# Apresentacao_stat4good

## Introdução

Foi desenvolvido no artigo "A new family of life distributions" por Birnbaum e Saunders a distribuição que ora apresento. A mesma modela o tempo de vida de materiais e equipamentos sujeitos a cargas dinâmicas através de modelos de dano acumulado e tem sido amplamente utilizada na área de engenharia, na indústria, em negócios, na análise de confiabilidade, na análise de sobrevivência, em ciências ambientais e ciências médicas e em diversas outras áreas, pois possui propriedades interessantes e uma relação próxima com a distribuição normal, o que a torna, do ponto de vista de aplicação, uma alternativa mais atraente para as bem conhecidas distribuições Weibull, log-logística, log-normal, gama e modelos inversos Gaussianos.

## Função de Distribuição Acumulada Birnbaum-Saunders

Suponha que $T$ seja uma variável aleatória que representa o tempo total até que ocorra a falha, então a distribuição de $T$ proposta por Birnbaum e Saunders (1969a) tem função de distribuição acumulada (fda) dada por:

$$F_T(t;\alpha,\beta) = P(T \leq t) = \Phi\left[\frac{1}{\alpha}\left(\sqrt{\frac{t}{\beta}} - \sqrt{\frac{\beta}{t}}\right)\right],\ t > 0 \tag{1}$$

em que $\Phi(.)$ é a fda de uma distribuição normal padrão. Dizemos que $T$ segue uma distribuição BS, com parâmetros de forma $\alpha > 0$ e de escala $\beta > 0$, que é usualmente denotada por $T \sim BS(\alpha,\beta)$.

## Função Densidade

Considerando a distribuição acumulada da variável aleatória $T$ dada em (1), a sua correspondente função densidade de probabilidade (fdp) é dada por

$$f_T(t) = \frac{t^{-\frac{3}{2}}(t+\beta)}{2\sqrt{2\pi}\alpha\sqrt{\beta}}\exp\left[-\frac{1}{2\alpha^2}\left(\frac{t}{\beta} + \frac{\beta}{t} - 2\right)\right] \tag{2}$$

em que $\alpha > 0$ e $\beta > 0$.

## Função de distribuição acumulada, função densidade e seus respectivos gráficos no R.

```
#Shape=alpha, scala=beta
acbs<-function(t,alpha,beta){
  ff<-pnorm((1/alpha)*(sqrt(t/beta)-sqrt(beta/t)))
  return(ff)
}

#Função densidade
bs<-function(t,alpha,beta){
  f<-((t+beta)/(2*alpha*sqrt(2*pi*beta)))*(t^(-3/2))*exp((-1/(
    2*alpha^2))*((t/beta)+
                    (beta/t)-2))
```

```
  return(f)
}
```

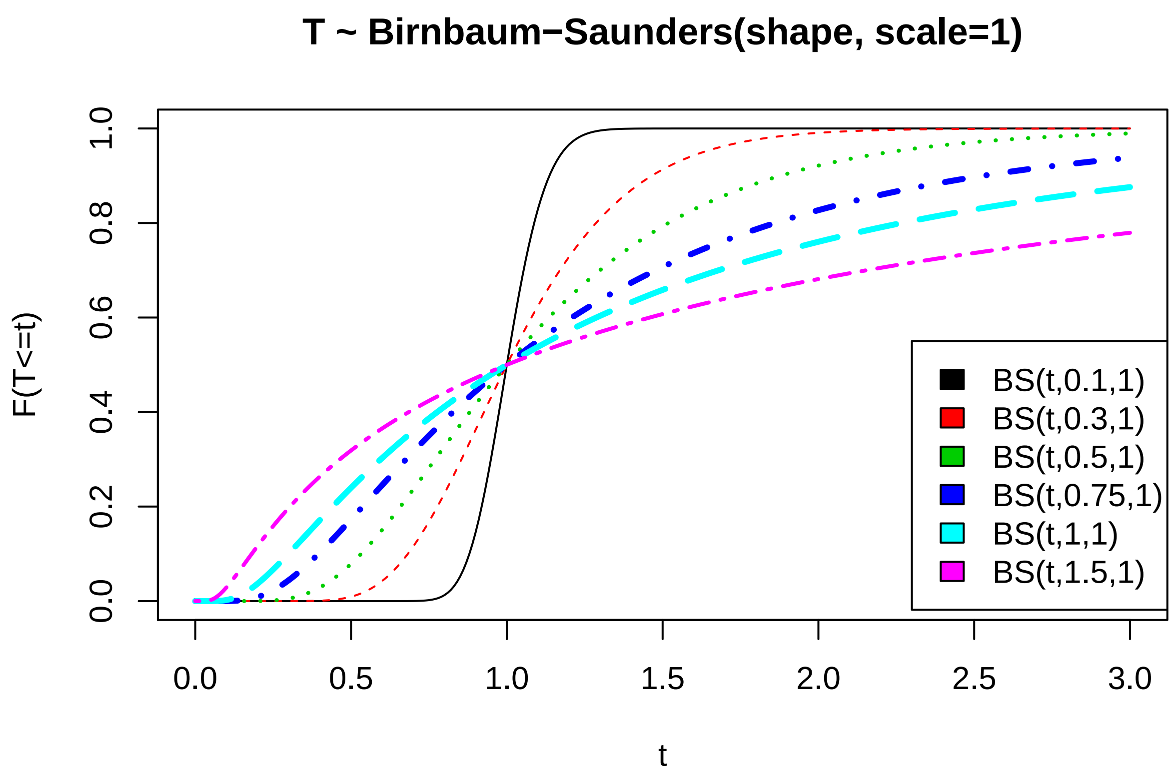
Conta simples para verificar se a integral da função densidade é igua a 1. Isso é somente uma forma de analisar se a digitação da densidade esta correta.

```
alpha=10
beta=2
integrand <- function(t){((t+beta)/(2*alpha*sqrt(2*pi*beta)))*(t^(-3/2))*
    exp((-1/(2*alpha^2))*((t/beta)+(beta/t)-2))}
integrate(integrand, lower = 0, upper = Inf)
```

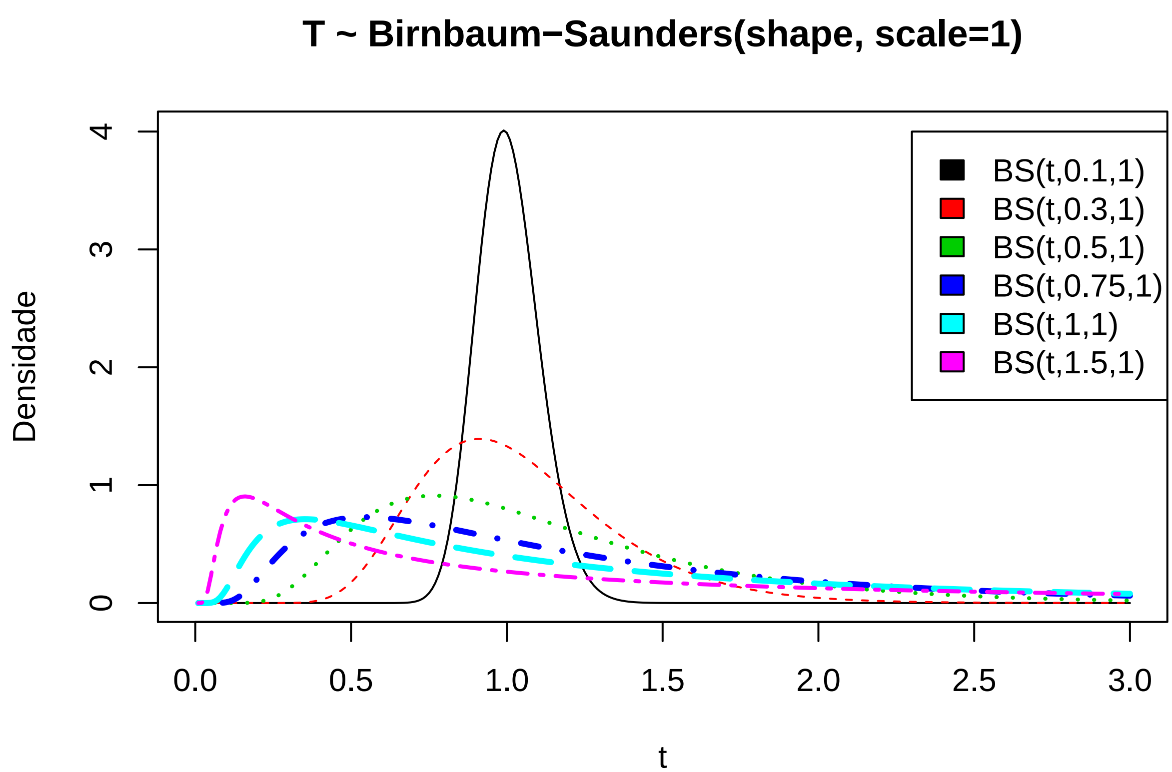
```
## 1 with absolute error < 4.1e-05
```

