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

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

1. Seja $x_1, \ldots x_n$ 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

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

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

$$Y = g(X) \Rightarrow \qquad f_Y(y) = f_X(g^{-1}(y)) \left| \frac{\mathrm{d}}{\mathrm{d}y} g^{-1}(y) \right|$$

Distribuições de Probabilidade

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

$$Z = (X - \mu)/\sigma \qquad \Gamma(\alpha) = \int_0^\infty x^{\alpha-1}\; \exp\{-x\}\; dx \qquad \Gamma(\alpha) = (\alpha - 1)! \qquad \Gamma(\alpha + 1) = \alpha\, \Gamma(\alpha)$$

Solução:

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

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

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

   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 $\mathrm{Var}(\hat{\sigma}^2) = I^{-1}(\hat{\sigma}^2)$ são obtidos analiticamente neste caso, com expressões

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

   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 ; sd.MLE <- sqrt(Var.MLE)
> c(H.MLE, Io.MLE, Var.MLE)
[1] -1.3360  1.3360  0.7485
```

   As funções $l(\sigma^2)$, $U(\sigma^2)$ e $I_o(\sigma^2)$ podem ser definidas como se seguem[1].

```
> 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)))
+ }
```

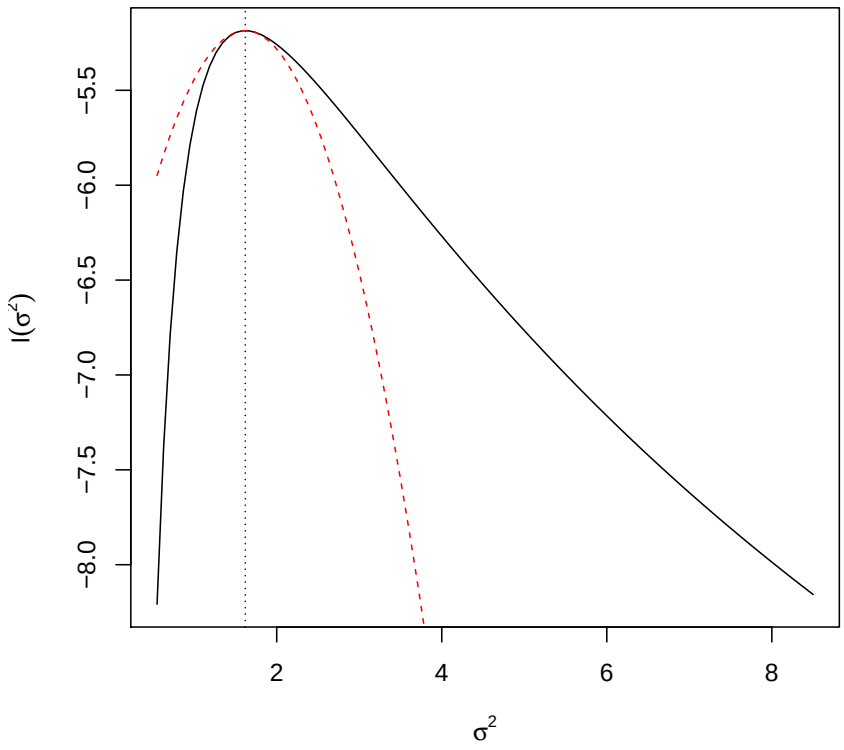
   Verificando os valores obtidos.

```
> 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
```

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

```
> 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)
> #
> llaprox <- 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(llaprox(x, dados=y, mu=10), from=0.55, to=8.5, add=T, col=2, lt=2)
```

[1] para 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.



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 algorítmo de Newton-Raphson, podem também ser utilizados. O algorítmo 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 internamente na função, uma vez que $\sigma^2 > 0$. As funções $U(\cdot)$ e $H(\cdot)$ são redefinidas sob esta reparametrização e omite-se os termos constantes 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(x) (1 - SQn * exp(-x))
+   H.lpar <- function(x) SQn * exp(-x)
+   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(exp(l.par), iteracoes = it))
+ }
> 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
[1] 1.619
attr(,"iteracoes")
[1] 5
```

