REVSTAT – Statistical Journal
Volume 20, Number 5, October 2022, Supplementary Material, 1–9

Supplementary Material to the Paper Titled "Likelihood-Based Finite Sample Inference for Synthetic Data from Pareto Model"

Authors: Nutan Mishra ✉
– Department of Mathematics and Statistics,
University of South Alabama,
Mobile, AL 36688, USA
nmishra@southalabama.edu

Sandip Barui
– Quantitative Methods and Operations Management Area,
Indian Institute of Management Kozhikode,
Kozhikode, Kerala 673570, India
sandipbarui@iimk.ac.in

Received: November 2019 Revised: March 2021 Accepted: March 2021

The notations given in the main article titled "Likelihood-Based Finite Sample Inference for Synthetic Data from Pareto Model" are to be followed. In the supplementary file, we provide necessary details of the algorithms in form of codes written in `Mathematica` and `R`.

1. ALGORITHM 1

Table 1 in the main article contains the results of the simulations to construct exact shortest $(1-\gamma)\%$ confidence interval for C when ψ is known. Recall that the formula is $\left(\tilde{C}e^{-\kappa_2}, \tilde{C}e^{-\kappa_1}\right)$.

1. Assign values of n, γ and ψ according to the parameter setting.
2. Compute κ_1 and κ_2 using the following R-code.

✉ Corresponding author.

R Code 1:

```
library(cubature)
library(nleqslv)
library(rootSolve)

    numsol=function(w_all){
    w1=w_all[1]; w2=w_all[2];
    f1=function(t){
        a1=(n^2)*(psi^2)*(t)*exp(-n*psi*t)
        return(a1)
    }
    A1=integrate(f1,0,w1)$value
    A2=integrate(f1,0,w2)$value
    k1=A2-A1-(1-gam)
    k2=exp(w2)*f1(w2)-exp(w1)*f1(w1)
    return(c(k1,k2))
    }
    nleqslv(c(0.001,0.01),numsol)
```

3. Assign value to C for simulation.
4. Draw a sample of size n from Pareto with parameter C and ψ.
5. Compute the maximum likelihood (ML) estimates of C and ψ. Generate a synthetic data of size n from Pareto with the ML estimates of C and ψ.
6. Compute estimate of C using the generated synthetic data.
7. Compute the unbiased estimate of C, estimate of the standard error, bias, MSE, lower and upper confidence limits using the following Mathematica code.

Mathematica Code 1:

```
ParetoCcurl[n0_, c0_, psi0_]:=
Module[{n = n0, c = c0, psi = psi0},
OriginaldataX = RandomVariate[ParetoDistribution[c, psi], n];
MLEs = FindDistributionParameters[OriginaldataX, ParetoDistribution[g, h]];
MLEC = Values[MLEs[[1]]]; MLET = Values[MLEs[[2]]];
y = Sort[RandomVariate[ParetoDistribution[MLEC, MLET], n]];
Ccurl = y[[1]];
Ccurlunbiased = ((n*psi-1)^2)/((n*psi)^2) y[[1]];
VaCcurl = N[((n*psi)^2)/((n*psi-1)^2(n*psi-2)^2) * (1 + (n*psi(n*psi-2))/((n*psi-1)^2)) * Ccurl^2];
VaCcurlunbiased = ((n*psi-1)^4)/((n*psi)^4) VaCcurl;
SDCcurl = N[VaCcurl^.5];
SDCVaCcurlunbiased = N[VaCcurlunbiased^.5];
mybias = c - Ccurlunbiased;
myMSE = mybias^2 + VaCcurlunbiased;
myU = Ccurl * Exp[-k1];
myL = Ccurl * Exp[-k2];
success = Boole[LC < c&&c < UC];
Bias = CstarUnbiased - c;
{Ccurlunbiased, SDCVaCcurlunbias success, myMSE}];
```

2. ALGORITHM 2

In Table 2 of the main article, we presented simulation results for estimating C when ψ is unknown. The symmetric exact $(1-\gamma)100\%$ confidence interval is given by $\left(\tilde{C}e^{-\kappa_2/\psi^*}, \tilde{C}e^{-\kappa_1\psi^*}\right)$.

