Project on Bootstrap

Luyan Hao

27 October 2019

1 Generate 10000 points from a N(0,1) distribution and throw away any point with value > 1. How many points do you expect to throw away? How many points did you throw away? (Call the points that remain – those N(0,1)s that you generated with values <= 1 – the residual data set.)

```
> # generate points from normal distribution with a settled seed 321
> set.seed(321)
> pointsGenerate <- function(n, m, s){
+   ListOfPoints <- rnorm(n,m,s)
+   ListOfPoints
+ }
> A <- pointsGenerate(10000,0,1)
> # generate residual points
> residualGenerate <- function(l){
+   rsd = l [l <= 1]
+   n =  length(l)-length(rsd)
+   print(sprintf("Number of points we expected to throw away: %s", floor(10000*(1-pno
+   print(sprintf("Number of points we throw away: %s", n))
+   print(sprintf("Number of points remain in the residual set: %s", length(rsd)))
+   rsd
+ }
> R <- residualGenerate(A)

[1] "Number of points we expected to throw away: 1586"
[1] "Number of points we throw away: 1572"
[1] "Number of points remain in the residual set: 8428"
```

Our full data set for 10000 points from normal distribution are stored as A, and the residual set with elements less than 1 are stored as R.

2 Compute (and report) the sample mean and sample standard deviation of the residual data set.

```
> print(sprintf("Mean of the residual set: %s", mean(R)))

[1] "Mean of the residual set: -0.280218885473395"

> print(sprintf("Standard Deviation of the residual set: %s", sd(R)))

[1] "Standard Deviation of the residual set: 0.790269442333382"
```

Answer: From the result we see that the mean of our residual set is $m = -0.280218885473395$, the standard deviation is $sd = 0.790269442333382$.

3 Note that the data in the residual data set do not have a N(0,1) distribution. Plot a histogram of the data in the residual data set and write down a formula for the density of the data in the residual data set. (The formula should involve the density and distribution function of a standard normal random variable.)

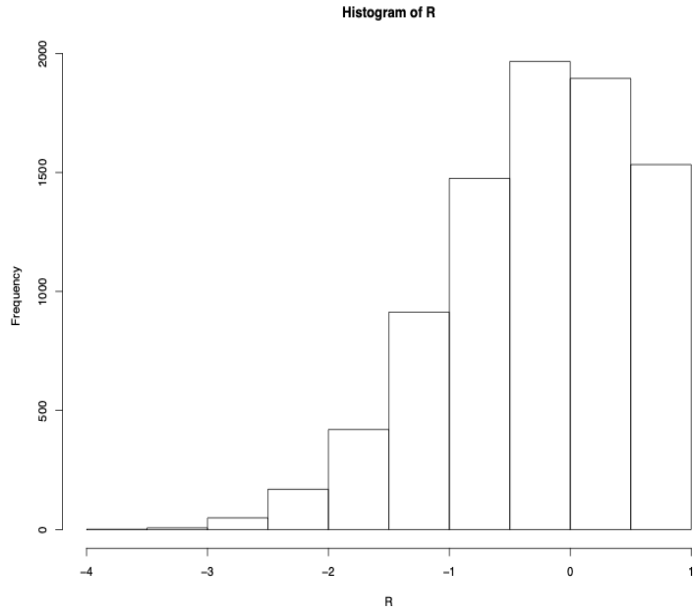
```
> h = hist(R,main="Histogram of the Residual set",col="darkmagenta")

> pdf("figure1.pdf",width=11,height=8.5) ;
> plot(h)
```

Answer: The result pdf is devided into two parts. When x > 1, f(x) = 0. When x <= 1, f(x) = p(x)/P(1). In the above function, the p(x) is the pdf funciton for N(0,1), P(x) is the cdf funciton for N(0,1).