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

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

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

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

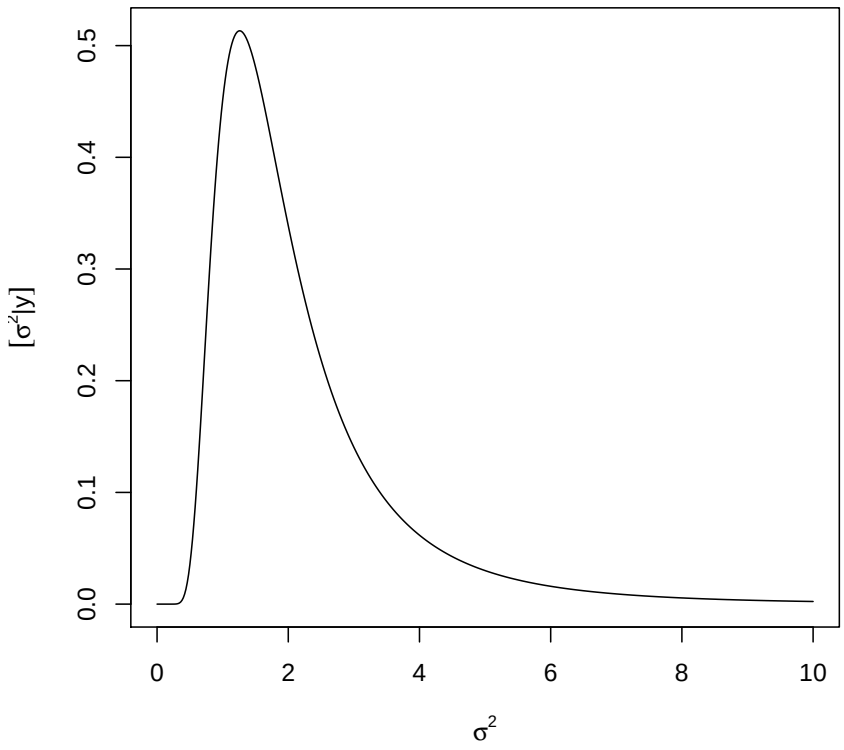
(d) Para o conjunto de dados temos que

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

```
> a.post <- length(y)/2
> b.post <- 2/sum((y-10)^2)
> c(a.post, b.post)
```

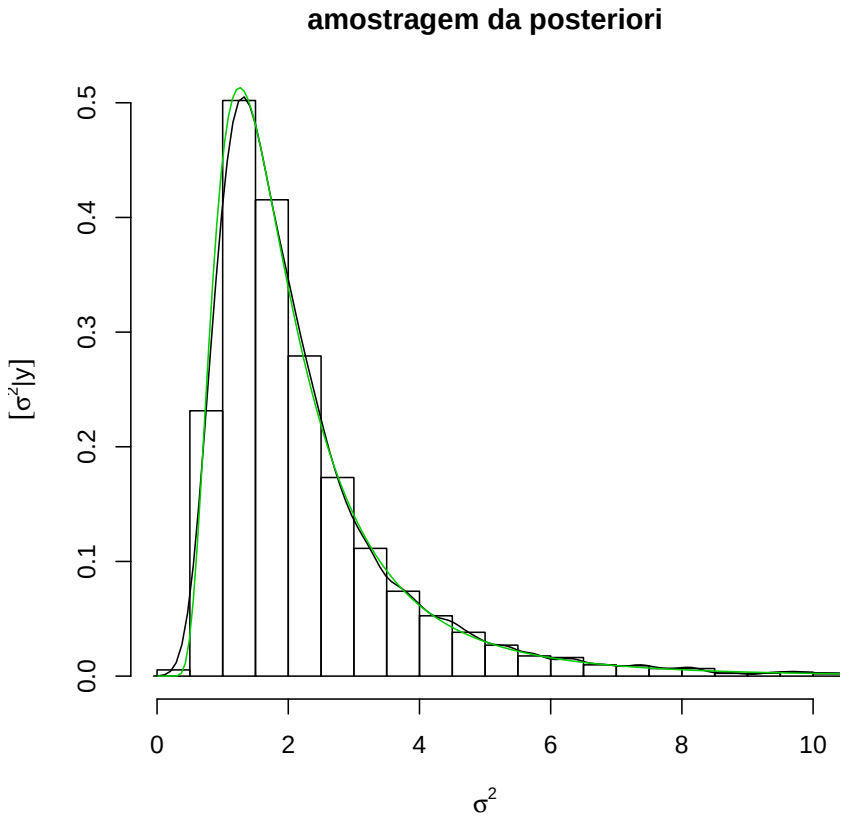
```
[1] 3.5000 0.1765
```

```
> 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)
+ }
> 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),"]")))
```



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.

```
> 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)),
+      main="amostragem da posteriori", xlab=expression(sigma^2),
+      ylab=expression(group("[",paste(sigma^2,"|",y),"]")), 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)
```



(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.).

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

```
> 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.

- 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 algorítmo 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.
- 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 função anterior desnecessária pois pode-se simplesmente tomar inversos dos quantis da distribuição gama.

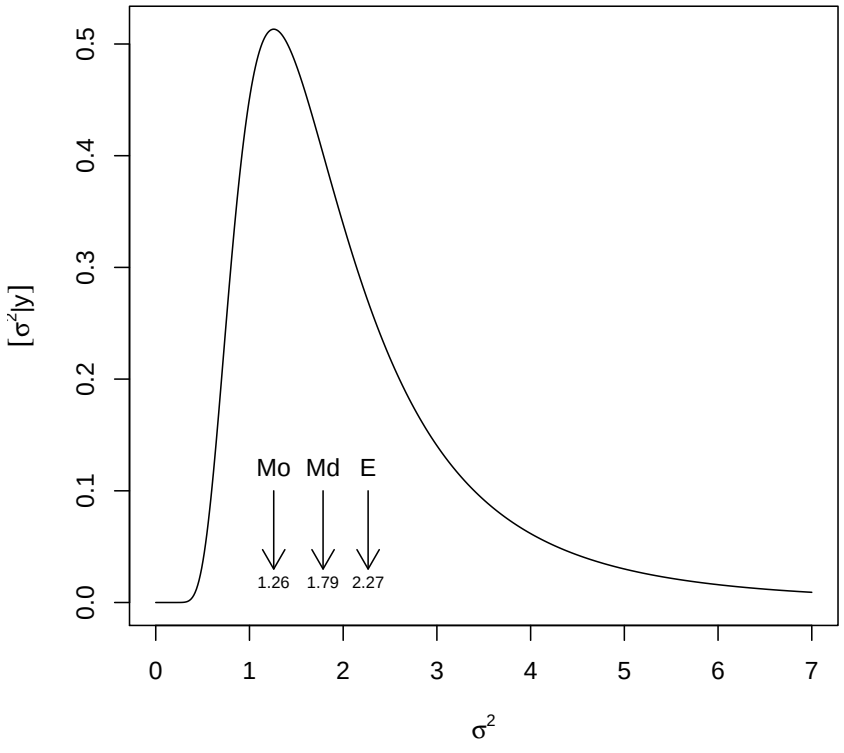
```
> 1/qgamma(1-0.5, shape=a.post, scale=b.post)
[1] 1.785
```

  Se desejar ter uma função específica basta fazer:

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

- Simulação

