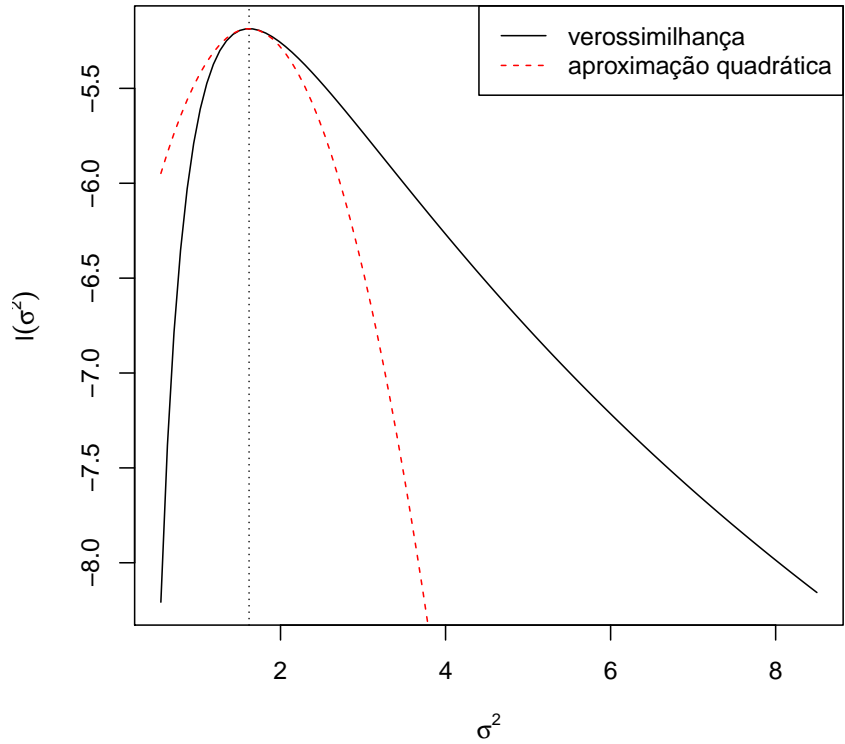


Figura 1: Função de verossimilhança para σ^2 e a respectiva aproximação quadrática

Os gráficos da função de log-verossimilhança e de sua aproximação quadrática são mostrados a seguir.

Embora as expressões analíticas para as quantidades de interesse mencionadas acima sejam disponíveis neste exemplo, vamos ilustrar como estimativas podem ser obtidas por métodos numéricos, que devem ser utilizados quando as expressões analíticas não podem ser obtidas.

Um código baseado na maximização da função de verossimilhança é apresentado a seguir e o hessiano numérico é também extraído.

```
> (MLEnum <- optimize(llfun, interval=c(0, 10), mu=10, dados=y,
+                    log=TRUE, maximum=TRUE))
```

```
$maximum
[1] 1.619
```

```
$objective
[1] -5.185
```

```
> optimHess(MLEnum$maximum, llfun, dados=y, mu=10)
```

```
      [,1]
[1,] -1.336
```

```
> -1/optimHess(MLEnum$maximum, llfun, dados=y, mu=10)
```

```
      [,1]
[1,] 0.7485
```

```
> sqrt(-1/optimHess(MLEnum$maximum, llfun, dados=y, mu=10))
```

```
      [,1]
[1,] 0.8652
```

Outra opção é encontrar a raiz da função escore.

```
> uniroot(Ufun, interval=c(0.1, 20), dados=y, mu=10)$root
```

```
[1] 1.619
```

Outros métodos, como por exemplo o algoritmo de Newton-Raphson, podem também ser utilizados. O algoritmo de Newton-Raphson exige que sejam fornecidas as funções escore e hessiana. Além disto, como o algoritmo não impõe restrição ao parâmetro, utilizamos neste caso uma transformação logarítmica $\phi = \log(\sigma^2)$ internamente na função, uma vez que $\sigma^2 > 0$. As funções $U(\cdot)$ e $H(\cdot)$ são redefinidas sob esta reparametrização e omite-se as constantes $(-n/2)$ que se cancelam na divisão $U(\cdot)/H(\cdot)$.

```
> NR <- function(ini, dados, mu, maxit = 100, tol = 1e-8, trace=FALSE){
+   l.ini <- log(ini)
+   it <- 0; delta <- 1
+   SQn <- sum((dados-mu)^2)/length(dados)
+   U.lpar <- function(phi) (1 - SQn * exp(-phi))
+   H.lpar <- function(phi) SQn * exp(-phi)
+   while(it < maxit & abs(delta) > tol){
+     l.par <- l.ini - U.lpar(l.ini)/H.lpar(l.ini)
+     delta <- l.par - l.ini
+     l.ini <- l.par; it <- it+1
+     if(trace) print(c(iteração = it, par = exp(l.par)))
+   }
+   return(structure(c(logpar=l.par, par=exp(l.par)),
+                    iteracoes = it, hessian.logpar=H.lpar(l.par)))
+ }
> NR(2, dados=y, mu=10, trace=T)
```

```
iteração    par
      1.00    1.58
iteração    par
      2.000   1.618
iteração    par
      3.000   1.619
iteração    par
      4.000   1.619
iteração    par
      5.000   1.619
```

¹para maior eficiência evitando cálculos repetitivos em chamadas das funções, elas poderiam receber diretamente o valor da soma de quadrados como argumento no lugar dos dados.

```
logpar    par
0.4815 1.6186
attr(,"iteracoes")
[1] 5
attr(,"hessian.logpar")
[1] 1
```

(b) A expressão da distribuição a posteriori é obtida da forma:

$$\begin{aligned} [\sigma^2|y] &\propto [\sigma^2] \cdot L(\sigma^2; y) \\ &= (\sigma^2)^{-1} \cdot (2\pi)^{-n/2} (\sigma^2)^{-n/2} \exp \left\{ -\frac{1}{2\sigma^2} \sum_{i=1}^n (y_i - \mu)^2 \right\} \\ &\propto (\sigma^2)^{-(n/2)-1} \exp \left\{ -\frac{\sum_{i=1}^n (y_i - \mu)^2}{2\sigma^2} \right\} \end{aligned}$$

(c) E a expressão anterior, como uma função de σ^2 , corresponde ao núcleo de uma densidade “gama inversa” de parâmetros:

$$\alpha = \frac{n}{2} \quad \text{e} \quad \beta = \frac{2}{\sum_{i=1}^n (y_i - \mu)^2}$$

(d) Para o conjunto de dados temos que

$$\alpha = \frac{n}{2} = \frac{7}{2} = 3,5 \quad \text{e} \quad \beta = \frac{2}{\sum_{i=1}^n (y_i - \mu)^2} = \frac{2}{11,33} = 0,1765.$$

```
> a.post <- length(y)/2
> b.post <- 2/sum((y-10)^2)
> c(a.post, b.post)
[1] 3.5000 0.1765
```

Definindo uma densidade gama-inversa.

```
> dinvgamma <- function(x, a, b, log=FALSE){
+   res <- ifelse(x > 0,
+               - a * log(b) - log(gamma(a)) - (a+1)*log(x) - 1/(b*x), -Inf)
+   if(!log) res <- exp(res)
+   return(res)
+ }
```

Se $X \sim \text{Ga}(\alpha, \beta)$ então $Y = 1/X \sim \text{InvGa}(\alpha, \beta)$. Portanto, para simular de uma distribuição gama inversa basta tomar o inverso de valores simulados de uma distribuição gama com os os mesmos parâmetros.

(e) i. Resumos pontuais

A. Média da posteriori. Neste caso se tem a expressão analítica e que portanto deve ser utilizada. Entretanto, como em muitos casos pode não ser disponível, ilustra-se também a obtenção por integração numérica e por simulação.

- Expressão analítica

$$E[\sigma^2|y] = \frac{1}{\beta(\alpha - 1)} = 2.27$$

```
> (e.post <- 1/(b.post*(a.post-1)))
[1] 2.266
```

- Integração numérica (a partir da definição de esperança de uma v.a.).

$$E[\sigma^2|y] = \int_0^{\infty} \sigma^2 f(\sigma^2|y) d\sigma^2$$

```
> Epost <- function(par, ...) par * dinvgamma(par, ..., log=FALSE)
> integrate(Epost, lower=0, upper=50, a=a.post, b=b.post)
2.263 with absolute error < 6.6e-06
```

- Simulação.

```
> mean(sim)
[1] 2.28
```

B. Moda da posteriori. Assim como no caso da média a expressão analítica da moda é conhecida mas ilustra-se também a obtenção por otimização numérica e por simulação.

```
> curve(dinvgamma(x, a=a.post, b=b.post), from=0, to=10, n=501,
+       xlab=expression(sigma^2), ylab=expression(group("[",paste(sigma^2,"|",y),"]")))

```

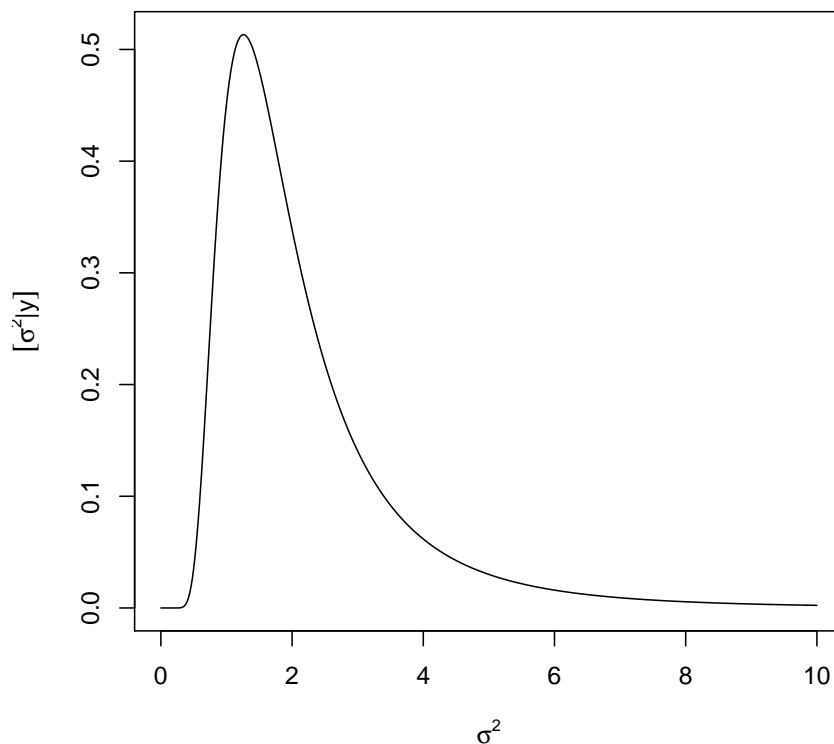


Figura 2: Densidade (gama-inversa) de σ^2 para o exemplo.