4 Assume that the data in the residual data set came f corded. Write down a formula for the density of such a random variable.

```
> denFun <- function(x,m,s){
+   if(x <= 1){
+   numerator = dnorm(x, mean = m, sd = s)
+   deno = pnorm(1, mean = m, sd = s)
+   output = numerator/deno}
```




```
+   else{output = 0}
+   output
+ }
> d = denFun(1,0,1)
> d

[1] 0.2876
```

Answer: Above is the R code for the pdf expression with m,s parameter. It's the same function in the Q3 with replace the N(0,1) into a distribution N(m,s)

5 Use maximum likelihood to estimate the parameters m and s from the residual data set, using the density you derived in question 4.

```
> MLECalculateResidual <- function(R){
+   # The function Loglikelihood1 calculate the sum of all log-values for the given de
+   Loglikelihood1 <- function(R,m,s){
+     logVector = log(denFun(R,m,s))
+     output = sum(logVector)
+     output
```

```
+   }
+   # f is used to prepare our log-function to become a better form in order to use in
+   # Since nlm only calculate minmum of the function, we take negative before the log
+   f = function(p){
+     f = - Loglikelihood1(R,p[1],p[2])
+   }
+   p = c(0.5,1)
+   model = nlm(f, p, hessian = FALSE)
+   # debug print line:
+   # print(sprintf('The maximum likelihood attended at (m = %s,s = %f)' , model$estim
+   model$estimate
+ }
> MLECalculateResidual(R)

[1] 0.008895827 0.997287083

> print(sprintf('The maximum likelihood attended at (m = %s,s = %f)' , MLECalculateRes

[1] "The maximum likelihood attended at (m = 0.00889582730134702,s = 0.997287)"
```

Answer: Above is the R code for the calculating the MLE based on the residual data set.

6 Use the bootstrap to compute standard errors for the MLEs you computed in question 5.

```
> bootStrapGenerate <- function(dataSet,rep){
+   n = length(dataSet)
+   A = matrix(data = NA,nrow = n,ncol = rep)
+   for(i in 1:rep){
+     A[,i] = sample(dataSet, n, replace = TRUE, prob = NULL)
+   }
+   A
+ }
> B = 1
> M = bootStrapGenerate(R,B)
> mlist = rep(0,B)
> slist = rep(0,B)
> # Use this for loop to calculate the m,s with each column in our bootstrap matrix
> for(i in 1:ncol(M))
+ {
+   mlist[i] = MLECalculateResidual(M[,i])[1]
+   slist[i] = MLECalculateResidual(M[,i])[2]
+ }
> # Calculate the Sample Standard Deviation
```

```
> msigma_hat = mean(mlist)
> ssigma_hat = mean(slist)
> # This msigma is our bootstrap estimate of standard error for m
> msigma = sqrt(sum((mlist - msigma_hat)^2)/(B-1))
> print(sprintf('The bootstrap estimate of standard error for m = %s)' , msigma))

[1] "The bootstrap estimate of standard error for m = NaN)"

> # This ssigma is our bootstrap estimate of standard error for s
> ssigma = sqrt(sum((slist - ssigma_hat)^2)/(B-1))
> print(sprintf('The bootstrap estimate of standard error for s = %s)' , ssigma))

[1] "The bootstrap estimate of standard error for s = NaN)"

>
```

Answer: The bootstrap Generator function takes our data vector and repeat time as parameter, with the output a matrix which have the coloumn by bootstrap vectors. Usually rep is settled between 50 to 200 based on the paper.

7 Show that, if the heights of the rank 9 elliptic curves are supposed to have a density of the form, where h is the height, then the log of the heights has a Gamma distribution with shape and scale depending on a and b; that is, write down a formula for the density of the log-heights.

```
> # First we input the data
> Tel <- read.csv("/Users/lulu/Downloads/tmp-20190928.csv", header = TRUE)
> attach(Tel)
> TestData = Tel[,]
> TestData = log(TestData)
> # Answer: # TestData is the log-height of our data, which suppose to have a gamma di
> # The Gamma distribution with parameters shape = shape and scale = scale has density
> # f(x)= 1/(scale^shape Gamma(shape))*x^(shape-1)*e^-(x/scale) for x >= 0, shape > 0 .
> # where Gamma is defined by Gamma(x) = integral_0^Inf t^(x-1)*exp(-t)dt
> # If we wirte them into the form of R code
> # dgamma(x, shape = shape, scale = scale, log = FALSE)
> # pgamma(q = x, shape = shape, scale = scale)
>
> # x = lgh:  x ~ (1/gamma(a)*b^a)*x^(a-1)*e^(-a/b)
> # f(h) = h^a*(log h)^b
```

```
> # then y = g(h) = ln(h)
> # g^-1(h) = e^y
> # pdf for /y = ln(h)/ : e^(y*(a+1))*y^b
> # since it is of the form of Gamma distribution
> # a = -1 - 1/theta
> # b = k - 1
> # as k be the shape
> # theta as the scale
```

