Distribuição Gamma

Bruno Normande

4 de Janeiro de 2014

1 Introdução

Esse trabalho faz uma análise do comportamento de três estimadores para distribuição Gamma, quando estimados usando o método de Monte Carlo comparando-os com suas versões usando *bootstrap*.

Os estimadores usados nesse trabalho foram:

- Estimador por máxima verossimilhança;
- Estimador pelo primeiro momento;
- Estimador pelo segundo momento central.

2 Características da Distribuição Gamma

A distribuíção Gamma é uma distribuíção que é caracterizda por dois parâmentros, *shape* (k) e *scale* (θ). Ela possui a seguinte função densidade probabilidade:

$$f(w;k,\theta) = \frac{w^{k-1}e^{\frac{w}{\theta}}}{\theta^k\Gamma(w)} \tag{1}$$

para

$$k,\theta > 0 \tag{2}$$

Essa distribuição é usada muitas vezes para modelar o tempo de espera, como por exemplo em teste de vida a distribuição Gamma é usada para modelar o tempo até a morte. A figura 1 mostra o comportamento de Gamma para diferentes valores de k e θ.

Obtemos a esperança e a variância da distribuição com as seguintes fórmulas:

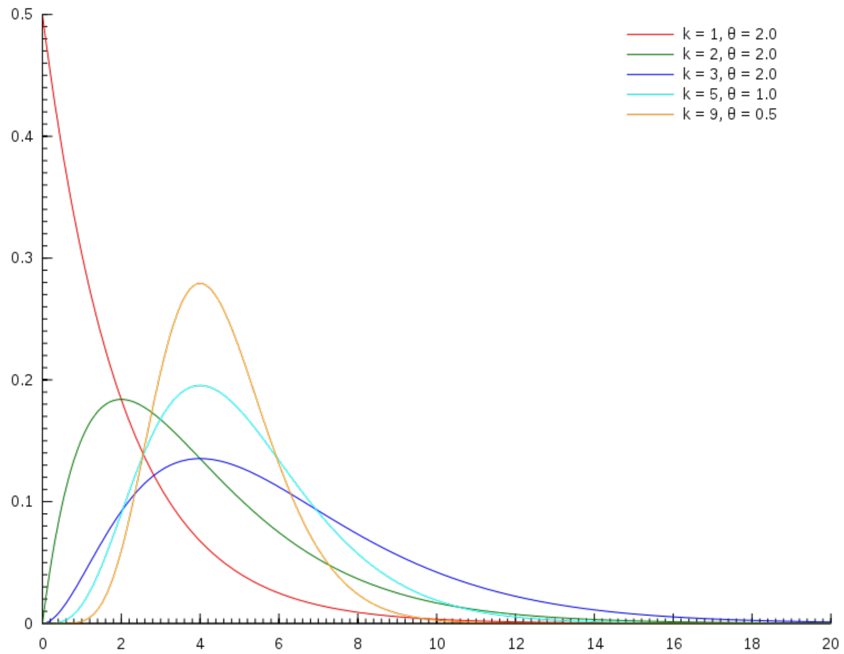

Figura 1: Função probabilidade Densidade de Gamma para diferentes parâmetros (imagem retirada de `http://en.wikipedia.org/wiki/Gamma_distribution`)

$$E[W] = k\theta \tag{3}$$
$$Var[W] = k\theta^2 \tag{4}$$

3 Estimadores

Nesse trabalho foram analisados 6 estimadores para a distribuição Gamma. Em todos foi usado $\theta = 1$ para manter a simplicidade dos teste. Os estimadores usados foram:

3.1 Estimador por Máxima Verossimilhança

Para estimar pela Máxima Verossimilhança é preciso maximizar a sua função log-verossimilhança de $\Gamma(w; k, 1)$

$$log(p(W|k,1)) = n(k-1)\overline{log(x)} - nlog(\Gamma(k)) - nklog(\overline{x}) + nklog(a) - nk \tag{5}$$

Que podemos resolver numericamente iterando sobre k em:

$$\frac{1}{k_n} = \frac{1}{k_{n-1}} + \frac{\overline{log(x)} - log(\overline{x}) + log(k_{n-1}) - \psi(k_{n-1})}{k_{n-1}^2(\frac{1}{k_{n-1}} - \psi'(k_{n-1}))} \tag{6}$$

até o momento em que

$$k_n \approx k_{n-1} \tag{7}$$

Como k inicial podemos usar a seguinte aproximação

$$\hat{k_0} = \frac{0.5}{log(\overline{x}) - \overline{log(x)}} \tag{8}$$

Nesse trabalho usaremos a notação $\hat{k}_0$ para o estimador por máxima verossimilhança.

