



Bootstrap Confidence Interval for Parameter 'p' of Truncated Negative Binomial Distribution

Muhammad Ibrahim Shamsi*

Department of Humanities & Science NUCES-FAST, Karachi Campus

Ghulam Hussain*

Department of Statistics, University of Karachi

ABSTRACT

Abstract: In formula based inferential statistics, many problems deal with the estimation of unknown parameters. This paper considers interval estimation. Two bootstrap confidence intervals for the parameter 'p' of an unknown population are discussed. They are obtained by the residual bootstrap and the two-stage wild bootstrap method. The results are illustrated with an example in which the investigated variable has the Truncated Negative Binomial distribution. We do not have to know the population distribution for determining the bootstrap confidence intervals for the parameters. This is the great advantage of bootstrap methods. The authors have developed a computer program that computes confidence limits using the procedure in this paper.

Keywords : Residual Bootstrap, Wild Bootstrap, Confidence Interval, Truncated Negative Binomial Distribution.

1) INTRODUCTION

Bootstrapping is a procedure where repeated samples are drawn from the sample, discriminant analysis is conducted on the samples drawn, and an error rate is computed. The overall error rate and its sampling distribution are obtained from the error rates of the repeated samples that are drawn. Bootstrapping techniques require a considerable amount of computer time. However, with the advent of fast and cheaper computing, they are gaining popularity as a viable procedure for obtaining sampling distribution of statistics whose theoretical sampling distributions are not known.

In the section 2 we briefly review the different methods of bootstrapping i.e. residuals method and wild bootstrap and their application for finite populations. Most of bootstrap literature is concerned with bootstrap implementations of tests, confidence intervals and application for estimation problems. It has been argued that for these problems bootstrap can be better understood if it is described as a plug-in method.

The motivation for this paper is to get rid of formula based inferential statistics, to approximate the sampling distribution for an unknown finite discrete population and to estimate their parameters.

* The material presented by the author does not necessarily portray the viewpoint of the editors and the management of the Institute of Business & Technology (BIZTEK) or Department of Humanities & Science NUCES-FAST, Karachi Campus and Department of Statistics, University of Karachi

* Muhammad Ibrahim Shamsi : ibrahim.shamsi@nu.edu.pk

* Ghulam Hussain : ghussain1212004@yahoo.com

2) LITERATURE REVIEW

The Bootstrap Technique

Bradly Efron (1979) has developed a new and major subpart of resampling procedure name as 'Bootstrap'. It has swept the field of statistics to an extra ordinary extent. Its idea is to develop a setup to generate more data using the information of the original data. True underlying sample properties are reproduced as closely as possible and unknown model characteristics are replaced by sample estimates.

$$Y_i = \beta X_i + \epsilon_i \quad ; \quad i = 1, 2, 3, \dots, N \quad (1)$$

Let us consider a finite population that can view as a realization of a certain super population model.

Where, Y_i 's are the measurement taken observations, β represents the regression parameter, X_i 's are assumed to be auxiliary quantities or past experiences. Also $\beta X_i = \mu_i$ are the true values of interest. The measurement errors ϵ_i 's are the identical and independent (i.i.d.) random variable such that

- $E(\epsilon_i) = 0$ and
- The variation is independent of μ , i.e. $E(\epsilon_i^2) = \sigma^2$

It cannot generally be assumed that the error distribution follows Normal with mean zero and variance σ^2 . To perform statistical test and to calculate confidence intervals the distribution of error ϵ_i has to be known. So the great challenge is how to gain information on the error distribution. Our motivation is to introduce the technique of Bootstrapping that aids us to extract the information about the residuals and to find the characteristics of the unknown population.

There are different types of bootstrap technique like;

1. Residuals bootstrap

Let $X = (X_1, X_2, \dots, X_n)$ be a random sample of size n drawn from the super population model (1). The bootstrap samples are defined as $\{(X_i, Y_i^*)\}$ with $Y_i = \hat{Y}_i + \hat{\epsilon}_i^*$, where \hat{Y}_i is an estimate of $\mu_i = \beta X_i$ and $\hat{\epsilon}_i$ is resampling from the residuals $\hat{\epsilon}_i = Y_i - \hat{Y}_i$. It cannot applicable under the Heteroscedastic error. So a very strong assumption made here is that the errors should be independently identical and hence uncorrelated with the design points X_i 's. [Chien Feng Chen (1998)].