```
> median(sim)
[1] 1.794
```



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

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

i. Intervalo de credibilidade – quantis

A. Por otimização numérica

```
> (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 – HPD[3]

A. Otimização numérica. A função a seguit é apenas um exemplo ilustrativo e pode ser escrita de outrasd formas. 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. Simulações

```
> require(hdrcde)
> ## baseado nas amostras
> hdr(sim, prob=95)$hdr
      [,1]  [,2]
95% 0.3314 5.318
> ## baseado em uma aproximação discreta da densidade
> par.vals <- seq(0, 30, length=2000)
> d.vals <- dinvgamma(par.vals, a=a.post, b=b.post)
> hdr(den=list(x=par.vals, y=d.vals), prob=95)$hdr
      [,1]  [,2]
95% 0.4828 5.119
```

Alguns exemplos de funções disponíveis em outros pacotes.

```
> ## outras funções/pacotes
> require(TeachingDemos)
> emp.hpd(sim)
> hpd(qinvgamma, a=a.post, b=b.post)
> detach(package:TeachingDemos)
> ##
> require(BEST)
```

[3]diversos pacotes do *R* fornecem algorítmos 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.

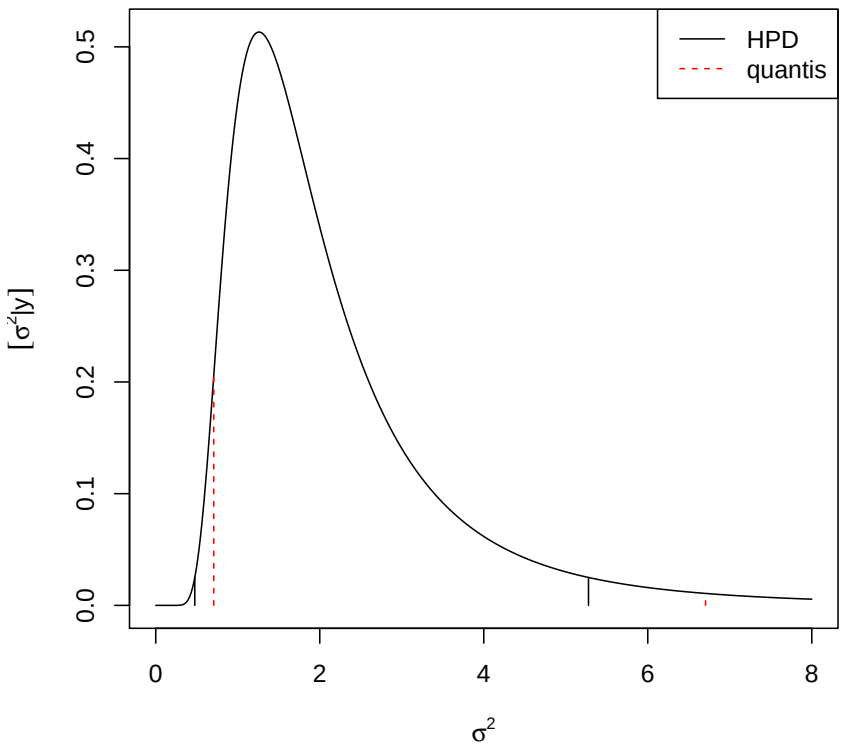
```
> hdi(sim, credMass=0.95)
> hdi(qinvgamma, credMass=0.95, a=a.post, b=b.post)
> detach(package:BEST)
> ##
> require(LaplacesDemon)
> p.interval(sim)
> detach(package:LaplacesDemon)
> ##
> require(emdbook)
> tcredint("invgamma", list(a=a.post, b=b.post)) # problema aqui
> detach(package:emdbook)
>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

Pode-se verificar que o IC-HPD é mais curto.

