eman ta zabal zazu

Universidad del País Vasco　Euskal Herriko Unibertsitatea

Activity 2

1 Synopsis.

What this session is about. In Seminar 1 and subsequent Activity 1 you learnt how to estimate things using the Monte Carlo method. In Seminar 2 we saw how to compute maximum likelihood estimates even in cases when direct solution of the likelihood equations is not possible by hand. You now will use both techniques to examine relative performance of moment and maximum likelihood estimators.

You will see on a simple example that, as expected from the theory, MLE estimators outperform moment estimators, at least for large sample size. For small sample sizes, this may not be the case.

What you need. You need to be fully acquainted with the content of previous seminars and practice assignment. You will also need access to a computer equiped with R. Depending on how ambitious you are, your simulations may take a long time to run: start small, and only try a substantial number of replcations when your code appears to be right.

2 Background

The beta distribution depends on two parameters, that we will name r and s. It is denoted by $B(r,s)$. The density function is

$$f_X(x;r,s) = \frac{\Gamma(r+s)}{\Gamma(r)\Gamma(s)} x^{r-1}(1-x)^{s-1} \tag{1}$$

and is defined for $0 < x < 1$. $\Gamma()$ is the familiar gamma integral, which also appears in the gamma density and many others. This distribution has many uses, in particular when modelling a proportion which takes values between zero and one[1].

Choice of the values of r and s give tremendous flexibility. The $B(r,s)$ can take many shapes. If you type in R

[1]In fact, given that $\Gamma(r) = (r-1)!$, you can easily see that for adequate (natural) values of r and s the $B(r,s)$ specializes to the binomial distribution.

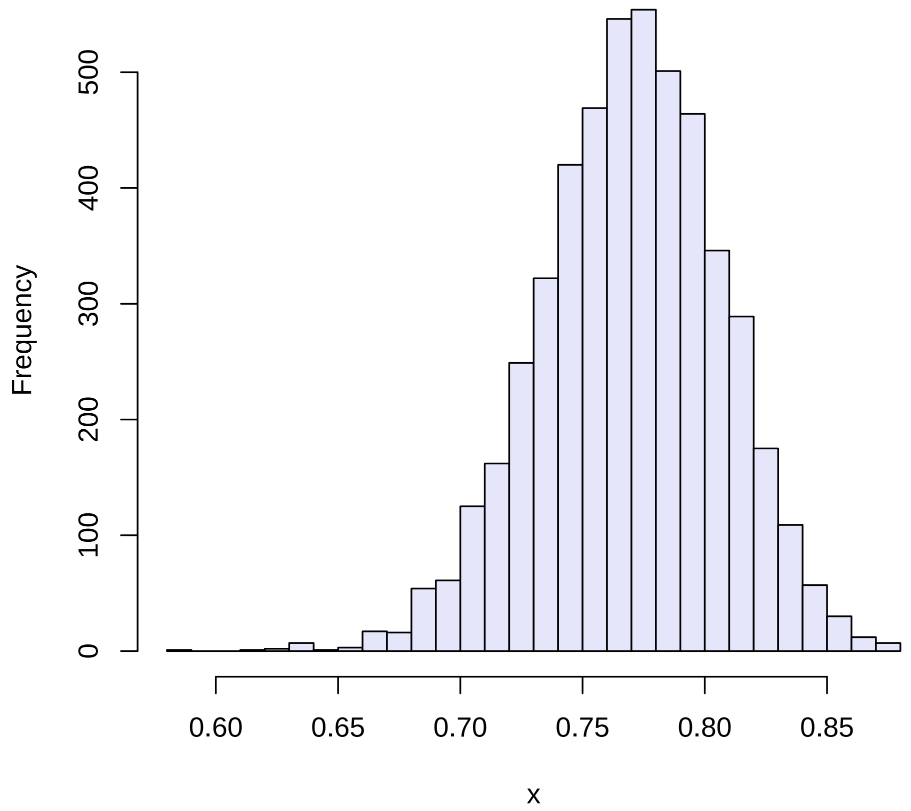
```
> example(dbeta)
```

you will see different shapes (and examples of use of the function `dbeta`).

You can generate random values from a beta using the function `rbeta`. See for instance the following code which generates the histogram in Figure 1.

```
> set.seed(12345)
> x <- rbeta(5000, shape1=100, shape2=30)
> hist(x, col="lavender", breaks=30, main="")
```

Figure 1: 1000 observations from a $B(r = 100, s = 30)$



2.1 Moment estimators of r and s

With some work we can obtain that the first and second order moments of the $B(r,s)$ distribution are:

$$\alpha_1 = \frac{r}{r+s} \tag{2}$$

$$\alpha_2 = \frac{r(r+1)}{n(r+s)(r+s+1)} \tag{3}$$

The variance is:

$$\sigma^2 = \alpha_2 - \alpha_1^2 = \frac{rs}{(r+s)^2(r+s+1)} \tag{4}$$

Equating (2) to the first order sample moment $\overline{X}$ we obtain:

$$s = \frac{r}{\overline{X}} - r \tag{5}$$

Replacing that value in (4) and equating to the sample variance S^2 we finally have that:

$$\hat{r}_{\mathrm{M}} = \overline{X}\left(\frac{\overline{X}(1-\overline{X})}{S^2} - 1\right) \tag{6}$$

$$\hat{s}_{\mathrm{M}} = (1-\overline{X})\left(\frac{\overline{X}(1-\overline{X})}{S^2} - 1\right) \tag{7}$$