- Expressão analítica.

$$E[\sigma^2|y] = \frac{1}{\beta(\alpha + 1)} = 1.26$$

```
> (mo.post <- 1/(b.post*(a.post+1)))

```

```
[1] 1.259
```

- Otimização numérica, maximizando a densidade (ou, equivalentemente, a log-densidade, o que em geral é mais estável numericamente).

```
> optimize(dinvgamma, lower=0, upper=50, a=a.post, b=b.post, log=FALSE, maximum=TRUE)

```

```
$maximum
```

```
[1] 1.259
```

```
$objective
```

```
[1] 0.5133
```

```
> optimize(dinvgamma, lower=0, upper=50, a=a.post, b=b.post, log=TRUE, maximum=TRUE)

```

```
$maximum
```

```
[1] 1.259
```

```
$objective
```

```
[1] -0.6669
```

- Simulação. Utiliza-se aqui um algoritmo simples tomando-se o ponto de máximo de uma suavização da densidade.²

```
> sim.den <- density(sim, n=1024)

```

```
> sim.den$x[which.max(sim.den$y)]

```

```
[1] 1.327
```

C. Mediana da Posteriori

- Expressão analítica: não disponível.

²veja outro algoritmo mais adiante com o uso da função `hdrcd::hdr()`.

```

> set.seed(123)
> sim <- 1/rgamma(10000, shape=a.post, scale=b.post)
> hist(sim, prob=TRUE, ylim=c(0,0.5), br=c(seq(0,10, by=0.5), seq(10,60,by=2)),
+       xlab=expression(sigma^2), ylab=expression(group("[",paste(sigma^2,"|",y),"]")),
+       main="", xlim=c(0, 10))
> lines(density(sim))
> curve(dinvgamma(x, a=a.post, b=b.post), from=0.01, to=30, add=T, col=3, n=501)

```

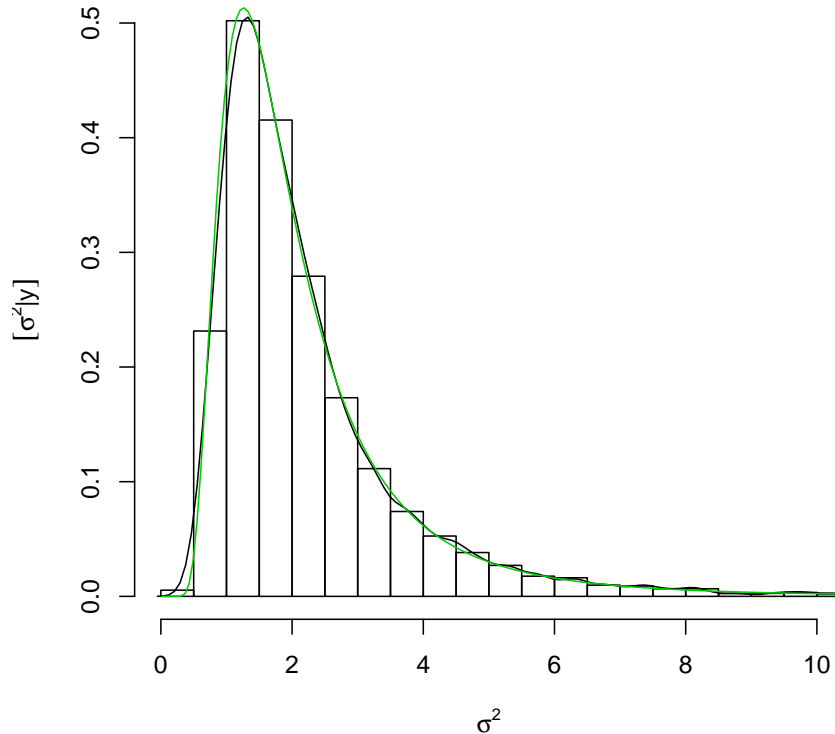


Figura 3: Histograma e densidade estimada das amostras da posteriori comparados com a expressão analítica da posteriori.

- Otimização numérica.

Pode-se definir uma função que retorne quantis da posteriori gamma inversa.

```

> qinvgamma <- Vectorize(function(p, a, b, ...){
+   q.f <- function(x, a, b)
+     integrate(dinvgamma, lower=0, upper=x, a=a, b=b)$value - p
+   uniroot(q.f, a=a, b=b, ...)$root
+ })
> (md.post <- qinvgamma(0.5, a=a.post, b=b.post, interval=c(0,50)))
[1] 1.785

```

Entretanto, para o cálculo de quantis neste exemplo, a definição da função anterior é desnecessária, pois, pode-se simplesmente tomar inversos dos quantis da distribuição gama.

```

> 1/qgamma(1-0.5, shape=a.post, scale=b.post)
[1] 1.785

```

Portanto desejando-se ter uma função específica para a gama-inversa, basta definir:

```

> qinvgamma <- function(p, a, b, ...)
+   1/qgamma(1-p, shape = a, scale=b, ...)

```

- Simulação

```

> median(sim)
[1] 1.794

```

ii. Resumos intervalares (usando $1 - \alpha = 0,95$)

i. Intervalo de credibilidade – quantis

A. Por otimização numérica

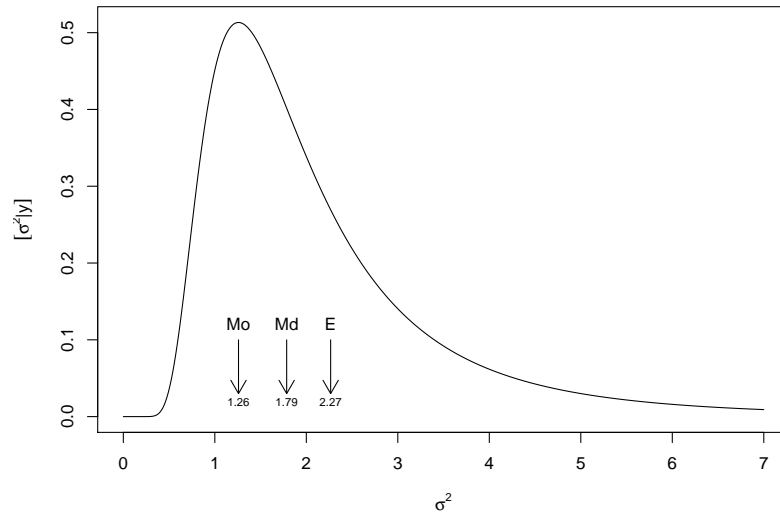


Figura 4: Média, mediana e moda na distribuição a posteriori do exemplo.

```
> (ICq <- qinvgamma(c(0.025, 0.975), a=a.post, b=b.post))
[1] 0.7076 6.7047
```

B. Por simulação

```
> quantile(sim, prob=c(0.025, 0.975))
 2.5% 97.5%
0.7075 6.8346
```

ii. Intervalo de credibilidade – HPD³

A. Otimização numérica. A função a seguir é apenas um exemplo ilustrativo e pode ser escrita de outras formas (o que fica como exercício). Para rotinas mais eficientes vejam as fornecidas por pacotes especializados.

```
> # Função para obter o HPD de uma função (f.d.p.) *unimodal*
> # fç nao totalmente geral devido a valor superior fixado no argumento "interval"
> hpdinvgamma <- function(prob, a, b){
+   require(rootSolve)
+   moda <- 1/(b*(a+1))
+   dmax <- dinvgamma(moda, a=a, b=b)
+   #
+   f.int <- function(corte, a, b, prob){
+     f.corte <- function(x) dinvgamma(x, a=a, b=b) - (dmax - corte)
+     int <- uniroot.all(f.corte, interval=c(0,100))
+     p1 <- integrate(dinvgamma, lower=0, upper=int[1], a=a, b=b)$value
+     p2 <- integrate(dinvgamma, lower=0, upper=int[2], a=a, b=b)$value
+     return(((p2-p1) - prob)^2)
+   }
+   corte.int <- optimize(f.int, interval=c(0, dmax), a=a, b=b, prob=prob)
+   f.corte <- function(x)
+     dinvgamma(x, a=a, b=b, log=FALSE) - (dmax - corte.int$minimum)
+   int <- uniroot.all(f.corte, interval=c(0,100))
+   return(int)
+ }
> (IChpd <- hpdinvgamma(0.95, a=a.post, b=b.post))
[1] 0.4761 5.2770
> integrate(dinvgamma, lower=IChpd[1], upper=IChpd[2], a=a.post, b=b.post)
0.95 with absolute error < 8e-08
```