```
> 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.

```
> 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))
```



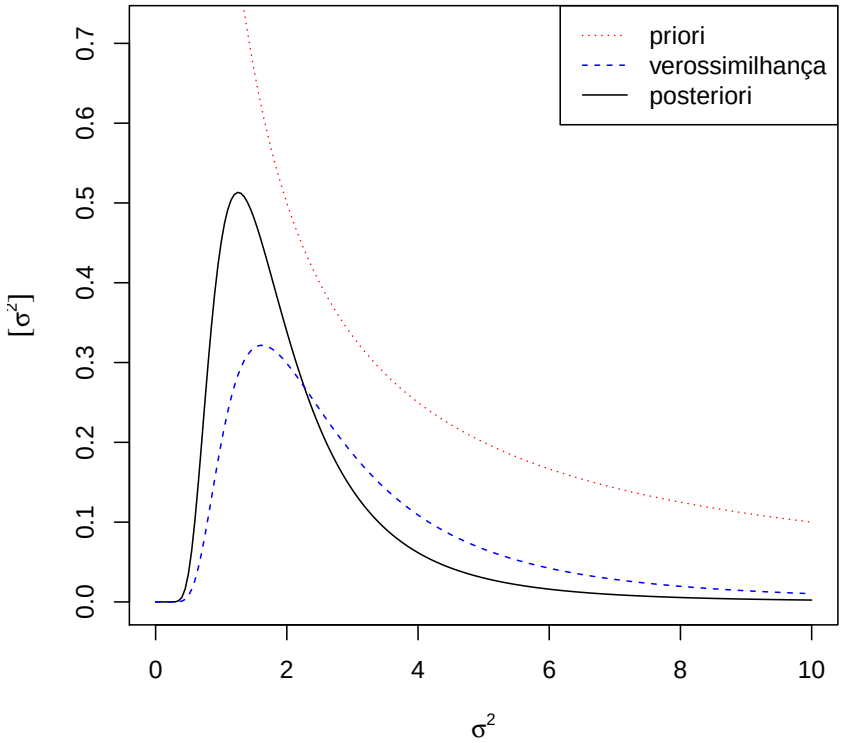
Gráficos da priori, posteriori e verossimilhança.

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)
+         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))))
+ }

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

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



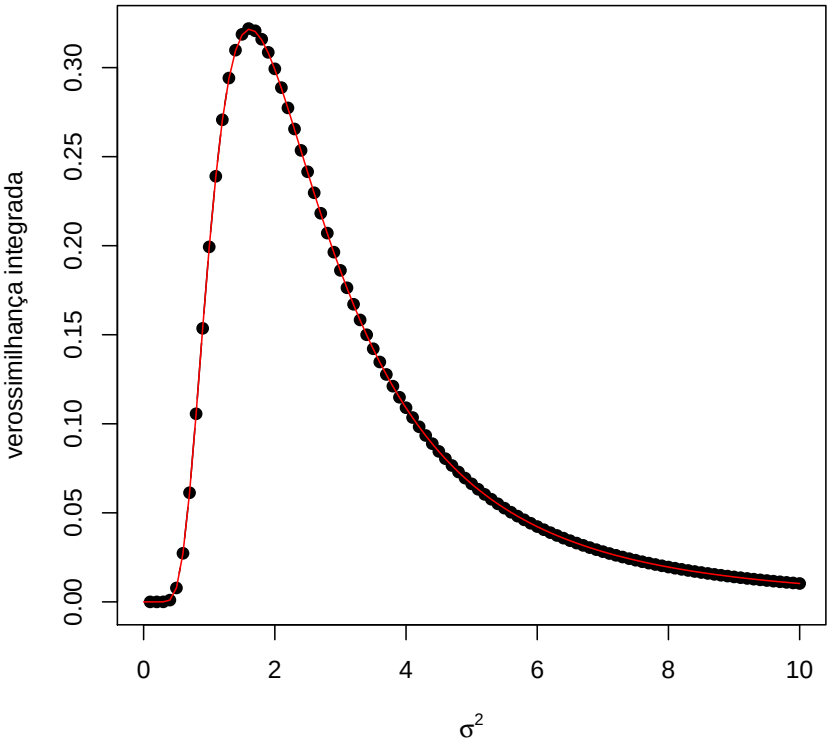
Para (ii) a verossimilhança integrada (numericamente) pode ser obtida da forma a seguir, que pode ser usada mesmo que a expressão da verossimilhança não tenha uma forma proporcional à de uma distribuição de probabilidades conhecida.

```
> 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)
+ }
> ## verificando...se integra 1
> integrate(llfun.int, mu=10, dados=y, lower=0, upper=50)

1 with absolute error < 2.9e-06
```

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

```
> 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)
```



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

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

```
[1] -2.839
```

```
[1] 0.3522
```

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:

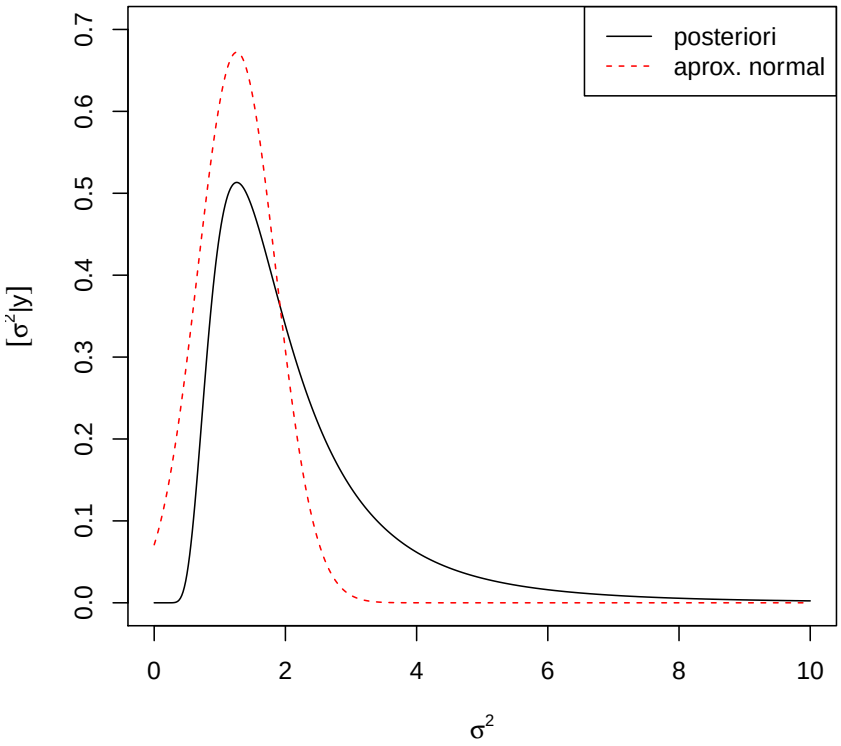
$$[\sigma^2|y] \sim \mathrm{N}(\hat{\sigma}^2, -H^{-1}(\hat{\sigma}^2)) \sim \mathrm{N}(1.26, 0.352)$$

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")]
```

Uma forma alternativa mais simples e direta de definir a log-posteriori, sem utilizar as derivações algébricas das expressões, (a uma consta `R` é a seguinte.

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

> 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))
```



No caso de se usarem estimativas numéricas a densidade da normal fica como a seguir, notanto que não toma-se o negativo do hessiano pois no códgo anterior foi minizada o negativo da log-posteriori.

```
> dnorm(x, m=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 os inversos dos valores simulados da distribuição gamma correspondente. Entretanto, a título de ilustração será exemplificado a seguir um algorítmo de *Cadeias de Markov via Monte Carlo* (MCMC) para obter amostras da posteriori.

O algorítmo 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 $U \sim \mathrm{U}(0,1)$ e

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

Uma versão simplificada deste algorítmo (chamada de algorítmo de Metrópolis) é obtida quanto $q(\cdot)$ 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 $\pi(\theta)$ e a distribuição objetivo, da qual se deseja simular, o 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 contante de proporcionalidade uma vez que tal constante é cancelada no cálculode 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 recalcular 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 $\theta_* \sim \mathrm{U}[\theta_{t-1} - a, \theta_{t-1} + a]$.

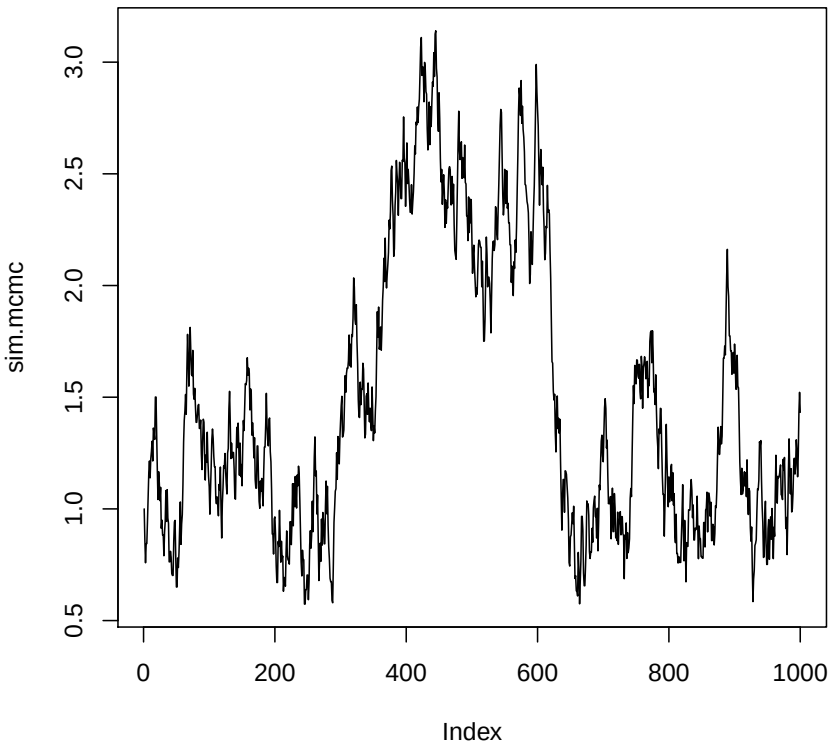
```
> MCMC <- function(N, init, n, SQ, a){
+     sim <- numeric(N)
+     sim[1] <- init
+     aceita <- 0
+     for(i in 2:N){
+         nv <- runif(1, sim[i-1]-a, sim[i-1]+a)
+         alpha <- min(1, exp(logdpost(nv, n=n, SQ=SQ)-logdpost(sim[i-1], n=n, SQ=SQ)))
+         if(runif(1) < alpha) {sim[i] <- nv; aceita <- aceita+1}
+         else sim[i] <- sim[i-1]
+     }
+     attr(sim, "taxa aceitação") <- aceita/(N-1)
+     return(sim)
+ }
```

E rodando o algorítmo para o exemplo temos os seguintes resultados.

