Binomial Inference

Pedro Victor Brasil Ribeiro

2021-12-19 - Last changed in 2022-01-19

Supose you have a sample that comes from a binomial distribution, and the sample size is igual to m. How can you estimate the parameter p and n With only that sample?

To solve this problem we use a estimator deriving from the likelihood function, which is define as:

$$L(x;\theta) = \prod_{i=1}^{n} f(x_i;\theta) \tag{1}$$

And then we find the maximum point of the function, or in other words, the point that make the derivative of $f(x;\theta)$ igual to 0.

The Problem

Let $X_1, X_2, \cdots, X_m$ be a random variable i.i.d. (Independent and identically distributed) a binomial sample of m observations with parameters n and p, in other words $X \sim Bin(n,p)$. Then we know that the density function of a binomial is:

$$f(x;\{n,p\}) = \binom{n}{x} p^x (1-p)^{n-x}; \qquad n \in \mathbb{Z}^* \quad p \in (0,1) \tag{2}$$

So first of all we need to find the likehood function for a binomial distribution so using the equation 1, where $f(x;\theta)$ for $\theta = \{n,p\}$, expressed in the equation 2.

Estimation

Usually is more convenient to work with the derivative of $log[(L(x;\theta)]$, note that the maximum point of $L(x;\theta)$ and $log[L(x;\theta)]$ is the same point.

$$\begin{aligned}
L(x;\theta) &= \prod_{i=1}^{n} f(x_i;\theta) \\
&= \prod_{i=1}^{n} \binom{n}{x_i} p^{x_i} (1-p)^{n-x_i} \\
\Rightarrow l(x;\theta) &= \sum_{i=1}^{m} log\Big[\binom{n}{x_i}\Big] + \sum_{i=1}^{m} x_i log(p) + \sum_{i=1}^{m} (n-x_i) log(1-p) \\
&= \sum_{i=1}^{m} log\Big[\binom{n}{x_i}\Big] + m\bar{X} log(p) + (mn - m\bar{X}) log(1-p)
\end{aligned}$$

Estimate p, with n known

Supposing that the parameter n is know, in order to estimate the parameter p we can derivate $l(x;\theta)$ in relation to p. We have:

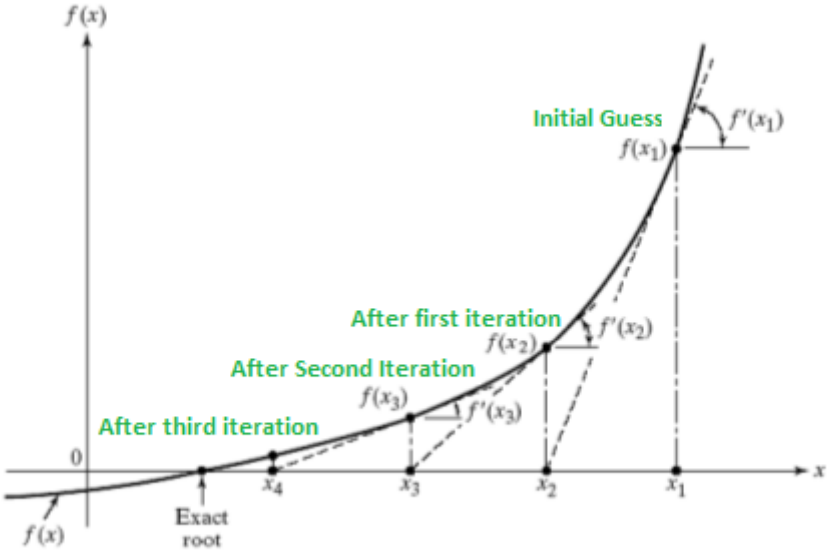
$$\begin{aligned}\frac{\partial l(x;\theta)}{\partial p} =& \frac{\partial}{\partial p}\Big[\sum_{i=1}^{m} log\Big[\binom{n}{x_i}\Big] + m\bar{X}log(p) + (mn - m\bar{X})log(1-p)\Big] \\ \Rightarrow& \frac{\bar{X}}{\hat{p}} - \frac{n-\bar{X}}{1-\hat{p}} = 0 \end{aligned} \tag{3}$$

There's no explict formular for the estimator of p, so we need to use some computation way to find the maximum root of the derivative of $l(x;\theta)$. A very good way of doing it is know as the Newton-Raphoson Method, there're mode way to solve the problem, such as the secant method, Bisection method or even finding the maximum point of the likehood function (which will be done in the last section). In this document we will use the Newton-Raphson method.