B. HDR obtido a partir de simulações da posteriori.

```
> require(hdrcde)
> ## baseado nas amostras
> hdr(sim, prob=95)$hdr
```

³diversos pacotes do R fornecem algoritmos para cálculos de intervalos HPD. Em geral são fornecidas funções para obtenção de intervalos HPD a partir de um objeto como amostras (simulação) da posteriori, mas alguns pacotes fornecem também funções para cálculo a partir de posterioris na forma de função.

```

      [,1] [,2]
95% 0.3314 5.318
> ## baseado em uma aproximação discreta da densidade
> par.vals <- seq(0, 30, length=2000)
> d.vals <- dinvgamma(par.vals, a=a.post, b=b.post)
> hdr(den=list(x=par.vals, y=d.vals), prob=95)$hdr

```

```

      [,1] [,2]
95% 0.4828 5.119

```

Alguns exemplos de funções disponíveis em outros pacotes.

```

> require(TeachingDemos)
> emp.hpd(sim)
> hpd(qinvgamma, a=a.post, b=b.post)
> detach(package:TeachingDemos)
> ##
> require(BEST)
> hdi(sim, credMass=0.95)
> hdi(qinvgamma, credMass=0.95, a=a.post, b=b.post)
> detach(package:BEST)
> ##
> require(LaplacesDemon)
> p.interval(sim)
> detach(package:LaplacesDemon)
> ##
> require(emdbook)
> tcredint("invgamma", list(a=a.post, b=b.post)) # problema aqui
> detach(package:emdbook)
> #
> require(coda)
> HPDinterval(as.mcmc(sim))
> detach(package:coda)

```

Pode-se verificar a propriedade de que o IC-HPD é mais curto do que o de quantis. A diferença torna-se mais acentuada quanto mais assimétrica for a distribuição.

```

> diff(ICq)
[1] 5.997
> diff(IChpd)
[1] 4.801

```

O gráfico da função com os dois tipos de intervalos mostra que eles são bem distintos neste caso.

Gráficos da priori, posteriori e verossimilhança.

Para incluir o gráfico da verossimilhança na mesma escala da priori e posteriori é necessário utilizar a função padronizada (de forma a integrar 1). Há duas formas que serão ilustradas a seguir: (i) reconhecendo o núcleo de alguma distribuição conhecida (quando possível), (ii) integrando explicitamente a verossimilhança (analítica ou numericamente).

Neste exemplo a opção (i) é possível pois, para a verossimilhança, tem-se que:

$$L(\sigma^2|y) \propto IG(a = (n/2) - 1, b = 2 / \sum_{i=1}^n (y_i - \mu)^2).$$

```

> post <- function(dados, mu, plot=TRUE, from, to){
+   a <- length(dados)/2
+   SQ <- sum((dados-mu)^2)
+   b <- 2/SQ
+   MLE <- SQ/length(dados)
+   if(plot){
+     par.seq <- seq(from, to, length=201)
+     vero.seq <- dinvgamma(par.seq, a=a-1, b=b)
+     post.seq <- dinvgamma(par.seq, a=a, b=b)
+     max.seq <- max(c(max(vero.seq), max(post.seq)))
+     plot(par.seq, post.seq, type="l", ylim=c(0, 1.4*max.seq),
+          xlab=expression(sigma^2), ylab=expression(group("[", sigma^2, "]")))
+     lines(par.seq, vero.seq, lty=2, col=4)
+     lines(par.seq, 1/par.seq, lty=3, col=2)
+   }
+ }

```

```

> curve(dinvgamma(x, a=a.post, b=b.post), from=0, to=8, n=501,
+       xlab=expression(sigma^2), ylab=expression(group("[",paste(sigma^2,"|",y),"]")))
> segments(ICq, 0, ICq, dinvgamma(ICq, a=a.post, b=b.post), col=2, lty=2)
> segments(IChpd, 0, IChpd, dinvgamma(IChpd, a=a.post, b=b.post))
> legend("topright", c("HPD","quantis"), col=c(1,2),lty=c(1,2))

```

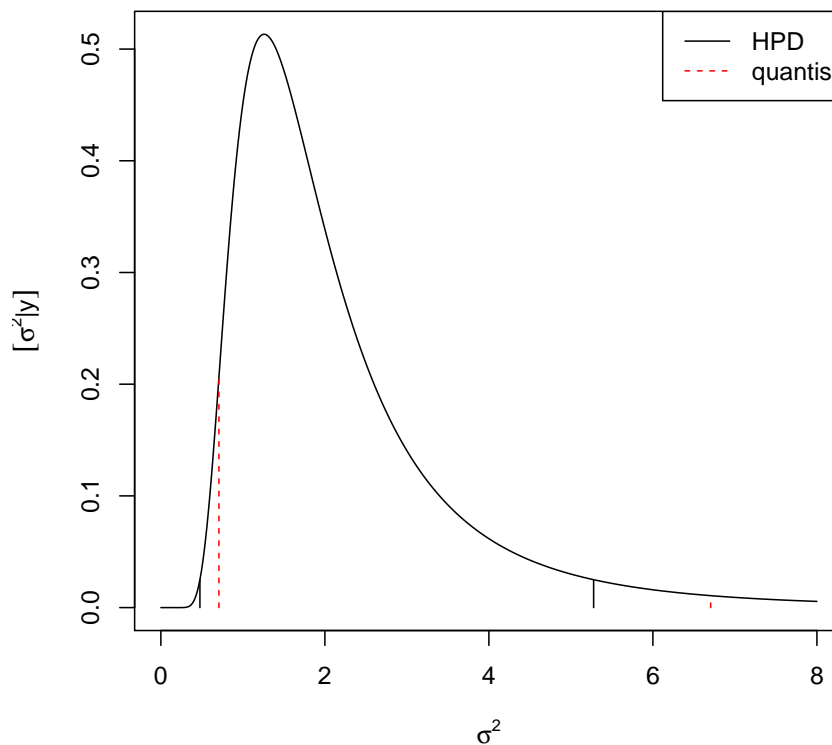


Figura 5: Comparação de intervalos de quantis e HDR.

```

+       legend("topright", c("priori","verossimilhança","posteriori"),
+             lty=c(3,2,1), col=c(2,4,1))
+   }
+   return(c(a=a, b=b, MLE=MLE, media=1/(b*(a-1)), moda=1/(b*(a+1))))
+ }

```

Para (ii) a verossimilhança integrada (numericamente) pode ser obtida da forma a seguir, que pode ser usada mesmo que a expressão da verossimilhança não tenha uma forma proporcional à de uma distribuição de probabilidades conhecida.

```

> llfun.int <- function(par, ...){
+   C <- integrate(llfun, low=0, up=50, ..., log=FALSE)$value
+   res <- llfun(par, ..., log=FALSE)/C
+   attr(res, "C") <- C
+   return(res)
+ }
> integrate(llfun.int, mu=10, dados=y, lower=0, upper=50)

```

1 with absolute error < 2.9e-06

Verificando os cálculos dos valores da densidade e da constante normalizadora temos:

O gráfico a seguir mostra que (i) e (ii) são equivalentes.

Aproximação normal da posteriori Sob certas condições, a posteriori pode ser aproximada por uma distribuição normal com média igual a moda da posteriori e variância (inversamente) relacionada ao Hessiano. Mais especificamente, obtém-se a log-densidade da posteriori e dela se obtém o máximo (moda) e o valor do hessiano avaliado neste ponto de máximo. Em certos casos particulares estes valores podem ser obtidos analiticamente, porém, de forma mais geral são obtidos numericamente. A

```
> post(dados=y, mu=10, from=0, to=10)
```

```
      a      b      MLE  media  moda
3.5000 0.1765 1.6186 2.2660 1.2589
```

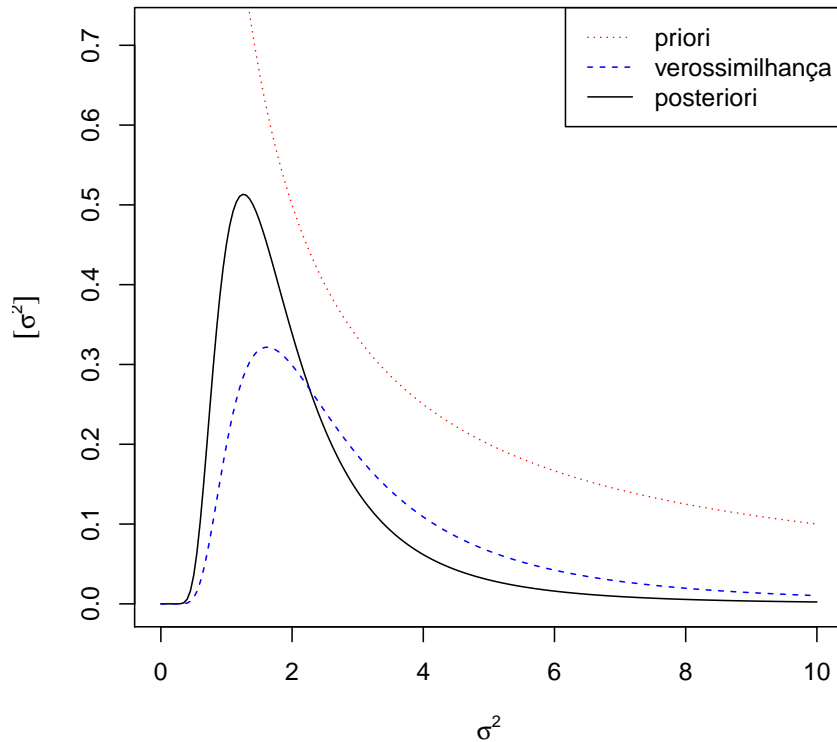


Figura 6: Priori, verossimilhança e posteriori para o exemplo.