```
> 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)
> plot(sim.mcmc, ty="l")
> attributes(sim.mcmc)

$`taxa aceitação`
[1] 0.9439
```

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

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



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

```
> 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)
> lines(density(sim.mcmc), col=2)
```

O algorítmo MCMC requer cuidados. Entre eles destacam-se: (i) é necessário que a distribuição estácionária 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 algorítmo (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 tantod valores gerados (por exemplo 1 a cada 10) o que reduz a memória computacional utilizada para armazenamento e reduz a autocorrelação dos valores a serem armazenados e a este procedimento denomina-se raleamento (*thinning*). Com estes pontos em mente o algorítmo `MCMC` anterior deve ser adaptado e calibrado (por tentativa e erro na especicicação de `a` para atingir a taxa aceitação recomendada.

Existem ainda na literatura diversos "testes convergência" que visam verificar se há evidências de que a cadeia tenha atingido seu estado estacionário.

Reparametrização

Tanto na análise de verossimilhança, quanto na aproximação nomal da posteriori e também nos algorítmos numéricos, é vantajoso para a qualidade das aproximações e para o desempenho dos algorítmos 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 possível seria $\phi = \log(\sigma)$).

O valor da verossimilhança maximizada é o mesmo (cf princípio da invariância da verossimilhança). A função de verossimilhança escrita anteriormente pode ser facilmente adaptada para permitir

```
> (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
+ }
```

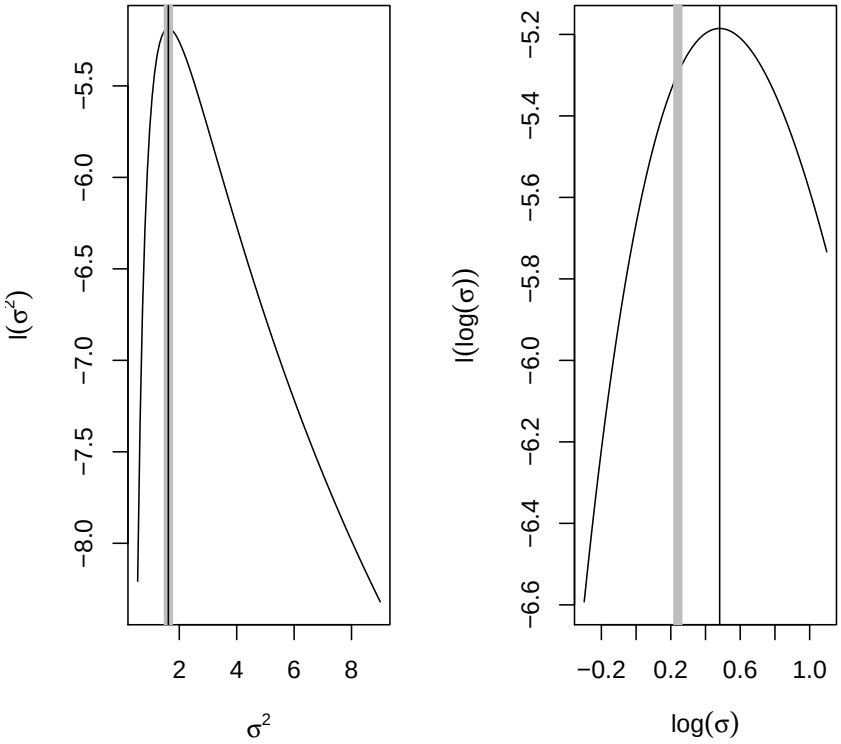
Comparando-se os resultados sem e com reparametrização.

```
> ## 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
```

```
> exp(MLEnumR$par)

[1] 1.619
```

As funções de verossimilhança estão na figura a seguir. Note a inveriância nos valores do eixo Y (valores da verossimilhança).

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

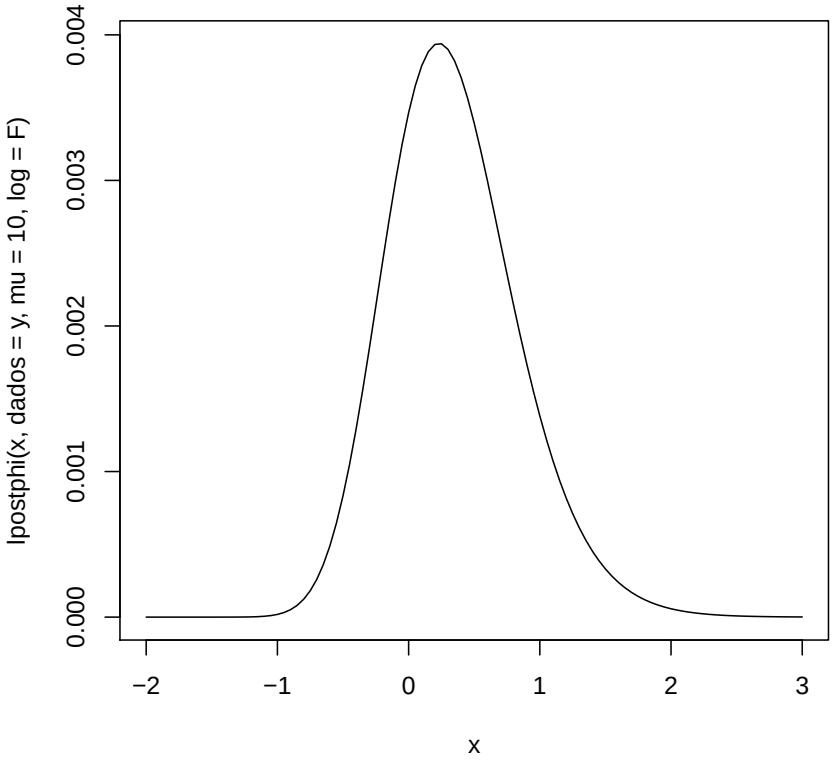


A distribuição a posteriori (não normalizada) tem a expressão:

$$f(\phi|y) = \ldots$$

e o gráfico pode ser obtido como a seguir. Comparado com o anteriormente mostrado para σ^2 é nítido o "melhor" comportamento sobre a reparametrização.