3.2 Estimador pelo Primeiro Momento

Usando o método dos momentos podemos estimar k a partir do primeiro momento da seguinte maneira:

$$E[W] = k\theta \tag{9}$$

$$\hat{k} = \frac{1}{n}\sum_{i=1}^{n} w_i \tag{10}$$

Nesse trabalho usaremos a notação $\hat{k}_1$ para o estimador pelo primeiro momento.

3.3 Estimador pelo Segundo Momento Central

De maneira similar podemos estimar k a partir do segundo momento central da seguinte maneira:

$$Var[W] = k\theta^2 \tag{11}$$

$$\hat{k} = Var(w) \tag{12}$$

Nesse trabalho usaremos a notação $\hat{k}_2$ para o estimador pelo segundo momento central.

3.4 Estimadores com *bootstrap*

Todos os estimadores mencionados àcima foram testados também contra suas versões com *bootstrap*. Dessa maneira foi possível observar se usando o método *bootstrap* poderiamos diminuir o viés desses estimadores.

Como notação para identificar estes estimadores foi usado um til no lugar do chápeu:

$$\tilde{k}_0, \tilde{k}_1 \ e \ \tilde{k}_1^2$$

4 Resultados

As tabelas a seguir comparam os estimadores usados com suas versões *bootstraped*.

Comparação dos Estimadores $\hat{k}$ e $\tilde{k}$			
n	k	$\lvert B(\hat{k})\rvert > \lvert B(\tilde{k})\rvert$	$EQM(\hat{k}) > EQM(\tilde{k})$
100	1	TRUE	TRUE
1000	1	FALSE	FALSE
10000	1	FALSE	TRUE
100000	1	FALSE	FALSE
100	2	TRUE	TRUE
1000	2	TRUE	FALSE
10000	2	FALSE	FALSE
100000	2	FALSE	FALSE
100	3	TRUE	TRUE
1000	3	FALSE	FALSE
10000	3	FALSE	FALSE
100000	3	TRUE	FALSE
100	5	TRUE	TRUE
1000	5	FALSE	FALSE
10000	5	FALSE	TRUE
100000	5	FALSE	TRUE
100	9	FALSE	TRUE
1000	9	TRUE	TRUE
10000	9	FALSE	FALSE
100000	9	TRUE	TRUE

Tabela 1: Estimadores de máxima verossimilhança $\hat{k}$ e $\tilde{k}$.

		Comparação dos Estimadores $\hat{k}_1$ e $\tilde{k}_1$	
n	k	$\|B(\hat{k}_1)\| > \|B(\tilde{k}_1)\|$	$EQM(\hat{k}_1) > EQM(\tilde{k}_1)$
100	1	FALSE	FALSE
1000	1	FALSE	FALSE
10000	1	TRUE	FALSE
100000	1	FALSE	FALSE
100	2	FALSE	FALSE
1000	2	TRUE	FALSE
10000	2	FALSE	FALSE
100000	2	FALSE	FALSE
100	3	FALSE	FALSE
1000	3	FALSE	FALSE
10000	3	FALSE	FALSE
100000	3	FALSE	FALSE
100	5	FALSE	FALSE
1000	5	FALSE	FALSE
10000	5	FALSE	FALSE
100000	5	FALSE	FALSE
100	9	FALSE	FALSE
1000	9	FALSE	FALSE
10000	9	TRUE	FALSE
100000	9	FALSE	FALSE

Tabela 2: Estimadores de primeiro momento $\hat{k}_1$ e $\tilde{k}_1$.

		Comparação dos Estimadores $\hat{k}_2^0$ e $\tilde{k}_2^0$	
n	k	$\lvert B(\hat{k}_2^0)\rvert > \lvert B(\tilde{k}_2^0)\rvert$	$EQM(\hat{k}_2^0) > EQM(\tilde{k}_2^0)$
100	1	FALSE	FALSE
1000	1	TRUE	FALSE
10000	1	FALSE	FALSE
100000	1	FALSE	FALSE
100	2	FALSE	FALSE
1000	2	TRUE	FALSE
10000	2	FALSE	FALSE
100000	2	FALSE	FALSE
100	3	FALSE	FALSE
1000	3	TRUE	FALSE
10000	3	FALSE	FALSE
100000	3	FALSE	FALSE
100	5	FALSE	FALSE
1000	5	FALSE	FALSE
10000	5	FALSE	FALSE
100000	5	FALSE	FALSE
100	9	TRUE	FALSE
1000	9	FALSE	FALSE
10000	9	TRUE	FALSE
100000	9	FALSE	FALSE

Tabela 3: Estimadores de segundo momento central $\hat{k}_2^0$ e $\tilde{k}_2^0$.