2. Wild bootstrap

It is basically a two stage resampling scheme, that was first suggested by Wu (1986), modified by Liu (1988) and finally proposed and given its name by Hardle and Mammen (1993). It is a variant of residual method but in this procedure i.i.d. observations are drawn from an external random variable with first moment zero and second and third moments both being one. The wild bootstrap residuals are constructed through multiplying these i.i.d. observations by the residuals in place. This method has more freedom for the assumption of heteroscedasticity.

Chen and Sitter (1993) have proposed a two stage resampling scheme; first simple random sampling without replacement in each stratum and then show that the resulting bootstrap sample is second order efficient.

In our study we follow the same technique of two-stage wild bootstrapping. That is given a sample $S = \{i_1, i_2, \dots, i_n\}$ from a population u and any estimator $\hat{\beta}$ as define on model (1). Estimate the residuals $\hat{\epsilon}_i = Y - \hat{\beta} X_i$; $i \in S$, then generate n independent wild

bootstrap components $\{Z_1, \dots, Z_n\}$ of a random variable Z with $E(Z)=0$ and $E(Z^2)=1$ and set $Y_i^* = \beta X_i + \varepsilon_i^*$ with $S \varepsilon_i^* = \hat{\varepsilon}_i Z_i; i$ (2)
 Technically speaking, each wild bootstrap observation is drawn from a distribution that mimics its corresponding sampling distribution through matching up the first three moments.

3) METHODOLOGY & RESEARCH DESIGN

Our focus is to implement bootstrapping on truncated negative binomial distribution, to estimate its parameter ‘p’ and confidence interval for the same parameter. Let us define first the truncated negative binomial distribution.

The Truncated Negative Binomial Distribution

The Negative binomial distribution is a two parameter discrete distribution. It is use extensively for the description of data that are too heterogeneous to be fitted by the Poisson distribution. It can be defined in terms of the expansion of the negative binomial expansion $(Q-P)^{-k}$, where $Q = 1 + P, P > 0$, and k is positive real; the $(x + 1)^{th}$ term in the expansion yields $Pr[X = x]$.

Thus the negative binomial distribution with parameters k, P , is the distribution of the random variable X for which

$$Pr[X = x] = \binom{k+x-1}{k-1} \left(\frac{P}{Q}\right)^x \left(1 - \frac{P}{Q}\right)^k ; \quad x = 0, 1, 2, \dots \quad (3)$$

Where $Q = 1 + P, P > 0$ and $k > 0$.
 The mean and variance are

$$\mu = kP \quad \text{and} \quad \mu_2 = kP(1 + P). \quad (4)$$

This parameterization (but with the symbol p instead of P) is the one introduced by Fisher (1941). You may relate Q and P as; $Q = 1 / p$ and $P = q / p$ (i.e., $p = 1 / Q$ and $q = P / Q$).

Our aim is to apply the technique of bootstrapping on truncated negative binomial distributions, to estimate the parameter p and to construct the 95% confidence band. Then compare these results with the classical approach like maximum likelihood method, normal approximation.

In most common form of truncation, the zeroes are not recorded, thus the equation (3) becomes,

$$Pr[X = x] = (1 - Q^{-k})^{-1} \binom{k+x-1}{k-1} \left(\frac{P}{Q}\right)^x \left(1 - \frac{P}{Q}\right)^k ; \quad x = 1, 2, \dots, \quad (5)$$

With mean

$$\mu = E(X) = kP(1 - Q^{-k})^{-1} \quad (6)$$

and variance

$$\mu_2 = \frac{(kPQ + k^2 P^2)}{(1 - Q^{-k})} - \frac{k^2 P^2}{(1 - Q^{-k})^2} \quad (7)$$

Since the parameters of Negative binomial distribution is not easy to estimate. Its calculation

is often lengthy and time consuming. Different statisticians proposed some iterative procedures to overcome this problem. David and Johnson (1952) proposed an explicit estimator based on the first three sample moments. Sampford (1955) developed a reasonably rapid iterative technique for solving the two moment estimating equations but ultimately concluded that resulting estimates might only be suitable for use as first approximation in an iterative solution of Maximum likelihood estimating equations.