2.2 Maximum likelihood estimators of r and s

The joint density of a sample of n independent values $x_1, \ldots, x_n$ is from (1):

$$f_{\vec{X}}(\vec{x}; r, s) = \prod_{i=1}^{n} \frac{\Gamma(r+s)}{\Gamma(r)\Gamma(s)} x_i^{r-1}(1-x_i)^{s-1} \tag{8}$$

Taking the log we obtain the log likelihood function associated to the sample $x_1, \ldots, x_n$:

$$\ell(r,s;\vec{x}) = n\log\Gamma(r+s) - n\log\Gamma(r) - n\log\Gamma(s) + (r-1)\sum_{i=1}^{n}\log(x_i) + (s-1)\sum_{i=1}^{n}\log(1-x_i) \tag{9}$$

3 Questions

Taking the derivatives of (9), equating them to zero and solving to obtain $\hat{r}_{\mathrm{MLE}}$ and $\hat{s}_{\mathrm{MLE}}$ is not promising (try and you will see why). Therefore, we have to resort to direct maximization of the likelihood if we want maximum likelihood estimators.

1. Write a function which receives as its only argument a vector of with parameter values r and s and computes (9).
2. Write a loop which does the following N times:
   (a) Generate a sample of size n from a beta distribution, $B(r = 10, s = 20)$.
   (b) Compute the moment estimators $\hat{r}_{\mathrm{M}}$ and $\hat{s}_{\mathrm{M}}$, using formulae (6)–(7).

(c) Compute maximum likelihood estimators $\hat{r}_{\text{MLE}}$ and $\hat{s}_{\text{MLE}}$ by direct maximization of the funcion you wrote in step 1,

(d) Store all estimates in the i-th row of a matrix with four columns (two moment estimators, two MLE estimators) and N rows.

3. After you exit the loop, compute the averages of all estimates and compare to the known values of r and s. What do your results suggest regarding unbiasedness of the respective estimators?

4. Estimate the mean square error, of the respective estimators. Use boxplots to compare the empircial distributions of the moment and MLE estimators. What are your conclusions? Which estimator appears to be best?

4 Hints and comments

1. Although the derivative of $\Gamma(t)$ with respect to t does not have a closed form, it is a well known function, the $\psi(t)$ or digamma function. It can be evaluated in R with function `digamma` (refer to the handout for Seminar 2). But this will not help much with the likelihood equations, were you to try to solve them by hand.

2. The mean square error of estimator $\hat{p}$ of p is defined as: $E(\hat{p}-p)^2$. If you have a vector `phat` containing the estimates $\hat{p}$ in the N replications, you can easily obtain an approximation to $E(\hat{p}-p)^2$. Assume the true value of p (which you know, since you generated the observations) is 10. You could compute:

   ```
   > MSEp <- sum((phat-10)^2) / length(phat)
   ```

   or, more simply,

   ```
   > MSEp <- mean((phat-10)^2)
   ```

3. Start with small N, only when everything is working properly switch to a larger value of N. Bear in mind that (unlike in Seminar and Activity 1) there is fairly substantial work to do at each iteration (one non-linear maximization, by `optim`), so don't choose N in the tens of thousands or millions.

   You might try for instance $N = 500$ replications and sample sizes n anywhere from 50 to 1000. That should finish in a reasonable time. In theory, MLE estimators should be optimal, at least asymptotically and among unbiased estimators, but for any finite n the reverse might be true. Further, we do not know whether either or both estimators are unbiased. On the other hand, "optimal" doesn't mean that the MLE should be substantially better: it may happen, and often happens, that the difference between moment and MLE estimators does not justify the extra effort needed to compute the MLE.

4. You may wonder where moments in formulae (2)–(3) as well as the moment estimators in (6)–(7) come from. Obtention is conceptually easy but laborious. For this and any other matters related to proporties of distributions, you may turn to any manual, check the Wikipedia or turn to [?], chap. 25.