5 Conclusão

Analisando os resultados podemos ver que para a distribuição Gamma o método *bootstrap* não melhorou o viés dos estimadores o que nos leva à conclusão de que esse método não deve ser usado para essa distribuição.

Para confirmar essa conclusão futuros estudos devem ser feitos usando essa distribuição e mais valores de n e k.

A Código em R Usados no Trabalho

Estimadores.

```r
## Esse arquivo contém os estimadores para gamma

# Estimador pelo primeiro momento
# Primeiro momento = E[X] = mean(x)
# E[X] = shape/rate , rate = 1
# shape_hat = mean(x)
first_moment = function(x, d)
{
  return(mean(x[d]))
}

## Estimador pela segundo momento central
# var[X] = shape/rate^2
# rate = 1
# shape_hat = var(x)
second_cent_moment = function(x, d)
{
  return(var(x[d]))
}

# temos que maximizar:
# log(p(D|a,^b)) = n*(a-1)*mean(log(x)) - n*log(G(a)) - n*a*log(mean(x))+ n*a*log(a)-n*a
# log(p(D|a,^b)) >= n*(a-1)*mean(log(x)) - n*log(G(a)) - n*a*log(mean(x))
#                                        + n*(1 + log(a_0))*(a - a_0) + n*a_0*log(a_0) - n*a
# Usando generalized Newton fazer a seguinte aproximação
# log(p(D|a,^b)) aproxx c0 + c1*a + c2*log(a)
# 1/a = 1/a_0 + (mean(log(x)) - log(mean(x)) + log(a_0) - digamma(a_0)) /
#                                                                     a_0^2*(1/a_0 - trigamma(a_0))
max_ver = function(x, d){
  EPSILON = 0.00001
  dist_sample = x[d]
  # inicializando a
  log_meanx = log(mean(dist_sample))
  mean_logx = mean(log(dist_sample))
  a_hat = 0.5/(log_meanx - mean_logx)

  while(digamma(a_hat) != mean_logx - log_meanx + log(a_hat)){
    aux = mean_logx - log_meanx + log(a_hat) - digamma(a_hat)
    aux = aux/(a_hat^2 * (1/a_hat - trigamma(a_hat)))
    aux = aux + 1/a_hat
```

```r
    aux = 1/aux
    if(abs(a_hat - aux) < EPSILON){
      break
    }
    a_hat = aux
  }

  return(a_hat)
}
```