aproximação é boa quando (i) o parâmetro é interior ao espaço paramétrico, para valores do parâmetro na proximidade da moda e para grandes tamanhos de amostra.

$$\begin{aligned}
 [\sigma^2|y] &\propto (\sigma^2)^{-(n/2)-1} \exp\left\{-\frac{\sum_{i=1}^n (y_i - \mu)^2}{2\sigma^2}\right\} \\
 \log[\sigma^2|y] &\propto -(n/2) - 1 \log(\sigma^2) - \frac{\sum_{i=1}^n (y_i - \mu)^2}{2} (\sigma^2)^{-1} \\
 \frac{d \log[\sigma^2|y]}{d\sigma^2} &\propto -(n/2) - 1 (\sigma^2)^{-1} + \frac{\sum_{i=1}^n (y_i - \mu)^2}{2} (\sigma^2)^{-2} \\
 H(\sigma^2) = \frac{d^2 \log[\sigma^2|y]}{d(\sigma^2)^2} &\propto -(-(n/2) - 1)(\sigma^2)^{-2} - \sum_{i=1}^n (y_i - \mu)^2 (\sigma^2)^{-3}
 \end{aligned}$$

E a variância para a aproximação normal é dada por $H(\hat{\sigma}^2) = -2.84$, em que $\hat{\sigma}^2 = 1/(\beta(\alpha + 1)) = 1.26$ é a moda da posteriori. Portanto a variância a aproximação normal é da posteriori:

$$[\sigma^2|y] \sim N(\hat{\sigma}^2, -H^{-1}(\hat{\sigma}^2)) \sim N(1.26, 0.352)$$

Neste caso, ambos, $\hat{\sigma}^2$ e $-H^{-1}(\hat{\sigma}^2)$ puderam ser obtidos analiticamente. Caso isto não fosse possível aproximação obtidas numericamente podem ser obtidas pela maximização da log-posteriori (ou minimização do negativo da log-posteriori).

```
> neglogpost <- function(par, y, mu)
+   (0.5*length(y)+1) * log(par) + 0.5 * sum((y-mu)^2)/par
> (APnum <- optim(1, neglogpost, y=y, mu=10, method="L-BFGS-B", lower=0, hessian=TRUE)[c("par", "hessian")])

$par
[1] 1.259

$hessian
[,1]
[1,] 2.839
```

```

> curve(llfun.int(x, mu=10, dados=y), from=0, to=10, type="b", pch=19,
+       xlab=expression(sigma^2), ylab="verossimilhança integrada")
> curve(dinvgamma(x, a=a.post-1, b=b.post), from=0, to=10, col=2, add=T)

```

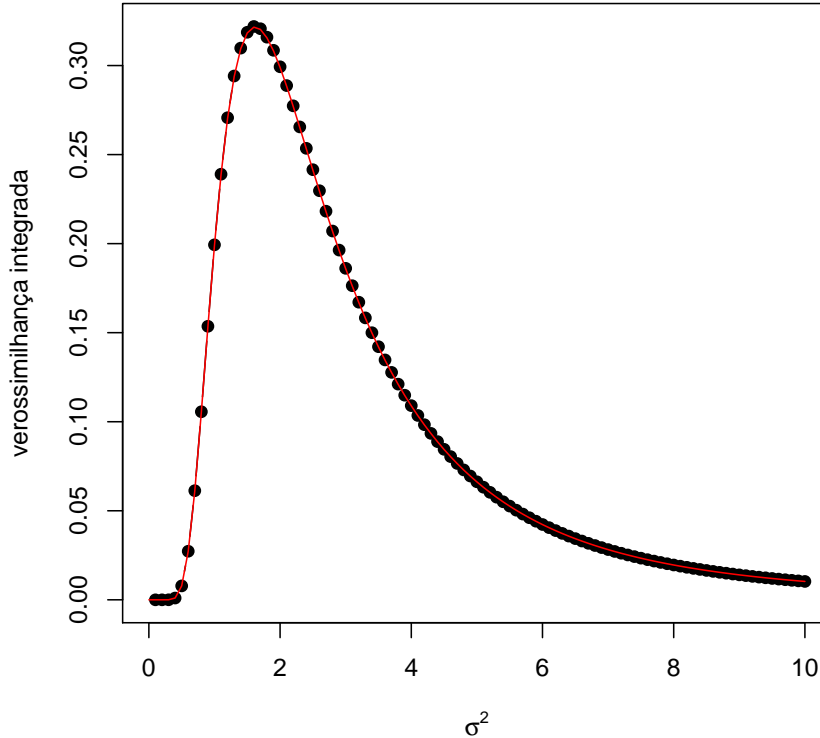


Figura 7: Verossimilhanças integradas obtida de duas formas diferentes: (i) por integração numérica (pontos) (ii) pela expressão da gama-inversa (linha).

Uma forma alternativa mais simples e direta de definir no R a log-posteriori (a uma constante), sem utilizar as derivações algébricas das expressões, é dada a seguir.

```

> neglogpost <- function(par, y, mu)
+   -(sum(dnorm(y, mean=mu, sd=sqrt(par), log=TRUE)) + log(1/par))

```

No caso de se usar estimativas numéricas, a densidade da normal fica como no comando a seguir. Não toma-se aqui o negativo do hessiano pois, no código anterior, foi minimizado o negativo da log-posteriori.

```

> dnorm(x, mean=APnum$par, sd=sqrt(1/APnum$hessian))

```

Amostragem (MCMC) Como já visto anteriormente a posteriori do exemplo possui forma analítica conhecida e amostras da posteriori podem ser obtidas diretamente, tomando-se os inversos de valores simulados da distribuição gamma correspondente. Entretanto, a título de ilustração, será exemplificado a seguir um algoritmo de *Cadeias de Markov via Monte Carlo* (MCMC) para obter amostras da posteriori.

O algoritmo de Metropolis-Hastings para amostrar de uma distribuição $\pi(\theta)$ segue os seguintes passos:

1. Toma-se um valor inicial θ_0 (arbitrário) para o parâmetro.
2. Escolhe-se uma função $q(\theta_*|\theta_{t-1})$ para sortear um novo valor θ_* “proposto” para o parâmetro.
3. Geram-se valores em uma “cadeia” (*suficientemente grande*) segundo as regras:

(a) para cada valor proposto calcula-se $\alpha = \min \left\{ 1, \frac{\pi(\theta_*)q(\theta_{t-1}|\theta_*)}{\pi(\theta_{t-1})q(\theta_*|\theta_{t-1})} \right\}$,

(b) aceita-se que a cadeia mova-se para o valor proposto com probabilidade α , ou seja, sorteia-se $U \sim U(0, 1)$ e

$$\theta_t = \begin{cases} \theta_* & \text{se } u < \alpha \\ \theta_{t-1} & \text{se } u \geq \alpha \end{cases} .$$

```

> curve(dinvgamma(x, a=a.post, b=b.post), from=0, to=10, n=501, ylim=c(0, 0.7),
+       xlab=expression(sigma^2), ylab=expression(group("[",paste(sigma^2,"|",y),"]")))
> curve(dnorm(x, m=mo.post, sd=sqrt(-1/H)), from=0, to=10, n=501, col=2, add=T, lty=2)
> legend("topright", c("posteriori", "aprox. normal"), lty=c(1,2), col=c(1,2))

```

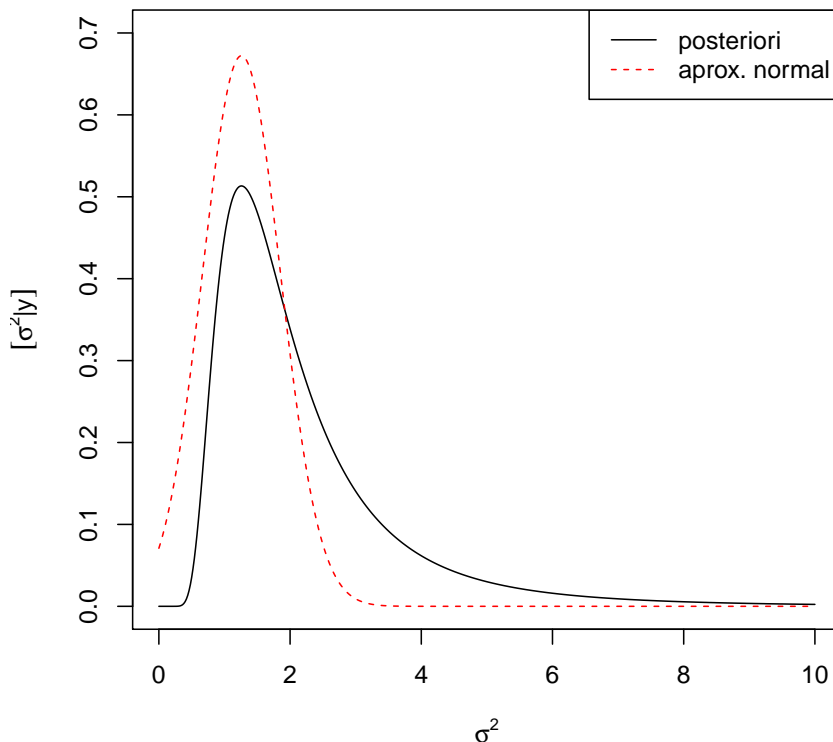


Figura 8: Densidade a posteriori e a respectiva aproximação normal.