Note que com a mudança de alpha, mantendo beta fixo, há uma alteração na assimetria do gráfico, veja os gráficos da distribuição acumulada e da densidade:

```
t=seq(0,3,by=0.01)
plot(t,acbs(t,0.1,1),main='T ~ Birnbaum-Saunders(shape, scale=1)',ylab='F(T<=t)',type='l')
lines(t,acbs(t,0.3,1),col=2,lty=2, lwd=1)
lines(t,acbs(t,0.5,1),col=3,lty=3, lwd=2)
lines(t,acbs(t,0.75,1),col=4,lty=4, lwd=3)
lines(t,acbs(t,1,1),col=5,lty=5, lwd=3)
lines(t,acbs(t,1.5,1),col=6,lty=6, lwd=2)
legend(2.3, 0.55, c("BS(t,0.1,1)","BS(t,0.3,1)","BS(t,0.5,1)","BS(t,0.75,1)","BS(t,1,1)",
                    "BS(t,1.5,1)"), fill=1:6)
```

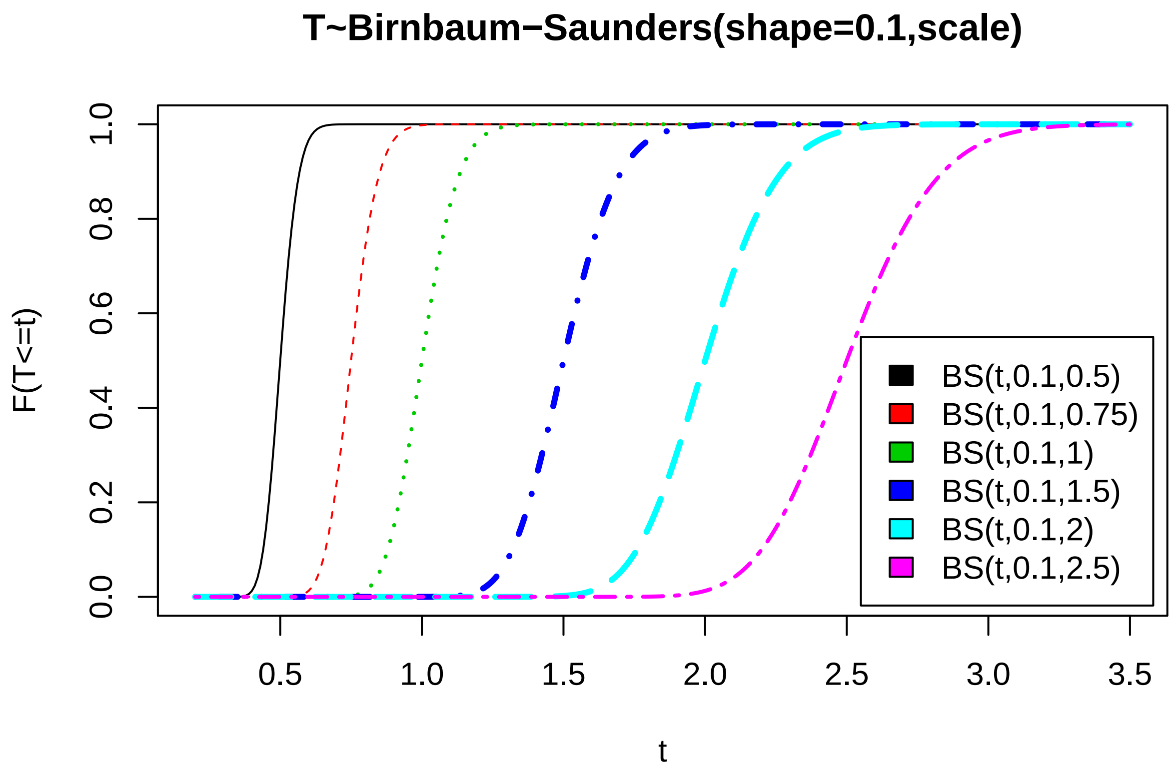


```
#
t=seq(0,3,by=0.01)
plot(t,bs(t,0.1,1),main='T ~ Birnbaum-Saunders(shape, scale=1)',ylab='Densidade',type='l')
lines(t,bs(t,0.3,1),col=2,lty=2, lwd=1)
lines(t,bs(t,0.5,1),col=3,lty=3, lwd=2)
lines(t,bs(t,0.75,1),col=4,lty=4, lwd=3)
lines(t,bs(t,1,1),col=5,lty=5, lwd=3)
lines(t,bs(t,1.5,1),col=6,lty=6, lwd=2)
legend(2.3, 4, c("BS(t,0.1,1)","BS(t,0.3,1)","BS(t,0.5,1)","BS(t,0.75,1)","BS(t,1,1)",
                 "BS(t,1.5,1)"), fill=1:6)
```