Illustrative Example

To illustrate estimation in the truncated negative binomial distribution using bootstrap method and other classical method, we considered a sample of chromosome breakage that was originally given by Sampford (1955). Where, the observed samples may be truncated, in the sense that the number of individuals falling into the zero class cannot be determined. For example, if chromosome breaks in irradiated tissue can occur only in those cells which are at a particular stage of the mitotic cycle at the time of irradiation, a cell can be demonstrated to have been at that stage only if breaks actually occur. Thus in the distribution of breaks per cell, cells not susceptible to breakage are indistinguishable from susceptible cells in which no breaks occur. The sample data are as follows:

X	1	2	3	4	5	6	7	8	9	10	11	12	13
Obs	11	6	4	5	0	1	0	2	1	0	1	0	1

The sample mean = 3.438, and sample variance = 9.931. The moment estimates of k and P are 0.632842 and 3.26216 respectively.

These results are obtained by the program coded on Mathematica.

```
d=FindRoot[{m==kp(1-(1+p)^(-k))^(-1),m2 (kp(1+p)+k^2 p^2)/(1-(1+p)^(-k))-(k^2 p^2)/(1-(1+p)^(-k))^2},{k,4},{p,3}]
```

```
{kg 0.632842,pg 3.26216}
```

Since $p = 1 / Q$ and $Q = 1 + P$, therefore $= 0.23462282$ (8)

Estimation of confidence interval

Brass Estimate

For this example Cohen (1965) calculated the estimate for p as

$$\hat{p}^b = 0.2345 \tag{9}$$

and

$$V(\hat{p}^b) = 0.0098628$$

And the 95% confidence interval for \hat{p}^b is:

$$0.2345 \pm (1.96)(0.09931163) = (0.039849, 0.429151) \tag{10}$$

Maximum Likelihood Estimate

The maximum likelihood estimate for this example, calculated by Sampford (1955) as,

$$\hat{p}^b = 0.2113 \tag{11}$$

and

$$V(\hat{p}^b) = 0.0097231$$

And the 95% confidence interval for \hat{p}^b is:

$$0.2113 \pm (1.96)(0.09860578) = (0.018033, 0.404567) \tag{12}$$

4) RESULTS & CONCLUSION

The purpose of this paper is to describe some methods of constructing confidence intervals for the parameter p of Truncated Negative Binomial Distribution. Various methods, classical as well as bootstrap, have been described with example illustrating Sampford (1955) the application of each procedure. The algorithm for bootstrapping is designed on C++. For sample one program code for wild bootstrap confidence interval is given on appendix. Others are with author and can be shown on demand. In order to assist the reader in assessing the options of the methods for construction of confidence intervals, the results of the example considered in the article are summarized below:

Methods of Estimate		95% Confidence Interval for true parameter 'p'		Length of Confidence Interval
		LL	UL	
Classical	Brass Estimate	0.039849	0.429151	0.389302
	Maximum Likelihood	0.018033	0.404567	0.386534
Bootstrap	Standard Bootstrap	0.100502	0.423390	0.322888
	Residual Bootstrap	0.123139	0.389920	0.266781
	Wild Bootstrap	0.122231	0.356407	0.234176

Where, LL = Lower limit and UL = Upper Limit

By Comparing the classical method with bootstrap, it is vivid that wild bootstrap provides the confidence bound approximately as precise as the classical method, whereas standard bootstrap method has a little bit wide confidence interval. And overall we can conclude that the bootstrapping provides more easy and sufficient method for finding the confidence intervals.

REFERENCES

- CHEN, J. AND SITTE, R.R. (1993), "Edge worth expansion and the Bootstrap for the stratified without replacement from a finite population", Canadian Journal of Statistics, 21, 347-357
- CHIEN-FENG, C. (1998), "Bootstrapping the order selection test", Dissertation, National Chengchi University, Taipei, Taiwan.
- COHEN, A.C. (1965), "Estimation in the Negative Binomial Distribution", T.R.14, Department of Statistics, University of Georgia, Athens
- DAVID, F.N., AND JOHNSON, N.L. (1952), "The truncated Poisson", Biometrics, 8, 275-285
- EFRON, B. (1979), "Bootstrap Methods: Another Look at the Jackknife", the Annals of Statistics, 7, 1-26.
- FISHER, R.A. (1941), "The Negative Binomial Distribution", Annals of Eugenics, London, 111 182-187
- HARDLE, W., AND MAMMEN, E. (1993), "Comparing Nonparametric Versus Parametric Regression Fits", The Annals of Statistics, 21, 1926-1947