Uma versão simplificada deste algoritmo (chamada de algoritmo de Metrópolis) é obtida quando $q(\cdot)$ é simétrica. Neste caso $q(\theta_*|\theta_{t-1}) = q(\theta_{t-1}|\theta_*)$ e o cálculo de α se reduz a

$$\alpha = \min \left\{ 1, \frac{\pi(\theta_*)}{\pi(\theta_{t-1})} \right\}.$$

A distribuição $\pi(\theta)$ é a distribuição objetivo, da qual se deseja simular, no caso, a posteriori $[\sigma^2|y]$. É importante notar que a função não precisa ser completamente especificada, ou seja, pode ser conhecida somente a uma constante de proporcionalidade uma vez que tal constante é cancelada no cálculo de α . Portanto, para simular de uma posteriori precisamos somente definir em $\pi(\cdot)$ a função obtida por

$$[\sigma^2|y] \propto [\sigma^2] \cdot L(\sigma^2; y).$$

Na definição da função usaremos agora os argumentos n e SQ para evitar recalculer quantidades fixas a cada iteração.⁴ Usa-se ainda o fato que $\frac{\pi(\theta_*)}{\pi(\theta_{t-1})} = \exp[\log(\pi(\theta_*) - \pi(\theta_{t-1}))]$ ⁵.

```

> dpost <- function(par, n, SQ)
+   ifelse(par > 0, exp(-(0.5*n+1) * log(par) - 0.5 * SQ/par), 0)
> logdpost <- function(par, n, SQ)
+   ifelse(par > 0, -(0.5*n+1) * log(par) - 0.5 * SQ/par, -Inf)

```

O algoritmo a seguir utiliza uma $q(\cdot)$ simétrica uniforme ao redor do valor atual do parâmetro tal que $\theta_* \sim U[\theta_{t-1} - a, \theta_{t-1} + a]$.

```

> MCMC <- function(N, init, n, SQ, a){
+   sim <- numeric(N)
+   sim[1] <- init
+   aceita <- 0
+   for(i in 2:N){

```

⁴Funções definidas anteriormente como a de verossimilhança também poderiam ser definidas desta forma para maior eficiência computacional.

⁵Outras simplificações poderiam ser obtidas para este exemplo considerando a razão para o cálculo de α .

```

+     nv <- runif(1, sim[i-1]-a, sim[i-1]+a)
+     alpha <- min(1, exp(logdpost(nv, n=n, SQ=SQ)-logdpost(sim[i-1], n=n, SQ=SQ)))
+     if(runif(1) < alpha) {sim[i] <- nv; aceita <- aceita+1}
+     else sim[i] <- sim[i-1]
+   }
+   attr(sim, "taxa aceitação") <- aceita/(N-1)
+   return(sim)
+ }

```

Rodando o algoritmo para o exemplo temos os resultados a seguir.

```

> n.y <- length(y)
> SQ.y <- sum((y-10)^2)
> sim.mcmc <- MCMC(1000, init=1, n=n.y, SQ=SQ.y, a=0.2)
> attributes(sim.mcmc)

$`taxa aceitação`
[1] 0.9439

> plot(sim.mcmc, ty="l", xlab="simulação", ylab=expression(sigma^2[sim]))

```

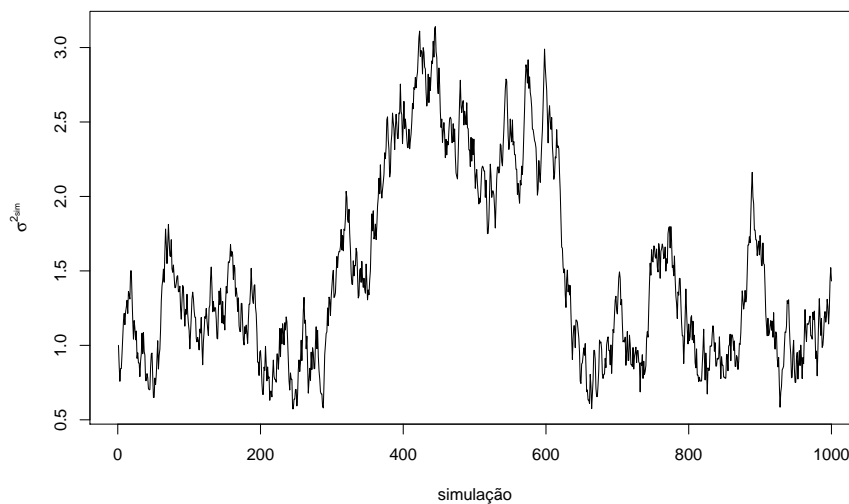


Figura 9: Sequência de valores em uma amostras da posteriori obtida por um algoritmo MCMC.

A figura que sobrepõe a densidade empírica mostra que esta amostra é claramente inadequada!

A mensagem é clara, embora simples, o algoritmo MCMC requer cuidados na sua implementação. Entre eles destacam-se: (i) é necessário que a distribuição estacionária seja atingida o que em geral não é válido para valores iniciais da cadeia que devem ser descartados sendo considerados um período de “aquecimento” da cadeia (*burn-in*); (ii) a distribuição $q(\cdot|\cdot)$ deve ser calibrada (no exemplo pela escolha do valor argumento a) para fornecer uma taxa de aceitação compatível com o algoritmo (no exemplo seria por volta de 30%); (iii) as amostras podem ser muito correlacionadas o que requer um maior número de valores a serem gerados (N). Além disto, pode-se armazenar apenas um a cada tantos valores gerados (por exemplo 1 a cada 10) o que reduz a memória computacional utilizada para armazenamento e a autocorrelação dos valores a serem armazenados. A este procedimento denomina-se raleamento (*thinning*); (iv) uma parametrização adequada pode assegurar uma melhor convergência, no exemplo poderia-se ter um melhor resultado com $\log(\sigma^2)$. Neste último caso os valores obtidos na cadeia podem ser diretamente transformados ao final para a parametrização desejada para inferências. Com estes pontos em mente o algoritmo MCMC anterior deve ser adaptado e calibrado (por tentativa e erro na especificação de a para atingir a taxa aceitação recomendada. Vamos rodar a cadeia novamente considerando apenas (i), (ii) e (iii).

```

> sim.mcmc <- MCMC(110000, init=1, n=n.y, SQ=SQ.y, a=5)
> attributes(sim.mcmc)

$`taxa aceitação`
[1] 0.3012

> sim.mcmc <- (sim.mcmc[-(1:10000)])[(1:10000)*10] ## thinning

```

```

> curve(dinvgamma(x, a=a.post, b=b.post), from=0, to=10, n=501, ylim=c(0, 0.9),
+       xlab=expression(sigma^2), ylab=expression(group("[",paste(sigma^2,"|",y),"]")))
> hist(sim.mcmc, prob=T, add=T)
> lines(density(sim.mcmc), col=2)

```

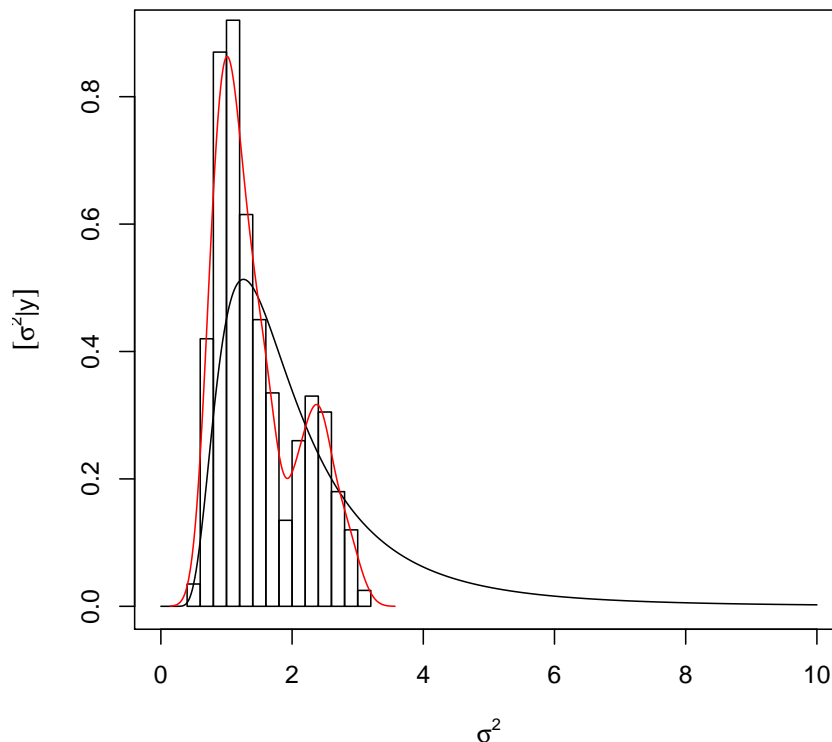


Figura 10: Histograma e densidade estimada da amostra MCMC comparadas com a posteriori analítica.