Newton-Raphson method

Newton-Raphson method is an interactive equation that start with a initial guess of the root. The Newton-Raphson method is an approch to find the roots of non-linear equations, is a well-known and widely used for his simplicity and his speed for convergency.

In this document we will not explain the thery behind the method, but the image bellow is a pretty good way to have a hint on how it works. Basically, given a start point (first guess), is calculated the derivative, so is found the point the the tangent line "touch" the x-axis, then is calculated his image on y-axis and the the derivative on the point, and so on. Until in one time the difference on the two point is lower than the tolerance accepted for the user.



$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{4}$$

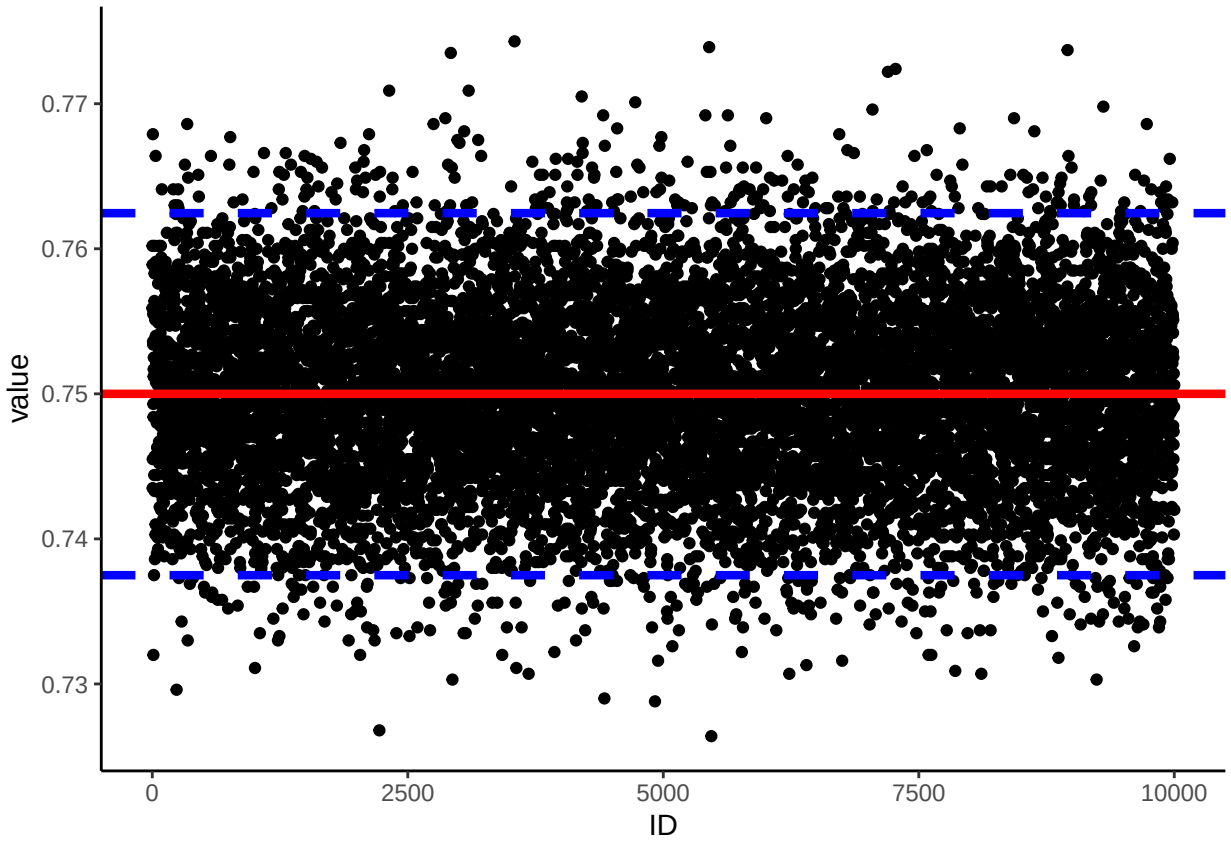
The code used to find the root using the Newton-Raphson method were made by Aaron Schlegel on the LINK, as it follows:

```r
## Newton-Raphson

newton.raphson <- function(f, a, b, tol = 1e-5, n = 1000) {
  require(numDeriv) # Package for computing f'(x)

  x0 <- a # Set start value to supplied lower bound
  k <- n # Initialize for iteration results

  # Check the upper and lower bounds to see if approximations result in 0
  fa <- f(a)
  if (fa == 0.0) {
    return(a)
  }

  fb <- f(b)
  if (fb == 0.0) {
    return(b)
  }

  for (i in 1:n) {
    dx <- genD(func = f, x = x0)$D[1] # First-order derivative f'(x0)
    x1 <- x0 - (f(x0) / dx) # Calculate next value x1
    k[i] <- x1 # Store x1
    # Once the difference between x0 and x1 becomes sufficiently small, output the results.
    if (abs(x1 - x0) < tol) {
      root.approx <- tail(k, n=1)
      res <- list('root_approximation' = root.approx, 'iterations' = k)
      return(res)
    }
    # If Newton-Raphson has not yet reached convergence set x1 as x0 and continue
    x0 <- x1
  }
  print('Too many iterations in method')
}
```

