Binomial Inference

Pedro Victor Brasil Ribeiro

2021-12-19 - Last changed in 2021-12-19

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:

$$\frac{\partial l(x;\theta)}{\partial p} = \frac{\partial}{\partial p}\Big[\sum_{i=1}^{m} log\Big[\binom{n}{x_i}\Big] + m\bar{X}log(p) + (mn - m\bar{X})log(1-p)\Big]$$

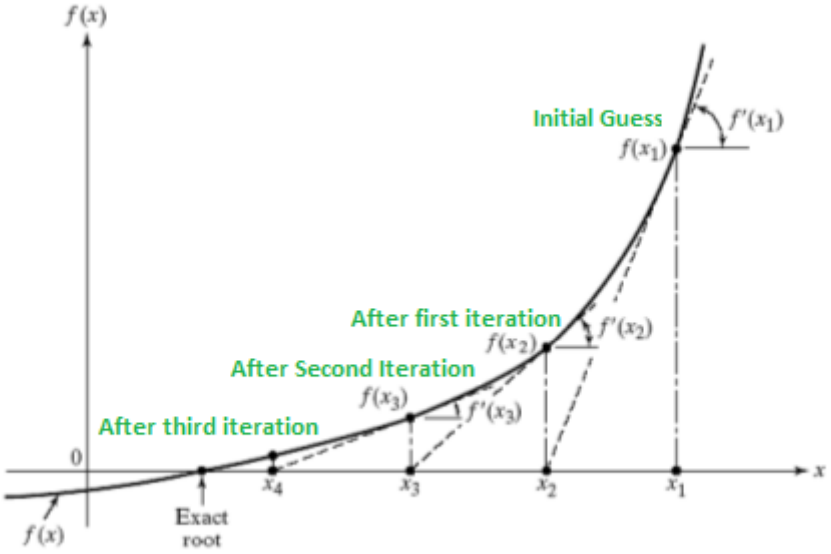
$$\Rightarrow \frac{m\bar{X}}{\hat{p}} - \frac{n-\bar{X}}{1-\hat{p}} = 0$$

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 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.



$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{3}$$

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

```
## 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')
}
```

Estimate n, with p know

Both n and p unknow