1. Choose n and γ.
2. Compute κ_1 and κ_2 using the following R-code.

R Code 2:

```
library(cubature)
library(nleqslv)
library(rootSolve)

cov=0;
for(j in 1:nsim){
print(j)
U=runif(n,0,1)
X=C/((1-U)^(1/psi))
Chat=min(X)
psihat=n/(sum(log(X/Chat)))
Y=Chat/((1-U)^(1/psihat))
psist=n/(sum(log(Y/min(Y))))
f1xci=function(var){
  k=var[1]; tx=var[2];
  a1=(n-1)*log(n)+log(n-1)-log(gamma(n-1))
  a2=exp(-n/tx)/((tx-1))
  a3=(tx+k)^(-n) - ((tx*(k+1))^(-n))
  f=exp(a1)*a2*a3
  return(f)
}
numsol1=function(w_val){
  A3=adaptIntegrate(f1xci, lowerLimit =c(0,0), upperLimit =
  c(w_val,Inf))$integral-(gam/2)
  return(A3)
}
numsol2=function(w_val){
  A3=adaptIntegrate(f1xci, lowerLimit =c(0,0), upperLimit =
  c(w_val,Inf))$integral-(1-gam/2)
  return(A3)
}
lclval=nleqslv(0.001,numsol1)$x
uclval=nleqslv(0.005,numsol2)$x
lcl=min(Y)*exp(-uclval/psist)
ucl=min(Y)*exp(-lclval/psist)
if(lcl<=C&&ucl>=C){cov=cov+1}
print(c(lcl,ucl))
}
```

3. Choose C and ψ for simulation, and run the following Mathematica code.

Mathematica Code 2:

ParetoCInterval[n0_, c0_, Psi0_]:=
Module[{n = n0, c = c0, Psi = Psi0},
OriginaldataX = RandomVariate[ParetoDistribution[c, Psi], n];
MLEs = FindDistributionParameters[OriginaldataX, ParetoDistribution[g, h]];
MLEC = Values[MLEs[[1]]]; MLET = Values[MLEs[[2]]];
y = Sort[RandomVariate[ParetoDistribution[MLEC, MLET], n]]; Ccurl = y[[1]];
Psistar = $\frac{n}{\text{Log}[\ \prod_{i=1}^{n}(y[[i]]/y[[1]])]}$;
I1 = NIntegrate $\left[\frac{\text{Exp}[-n/t]}{(t\text{^}(n-1))*((n*\text{Psistar}*t)-1)}, \{t, 1/(n*\text{Psistar}), \infty\}\right]$;
I2 = NIntegrate $\left[\frac{\text{Exp}[-n/t]}{(t\text{^}(n-1))*((n*\text{Psistar}*t)-2)}, \{t, 2/(n*\text{Psistar}), \infty\}\right]$;
CstarUnbiased = $\frac{(\text{Gamma}[n-1]*(n*\text{Psistar}-1))}{(n\text{^}(n+1)*\text{Psistar}\text{^}2*\text{I1})}$ * Ccurl;
VaCstarUnbiased = $\left(\frac{\text{Gamma}[n-1]*(n*\text{Psistar}-1)\text{^}2*\text{I2}}{n\text{^}(n+1)*(n*\text{Psistar}-2)*\text{Psistar}\text{^}2*(\text{I1})\text{^}2} - 1\right)$ Ccurl^2;
SDCstarUnbiased = VaCstarUnbiased^.5;
LC = Ccurl * Exp[−w2/Psistar]; UC = Ccurl * Exp[−w1/Psistar];
success = Boole[LC < c&&c < UC];
Bias = CstarUnbiased − c;
myMSE = Bias^2 + VaCstarUnbiased;
{CstarUnbiased, SDCstarUnbiased, Bias, myMSE, success}];

3. ALGORITHM 3

Table 3 in the main article deals with estimation of ψ when C is known. The exact shortest $(1-\gamma)100\%$ confidence interval is given by $\left(\frac{n(\log A)^{-1}}{\kappa_2}, \frac{n(\log A)^{-1}}{\kappa_1}\right)$ where $A = C^{-n}\prod_{i=1}^{n} Y_i$.