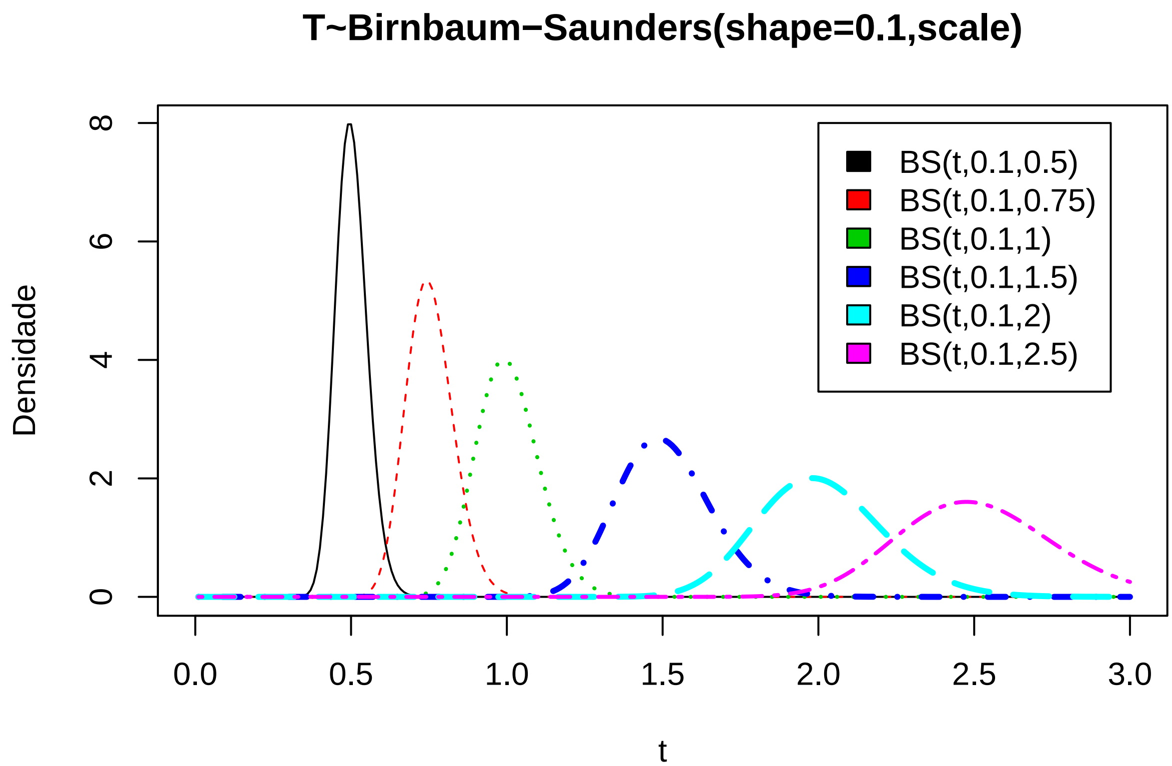



Enquanto que ao manter alpha fixo e mudar o beta há uma mudança na média e na variança da variável aleatória.

```
t=seq(0.2,3.5,by=0.01)
plot(t,acbs(t,0.1,0.5),main='T~Birnbaum-Saunders(shape=0.1,scale)',ylab='F(T<=t)',type='l')
lines(t,acbs(t,0.1,0.75),col=2,lty=2, lwd=1)
lines(t,acbs(t,0.1,1),col=3,lty=3, lwd=2)
lines(t,acbs(t,0.1,1.5),col=4,lty=4, lwd=3)
lines(t,acbs(t,0.1,2),col=5,lty=5, lwd=3)
lines(t,acbs(t,0.1,2.5),col=6,lty=6, lwd=2)
legend(2.55, 0.55, c("BS(t,0.1,0.5)","BS(t,0.1,0.75)","BS(t,0.1,1)","BS(t,0.1,1.5)",
                     "BS(t,0.1,2)","BS(t,0.1,2.5)"), fill=1:6)
```




```r
t=seq(0,3,by=0.01)
plot(t,bs(t,0.1,0.5),main='T~Birnbaum-Saunders(shape=0.1,scale)',ylab='Densidade',type='l')
lines(t,bs(t,0.1,0.75),col=2,lty=2, lwd=1)
lines(t,bs(t,0.1,1),col=3,lty=3, lwd=2)
lines(t,bs(t,0.1,1.5),col=4,lty=4, lwd=3)
lines(t,bs(t,0.1,2),col=5,lty=5, lwd=3)
lines(t,bs(t,0.1,2.5),col=6,lty=6, lwd=2)
legend(2, 8, c("BS(t,0.1,0.5)","BS(t,0.1,0.75)","BS(t,0.1,1)","BS(t,0.1,1.5)",
               "BS(t,0.1,2)","BS(t,0.1,2.5)"), fill=1:6)
```




## Observações:

- A função de distribuição acumulada da $BS(\alpha, \beta)$ é identificável.
- $F_T(\beta) = 0.5$, ou seja, $\beta$ é a mediana da $BS(\alpha, \beta)$
- Se $T \sim BS(\alpha, \beta)$, então
$$a > 0,\ aT \sim BS(\alpha, a\beta) \text{ e } T^{-1} \sim BS(\alpha, \beta^{-1})$$
- A partir da FDA, fazendo $Z = \dfrac{1}{\alpha}\left(\sqrt{\frac{t}{\beta}} - \sqrt{\frac{\beta}{t}}\right)$ temos:
$$T = \frac{\beta}{4}\left[\alpha Z + \sqrt{(\alpha Z)^2 + 4}\right]^2 \tag{3}$$
Essa relação é extremamente útil e pode ser usada para obtenção de números pseudo-aleatórios.

a média, a variância, o coeficiente de variação e os coeficientes de assimetria ($\mu_3$) e curtose ($\mu_4$) da distribuição BS são, respectivamente:

$$E(T) = \beta\left(1 + \frac{\alpha^2}{2}\right),\ Var(T) = (\alpha\beta)^2\left(1 + \frac{5\alpha^2}{4}\right)$$

$$CV(T) = \frac{\sqrt{5\alpha^4 + 4\alpha^2}}{\alpha^2 + 2}$$

$$\mu_3 = \frac{16\alpha^2(11\alpha^2 + 6)}{(5\alpha^2 + 4)^3},\ \mu_4 = 3 + \frac{6\alpha^2(93\alpha^2 + 41)}{5 + \alpha^2 + 4}$$

## A Função de Verossimilhança

Se $T_1, \cdots, T_n$ é uma amostra de n observações independentes da distribuição $BS(\alpha, \beta)$. Então, o logaritmo da função de verossimilhança para $\theta = (\alpha, \beta)$, possui a seguinte forma:

$$l(\theta) = -\frac{3}{2}\sum_{i=1}^{n}\log(t_i) - n\log(2\alpha) - \frac{n}{2}\log(2\pi)\beta - \frac{1}{2\alpha^2}\sum_{i=1}^{n}\left(\frac{t_i}{\beta} + \frac{\beta}{t_i} - 2\right) + \sum_{i=1}^{n}(t_i + \beta)$$

### Estimadores de Máxima Verossimilhança

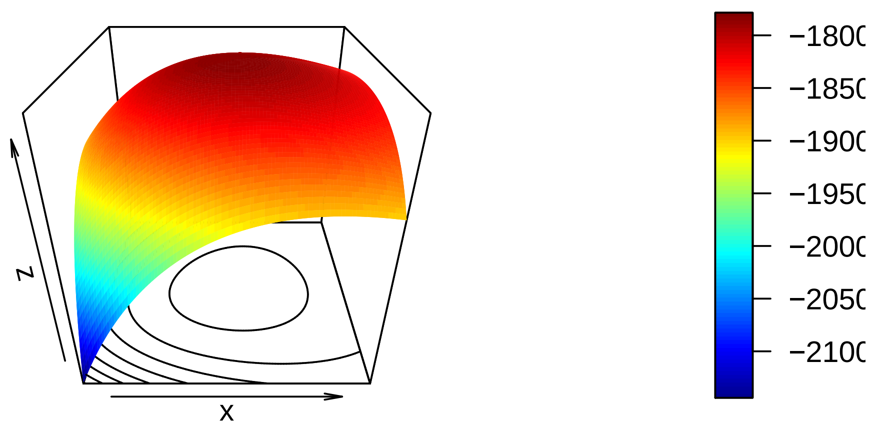
Os estimadores de máxima verossimilhança (EMV) de $\alpha$ e $\beta$ são obtidos maximizando $l(\theta)$ a partir das soluções das equações:

$$\frac{\partial l(\alpha, \beta)}{\partial \alpha} = -\frac{n}{\alpha}\left(1 + \frac{2}{\alpha^2}\right) + \frac{1}{\beta\alpha^3}\sum_{i=1}^{n} t_i + \frac{\beta}{\alpha^3}\sum_{i=1}^{n}\frac{1}{t_i} = 0$$