```
> lpostphi <- function(phi, dados, mu, log=TRUE){
+     n <- length(dados)
+     SQ <- sum((dados-mu)^2)
+     res <- (-(n+2)/2)*phi - (SQ/2) * exp(-phi)
+     if(!log) res <- exp(res)
+     return(res)
+ }
> curve(lpostphi(x, dados=y, mu=10, log=F), from=-2, to=3)
```



A distribuição normalizada (para integrar 1 em seu domínio) pode ser definida como a seguir.

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

1 with absolute error < 1.2e-06
```

Para obter a aproximação normal da posteriori neste exemplo é possível derivar expresões analíticas para a moda e hessiano.

$$f' = \ldots f'' \qquad = \ldots \hat{\phi} = \ldots H(\hat{\phi}) \qquad = \ldots$$

```
> (phi.moda <- log(sum((y-10)^2)/(length(y)+2)))

[1] 0.2302

> (phi.hess <- - (sum((y-10)^2)/2) * exp(-phi.moda))

[1] -4.5
```

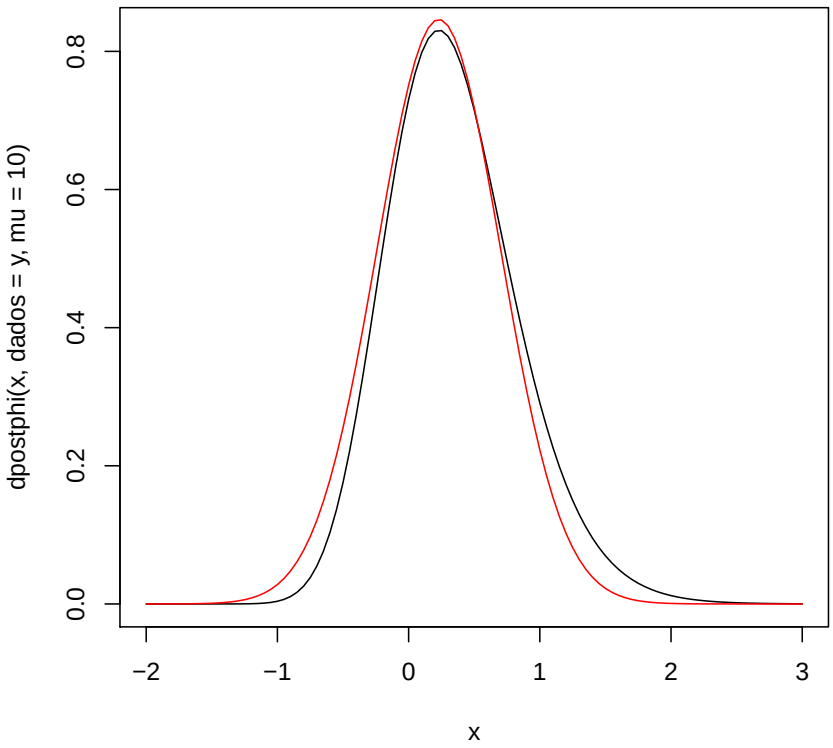
Entretanto vale ressaltar que nem sempre é possível obter tais expressões analiticamente e se este for o caso pode-se utilizar algorítmos numéricos.

```
> (phi.num <- optimize(lpostphi, interval = c(-20, 20), dados=y, mu=10, maximum=TRUE))

$maximum
[1] 0.2302

$objective
[1] -5.536

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

     [,1]
[1,] -4.5

> c(exp(phi.num$maximum), mo.post)
```

```
[1] 1.259 1.259
```