- Helmers, R., and Wegkamp M. (1998), "Wild Bootstrapping in finite populations", Scandinavian Journal of Statistics, USA, 25 383-399
- Ibrahim M. (2004), "Recent trends in Bootstrapping Technique – Wild Bootstrap" Report submitted in partial fulfillment of M.Sc., Department of Statistics University of Karachi, Pakistan.
- Khurshid A., Ageel M.I. and Raheeq A. (2005), "On confidence intervals for the negative binomial distribution", Investigaciones Operacionales, 26, 59-70.
- Liu, R.Y. (1988), "Bootstrap Procedures under some non i.i.d. models", The Annals of Statistics, 16, 1696-1708
- Sampford, M.R. (1955), "The truncated Negative Binomial Distribution", Biometrika, 42, 58-69
- Wu, C.F.J. (1986), "Jackknife, Bootstrap and other resampling methods in regression analysis", the Annals of Statistics, 14, 1261-1343

APPENDIX

The algorithm for bootstrapping is designed on C++. For sample one program code for wild bootstrap confidence interval is given below. Others are with authors and can be shown on demand.

```
#include<iostream>
#include<fstream>
using namespace std;

float B = -0.6263;

int power(int num , int pow)
{
    return (pow == 0) ? 1 : num * power(num,pow-1);
}

void Select_Sort(float arr[],int length)
{
    int min = 0;
    for (int i = 0 ; i < length ; i++)
    {
        min = i;
        for (int j = i ; j < length ; j++)
            min = (arr[min] >= arr[j]) ? j : min;
        if (min != i)
        {
            float temp = arr[min];
            arr[min] = arr[i];
            arr[i] = temp;
        }
    }
}

int main(int argno, char **argv)
{
```

```
srand(time(NULL));
if (argno == 4)
{
    int n = 13 , t = 0;
    cout << "Enter number of Bootstrap Samples : ";
    cin >> t;
    int *xi,*yi,*yci,*ei;
    float *mean,*variance,*p;
    float **esi;
    float **ysi;
    float **zi;
    xi = new int [n];
    yi = new int [n];
    yci = new int [n];
    ei = new int [n];
    mean = new float [t];
    variance = new float [t];
    p = new float [t];
    ifstream infile(argv[1]);
    int num = 0;
    char temp = 'a';
    while (temp != '\n')
    infile.get(temp);
    infile.get();

    for (int i = 0 ; i < n ; i++)
    {
        infile >> num;
        infile.get();
        infile >> num;
        xi[i] = num;
        infile.get();
        infile >> num;
        yi[i] = num;
        infile.get();
        infile >> num;
        yci[i] = num;
        infile.get();
        infile >> num;
        ei[i] = num;
        infile.get();
    }
    esi = new float * [t];
    ysi = new float * [t];
    zi = new float * [t];
    for (int i = 0 ; i < t ; i++)
    {
        *(esi + i) = new float [n];
        *(ysi + i) = new float [n];
        *(zi + i) = new float [n];
    }

    float arr[2] = {-0.6180,1.6180};

    int count = 0, cp = 0;
    float **xf = new float * [t] , **xsqf = new float * [t];
    for (int i = 0 ; i < t ; i++)
```