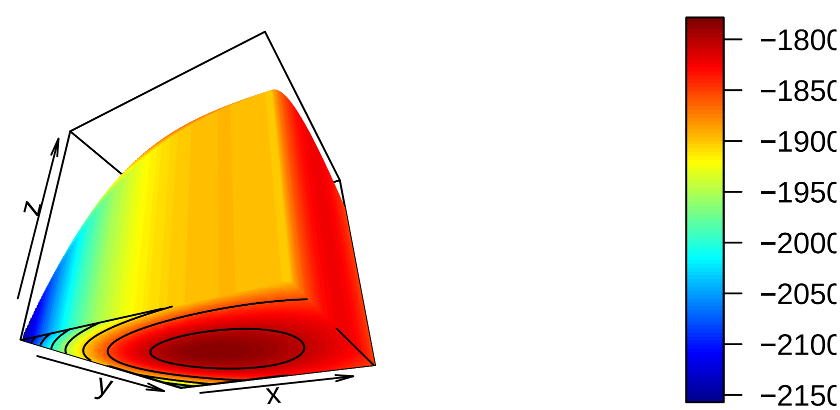
$$\frac{\partial l(\alpha, \beta)}{\partial \beta} = -\frac{n}{2\beta} + \frac{\sum_{i=1}^{n} t_i}{2\alpha^2\beta^2} + \sum_{i=1}^{n}\frac{1}{t_i + \beta} - \frac{1}{2\alpha^2}\sum_{i=1}^{n}\frac{1}{t_i} = 0$$

Antes de apresentar a simulação de Monte Carlo para estimação dos parâmetros veja o gráfico da função log de verossimilhança:

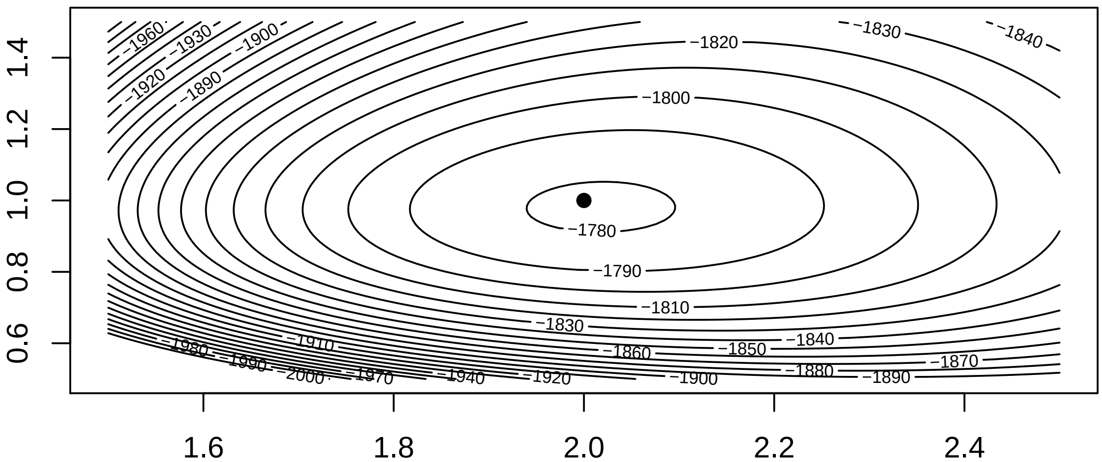
```
########################################
require(plot3D)
alpha=2
beta=1
n=1000
set.seed(355)
z1<-rnorm(1000,0,1)
dados<-beta*((alpha*z1/2)+sqrt((alpha*z1/2)^2+1))^2
alpha= seq(1.5, 2.5,l=100)
beta=seq(0.5,1.5,l=100)
# A função de verossimilhança para fazer o gráfico:
f<-function(alpha,beta){
  t=dados
  sum(log(t+beta))-n*log(2*alpha)-(n/2)*(log(2*pi*beta))-(3/2)*sum(log(t))-
    ((1/(2*alpha^2))*sum((t/beta)+(beta/t)-2))
}

f <- Vectorize(f)
z <- outer(alpha,beta, f)
persp3D(x=alpha, y=beta, z=z, contour=TRUE, facets=TRUE, curtain=F, phi=30,theta=0)
```




```
hist3D(x=alpha, y=beta, z=z, contour=TRUE, facets=TRUE, curtain=F, phi=-30,theta=30)
```




```
contour(x=alpha, y=beta, z=z,levels = pretty(c(-1780,-2000),20))
points(x=c(2,1),pch=19)
```




```
########################################
```

## Simulação de Monte Carlo

O método de Monte Carlo é um método de simulação estatística que utiliza sequencias de números aleatórios para desenvolver simulações. Em outras palavras, é visto como método numérico universal para resolver problemas por meio de amostragem aleatória.

## Possíveis problemas na estimação dos parâmetros

Alguns problemas que tive na implementação da verossimilhança e que merecem destaque:

Veja que apesar de `0.3-0.1=0.2` o R diz que isso não é verdade. Isso ocorre porque nesta diferença o R arredonda o resultado, veja abaixo que se aumentamos o número de dígitos a saída é um número próximo mas diferente de 2.

```r
0.3-0.1==0.2
```

```
## [1] FALSE
```

```r
isTRUE(0.3-0.1==0.2)
```

```
## [1] FALSE
```

```r
print(0.3-0.1,digits=17)
```

```
## [1] 0.19999999999999998
```

Outro problema possível é aparecer números muito grandes na simulação de forma que o R entende que o mesmo é infinito e a simulação para de forma brusca. Veja o menor número positivo que pode ser representado pela máquina, o maior número e outros valores importantes:

```r
.Machine
```

```
## $double.eps
## [1] 2.220446e-16
##
## $double.neg.eps
## [1] 1.110223e-16
##
## $double.xmin
## [1] 2.225074e-308
##
## $double.xmax
## [1] 1.797693e+308
##
## $double.base
## [1] 2
##
## $double.digits
## [1] 53
##
## $double.rounding
## [1] 5
##
## $double.guard
## [1] 0
##
## $double.ulp.digits
## [1] -52
##
## $double.neg.ulp.digits
## [1] -53
##
## $double.exponent
## [1] 11
##
## $double.min.exp
## [1] -1022
##
## $double.max.exp
## [1] 1024
##
## $integer.max
## [1] 2147483647
##
## $sizeof.long
## [1] 4
##
## $sizeof.longlong
## [1] 8
##
## $sizeof.longdouble
## [1] 16
##
## $sizeof.pointer
```

```
## [1] 8
```

Para contornar esse problema, podemos fazer o seguinte truque:

No exemplo específico da Birnbaum-Saunders ao definir a função log de verossimilhança mude alpha e beta para exp(lalpha) e exp(lbeta), respectivamente, onde lalpha=log(alpha) e lbeta=log(beta). Ao fazer esta mudança não altera-se em nada a função pois a substituição usa exp(log(alpha))=alpha e exp(log(beta))=beta, isso é um macete para o R não ter que calcular log de números maiores que o maior valor possível (~exp(709)).

Outra forma de resolver, caso o método de otimização funcione, mas aparece alguns valores NA, é manter alpha e beta e acrescentar na função optim o comando abaixo:

```
1-suppressWarnings(optim(start,fn=Loglik,method="BFGS",hessian=T)$par)
```

Neste caso escondemos os avisos. Os mesmos continuarão lá, porém ocultos, não é recomendado! Outra forma é caso na simulação apareça números negativos no argumento do log, podemos acrescentar o código:

```
if (any(c(t, beta, alpha, pi) < 0)) return(NA)
```