Os resultados com uma amostra maior, descartando amostras iniciais, tomando um valor a cada 10 e com uma taxa de aceitação adequada para o algoritmo são claramente melhores. Melhorias adicionais do algoritmo anterior ficam como exercício neste momento.

Existem ainda na literatura diversos “testes convergência” que visam verificar se há evidências de que a cadeia tenha atingido seu estado estacionário. Uma prática recomendada é obter não apenas uma, mas múltiplas cadeias o que assegura resultados mais confiáveis e é necessário para alguns testes de convergência.

Reparametrização

Tanto na análise de verossimilhança, quanto na aproximação normal da posteriori e também nos algoritmos numéricos, é vantajoso, para a qualidade das aproximações e para o desempenho dos algoritmos, que a verossimilhança e a posteriori sejam aproximadamente simétricas e, se possível, com curvas próximas às correspondentes da distribuição normal. Vamos considerar no exemplo em questão a reparametrização $\phi = \log(\sigma^2)$ (outra reparametrização comumente utilizada seria $\phi = \log(\sigma)$).

As função de verossimilhança, escores e hessiana, como funções de ϕ são dadas a seguir.

$$\begin{aligned}
 L(\phi = \log(\sigma^2)|y) &= \prod_{i=1}^n \frac{1}{\sqrt{2\pi e^\phi}} \exp\left\{-\frac{1}{2e^\phi}(y_i - \mu)^2\right\} \\
 &= (2\pi)^{-n/2} e^{-(n/2)\phi} \exp\left\{-\frac{1}{2e^\phi} \sum_{i=1}^n (y_i - \mu)^2\right\} \\
 l(\phi; y) &= \log\{L(\phi; y)\} = -(n/2)[\log(2\pi) + \phi + \frac{\sum_{i=1}^n (y_i - \mu)^2}{n} e^{-\phi}] \\
 U(\phi) &= \frac{dl(\phi; y)}{d\phi} = -(n/2)\left[1 - \frac{\sum_{i=1}^n (y_i - \mu)^2}{n} e^{-\phi}\right] \\
 H(\phi) &= \frac{d^2l(\phi; y)}{d(\phi)^2} = -\frac{\sum_{i=1}^n (y_i - \mu)^2}{2} e^{-\phi}
 \end{aligned}$$


```

> par(mfrow=c(1,2))
> plot(sim.mcmc, ty="l", xlab="simulação", ylab=expression(sigma^2[sim]))
> curve(dinvgamma(x, a=a.post, b=b.post), from=0, to=10, n=501, ylim=c(0, 0.7),
+       xlab=expression(sigma^2), ylab=expression(group("[",paste(sigma^2,"|",y),"]")))
> hist(sim.mcmc, prob=T, add=T, br=40)
> lines(density(sim.mcmc), col=2)

```

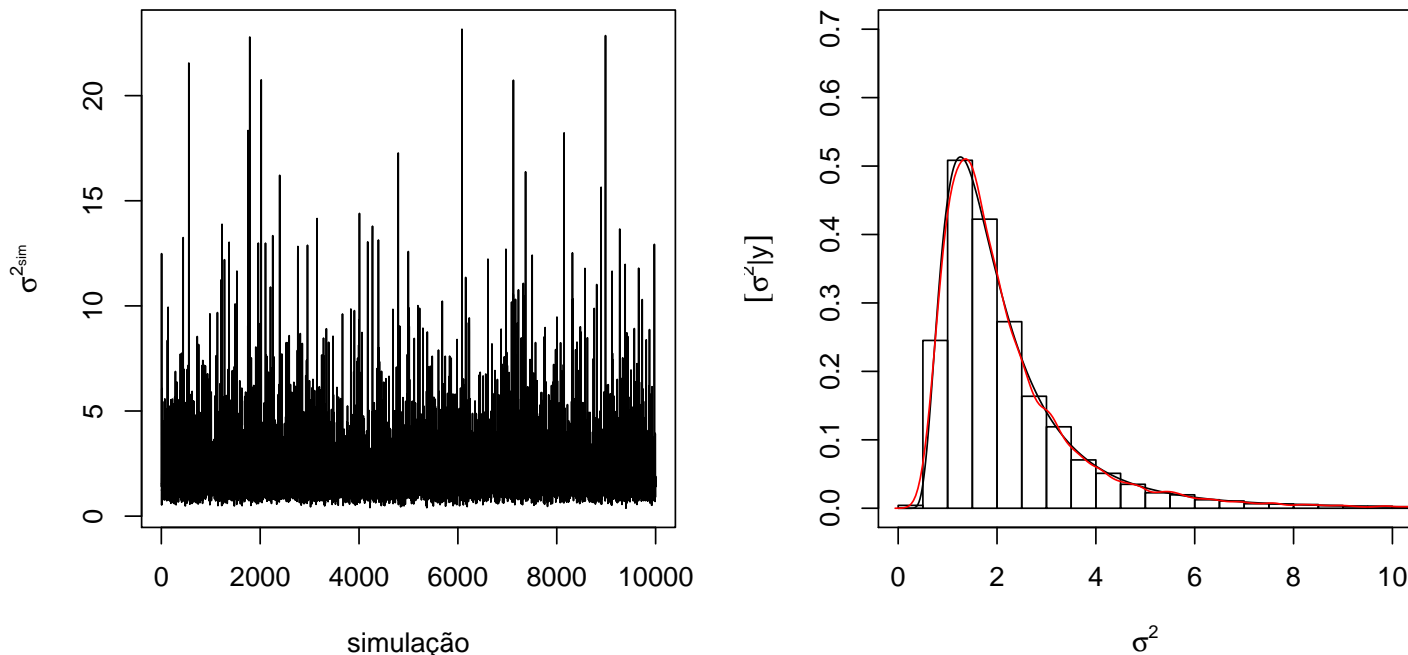


Figura 11: Novos resultados obtidos modificando opções no algoritmo MCMC.

O valor da verossimilhança maximizada é o mesmo, bem como o valor da função para cada ϕ é igual ao da função para o σ^2 correspondente, conforme princípio da invariância da verossimilhança. A função de verossimilhança escrita anteriormente pode ser facilmente adaptada para permitir a opção de se usar ou não a forma reparametrizada.

```

> (MLE <- sum((y-10)^2)/length(y))

```

```

[1] 1.619

```

```

> vero <- function(par, mu, dados, log=TRUE, repar=FALSE){
+   if(repar) par <- exp(par)
+   res <- -(length(dados)/2)*log(par) - sum((dados-mu)^2)/(2*par)
+   if(!log) res <- exp(res)
+   res
+ }

```

Os códigos a seguir comparam os resultados sem e com reparametrização.

```

> ## sem reparametrização
> (MLEnum <- optimize(vero, interval=c(0, 10), mu=10, dados=y, log=TRUE, maximum=TRUE))

```

```

$maximum

```

```

[1] 1.619

```

```

$objective

```

```

[1] -5.185

```

```

> (MLEnum <- optim(1, vero, mu=10, dados=y, log=TRUE, method="Brent",
+                 lower=0, upper=100, control=list(fnscale=-1), hessian=TRUE))

```

```

$par
[1] 1.619

$value
[1] -5.185

$counts
function gradient
      NA      NA

$convergence
[1] 0

$message
NULL

$hessian
      [,1]
[1,] -1.336

> ## com reparametrização
> (MLEnumR <- optimize(vero,interval=c(-20, 20), mu=10, dados=y, log=TRUE, maximum=TRUE, repar=TRUE))

$maximum
[1] 0.4816

$objective
[1] -5.185

> exp(MLEnumR$maximum)

[1] 1.619

> (MLEnumR <- optim(1, vero, mu=10, dados=y, log=TRUE, repar=TRUE, method="Brent",
+                   lower=-20, upper=20, control=list(fnscale=-1), hessian=TRUE))

$par
[1] 0.4815

$value
[1] -5.185

$counts
function gradient
      NA      NA

$convergence
[1] 0

$message
NULL

$hessian
      [,1]
[1,] -3.5

> c(MLE, MLEnum$par, exp(MLEnumR$par))

[1] 1.619 1.619 1.619

```

As funções de verossimilhança estão na figura a seguir. Note a invariância nos valores do eixo-Y do valores da verossimilhança.