gamma_estimators.R

Script de teste.

```r
## Essaio Monte Carlo
#   Distribuição Gamma
#   author: Bruno Normande

library(lattice)
library(boot)
source("gamma_estimators.R")

R = 100
N = c(100, 1000, 10000, 100000)
K = c(1, 2, 3, 5, 9)
BOOT_S = 200

nrows = R*length(K)*length(N)
each_estim = R*length(K)*length(N)
nrows_bias_eqm = length(N)*length(K)

estimatives = data.frame(N = rep(0,nrows),
                         K = rep(0,nrows),
                         k_hat = rep(0, nrows),
                         k_til = rep(0, nrows))
bias_eqm_1 = data.frame(N = rep(0, nrows_bias_eqm),
                        K = rep(0, nrows_bias_eqm),
                        Bias = rep(0, nrows_bias_eqm),
                        EQM = rep(0, nrows_bias_eqm),
                        Bias_til = rep(0, nrows_bias_eqm),
                        EQM_til = rep(0, nrows_bias_eqm))

estimatives2 = data.frame(N = rep(0,nrows),
                          K = rep(0,nrows),
                          k_hat = rep(0, nrows),
                          k_til = rep(0, nrows))
bias_eqm_2 = data.frame(N = rep(0, nrows_bias_eqm),
                        K = rep(0, nrows_bias_eqm),
                        Bias = rep(0, nrows_bias_eqm),
                        EQM = rep(0, nrows_bias_eqm),
```

```
                          Bias_til = rep(0, nrows_bias_eqm),
                          EQM_til = rep(0, nrows_bias_eqm))

estimatives3 = data.frame(N = rep(0,nrows),
                          K = rep(0,nrows),
                          k_hat = rep(0, nrows),
                          k_til = rep(0, nrows))
bias_eqm_3 = data.frame(N = rep(0, nrows_bias_eqm),
                          K = rep(0, nrows_bias_eqm),
                          Bias = rep(0, nrows_bias_eqm),
                          EQM = rep(0, nrows_bias_eqm),
                          Bias_til = rep(0, nrows_bias_eqm),
                          EQM_til = rep(0, nrows_bias_eqm))

i = 1
count_eq = 0
count_dif = 0
# para cada shape k
for(k in 1:length(K)){
  # para cada tamanho da amostragem
  for(n in 1:length(N)){
    print(sprintf("Shape: %d N: %d", K[k], N[n]))
    for(r in 1:R){
      dist = rgamma(N[n], K[k]) # rate = scale = 1 por default
      k_hat = max_ver(dist, 1:N[n])
      b = boot(dist, max_ver, BOOT_S)
      k_til = 2*k_hat - mean(b$t[,1])

      k_hat2 = first_moment(dist, 1:N[n])
      b = boot(dist, first_moment, BOOT_S)
      k_til2 = 2*k_hat - mean(b$t[,1])

      k_hat3 = second_cent_moment(dist, 1:N[n])
      b = boot(dist, second_cent_moment, BOOT_S)
      k_til3 = 2*k_hat - mean(b$t[,1])

      estimatives[i,] = c(N[n],K[k], k_hat, k_til)
      estimatives2[i,] = c(N[n],K[k], k_hat2, k_til2)
      estimatives3[i,] = c(N[n],K[k], k_hat3, k_til3)

      i = i + 1
    }
  }
}

get_bias_eqm = function(est, k)
{
  bias = mean(est) - k
  EQM = bias^2 + var(est)
```

```
  return(c(bias,EQM))
}

counter = 1
# Retirando Viés e EQM
# para cada shape k
for(k in 1:length(K))
{
  # para cada tamanho da amostragem
  the_k = K[k]
  for(n in 1:length(N))
  {
    the_n = N[n]
    print(sprintf("%d -- %d", the_k, the_n))
    # para k_hat
    est = estimatives[estimatives$N == the_n & estimatives$K == the_k
    ,]
    bias_eqm = get_bias_eqm(est$k_hat, the_k)
    bias_eqm_til = get_bias_eqm(est$k_til, the_k)
    bias_eqm_1[counter,] = c(the_n, the_k, bias_eqm, bias_eqm_til)

    # para k_hat2
    est = estimatives2[estimatives2$N == the_n & estimatives2$K ==
    the_k,]
    bias_eqm = get_bias_eqm(est$k_hat, the_k)
    bias_eqm_til = get_bias_eqm(est$k_til, the_k)
    bias_eqm_2[counter,] = c(the_n, the_k, bias_eqm, bias_eqm_til)

    # para k_hat3
    est = estimatives3[estimatives3$N == the_n & estimatives3$K ==
    the_k,]
    bias_eqm = get_bias_eqm(est$k_hat, the_k)
    bias_eqm_til = get_bias_eqm(est$k_til, the_k)
    bias_eqm_3[counter,] = c(the_n, the_k, bias_eqm, bias_eqm_til)

    counter = counter + 1
  }
}
pdf("k_hat.pdf")
bwplot( k_hat ~ N | K, data=estimatives, horizontal = FALSE)
dev.off()
pdf("k_til.pdf")
bwplot( k_til ~ N | K, data=estimatives, horizontal = FALSE)
dev.off()

pdf("k1_hat.pdf")
bwplot( k_hat ~ N | K, data=estimatives2, horizontal = FALSE)
dev.off()
pdf("k1_til.pdf")
```

```r
bwplot( k_til ~ N | K, data=estimatives2 , horizontal = FALSE)
dev.off()

pdf("k2_hat.pdf")
bwplot( k_hat ~ N | K, data=estimatives3 , horizontal = FALSE)
dev.off()
pdf("k2_til.pdf")
bwplot( k_til ~ N | K, data=estimatives3 , horizontal = FALSE)
dev.off()

get_row = function(x)
{
  return(sprintf("%d & %d & %s & %s\\", x[1], x[2],
                  (abs(x[3]) > abs(x[5])),
                   (x[4] > x[6])))
}

print("Maxima Verossimilhança")
table_rows = apply(bias_eqm_1,1,get_row)
for(r in table_rows){print(r)}

print("Primeiro Momento")
table_rows = apply(bias_eqm_2,1,get_row)
for(r in table_rows){print(r)}

print("Segundo Momento Central")
table_rows = apply(bias_eqm_3,1,get_row)
for(r in table_rows){print(r)}
```

ensaio_montecarlo.R