Vejamos como gerar valores aleatório com distribuição Birnbaum-Saunders e a estimação de alpha e beta usando uma simulação de Monte Carlo e as funções optim e nlimb.

```
#geração de valores de uma distribuição Birnbaum-Saunders(alpha,beta)

n<-100

#Gerando uma variável aleatória t com distribuição Birnbaum-Saunders(alpha,beta)
alpha=1
beta=2
truevalue=c(alpha,beta)
#Note que teremos N amostras de tamanho 100 distintas, portanto t
#deve estar dentro do loop do Monte Carlo, alpha e beta deve estar fora do
#Loop pois os mesmos são fixos para todas as amostras.
N=1000
 m=matrix(nrow=N,ncol=2)
m1=matrix(nrow=N,ncol=2)
for(i in 1:N){
z<-rnorm(n,0,1)
#Gerando uma variável aleatória t com distribuição Birnbaum-Saunders(alpha,beta)
t<-cbind(beta*((alpha*z/2)+sqrt((alpha*z/2)^2+1))^2)
#Verossimilhança
Loglik<-function(par,dados){
    lalpha=par[1]
    lbeta=par[2]
    t<-dados
    ll<-sum(log(t+exp(lbeta)))-n*log(2*exp(lalpha))-(n/2)*(log(2*pi*exp(lbeta)))-(3/2)*
      sum(log(t))-((1/(2*exp(lalpha)^2))*sum((t/exp(lbeta))+(exp(lbeta)/t)-2))
    return(-ll)
}

#Utilizando a verossimilhança e a função optim para estimar os parâmetros alpha e beta que
#deram origem as observações t observadas. Note que foi utilizado o log de alpha e
#beta como chutes iniciais.

lalpha_0=log(2)
lbeta_0=log(2)
start=c(lalpha_0,lbeta_0)
```

```r
m[i,]=exp(optim(start,fn=Loglik,method="BFGS",dados=t,hessian=T)$par)
m1[i,]=exp(nlminb(start,Loglik,dados=t)$par)


}

#Calculating the average of each column of the array of parameters m
mest=colMeans(m)
mest1=colMeans(m1)

#calculating the standard deviation of each column of the array of parameters m
dest=apply(m,2,sd)
dest1=apply(m1,2,sd)

#root mean square error in the calculation of each column of the array of parameters m in
#relation to the true value of the parameter
eqm=function(x,poisson_opt){
k=length(x)
sqrt(sum(((x-poisson_opt)^2))/k)}

eqm1=function(x,poisson_nlm){
k=length(x)
sqrt(sum(((x-poisson_nlm)^2))/k)}

#Estimated mean squared error of each parameter
eqmest=c(eqm(x=m[,1],poisson_opt=truevalue[1]),
         eqm(x=m[,2],poisson_opt=truevalue[2]))

#Estimated mean squared error of each parameter
eqmest1=c(eqm1(x=m1[,1],poisson_nlm=truevalue[1]),
          eqm1(x=m1[,2],poisson_nlm=truevalue[2]))

# Table with the true values of the parameters and the average
# Standard deviation and mean square error of the estimated parameters
tab=data.frame(truevalue,mean=mest,sd=dest,eqm=eqmest)
tab1=data.frame(truevalue,mean=mest1,sd=dest1,eqm=eqmest1)

tab
```

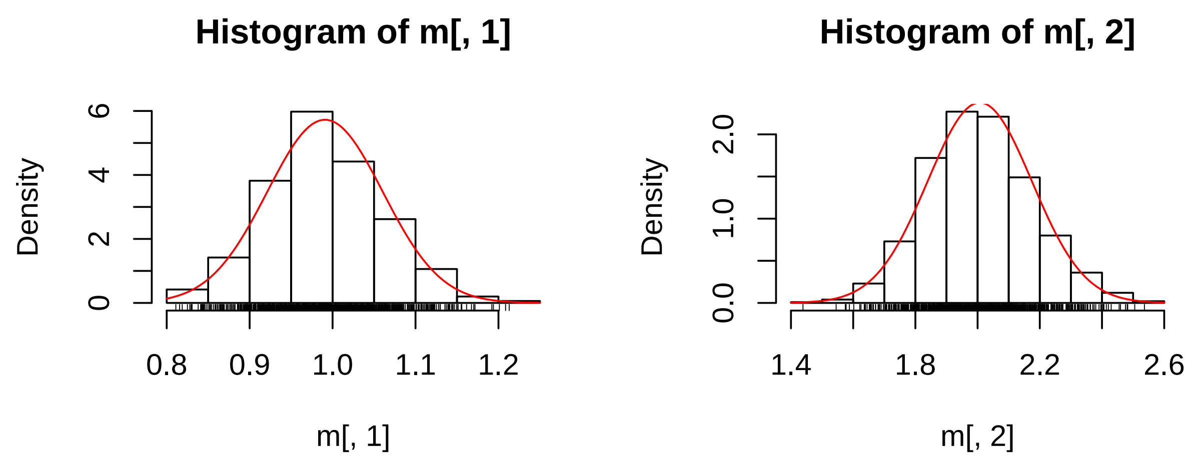
```
##   truevalue      mean        sd        eqm
## 1         1 0.9908571 0.0696564 0.07021934
## 2         2 2.0069399 0.1677734 0.16783301
```

```r
tab1
```

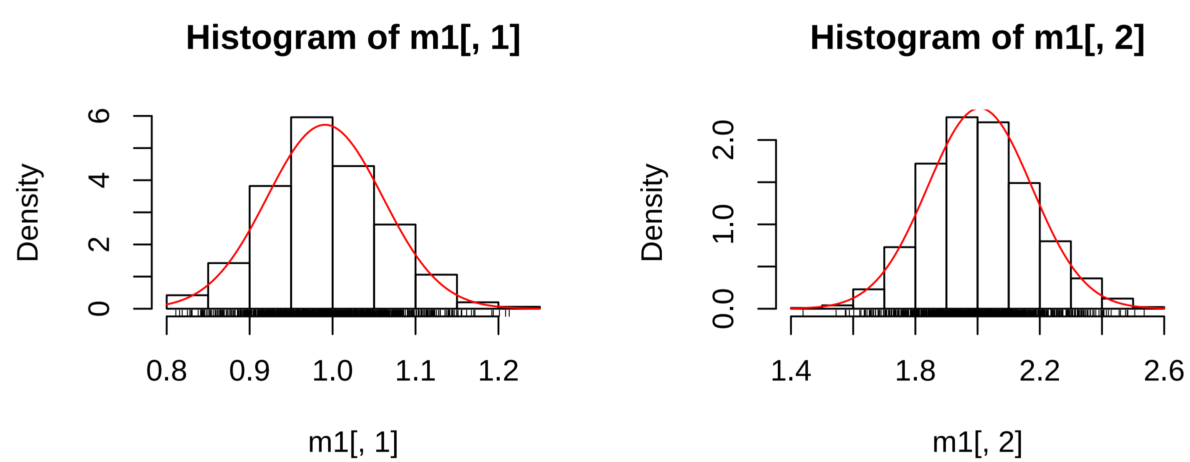
```
##   truevalue      mean         sd        eqm
## 1         1 0.9908604 0.06967077 0.07023314
## 2         2 2.0069489 0.16778293 0.16784292
```

```r
par(mfrow=c(1,2))
hist(m[,1],prob=T);
rug(m[,1])
curve(expr = dnorm(x,mean=mean(m[,1]),sd=sd(m[,1])),add=T, col="red")
hist(m[,2],prob=T);
rug(m[,2])
```

```r
curve(expr = dnorm(x,mean=mean(m[,2]),sd=sd(m[,2])),add=T, col="red")
```




```r
hist(m1[,1],prob=T);
rug(m1[,1])
curve(expr = dnorm(x,mean=mean(m1[,1]),sd=sd(m1[,1])),add=T, col="red")
hist(m1[,2],prob=T);
rug(m1[,2])
curve(expr = dnorm(x,mean=mean(m1[,2]),sd=sd(m1[,2])),add=T, col="red")
```




### Uma Reparametrização Importante

As vezes, reparametrizações são essenciais, pois facilitam o desenvolvimento analítico de algumas distribuições e também podem melhorar a eficiência em simulações, em determinadas situações, como em regressão, quando a distribuição da variável resposta não possui a média como um de seus parâmetros podemos proceder uma reparametrização de forma a atender essa condição e poder ajustar a média da variável resposta. Exemplos de distribuições em que são realizadas reparametrizações com sucesso são: distribuição beta (ver Ferrari e Cribari-Neto 2004) e distribuição gaussiana inversa (ver Tweedie 1957).