Transformações podem levar a funções de verossimilhança que possuem melhores aproximações quadráticas. Além disto, a aproximação na parametrização original pode ser melhor obtida transformando-se de volta os valores a partir da aproximação quadrática de função reparametrizada. Na figura a seguir, o gráfico da esquerda mostra a função de verossimilhança de ϕ sua aproximação quadrática. A gráfico da direita mostra a função de verossimilhança de σ^2 , a sua aproximação quadrática

```

> par(mfrow=c(1,2))
> curve(vero(x, mu=10, dados=y, log=TRUE), from=0.55, to=9,
+       xlab=expression(sigma^2), ylab=expression(l(sigma^2)))
> abline(v=c(MLE,MLEnum$par), col=c(8,1), lwd=c(6,1))
> curve(vero(x, mu=10, dados=y, log=TRUE, repar=TRUE), from=log(0.55), to=log(9),
+       xlab=expression(phi == log(sigma^2)), ylab=expression(l(phi)))
> abline(v=c(log(MLE),MLEnumR$par), col=c(8,1), lwd=c(6,1))

```

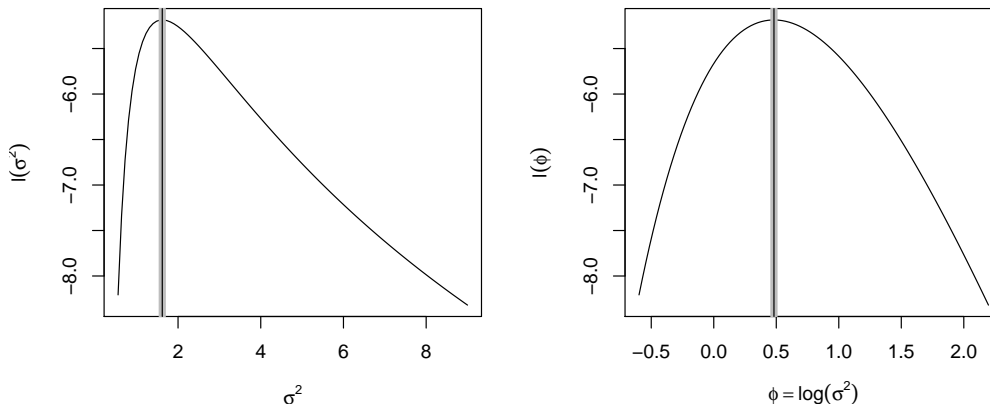


Figura 12: Funções de verossimilhança parametrizadas com σ^2 (esquerda) e $\phi = \log(\sigma^2)$ (direita).

como obtida anteriormente e a aproximação de $l(\sigma^2)$ obtida a partir da aproximação de $l(\phi)$. Esta última aproximação é claramente superior. Em resumo, o ideal é sempre usar a função de verossimilhança e evitar se possível o uso de aproximações. Entranto, se for necessário utilizar aproximações de-se aproximar a verossimilhança em uma parametrização conveniente, que forneça uma verossimilhança aproximadamente simétrica e quadrática e, a partir desta aproximação as funções aproximadas para outras parametrizações podem ser obtidas diretamente, (pela invariância da verossimilhança, apenas transformando os valores do parâmetro.

```

> Hfun.phi <- function(par, dados, mu){- (sum((dados-mu)^2)/2) * exp(-par)}
> logvero.approx.phi <- function(par, dados, mu, est){
+   vero(est, dados=dados, mu=mu, repar=TRUE) + 0.5 * (par - est)^2 * Hfun.phi(est, dados=dados, mu=mu)
+ }

```

```

> par(mfrow=c(1,2))
> curve(vero(x, mu=10, dados=y, log=TRUE, repar=TRUE), from=log(0.55), to=log(8.5),
+       xlab=expression(phi == log(sigma^2)), ylab=expression(l(phi)))
> curve(logvero.approx.phi(x, dados=y, mu=10, est=log(MLE)), from=log(0.55), to=log(8.5),
+       xlab=expression(phi == log(sigma^2)), ylab=expression(l(phi)), add=T, col=2, lty=2)
> legend("topright", c("verossimilhança", "aprox. quad."), lty=1:2, col=1:2)
> curve(llfun(x, mu=10, dados=y, log=TRUE), from=0.55, to=8.5,
+       xlab=expression(sigma^2), ylab=expression(l(sigma^2)))
> curve(llapprox(x, dados=y, mu=10), from=0.55, to=8.5, add=T, col=2, lt=2)
> sigma2.vals <- seq(0.5, 10, l=501)
> lines <- lines(sigma2.vals, logvero.approx.phi(log(sigma2.vals), dados=y, mu=10, est=log(MLE)),
+               lty=3, col=4)
> legend("topright", c("verossimilhança",
+               expression(paste("aprox. quad.", sigma^2)),
+               expression(paste("aprox. quad.", phi))), lty=1:3, col=c(1,2,4))

```

Figura 13: Esquerda: verossimilhança de ϕ e respectiva aproximação quadrática. Direita: verossimilhança de σ^2 e aproximações quadráticas obtidas a partir de $l(\sigma^2)$ e de $l(\phi)$.

No contexto bayesiano as reparametrizações seguem o teorema de transformação de variáveis. Revisando, se uma v.a. X possui densidade $f_X(x)$, então a v.a. $Y = h(X)$, sendo $h(\cdot)$ uma função monótona, tem densidade

$$f_Y(y) = f_X(h^{-1}(x)) \cdot \left| \frac{dh^{-1}(x)}{dy} \right|.$$

Para a expressão da posteriori de ϕ consideramos que a mudança de variável $\phi = h(\sigma^2) = \log(\sigma^2)$ e, equivalentemente $\sigma^2 = h^{-1}(\phi) = e^\phi$ é uma transformação segundo uma função monótona. Aplicando-se o teorema de transformação de variáveis aleatórias, a densidade da priori tem a expressão

$$\begin{aligned} f(\phi) &= f(h^{-1}(\phi)) \left| \frac{dh^{-1}(\pi)}{d\phi} \right| \\ &\propto \frac{1}{e^\phi} e^\phi \\ &\propto 1, \end{aligned}$$

ou seja, a priori (não normalizada, imprópria) é uma uniforme em toda reta real. A posteriori (não normalizada) tem a expressão,

$$f(\phi|y) \propto e^{-\frac{n}{2}\phi} \exp \left\{ -\frac{\sum_i (y_i - \mu)^2}{2} e^{-\phi} \right\},$$

e o gráfico pode ser obtido como a seguir. Comparado com o anteriormente mostrado para σ^2 é nítido o “melhor” (mais simétrico e próximo da normal) comportamento da posteriori sob a reparametrização.

```
> lpstphi <- function(phi, dados, mu, log=TRUE){
+   n <- length(dados)
+   SQ <- sum((dados-mu)^2)
+   res <- -(n/2)*(phi + (SQ/n) * exp(-phi))
+   if(!log) res <- exp(res)
+   return(res)
+ }
```

Esta função não é uma densidade, pois não é normalizada, ou seja, integrando 1 em seu domínio.

```
> integrate(lpstphi, lower=-10, upper=10, dados=y, mu=10, log=F)
```

0.00768 with absolute error < 2.4e-05

A distribuição normalizada é obtida calculando o denominador do Teorema de Bayes, $[Y] = \int [Y|\phi][\phi]d\phi$. No exemplo este termo é

$$\frac{(\sum_{i=1}^n (y_i - \mu)^2)^{n/2}}{\Gamma(n/2)}$$

que facilmente obtido, resolvendo a integral, ou, de forma mais fácil, tomando-se os termos constantes na densidade de σ^2 . A função para $[\phi|y]$ pode ser definida como forma a seguir.

```
> dpostphi <- function(phi, dados, mu, log=FALSE, cte=TRUE){
+   n <- length(dados)
+   SQ <- sum((dados-mu)^2)
+   res <- -(n/2)*(phi + (SQ/n) * exp(-phi))
+   if(cte) res <- res + (n/2)*log(SQ/2) - log(gamma(n/2))
+   if(!log) res <- exp(res)
+   return(res)
+ }
> integrate(dpostphi, lower=-10, upper=10, dados=y, mu=10, cte=TRUE)
```

1 with absolute error < 3.3e-05

Por simplicidade e generalidade, vamos aqui utilizar uma solução numérica que pode ser utilizado caso a constante normalizadora correspondente a $[Y]$ não possa ser obtida em forma fechada. Neste caso a função de densidade pode ser definida como a seguir.

```
> dpostphiN <- function(phi, ...){
+   CTE <- integrate(lpstphi, -Inf, Inf, ..., log=FALSE)$value
+   return(lpstphi(phi, ..., log=FALSE)/CTE)
+ }
> integrate(dpostphiN, low=-Inf, up=Inf, dados=y, mu=10)
```

1 with absolute error < 9.5e-07

```

> par(mfrow=c(1,2))
> curve(lpostphi(x, dados=y, mu=10, log=F), from=-2, to=3, xlab=expression(phi),
+ ylab=expression(group("[",paste(phi,"|",y),"]")))
> curve(dpostphi(x, dados=y, mu=10, log=F, cte=T), from=-2, to=3, xlab=expression(phi),
+ ylab=expression(group("[",paste(phi,"|",y),"]")), col=8, lwd=6)
> curve(dpostphiN(x, dados=y, mu=10), from=-2, to=3, xlab=expression(phi),
+ ylab=expression(group("[",paste(phi,"|",y),"]")), add=T)
> legend("topright", c("analítica","numérica"), lwd=c(6,1), col=c(8,1))

```

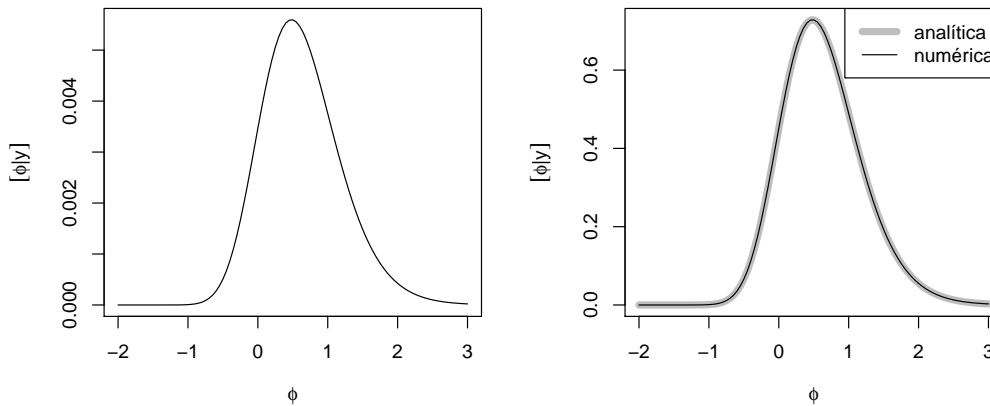


Figura 14: Densidades da posteriori $[\phi|y]$ não normalizada (esquerda) e normalizada (direita).