So let the parameter as previously defined (n = 17, p = 0.75), we will define the function 3 on hat and run the newton raphson method as shown in the equation 4.

So for the seed `18122021` we had a estimation on the p-value 0.7392513 and the root converged with 13, such that the tolerance was of 0.00001.

To confirm the efficiency of the estimator we will use a Monte-Carlo simulation, which basically means, that we will the code a bunch of times and see the estimator behavior. In this case we will run the code 10,000 times, and for we have the same result.



From the plot above we can see that the estimations in highly concentrated inside the interval of confidence (blue line). To check the coverage of the estimation, we can see the percentage of points that is inside the interval and the value have to be greater than or equal to the confidence chosen ($\gamma = 0.95$).

let's call "hit" the points that are within the confidence interval and "error" the ones that aren't.

coverage	n
error	0.048
hit	0.952

So, since the value is equal to 0.952, which is greater than the chosen γ, then the estimator have a good covarage, and the value converges quickly, with only 13 or 14 interactions (12.34% and 87.66%, respectively). So we have a great estimator

Estimate n, with p know

Now we will use the same theory to find the estimative for n, called $\hat{n}$, so first of all we will open the binomial on the log of the likelihood function:

$$\begin{aligned} l(x;\theta) &= \sum_{i=1}^{n} log\Big[\frac{n!}{(n-x_i)!x_i!}\Big] + n\bar{X}log(p) + (mn - m\bar{X})log(1-p) \\ &= \sum_{i=1}^{n} log(n!) - \sum_{i=1}^{n} log[(n-x_i)!] - \sum_{i=1}^{n} log(x_i!) + n\bar{X}log(p) + (mn - m\bar{X})log(1-p) \end{aligned} \tag{5}$$

One thing that we have to note is that "n" is a discrete number, therefore the function 5 doesn't have a derivative on respect to n, but we have a very interesting function that can help to solve this problem, that function is the gamma function.

$$\Gamma(t) = \int_0^\infty x^{t-1} e^{-x} dx \tag{6}$$

The gamma function have a very interesting propriety:

- $\Gamma(n) = (n-1)! \quad \Rightarrow \quad n! = \Gamma(n+1)$.

Which is a continuous function. So making the substitution of the equation 6 in the equation 5, and defining that the derivative of $log[\Gamma(n)]$ in relation to n is equal to $\psi^{(0)}(n)$.

$$\begin{aligned} l(x;\theta) =& \sum_{i=1}^{n} log(n!) - \sum_{i=1}^{n} log[(n-x_i)!] - \sum_{i=1}^{n} log(x_i!) + n\bar{X} log(p) + (mn - m\bar{X}) log(1-p) \\ \frac{\partial l(x;\theta)}{\partial n} =& m\psi^{(0)}(n+1) - \sum_{i=1} \psi^{(0)}(n - x_i + 1) + mlog(1-p) \end{aligned}$$