Seja $\mu = \beta(1 + \frac{\alpha^2}{2})$ e $\phi = \frac{2}{\alpha^2}$. Então,

$$\alpha = \sqrt{\frac{2}{\phi}} \text{ e } \beta = \frac{\mu}{(1 + \frac{1}{\phi})}. \tag{4}$$

A fda da $BS(\mu, \phi)$ é obtida substituindo os valores de $\alpha$ e $\beta$ definidos em (4) na expressão (1). De onde, temos:

$$F(t; \mu, \phi) = P(T \leq t) = \Phi\left[\sqrt{\frac{\phi}{2}}\left(\sqrt{\frac{(\phi+1)t}{\phi\mu}} - \sqrt{\frac{\phi\mu}{(\phi+1)t}}\right)\right], \tag{5}$$

onde $\phi > 0,\ \mu > 0,\ t > 0$.

Segue que a fdp é dada por:

$$f(t; \phi, \mu) = \frac{exp(\frac{\phi}{2})\sqrt{\phi+1}}{4\sqrt{\pi\mu}} t^{-\frac{3}{2}}\left[t + \frac{\phi\mu}{\phi+1}\right] exp\left\{-\frac{\phi}{4}\left(\frac{t(\phi+1)}{\phi\mu} + \frac{\phi\mu}{t(\phi+1)}\right)\right\}$$

A nova média e variância são:

$$E(T) = \mu,\ \text{e } Var(T) = \frac{g(\mu)}{h(\phi)} \tag{6}$$

A distribuição $BS(\mu, \phi)$ satisfaz a propriedade de escala e também satisfaz a propriedade recíproca.

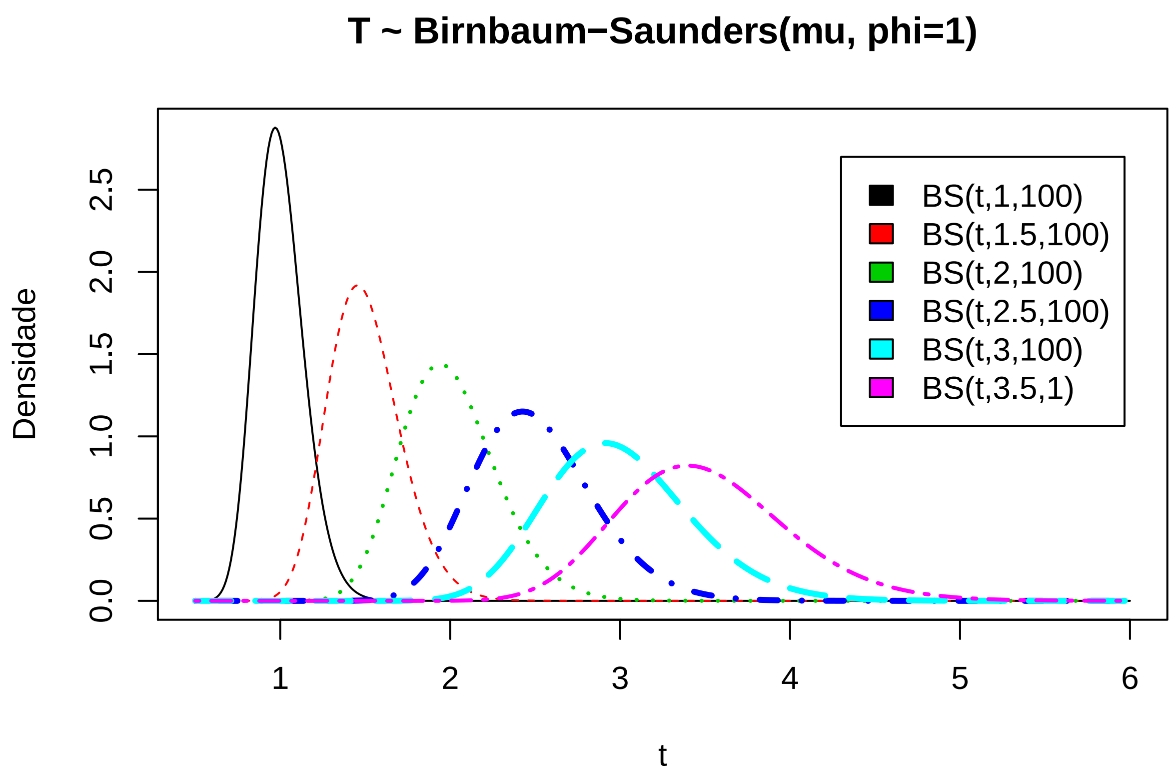
```
bs<-function(t,mu,phi){
  fdp=((exp(phi/2)*sqrt(phi+1))/(4*sqrt(pi*mu)*t^(3/2)))*(t+((phi*mu)/(phi+1)))*
    exp((-phi/4)*((t*(phi+1)/(phi*mu))+(phi*mu/(t*(phi+1)))))
  return(fdp)
}

#Teste para ver se a integral da densidade é igual a 1.
mu=1
phi=2
integrand <- function(t){((exp(phi/2)*sqrt(phi+1))/(4*sqrt(pi*mu)*t^(3/2)))*
    (t+((phi*mu)/(phi+1)))*exp((-phi/4)*((t*(phi+1)/(phi*mu))+(phi*mu/(t*(phi+1)))))}
integrate(integrand, lower = 0, upper = Inf)
```

```
## 1 with absolute error < 7.8e-05
```

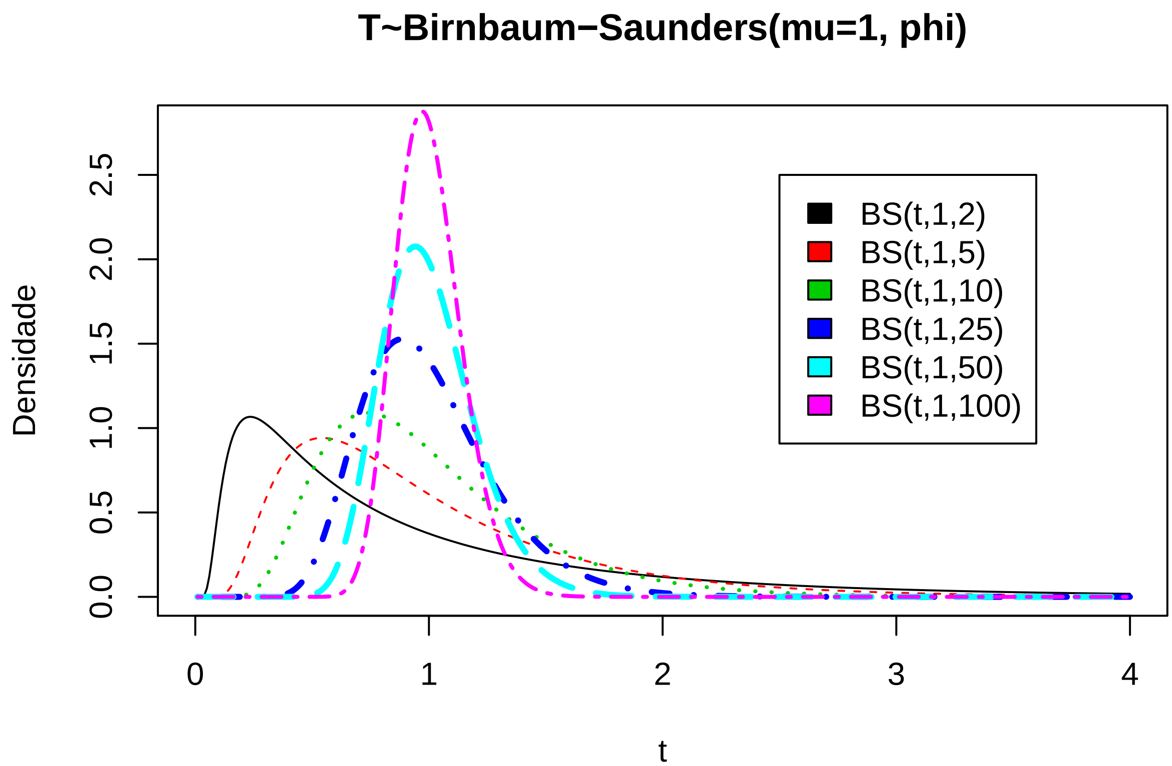
Note que com a mudança de mu, mantendo phi fixo, há uma alteração curtose da variável aleatória e com o aumento de beta há um aumento da média e da variança, ou seja, mudamos a assimetria do gráfico.

```
#
t=seq(0.5,6,by=0.01)
plot(t,bs(t,1,100),main='T ~ Birnbaum-Saunders(mu, phi=1)',ylab='Densidade',type='l')
lines(t,bs(t,1.5,100),col=2,lty=2, lwd=1)
lines(t,bs(t,2,100),col=3,lty=3, lwd=2)
lines(t,bs(t,2.5,100),col=4,lty=4, lwd=3)
lines(t,bs(t,3,100),col=5,lty=5, lwd=3)
lines(t,bs(t,3.5,100),col=6,lty=6, lwd=2)
```

```r
legend(4.3, 2.7, c("BS(t,1,100)","BS(t,1.5,100)","BS(t,2,100)","BS(t,2.5,100)",
                   "BS(t,3,100)","BS(t,3.5,1)"), fill=1:6)
```