Bootstrap Confidence Interval for Parameter 'p' of Truncated Negative Binomial Distribution

```
{
  *(xf+i) = new float [n];
  *(xsqf+i) = new float [n];
}
float *sumx,*sf,*sxf,*sxsqf,*syi,*sysi,*sei,*sesi,*szi;
sumx = new float [t];
sf = new float [t];
sxf = new float [t];
sxsqf = new float [t];
syi = new float [t];
sysi = new float [t];
sei = new float [t];
sesi = new float [t];
szi = new float [t];

float s1,s2,s3,s4,s5,s6,s7,s8,s9;
int psc = 0;

for ( ; cp < t ; )
{
  s1 = s2 = s3 = s4 = s5 = s6 = s7 = s8 = s9 = 0;
  for (int j = 0 ; j < n ; j++)
  {
    zi[cp][j] = arr[rand() % 2];
    esi[cp][j] = zi[cp][j] * ei[j];
    ysi[cp][j] = (xi[j] * B) + esi[cp][j];

    xf[cp][j] = xi[j] * ysi[cp][j];

    int pow = power(xi[j],2);
    xsqf[cp][j] = ysi[cp][j] * pow;
    s1 += xi[j];
    s2 += ysi[cp][j];
    s3 += xf[cp][j];
    s4 += xsqf[cp][j];
    s5 += yci[j];
    s6 += ysi[cp][j];
    s7 += ei[j];
    s8 += esi[cp][j];
    s9 += zi[cp][j];
  }
  sumx[cp] = s1;
  sf[cp] = s2;
  sxf[cp] = s3;
  sxsqf[cp] = s4;
  syi[cp] = s5;
  sysi[cp] = s6;
  sei[cp] = s7;
  sesi[cp] = s8;
  szi[cp] = s9;
  mean[cp] = sxf[cp]/sf[cp];
  variance[cp] = ((s4/s2) - (mean[cp] * mean[cp]));
  float check = 0;
  if (variance[cp] > 0)
  {
    check = (variance[cp]/mean[cp]);
    check -= 1;
  }
}
```

```
    }
    else
    check = -9999;

    if (check < 0)
    count++;
    else
    {
    p[psc++] = check;
    cp++;
    }
}

infile.close();
ofstream out(argv[2]);
ofstream out2(argv[3]);
out2 << "P's," << "Q = 1+P," << "p = 1/Q" << endl;

for (int i = 0 ; i < t ; i++)
{
    out << "Reading Number : " << (i+1) << endl << endl;
    out << "S.No.," << "Xi (x)," << "Yi," << "Yi* (f)," << "Ei," << "Ei*," << "Xf,"
    << "X2f," << "Zi" << endl;
    for (int j = 0 ; j < n ; j++)
    out << (j+1) << "," << xi[j] << "," << yci[j] << "," << ysi[i][j] << "," << ei[j] <<
    "," << esi[i][j] << "," << xf[i][j] << "," << xsqf[i][j] << "," << zi[i][j] << endl;
    out << "Total: ," << sumx[i] << "," << syi[i] << "," << sf[i] << "," << sei[i] << ","
    << sesi[i] << "," << sxf[i] << "," << xsqf[i] << "," << szi[i] << endl << endl;
    out << "Mean : ," << mean[i] << endl;
    out << "Variance : ," << variance[i] << endl;
    out << "P : ," << p[i];
    out << endl << endl << endl;
}

float sump = 0;
Select_Sort(p,t);
for (int i = 1 ; i < t ; i++)
{
    (p[i] >= 0) ? sump += p[i] : p[i];
    out2 << p[i] << "," << (1+p[i]) << "," << (1/(1+p[i])) << endl;
}

out2 << endl << "Sum : " << sump << endl << endl;
out2 << "Average : ," << (sump/t) << endl << endl;
out2 << endl << "There Were " << (count+cp) << " P's But " << count << " P's Were
Dropped As They Were Negative" << endl;

out2 << "P [25th] :," << (1/(1 + p[974]))/2.8 << endl;
out2 << "P [975th] :," << (1/(1 + p[24]))/2.8 << endl;
out2.close();
out.close();

for (int i = 0 ; i < t ; i++)
{
    delete[] *(xf+i);
    delete[] *(xsqf+i);
    delete[] *(ysi+i);
```

Bootstrap Confidence Interval for Parameter 'p' of Truncated Negative Binomial Distribution

```
        delete[] *(esi+i);
    }
    delete[] xf;
    delete[] xsqf;
    delete[] ysi;
    delete[] esi;
    delete[] xi;
    delete[] yi;
    delete[] yci;
    delete[] ei;
    delete[] mean;
    delete[] variance;
    delete[] p;

    cout << "Output Is Generated In : " << argv[2] << endl;
    cout << "And The List Of P's Are Generated In : " << argv[3] << endl;
}
else
{
    cout << "Usage Error ..... !" << endl;
    cout << "Use It As : ";
    cout << "Program.exe InputFile.csv OutputFile.csv Pfile.csv" << endl;
}
system("pause");
return 0;
}
```