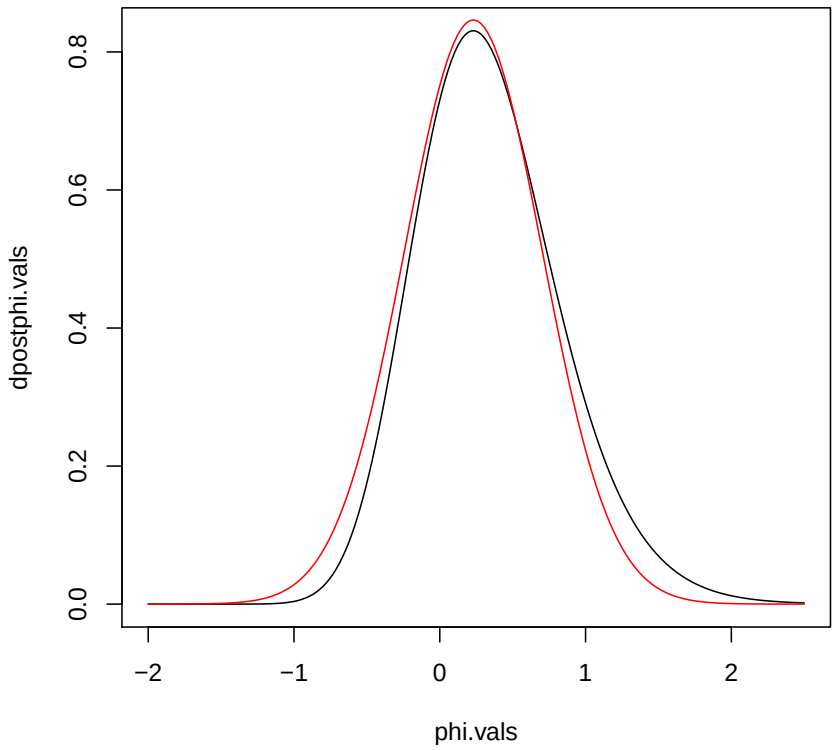
A aproximação normal da posteriori é mostrada na Figura a seguir.

```
> curve(dpostphi(x, dados=y, mu=10), from=-2, to=3)
> curve(dnorm(x, m=phi.moda, sd=sqrt(-1/phi.hess)), from=-2, to=3, col=2, add=T)
```



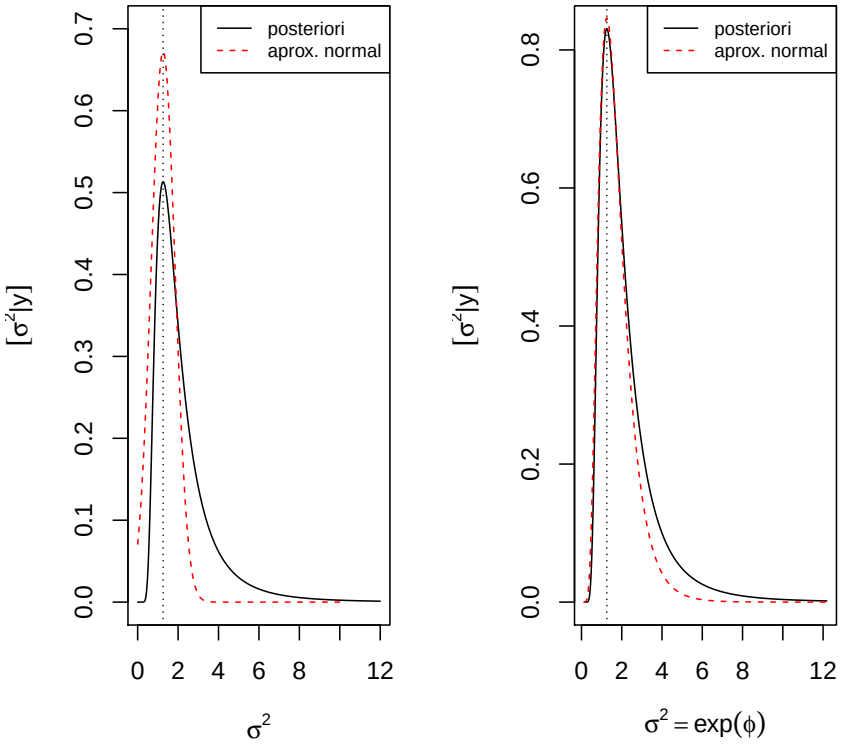
E nesta parametrização com ϕ a aproximação é claramente superior à obtida anteriormente para σ^2. Primeiro vamos notar que o gráfico anterior pode ser igualmente obtido com os comandos a seguir.

```
> 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)
```



E este última forma, embora aparetemente mais trabalhosa, é conveniente para obter as densidades (ou também verossimilhanças) após a reparametrização.

```
> 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)
> ##
>
> plot(exp(phi.vals), dpostphi.vals, ty="l",
+       xlab=expression(sigma^2 == exp(phi)), ylab=expression(group("[",paste(sigma^2,"|",y),"]")))
> #lines(exp(phi.vals), dpostphi.vals, col=4)
> lines(exp(phi.vals), dnorm.vals, col=2, lty=2)
> legend("topright", c("posteriori","aprox. normal"), lty=c(1,2), col=c(1,2), cex=0.75)
> abline(v=exp(phi.moda), lty=3)
> integrate(dinvgamma, low=0, up=Inf, a=a.post, b=b.post)

1 with absolute error < 0.00012

> integrate(dpostphi, low=-Inf, up=Inf, dados=y, mu=10)

1 with absolute error < 1.2e-06
```


posteriori
aprox. normal
[σ²|y]
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
2
4
6
8
12
σ²
posteriori
aprox. normal
[σ²|y]
0.0
0.2
0.4
0.6
0.8
0
2
4
6
8
12
σ² = exp(ϕ)