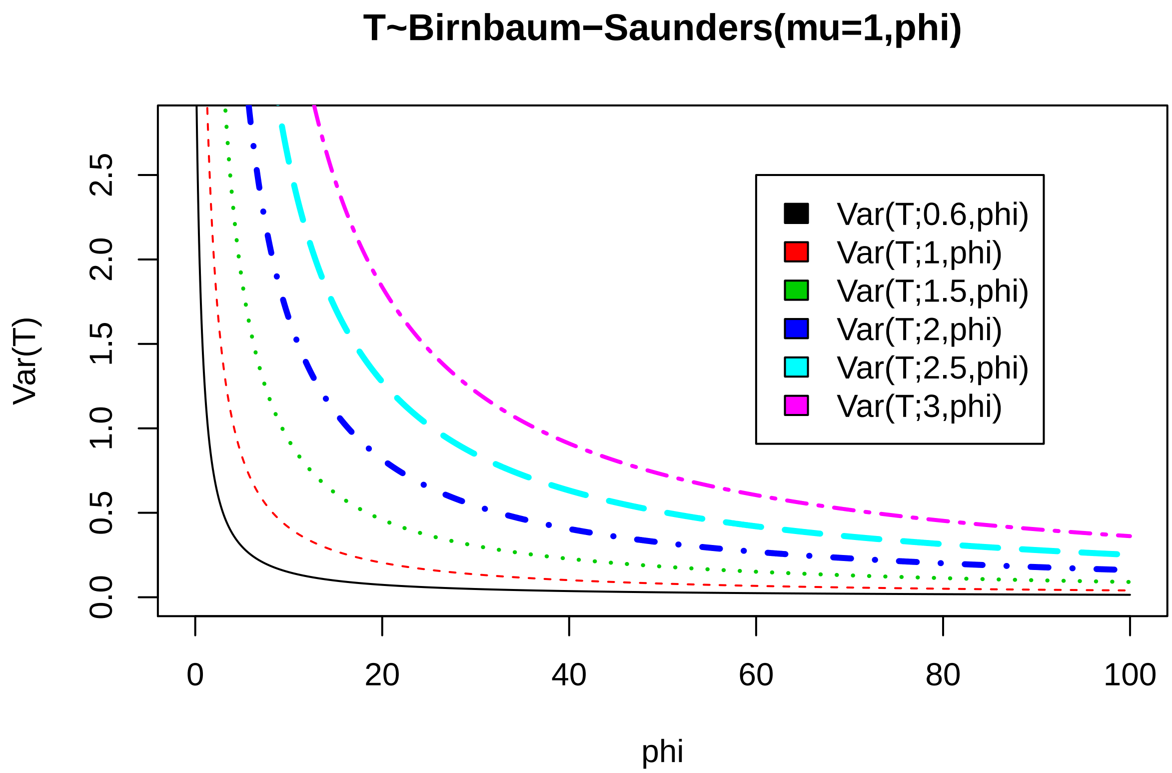



```r
t=seq(0,4,by=0.01)
plot(t,bs(t,1,2),ylim=c(0,2.8),main='T~Birnbaum-Saunders(mu=1, phi)',ylab='Densidade',type='l')
lines(t,bs(t,1,5),col=2,lty=2, lwd=1)
lines(t,bs(t,1,10),col=3,lty=3, lwd=2)
lines(t,bs(t,1,25),col=4,lty=4, lwd=3)
lines(t,bs(t,1,50),col=5,lty=5, lwd=3)
lines(t,bs(t,1,100),col=6,lty=6, lwd=2)
legend(2.5, 2.5,
       c("BS(t,1,2)","BS(t,1,5)","BS(t,1,10)","BS(t,1,25)",
         "BS(t,1,50)","BS(t,1,100)"), fill=1:6)
```




Ao manter mu fixo e aumentar phi a variância de T (Var(T)) tende a zero.

```
V=function(mu,phi){
  vt=2*(mu^2)*(2*phi+5)/((phi+1)^2)
  return(vt)
}

#
phi=seq(0.001,100,by=0.01)
plot(phi,V(0.6,phi),ylim=c(0,2.8),main='T~Birnbaum-Saunders(mu=1,phi)',ylab='Var(T)',type='l')
lines(phi,V(1,phi),col=2,lty=2, lwd=1)
lines(phi,V(1.5,phi),col=3,lty=3, lwd=2)
lines(phi,V(2,phi),col=4,lty=4, lwd=3)
lines(phi,V(2.5,phi),col=5,lty=5, lwd=3)
lines(phi,V(3,phi),col=6,lty=6, lwd=2)
legend(60, 2.5, c("Var(T;0.6,phi)","Var(T;1,phi)","Var(T;1.5,phi)",
                  "Var(T;2,phi)","Var(T;2.5,phi)","Var(T;3,phi)"), fill=1:6)
```




### Função Log de Verossimilhança da $BS(\mu, \phi)$

$$l(\mu,\phi;\mathbf{T}) = \frac{n\phi}{2}+\frac{n}{2}\log\left(\phi+1\right)-\frac{3}{2}\sum_{i=1}^{n}\log t_i - n\log\left(4\sqrt{\pi\mu}\right)+\sum_{i=1}^{n}\log\left[t_i+\frac{\phi\mu}{\phi+1}\right]-\frac{\phi}{4}\sum_{i=1}^{n}\left[\frac{t_i(\phi+1)}{\phi\mu}+\frac{\phi\mu}{t_i(\phi+1)}\right]$$

Os estimadores (ou estimativas) de máxima verossimilhança de $\mu$ e $\phi$ são obtidos maximizando essa função, a partir da solução das equações formadas com as derivadas parciais em relação $\mu$ e $\phi$. É possível mostrar que não é possível obter uma solução analítica para os estimadores de máxima verossimilhança e, portanto, métodos iterativos de otimização são utilizados.

