

Binomial Inference

Pedro Victor Brasil Ribeiro

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

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

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

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

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

The Problem

Let X_1, X_2, \dots, 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 $X \sim \text{Bin}(n, p)$. Then we know that the density function of a binomial is:

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

So first of all we need to find the likelihood 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 it 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. Since we are looking at the maximum point then we know that $L'(x; \theta) = 0$, so:

$$\begin{aligned} \log'[L(x; \theta)] &= \frac{\partial \log[L(x; \theta)]}{\partial \theta} \\ &= \frac{1}{L(x; \theta)} \frac{\partial L(x; \theta)}{\partial \theta}; \quad \text{but we know that in the maximum point } \frac{\partial L(x; \theta)}{\partial \theta} = 0 \\ \Rightarrow \frac{\partial \log[L(x; \theta)]}{\partial \theta} &= 0 \end{aligned}$$

$\forall x_0$ That $f(x_0; \theta) \neq 0$.

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

Estimate p, with n known

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

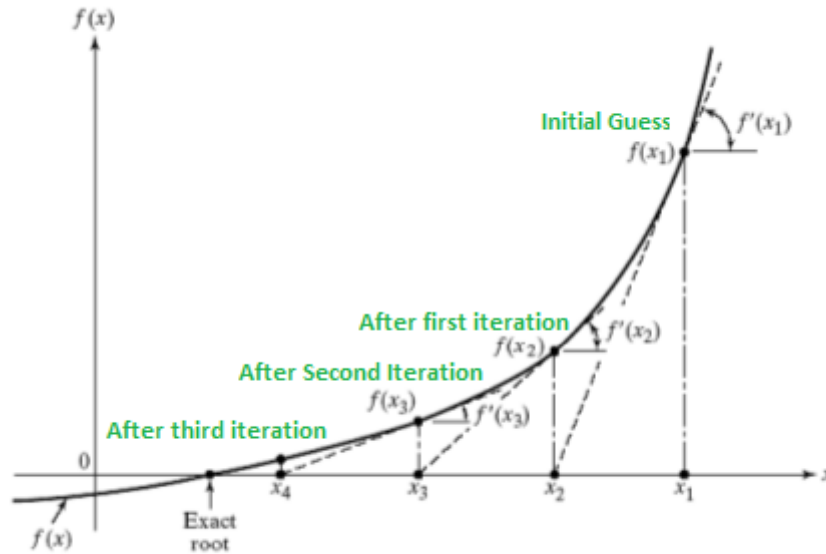
$$\begin{aligned}
\frac{\partial l(x; \theta)}{\partial p} &= \frac{\partial}{\partial p} \left[\sum_{i=1}^m \log \left[\binom{n}{x_i} \right] + m\bar{X} \log(p) + (mn - m\bar{X}) \log(1-p) \right] \\
\Rightarrow \frac{m\bar{X}}{\hat{p}} - \frac{n - \bar{X}}{1 - \hat{p}} &= 0
\end{aligned}$$

There's no explicit formula for the estimator of p, so we need to use some computational way to find the maximum root of the derivative of $l(x; \theta)$. A very good way of doing it is known as the Newton-Raphson Method, there are many ways to solve the problem, such as the secant method, Bisection method or even finding the maximum point of the likelihood 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 iterative that starts with an initial guess of the root. The Newton-Raphson method is an approach to find the roots of non-linear equations, is a well-known and widely used for its simplicity and its speed for convergence.

In this document we will not explain the theory behind the method, but the image below 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 where the tangent line "touch" the x-axis, then is calculated its image on y-axis and the derivative on the point, and so on. Until in one time the difference on the two points is lower than the tolerance accepted for the user.



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

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