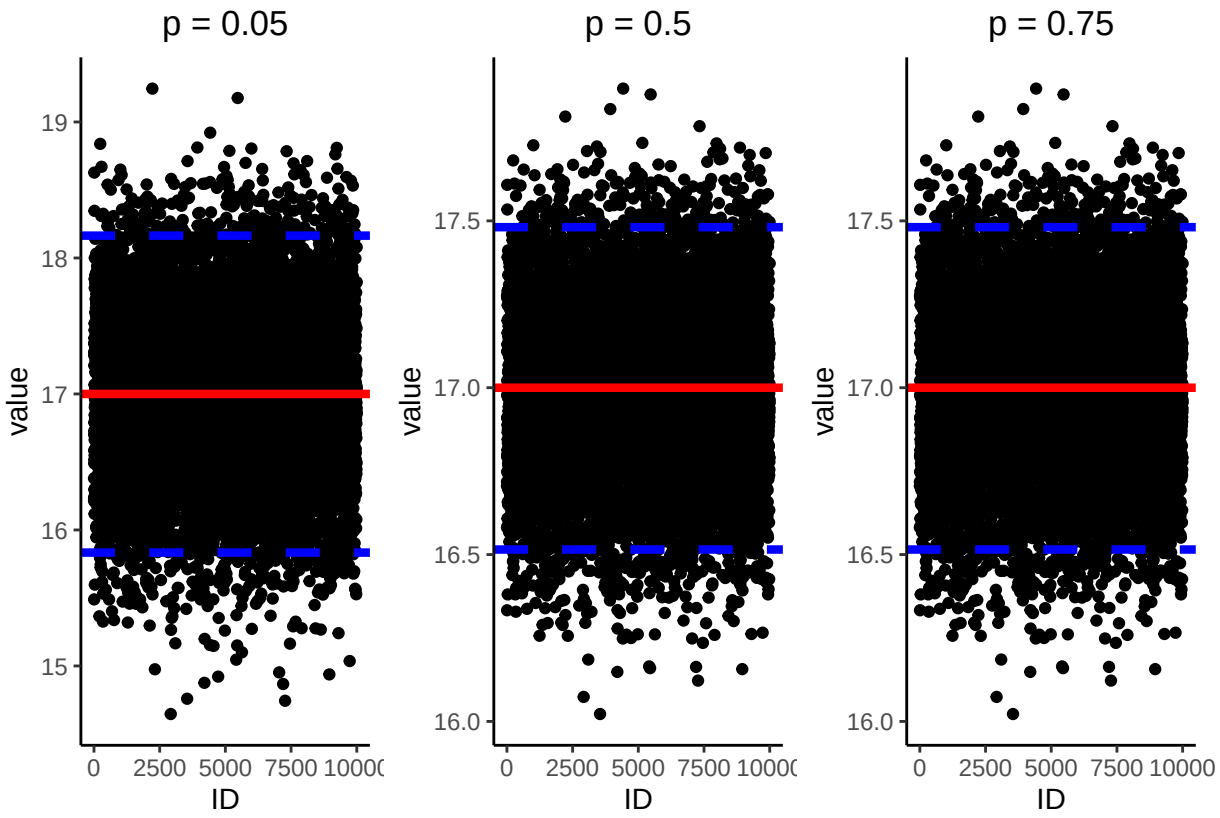
Then the maximum likelihood for n doesn't have a closed equation, similar to p. So we have the function.

$$m\psi^{(0)}(\hat{n}+1) - \sum_{i=1} \psi^{(0)}(\hat{n} - x_i + 1) + mlog(1-p) = 0 \tag{7}$$

We will find the the root of equation 7 using Newton-Raphson [4]. For this case we will make the estimation in three scenarios when the p is 0.15, 0.5 and 0.75, to check is there's no different behavior with is small, big or even in the point that maximize the variance of a binomial $[Var(X) = np(1-p)]$.

The estimator of n is a continuous, but n is discrete, the estimation will be rounded. With p = 0.15 we had the n = 18, with p = 0.5 n = 17 and with p = 0.75 n = 17.

So we'll do the same process to diagnose the quality of the estimator as done before. In order to not insert an error into it, the number will not be rounded.



coverage	p_0.15	p_0.5	p_0.75
error	0.0483	0.0491	0.0491
hit	0.9517	0.9509	0.9509

So we since the coverage for all values of p, we can say that the estimator defined is the equation 7 is a good estimator.

Both n and p unknown

For the case that we want to find both parameter, we have some options, but since both parameter don't have a closed equation for the parameters we'll probably fall into solving a system of equation. Which is very hard to solve computationally.