Criou-se agora uma estrutura de regressão para a média da distribuição $BS(\alpha, \beta)$ fazendo $g(\mu) = \beta_0 + \beta_1 * X$

```
rm(list=ls())
cat("\014")
```

```
N=1000
#m e m1 são as matrizes que receberão as estimativas no final do processo de estimação
m=matrix(ncol=2,nrow=N)
m1=matrix(ncol=2,nrow=N)
#Valores iniciais dos parâmetros usados para gerar t
#Ou seja, Valor verdadeiro dos parâmetros
beta0=2
beta1=1
truevalue=c(beta0,beta1)
#Tamanho das amostras
n=100
```

```r
#Vetor de parâmetros
beta=matrix(c(beta0,beta1),nrow=2,ncol=1)
#Vetor de 1's
const1 <- rep(1,n);
const <- cbind(const1);
#Vetor de cováriavel com distribuição unif(0,1)
X1=matrix(runif(n, 0, 1),nrow=n,ncol=1)
#Matriz de covariáveis
X <-matrix(c(const,X1),nrow=n,ncol=ncol(X1)+1)
#Número de colunas de X
p=ncol(X)
#Vetor de médias
mu=exp(X%*%beta)

#Gerando uma variável aleatória t com distribuição Birnbaum-Saunders(mu,phi)
phi=2
remove(beta,beta0,beta1)

#Monte Carlo para estimação dos parâmetros beta0, beta1 e beta2

for (i in 1:N){

z<-cbind(rnorm(n,0,1))
t<-((phi*mu)/(phi+1))*((z/sqrt(2*phi))+(sqrt((z/sqrt(2*phi))^2+1)))^2

Loglik<-function(beta,dados){
 p=ncol(X)
 mu=exp(X%*%beta[1:p])
 phi=phi
 lv=sum((phi/2)-(3/2)*log(t)+(1/2)*log(phi+1)-log(4*sqrt(pi*mu))+log(t+(phi*mu/(phi+1)))-
          (phi/4)*((t*(phi+1)/(phi*mu))+(phi*mu/(t*(phi+1)))))
return(-lv)
}
#Chute inicial para as funções de estimação
start=c(10,20)

#Estimation with function optim
bs_op=optim(start,Loglik,method="BFGS",dados=t,hessian = T)
m[i,]=bs_op$par
bs_nl=nlminb(start, Loglik)
m1[i,]=bs_nl$par
}
mest=colMeans(m)
mest1=colMeans(m1)

#calculating the standard deviation of each column of the array of parameters m
dest=apply(m,2,sd)
dest1=apply(m1,2,sd)

#root mean square error in the calculation of each column of the array of parameters m in
#relation to the true value of the parameter
eqm=function(x,bs_op){
  k=length(x)
```

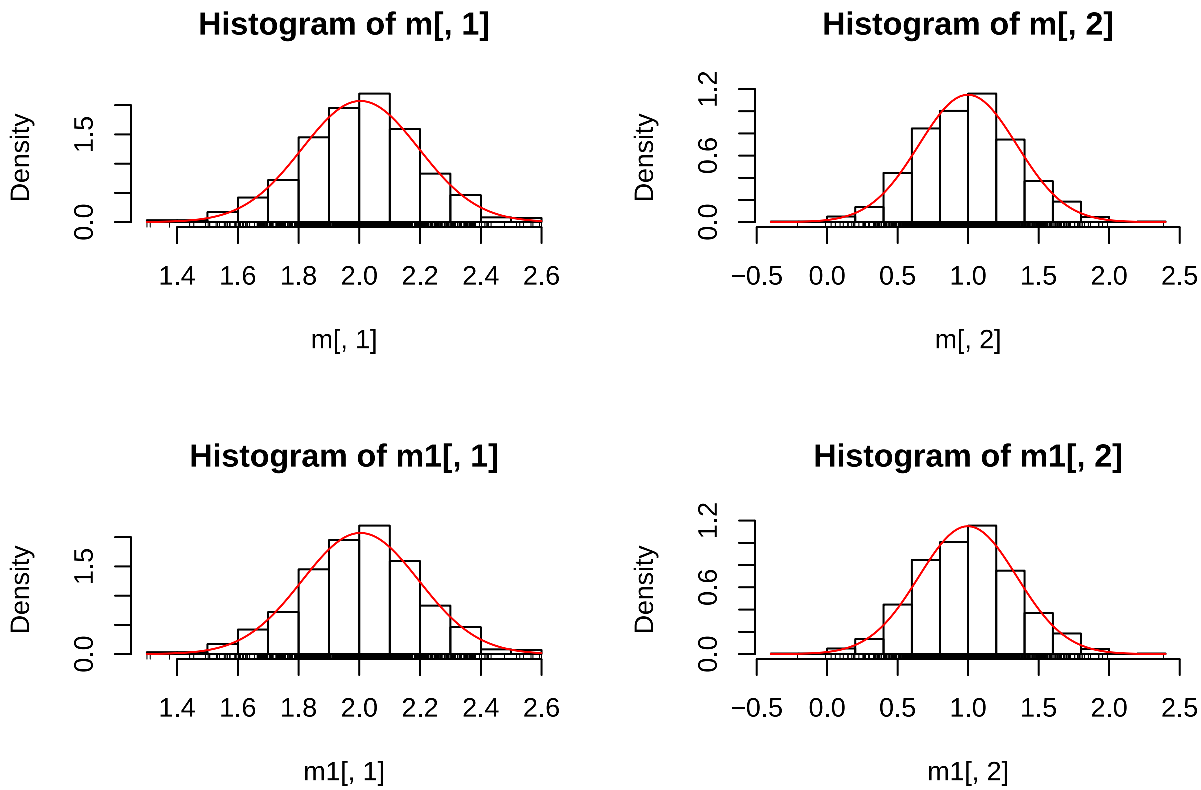
```r
  sqrt(sum(((x-bs_op)^2))/k)}

eqm1=function(x,bs_nl){
  k=length(x)
  sqrt(sum(((x-bs_nl)^2))/k)}

#Estimated mean squared error of each parameter

eqmest=c(eqm(x=m[,1],bs_op=truevalue[1]),
         eqm(x=m[,2],bs_op=truevalue[2]))

#Estimated mean squared error of each parameter
eqmest1=c(eqm1(x=m1[,1],bs_nl=truevalue[1]),
          eqm1(x=m1[,2],bs_nl=truevalue[2]))


# Table with the true values of the parameters and the average
# Standard deviation and mean square error of the estimated parameters
tab=data.frame(truevalue,mean=mest,sd=dest,eqm=eqmest)
tab1=data.frame(truevalue,mean=mest1,sd=dest1,eqm=eqmest1)

tab
```

```
##   truevalue      mean        sd       eqm
## 1         2 2.0038803 0.1923609 0.1923039
## 2         1 0.9980659 0.3470936 0.3469254
```

```r
tab1
```

```
##   truevalue      mean        sd       eqm
## 1         2 2.0038928 0.1923613 0.1923045
## 2         1 0.9980388 0.3470922 0.3469241
```

```r
par(mfrow=c(2,2))
hist(m[,1],prob=T);
rug(m[,1])
curve(expr = dnorm(x,mean=mean(m[,1]),sd=sd(m[,1])),add=T, col="red")
hist(m[,2],prob=T);
rug(m[,2])
curve(expr = dnorm(x,mean=mean(m[,2]),sd=sd(m[,2])),add=T, col="red")
hist(m1[,1],prob=T);
rug(m1[,1])
curve(expr = dnorm(x,mean=mean(m1[,1]),sd=sd(m1[,1])),add=T, col="red")
hist(m1[,2],prob=T);
rug(m1[,2])
curve(expr = dnorm(x,mean=mean(m1[,2]),sd=sd(m1[,2])),add=T, col="red")
```




## Referências

1-Z.W. Birnbaum & S.C. Saunders. A new family of life distribuitions. Journal of Applied Probability, 6 (1969), 319-327.