Project name – Find Bita in Weibull distribution using the Newton Rapson method.

Author – Y.S. Nimesh

Date – 2023.07.23

1. Introduction

The Weibull distribution is a type of continuous probability distribution that is frequently used in reliability engineering and survival analysis. It is frequently used to simulate the lifespan of items and materials. The shape parameter (beta, abbreviated as 'b') and the scale parameter are the two parameters of the Weibull distribution. In this project, we will use the Newton-Raphson method to get the value of the shape parameter 'b,' which is an iterative numerical approach for determining the roots of a real-valued function.

2. Newton-Raphson Method

The Newton-Raphson technique is a numerical iterative method for determining the roots (zeroes) of a real-valued function. Given a function f(x), the approach begins with an initial guess 'x0' and refines the guess repeatedly using the formula:

$$x1 = x0 - f(x0) / f'(x0)$$

f'(x0) is the derivative of the function evaluated at x0. The procedure is repeated until the difference between subsequent iterations is less than a set tolerance threshold or a maximum number of iterations is achieved.

3. Mathematical Background

The shape parameter 'b' of the Weibull distribution can be estimated by solving the following equation:

$$g(b, x) = (\Sigma(xi\hat{}b * \ln(xi))) / (\Sigma(xi\hat{}b)) - (1 / b) - (\Sigma(\ln(xi)) / n) = 0$$

where:

- x denotes an array of x-axis values (lifetime data)

- n is the number of array items - signifies summation

- ln(x) represents the natural logarithm of x;

- g(b, x) represents the objective function

The derivative of g(b, x) with respect to 'b' is given by:

dg(b, x) = (Σ(xi^b * ln(xi)^2)) / (Σ(xi^b * ln(xi))) - (1 / b^2) - (Σ(xi^b * ln(xi)) / Σ(xi^b))

4. Implementation

There's a package to solve this problem in python. Math package use for solve mathematical equations.

Importl math package in python environment.

```
In [1]: import math
```

4.1. Objective Function (g(b, x)package. Unction g(b, x) computes the objective function value given the shape parameter 'b' and an array of x-axis values 'x'. It returns the function's value, which we want to make zero in order to get the value of 'b'.

```
In [2]: def g(b, x):
            n = len(x)
            sum1 = sum([math.pow(i, b) * math.log(i) for i in x])
            sum2 = sum([math.pow(i, b) for i in x])
            sum3 = sum([math.log(i) for i in x])
            return (sum1 / sum2) - (1 / b) - (sum3 / n)
```

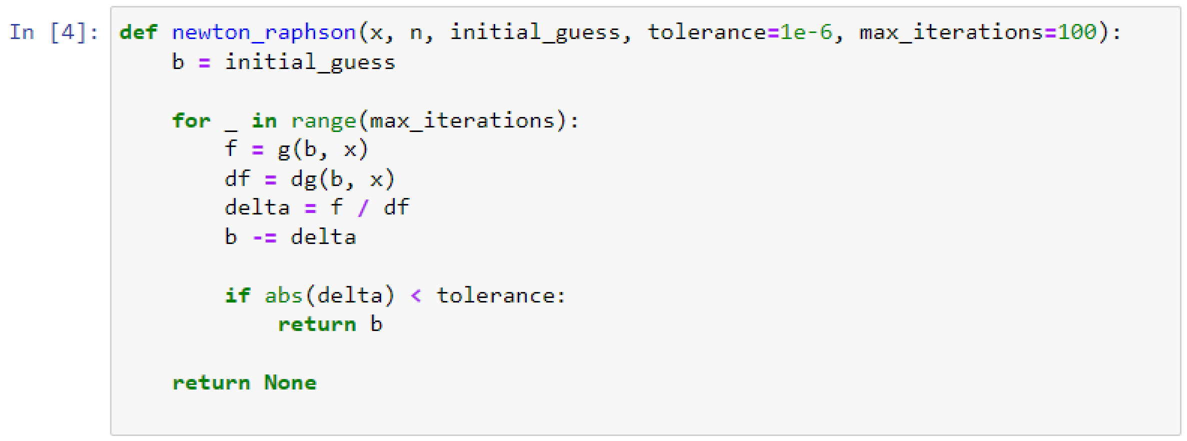
4.2. Derivative of Objective Function (dg(b, x))