1. Choose n and γ.
2. Compute κ_1 and κ_2 using the following R-code.

R Code 3:

```
library(cubature)
library(nleqslv)
library(rootSolve)

numsol=function(w_all){
  w1=w_all[1]; w2=w_all[2];
  f1=function(t){
       a1=(1/t)*((m)^(-n-1))
       a2=exp(-n*(1/t+t/m))
       a3=2*n*log(n)-2 *log(gamma(n))
```

```
      a=exp(a3)*a2*a1
      return(a)
  }
  F1=function(vari){
    t=vari[1];w=vari[2];
    a1=(1/t)*((w)^(-n-1))
    a2=exp(-n*(1/t+t/w))
    a3=2*n*log(n)-2 *log(gamma(n))
    a=exp(a3)*a2*a1
    return(a)
  }
  A1=adaptIntegrate(F1, lowerLimit = c(0, 0),
  upperLimit = c(Inf, w1))$integral
  A2=adaptIntegrate(F1, lowerLimit = c(0, 0),
  upperLimit = c(Inf, w2))$integral
  A2-A1
  m=w1
  if(m<=0){m=10^-3}
  B1=integrate(f1,0,10)$value
  m=w2
  if(m<=0){m=10^-2}
  B2=integrate(f1,0,10)$value
  k1=A2-A1-(1-gam)
  k2=(w2^2)*B2-(w1^2)*B1
  return(c(k1,k2))
}
nleqslv(c(0.7,1.6),numsol)
```

3. Choose the true values of C and ψ for simulation, and run the following Mathematica code.

Mathematica Code 3:

ParetoPsicurl[n0_, c0_, Psi0_]:=
Module[{n = n0, c = c0, Psi = Psi0},
OriginaldataX = RandomVariate[ParetoDistribution[c, Psi], n];
MLEs = FindDistributionParameters[OriginaldataX, ParetoDistribution[g, h]];
MLEC = Values[MLEs[[1]]]; MLET = Values[MLEs[[2]]];
y = Sort[RandomVariate[ParetoDistribution[MLEC, MLET], n]];
Psicurl = $\frac{n}{\text{Log}\left[\prod_{i=1}^{n}(y[[i]]/c)\right]}$;
Psicurlunbiased = $\frac{(n-1)\text{^}2}{(n)\text{^}2}\frac{n}{\text{Log}\left[\prod_{i=1}^{n}(y[[i]]/c)\right]}$;
VaPsicurlunbiased = $N\left[\left(\frac{(2n-3)}{(n-2)\text{^}2}\right) * \text{Psicurlunbiased}\text{^}2\right]$;
SDPsicurlunbiased = N[VaPsicurlunbiased^.5];
mybias = Psi − Psicurlunbiased;
myU = $\frac{n}{\text{Log}\left[\prod_{i=1}^{n}(y[[i]]/c)\right]}\text{w1}\text{^} - 1$;
myL = $\frac{n}{\text{Log}\left[\prod_{i=1}^{n}(y[[i]]/c)\right]}\text{w2}\text{^} - 1$;
myMSE = mybias^2 + VaPsicurlunbiased;
success = Boole[LC < c&&c < UC];
{Psicurlunbiased, SDPsicurlunbiased, mybias,myMSE, success}];

4. ALGORITHM 4

Table 4 in the main article estimates the parameter ψ when C is unknown. The exact shortest (1-γ)100% confidence interval is given by $\left(\frac{n(\log B)^{-1}}{\kappa_2}, \frac{n(\log B)^{-1}}{\kappa_1}\right)$ where $B = Y_{(1)}^{-n}\prod_{i=1}^{n} Y_i$.