Para obter a aproximação normal da posteriori neste exemplo, é possível obter expressões analíticas para a moda e hessiano como se segue, em que $\hat{\sigma}^2 = \sum_i (y_i - \mu)^2 / n$:

$$\begin{aligned}
 \log(f(\phi|y)) &\propto -\frac{n}{2}[\phi + \hat{\sigma}^2 e^{-\phi}] \\
 \frac{d \log(f(\phi|y))}{d\phi} &\propto -\frac{n}{2}[1 - \hat{\sigma}^2 e^{-\phi}] \\
 \frac{d \log(f(\phi|y))}{d\phi} &\propto -\frac{n}{2}\hat{\sigma}^2 e^{-\phi} \\
 \hat{\phi} &= \log(\hat{\sigma}^2) \\
 H(\hat{\phi}) &= -\frac{n}{2}.
 \end{aligned}$$

Para os dados o exemplo em questão, temos os seguintes valores:

```

> (phi.moda <- log(sum((y-10)^2)/(length(y))))
[1] 0.4815
> (phi.hess <- - (sum((y-10)^2)/2) * exp(-phi.moda))
[1] -3.5
> (phi.hess <- - length(y)/2)
[1] -3.5

```

Entretanto vale ressaltar que nem sempre é possível obter tais expressões analiticamente e, se este for o caso, pode-se utilizar algoritmos numéricos.

```

> (phi.num <- optimize(lpostphi, interval = c(-20, 20), dados=y, mu=10, maximum=TRUE))
$maximum
[1] 0.4816
$objective
[1] -5.185

```

```
> (phi.num.hess <- optimHess(phi.num$maximum, lpostphi, dados=y, mu=10))
```

```
[,1]
```

```
[1,] -3.5
```

A aproximação normal da posteriori $[\phi|y]$ é mostrada no gráfico da direita na Figura a seguir. Nesta parametrização a aproximação é claramente superior à obtida anteriormente para $[\sigma^2|y]$ que é mostrada novamente na figura à esquerda.

```
> par(mfrow=c(1,2))
> curve(dinvgamma(x, a=a.post, b=b.post), from=0, to=12, n=501, ylim=c(0, 0.7),
+       xlab=expression(sigma^2), ylab=expression(group("[",paste(sigma^2,"|",y),"]")))
> curve(dnorm(x, m=mo.post, sd=sqrt(-1/H)), from=0, to=10, n=501, col=2, add=T, lty=2)
> legend("topright", c("posteriori","aprox. normal"), lty=c(1,2), col=c(1,2), cex=0.75)
> abline(v=mo.post, lty=3)
> #
> curve(dpostphi(x, dados=y, mu=10), from=-2, to=3, ylim=c(0, 0.75),
+       xlab=expression(phi), ylab=expression(group("[",paste(phi,"|",y),"]")))
> curve(dnorm(x, m=phi.moda, sd=sqrt(-1/phi.hess)), from=-2, to=3, col=2, add=T)
> legend("topright", c("posteriori","aprox. normal"), lty=c(1,2), col=c(1,2), cex=0.75)
> abline(v=phi.moda, lty=3)
```

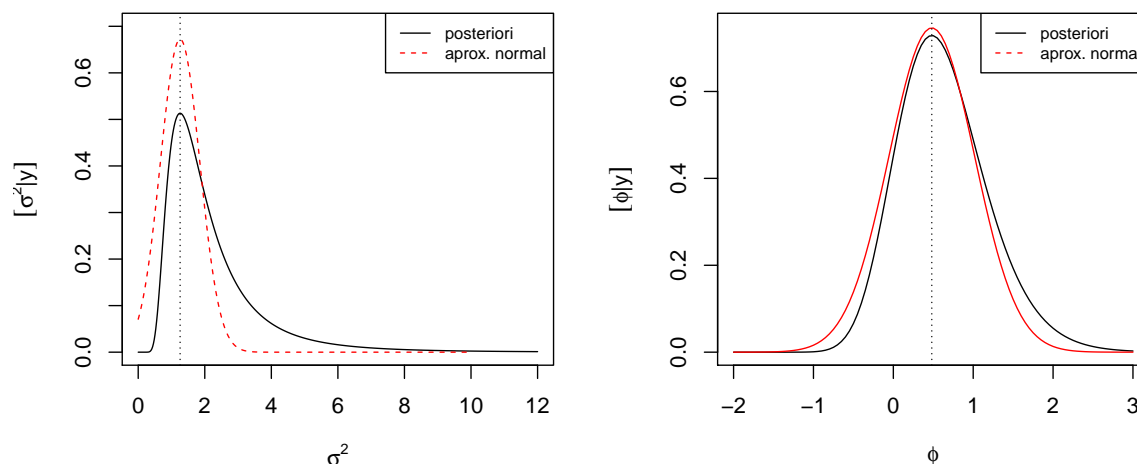


Figura 15: Comparação das aproximações normais das posterioris, na parametrização σ^2 original (esquerda) e com a reparametrização $\phi = \log(\sigma^2)$ (direita).

Portanto, assim como discutido para função de verossimilhança, uma melhor aproximação para $[\sigma^2|y]$ pode ser obtida tomando-se a transformação para σ^2 da aproximação de $[\phi|y]$. Em outras palavras, ao invés de aproximarmos diretamente $[\sigma^2|y]$, (i) obtemos a aproximação de $\tilde{f}(\phi|y)$ e, (ii) transformamos a aproximação para obter

$$\tilde{f}_{\sigma^2|y}(\sigma^2) = \tilde{f}_{\phi|y}(h^{-1}(\phi)) \left| \frac{dh^{-1}(\phi)}{d\sigma^2} \right|.$$

Aqui, por estarmos tratando de distribuições de probabilidades, é necessário utilizar o Jacobiano para escalonar corretamente a densidade, o que não era necessário na análise de verossimilhança.

Note-se que os gráficos anteriores podem ser igualmente obtidos com os comandos a seguir, definindo sequencias de valores e avaliando as funções. Embora aparentemente mais trabalhoso, os comandos desta forma são convenientes para obter diretamente as densidades após a reparametrização, sem a necessidade de definir as densidades transformadas.

```

> par(mfrow=c(1,2))
> phi.vals <- seq(-2, 2.5, l=200)
> dpostphi.vals <- dpostphi(phi.vals, dados=y, mu=10)
> dnorm.vals <- dnorm(phi.vals, m=phi.moda, sd=sqrt(-1/phi.hess))
> plot(phi.vals, dpostphi.vals, ty="l")
> lines(phi.vals, dnorm.vals, col=2)
> ##
> curve(dinvgamma(x, a=a.post, b=b.post), from=0, to=12, n=501, ylim=c(0, 0.7),
+       xlab=expression(sigma^2), ylab=expression(group("[",paste(sigma^2,"|",y),"]")))
> curve(dnorm(x, m=mo.post, sd=sqrt(-1/H)), from=0, to=10, n=501, col=2, add=T, lty=2)
> lines(exp(phi.vals), dnorm.vals*(1/exp(phi.vals)), col=4)

```

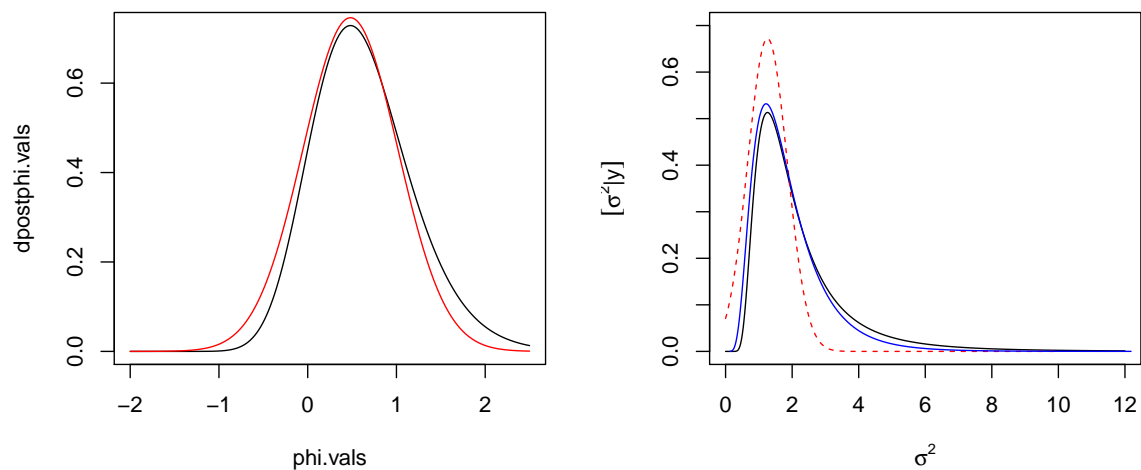


Figura 16: direita: posteriori $[\phi|y]$ e respectiva aproximação normal, esquerda: posteriori $[\sigma^2|y]$ (linha sólida, preta), aproximação normal de $[\sigma^2|y]$ (linha tracejada, vermelha) e transformação da aproximação normal de $[\phi|y]$ (linha pontilhada, azul).