5 Ejemplos de respuestas esperadas

1. La verosimilitud de la $Beta(r, s)$ podría escribirse así:

```
> betalik <- function(p,x) {
   n <- length(x)
   r <- p[1] ; s <- p[2]
   ll <- n*(log(gamma(r+s)) - log(gamma(r)) -
            log(gamma(s))) + (r-1)*sum(log(x)) +
         (s-1)*sum(log(1-x))
   return(ll)
  }
```

2. Una vez se dispone de la función calculando al verosimilitud asociada a una muestra `x` y unos valores de los parámetros `p`, el bucle prncipal de la simulación es simple:

```
> N <- 500
> n <- 1000
> r <- 10 ; s <- 20
> estim <- matrix(0,N,4)
> colnames(estim) <- c("rM", "sM", "rMLE", "sMLE")
> for (i in 1:N) {
      x    <- rbeta(n, 10, 20)
      Xbar <- mean(x)
      S2   <- var(x)
      estim[i,1] <- Xbar * ( (Xbar*(1-Xbar)) / S2 - 1)
      estim[i,2] <- (1- Xbar) * ( (Xbar*(1-Xbar)) / S2 - 1)
      res <- optim(par=c(8,15), fn=betalik, x=x,
                   control=list(fnscale=-1))
      estim[i,3:4] <- res$par
      }
```

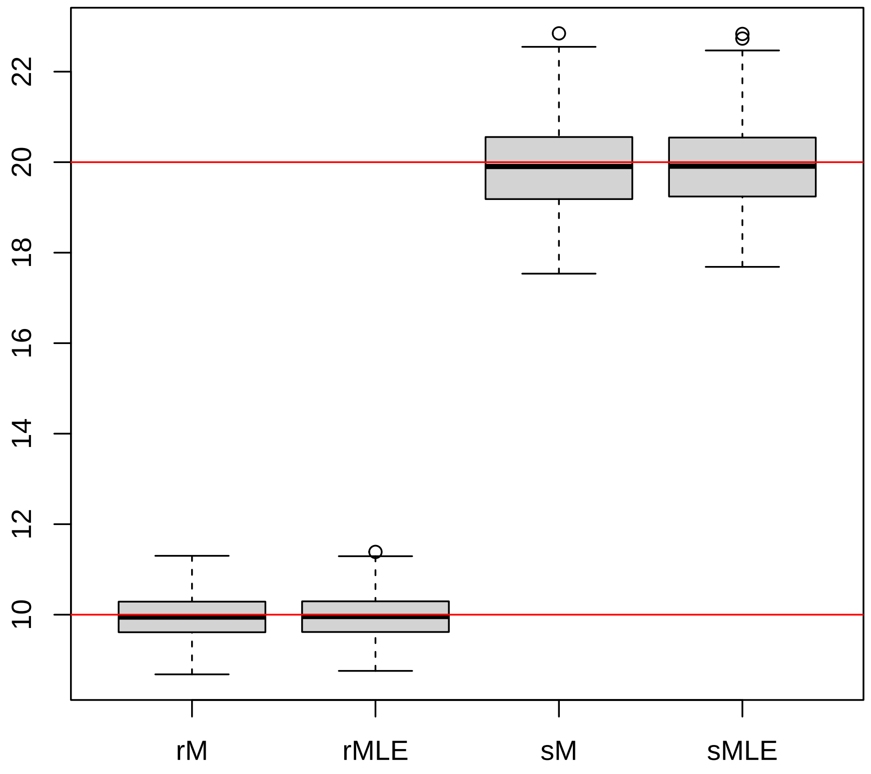
3. Acabada la ejecución del bucle, podemos examinar los valores simulados de los estimadores. Reordenamos las columnas en `estim` para facilitar la comparación:

```
> estim <- estim[,c(1,3,2,4)]
> summary(estim)

       rM               rMLE               sM
 Min.   : 8.681   Min.   : 8.757   Min.   :17.54
 1st Qu.: 9.612   1st Qu.: 9.617   1st Qu.:19.19
 Median : 9.947   Median : 9.960   Median :19.90
 Mean   : 9.967   Mean   : 9.978   Mean   :19.94
 3rd Qu.:10.287   3rd Qu.:10.293   3rd Qu.:20.55
 Max.   :11.301   Max.   :11.387   Max.   :22.85
      sMLE
 Min.   :17.69
 1st Qu.:19.24
 Median :19.91
 Mean   :19.96  
 3rd Qu.:20.54  
 Max.   :22.83  
```

y compararlos mediante sendos boxplots. Ponemos contiguos los estimadores de los mismos parámetros para facilitar la compración.

```
> boxplot(estim)
> abline(h=10, col="red")
> abline(h=20, col="red")
```



4. Podemos, finalmente, estimar el error cuadrático medio de los diferentes estimadores.

```
> Params <- matrix(c(r,r,s,s),N,4,byrow=TRUE)
> MSE    <- colMeans((estim-Params)^2)
> MSE

       rM      rMLE        sM      sMLE 
0.2323589 0.2283659 0.9416495 0.9282239 
```

Vemos que el MSE de los estimadores máximo verosímiles es un poco menor que el de los estimadores por momentos. Los sesgos estimados son

```
> Sesgos <- colMeans(estim-Params)
> Sesgos

         rM        rMLE          sM        sMLE 
-0.03292715 -0.02177571 -0.06206382 -0.03970390 
```

Los sesgos estimados de los estimadores máximo-verosímiles son apreciablemente menores para este tamaño muestral. Esto puede variar en diferentes simulaciones. Lo que esperamos permanezca, al menos para `n` suficientemente grande, es algún mejor comportanmiento del MLE. La diferencia es tan escasa que la moraleja aquí sería que no hay que molestarse en ir más allá de los estimadores por momentos en este caso, al menos para el tamaño muestral considerado.