1. Choose n and γ.
2. Apply following R-code to compute κ_1 and κ_2.

R Code 4:

```
library(cubature)
library(nleqslv)
library(rootSolve)

numsol=function(w_all){
  w1=w_all[1]; w2=w_all[2];
  f1=function(t){
    a1=(1/t)*((m)^(-n))
    a2=exp(-n*(1/t+t/m))
    a3=(2*n-2)*log(n)-2 *log(gamma(n-1))
    a=exp(a3)*a2*a1
    return(a)
  }
  F1=function(vari){
    t=vari[1];w=vari[2];
    a1=(1/t)*((w)^(-n))
    a2=exp(-n*(1/t+t/w))
    a3=(2*n-2)*log(n)-2 *log(gamma(n-1))
    a=exp(a3)*a2*a1
    return(a)
  }
  A1=adaptIntegrate(F1, lowerLimit = c(0, 0),
  upperLimit = c(Inf, w1))$integral
  A2=adaptIntegrate(F1, lowerLimit = c(0, 0),
  upperLimit = c(Inf, w2))$integral
  A2-A1
  m=w1
  if(m<=0){m=10^-3}
  B1=integrate(f1,0,10)$value
  m=w2
  if(m<=0){m=10^-2}
  B2=integrate(f1,0,10)$value
  k1=A2-A1-(1-gam)
  k2=(w2^2)*B2-(w1^2)*B1
  return(c(k1,k2))
}
nleqslv(c(0.7,1.6),numsol)
```

3. Choose true values of C and ψ, and run the following Mathematica code to get estimates.

Mathematica Code 4:

ParetoPsistar[n0_, c0_, Psi0_]:=
Module[{n = n0, c = c0, Psi = Psi0},
OriginaldataX = RandomVariate[ParetoDistribution[c, Psi], n];
MLEs = FindDistributionParameters[OriginaldataX, ParetoDistribution[g, h]];
MLEC = Values[MLEs[[1]]]; MLET = Values[MLEs[[2]]];
y = Sort[RandomVariate[ParetoDistribution[MLEC, MLET], n]];
Psistar = $\frac{n}{\text{Log}\left[\prod_{i=1}^{n}(y[[i]]/y[[1]])\right]}$;
Psistarunbiased = $\frac{(n-2)\text{\^{}}2}{(n)\text{\^{}}2}\frac{n}{\text{Log}\left[\prod_{i=1}^{n}(y[[i]]/y[[1]])\right]}$;
VaPsistarunbiased = $N\left[\left(\frac{(2n-5)}{(n-3)\text{\^{}}2}\right) * \text{Psistarunbiased\^{}}2\right]$;
SDPsistarunbiased = N[VaPsistarunbiased\^{}.5];
mybias = Psi − Psistarunbiased;
myU = $\frac{n}{\text{Log}\left[\prod_{i=1}^{n}(y[[i]]/y[[1]])\right]}$w1u\^{} − 1;
myL = $\frac{n}{\text{Log}\left[\prod_{i=1}^{n}(y[[i]]/y[[1]])\right]}$w2u\^{} − 1;
myMSE = mybias\^{}2 + VaPsistarunbiased;
success = Boole[LC < c&&c < UC];
{Psistarunbiased, SDPsistarunbiased,mybias, myMSE, success};]

5. ALGORITHM 5

Tables 5 and 6, respectively, provide $(1-\gamma)100\%$ Bayesian predictive shortest intervals for ψ when C is known. The confidence intervals are computed using the formula

$$\left(\tilde{\psi}\omega_2^{-1}, \tilde{\psi}\omega_1^{-1}\right).$$

The two tables are created with two sets of hyper parameter values needed in the prior distributions. Thus same algorithm is used to create these two tables. The algorithm steps are described as follows:

1. Choose the values of the hyper parameters, namely, d and c_0.

2. Choose n and γ.

3. Run the following R-code to compute ω_1 and ω_2 for the above settings.

R Code 5:

```
library(cubature)
library(nleqslv)
library(rootSolve)

loggam=function(q){
  return(sum(log(seq(1,q-1,1))))
}
numsol=function(w_all){
  w1=w_all[1]; w2=w_all[2];
  f1=function(t){
    a1=t^(2*n+c0-1)
    a2=exp(-(1/m)*(n*t/(1-t)))
    a3=1/(1-t)
    a4=1/(m^(n+1))
    a5=n*log(n)-2*loggam(n)+loggam(2*n+c0)-loggam(n+c0)
    a6=exp(a5)
    a=a1*a2*a3*a4*a6
    return(a)
  }
  F1=function(vari){
    t=vari[1]; wval=vari[2];
    a1=t^(2*n+c0-1)
    a2=exp(-(1/wval)*(n*t/(1-t)))
    a3=1/(1-t)
    a4=1/(wval^(n+1))
    a5=n*log(n)-2*loggam(n)+loggam(2*n+c0)-loggam(n+c0)
    a6=exp(a5)
    a=a1*a2*a3*a4*a6
    return(a)
  }
  A1=adaptIntegrate(F1, lowerLimit = c(0, 0),
  upperLimit = c(1, w1))$integral
  A2=adaptIntegrate(F1, lowerLimit = c(0, 0),
  upperLimit = c(1, w2))$integral
  A2-A1
  m=w1
  B1=integrate(f1,0,1)$value
  m=w2
  B2=integrate(f1,0,1)$value
  k1=A2-A1-(1-gam)
  k2=(w2^2)*B2-(w1^2)*B1
  return(c(k1,k2))
}
strN=nleqslv(c(0.7,1.6),numsol)
tw1=strN$x[1]
tw2=strN$x[2]
A1=adaptIntegrate(F1, lowerLimit = c(0, 0),
upperLimit = c(1,tw1 ))$integral
A2=adaptIntegrate(F1, lowerLimit = c(0, 0),
upperLimit = c(1, tw2))$integral
A2-A1
```

4. Using ω_1 and ω_2 computed above run the following Mathematica code to compute the estimates.

Mathematica Code 5:

ParetoPsiInterval[n0_, c0_, Psi0_]:=
Module[{n = n0, c = c0, Psi = Psi0},
OriginaldataX = Sort[RandomVariate[ParetoDistribution[c, Psi], n]];
u = Log [$\prod_{i=1}^{n}$(OriginaldataX[[i]]/OriginaldataX[[1]])] ;
Psistar = RandomVariate[GammaDistribution[n, 1/u], 1];
y = Sort[RandomVariate[ParetoDistribution[c, Psistar[[1]]], n]];
Psicurl = $\frac{n}{\text{Log}\left[\ \prod_{i=1}^{n}(y[[i]]/y[[1]])\right]}$;
M[i_]:=Integrate $\left[\frac{z\text{^}(n-1)*\text{Exp}[-z*\text{Psi}]}{(z+d)\text{^}i}, \{z, 0, \infty\}\right]$;
ExpectedPsicurl = $N\left[\frac{n(n+\text{c1})*\text{Psi}\text{^}n}{(n-1)\text{Gamma}[n]} * \left(\frac{\text{Gamma}[n-1]}{\text{Psicurl}\text{^}(n-1)}\right)\right]$;
I1 = $\frac{n\text{^}2*(n+\text{c1})*(n+\text{c1}+1)*\text{Psicurl}\text{^}n}{(n-1)\text{^}2*(n-2)*\text{Gamma}[n]} * \frac{\text{Gamma}[n-2]}{\text{Psicurl}\text{^}(n-2)}$;
I2 = $\frac{n\text{^}2*(n+\text{c1})*\text{Psicurl}\text{^}n}{(n-1)\text{^}2*\text{Gamma}[n]}$*
$\left(\left((n+\text{c1}+1) * \frac{\text{Gamma}[n-2]}{\text{Psicurl}\text{^}(n-2)}\right) - \frac{(n+\text{c1})*\text{Psicurl}\text{^}n}{\text{Gamma}[n]} * \left(\frac{\text{Gamma}[n-1]}{\text{Psicurl}\text{^}(n-1)}\right)\text{^}2\right)$;
VaPsicurl = I1 + I2;
SDPsicurl = VaPsicurl^.5;
LC = Psicurl/w2; UC = Psicurl/w1;
success = Boole[LC < Psi&&Psi < UC];
Bias = Psicurl − Psi;
myMSE = Bias^2 + VaPsicurl;
{Psicurl, SDPsicurl, Bias, myMSE, success}];