But note that, the definition of the maximum estimator from the likelihood function, is to find a point that maximize that same function, so a easier way so solve it, would be computationally finding that point. That way of viewing the problem would also be possible for the past problems, but personally i find easier to solve those cases with using newton raphson in the derivative of the likelihood function.

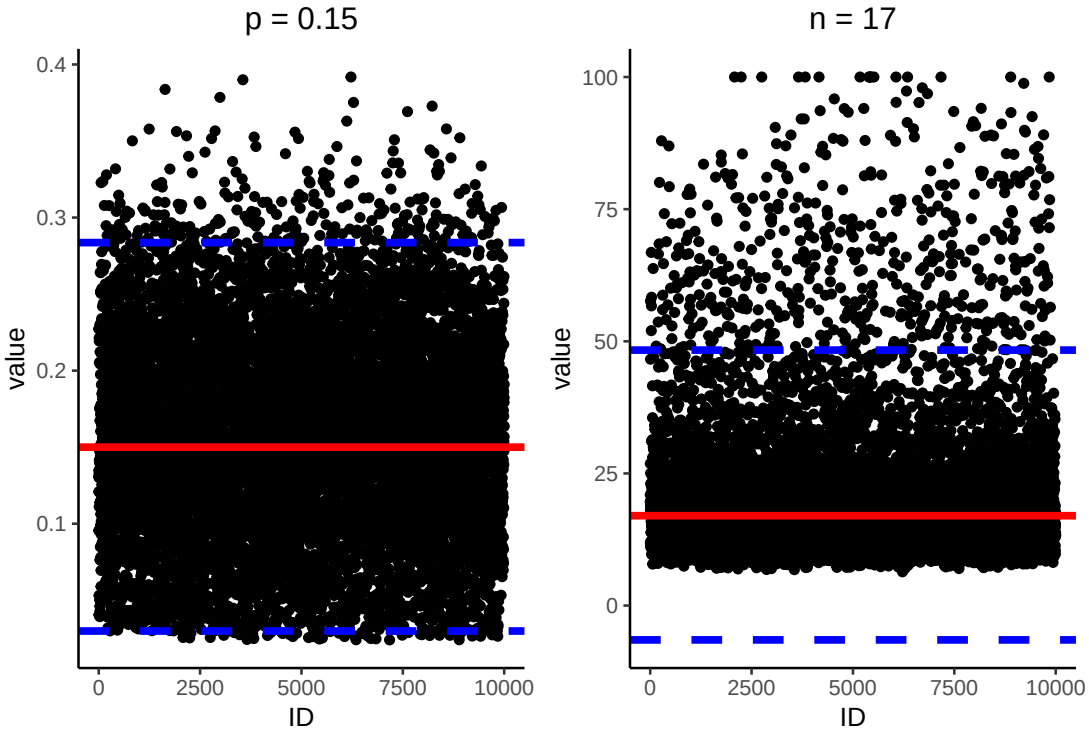
In this case it'll be used a function from R `optim()`, one thing to note is that the function find the minimum of the function , but unfurtunally there's no argument to find the maximum. So in order to find the maximum you can only define $-L(x;\theta)$.