Given the shape parameter 'b' and the array of x-axis values 'x,' the function dg(b, x) computes the derivative of the objective function with respect to 'b'. The Newton-Raphson technique requires this derivative to update the estimate in each iteration.

```
In [3]: def dg(b, x):
            n = len(x)
            sum1 = sum([math.pow(i, b) * math.log(i) * math.log(i) for i in x])
            sum2 = sum([math.pow(i, b) * math.log(i) for i in x])
            sum3 = sum([math.pow(i, b) for i in x])
            return (sum1 / sum2) - (1 / (b*b)) - (sum2 / sum3)
```

4.3. Newton-Raphson Method (newton Raphson)

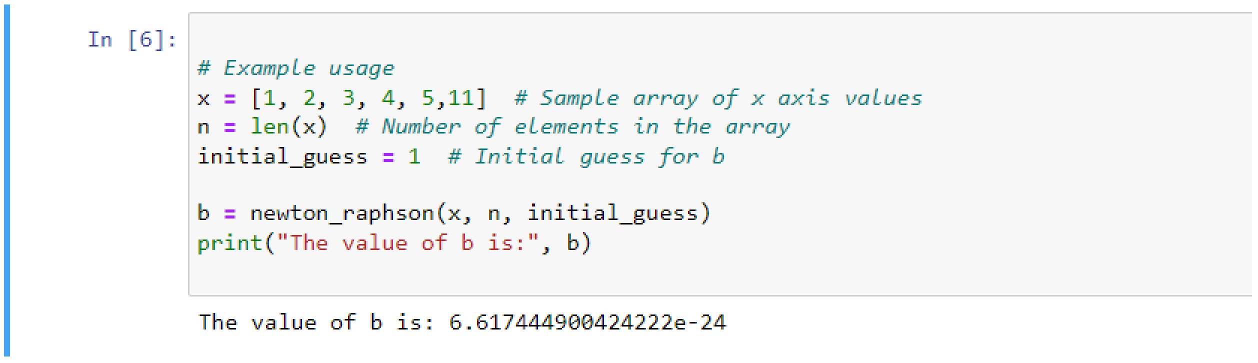
The Newton-Raphson technique is used to discover the value of 'b' in the function newton Raphson(x, n, initial guess, tolerance, max iterations). It requires an array of x-axis values 'x,' a number of elements 'n,' a starting guess for 'b,' a tolerance level to halt iterations, and a maximum number of iterations. The function refines the guess for 'b' iteratively until the difference between subsequent iterations is less than the set tolerance or the maximum number of iterations is achieved.



```
In [4]: def newton_raphson(x, n, initial_guess, tolerance=1e-6, max_iterations=100):
            b = initial_guess

            for _ in range(max_iterations):
                f = g(b, x)
                df = dg(b, x)
                delta = f / df
                b -= delta

                if abs(delta) < tolerance:
                    return b

            return None
```

5. Example Usage

We have a sample array of x-axis values 'x' and an initial guess for 'b' in the code's example use section. With these inputs, the newton raphson function is invoked to get the value of 'b,' which is then reported.



```
In [6]:
        # Example usage
        x = [1, 2, 3, 4, 5,11]  # Sample array of x axis values
        n = len(x)  # Number of elements in the array
        initial_guess = 1  # Initial guess for b

        b = newton_raphson(x, n, initial_guess)
        print("The value of b is:", b)

        The value of b is: 6.617444900424222e-24
```

6. Conclusion

The Newton-Raphson method is a suitable numerical approach for determining the roots of a function that is especially effective for solving complicated equations like the one used to estimate the shape parameter 'b' in the Weibull distribution. We can easily determine the value of 'b' from provided lifespan data by using this approach, which is useful in a variety of applications, including reliability studies.