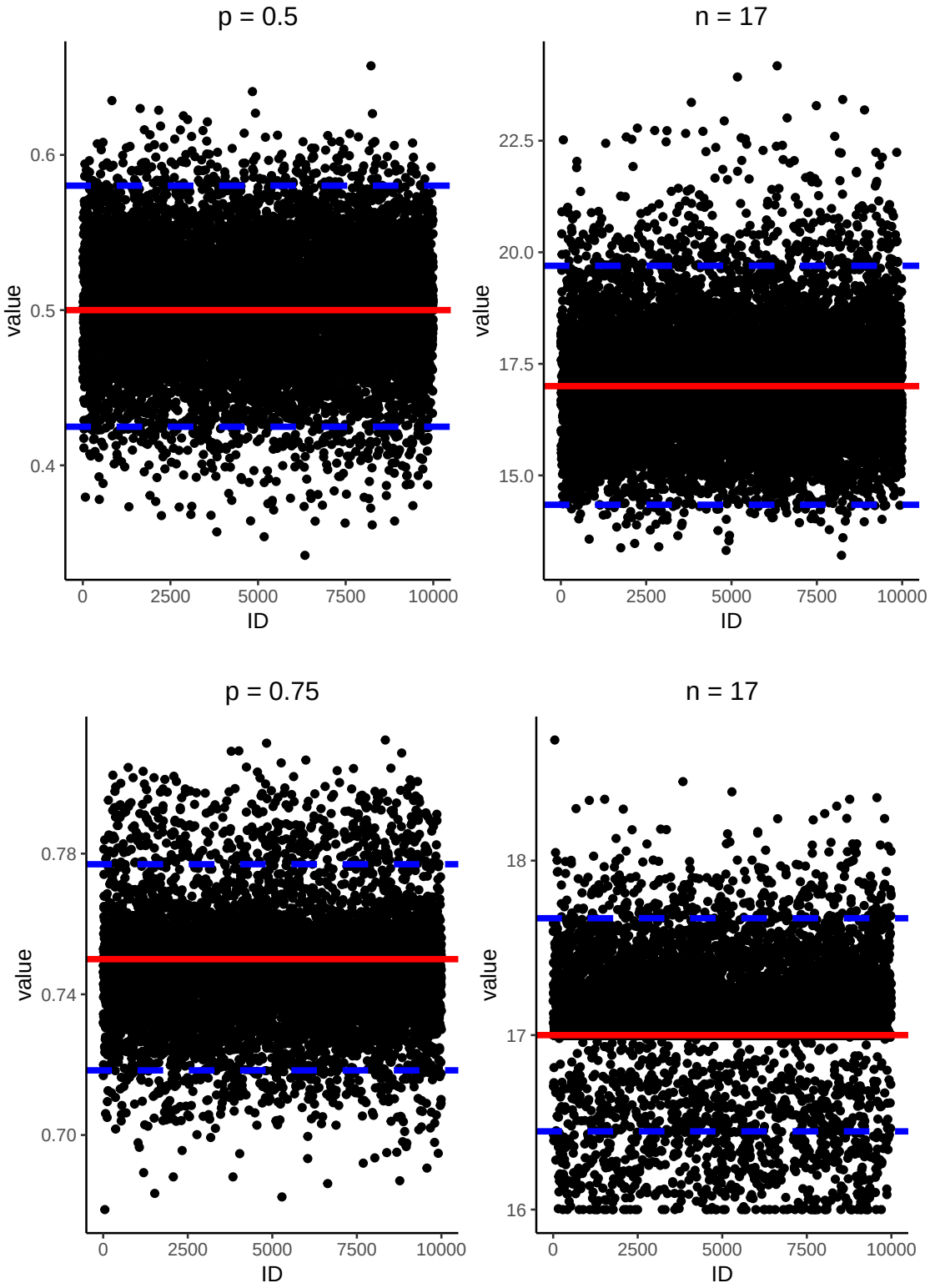
How we are using a mainly computational way to maximize the likelihood function, and there's two unknown parameter, i'll be more generous with the interpretation of the results.

It'll be used, in the optim, the method `L-BFGS-B`, because is the multiparametric method that allow the

definition of a minimum and a maximum for the possible parameter of the function, with is needed for p ($p \in (0, 1)$), and for n, since any value greater than the maximum of the data will diverge.

In order to evaluate the numerical algorithm to find the max of the likelihood function, we will use 3 different scenarios, changing the value of p. We`ll use p = 0.15, p = 0.5 and p = 0.75. And then as done previously` we`ll` plot the estimates of each loop made and then it'll be shown a table compiling the coverage for the 3 cases (p = 0,15; p = 0.5; p = 0.75).



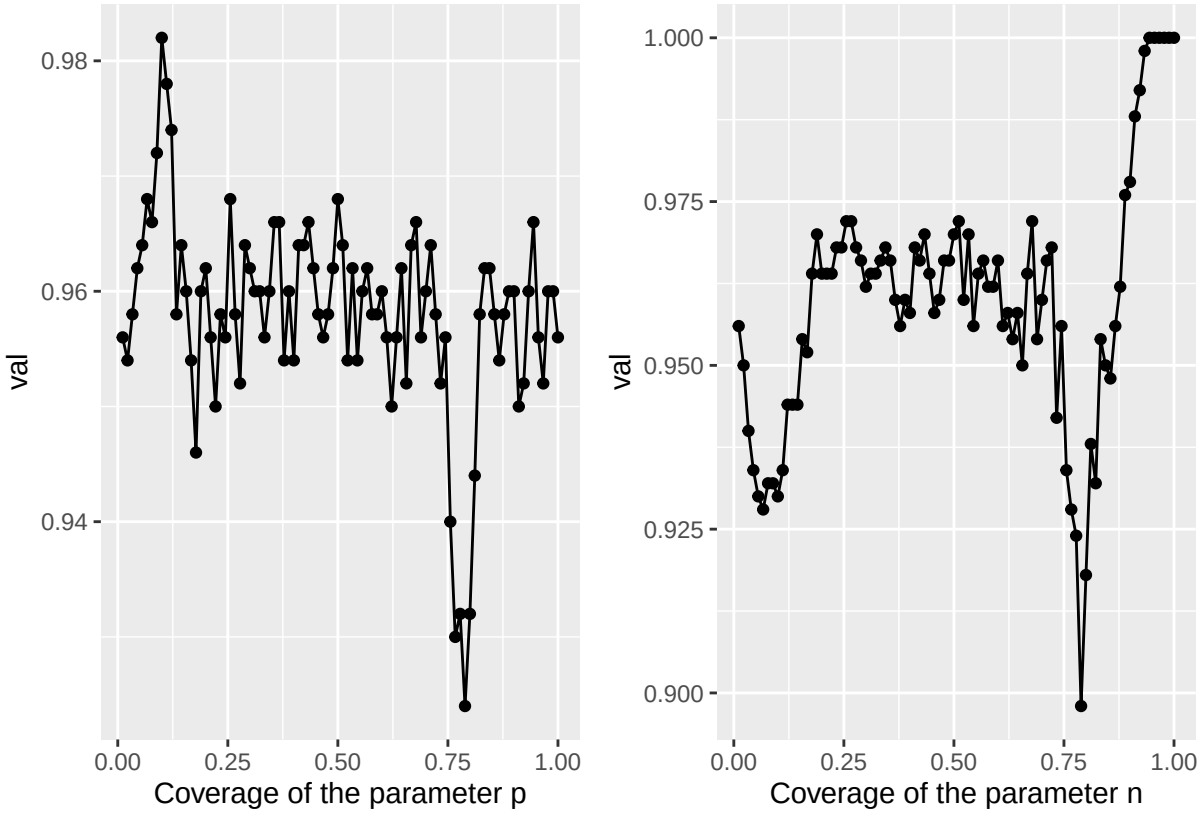


As we can see in the plot shown, the amount of points outside the confidence interval, is bigger for 0.5 than 0.15 and 0.75 is bigger than 0.5. Which wasn't expected, since 0.5 is the point that maximize the variance $[Var(X) = np(1-p)]$.

True values	Situation	p	n
p = 0.15; n = 17	error	0.339	0.0558
p = 0.15; n = 17	hit	0.966	0.944
p = 0.5; n = 17	error	0.0524	0.0488
p = 0.5; n = 17	hit	0.948	0.951
p = 0.75; n = 17	error	0.0766	0.0877
p = 0.75; n = 17	hit	0.923	0.912

With the table we can see that the estimation shown good results for p = 0.15 and p = 0.5, but the results wasn't that great for p = 0.75 (p = 0.923; n = 0.912), but as said previously we'll be more lenient with the multiparametric case. So we'll consider the estimation good for p = 0,15 and p = 0.5 and regular for p = 0.75.

But an interesting might be on your mind, how the covarege perform in different values of p? So we'll see it for 90 different on the range of 0.05 and 0.95, with the code `seq(0.05,0.95, length.out = 90)`. I'll leave to the reader to do the same type of analysis for diffent values of n



So we have a very interesting result here:

- Where we have a covarage between [0.95; 0.97] for $p \in [0.25; 0.75]$, for both parameter
- For $p < 0.25$ the covarage, for p, have a positive spike and, for n, have a negative spike.
- For $p \in [0.70; 0.80]$ a big reduction on both parameter;
- For $p > 0.8$ a covarage between [0.95; 0.97] for p, and a perfect or nearly perfect covarege for n.

Conclusion

By using the classic inference paradigm we had a good form to take the inference for the parameters on each case presented, remarking the points argued for the biparametric case.