8 Given that no height above 2^24 was recorded, write down a formula for the density of the data in the elliptic curve data set.(This formula should involve the density and distribution function of a Gamma random variable.)

```
> pdfforGammaWithTruncate <- function(x, shape, scale){
+   if(x < log(2^24))
+   {p = dgamma(x, shape = shape, scale = scale, log = FALSE)/pgamma(log(2^24), shape =
+   else{p = 0}
+   p
+ }
```

9 Apply maximum likelihood to estimate the shape and scale parameters for the Gamma from which the elliptic curve data (log heights) were most likely generated.

```
> # I use the initial guess calculate based on the mean and variance of our data 45 an
> MLECalculatorForlnh <- function(R){
+     Loglikelihood1 <- function(R,shape,scale){
+     logVector = -log(pdfforGammaWithTruncate(R,shape, scale))
+     output = sum(logVector)
+     output
+   }
+     f = function(p){
+     f = Loglikelihood1(R,p[1],p[2])
+   }
+     p = c(45,2)
+     model = nlm(f, p)
+     model$estimate
```

```
+ }
> shape = MLECalculatorForlnh(TestData)[1]
> scale = MLECalculatorForlnh(TestData)[2]
> print(sprintf('The maximum likelihood attended at (shape = %s,scale = %f)' , shape,

[1] "The maximum likelihood attended at (shape = 45.3503189358953,scale = 0.494349)"
```

10 Use these estimates to compute the MLEs of the parameters a and b of interest.

```
> a = -1 - 1/scale
> b = shape - 1
> print(sprintf('The maximum likelihood attended at (a = %s,b = %f)' , a, b))

[1] "The maximum likelihood attended at (a = -3.02286250894234,b = 44.350319)"
```

11 Apply the bootstrap to compute standard errors for both the MLEs of shape and scale and the MLEs of the original parameters, a and b.

```
> N = bootStrapGenerate(TestData,B)
> shapelist = rep(0,B)
> scalelist = rep(0,B)
> # Use this for loop to calculate the shape,scale with each column in our bootstrap m
> for(i in 1:ncol(N))
+ {
+   shapelist[i] = MLECalculatorForlnh(N[,i])[1]
+   scalelist[i] = MLECalculatorForlnh(N[,i])[2]
+ }
> alist = -1 - 1/(scalelist)
> blist = shapelist - 1
> # Calculate the Sample Standard Deviation
> shapesigma_hat = mean(shapelist)
> scalesigma_hat = mean(scalelist)
> # This shapesigma is our bootstrap estimate of standard error for shape
> shapesigma = sqrt(sum((shapelist - shapesigma_hat)^2)/(B-1))
> print(sprintf('The bootstrap estimate of standard error for shape = %s)' , shapesigm

[1] "The bootstrap estimate of standard error for shape = NaN)"

> # This scalesigma is our bootstrap estimate of standard error for scale
> scalesigma = sqrt(sum((scalelist - scalesigma_hat)^2)/(B-1))
> print(sprintf('The bootstrap estimate of standard error for scale = %s)' , scalesigm
```

```
[1] "The bootstrap estimate of standard error for scale = NaN)"
```

```
> asigma_hat = mean(alist)
> bsigma_hat = mean(blist)
> asigma = sqrt(sum((alist - asigma_hat)^2)/(B-1))
> print(sprintf('The bootstrap estimate of standard error for a = %s)' , asigma))
```

```
[1] "The bootstrap estimate of standard error for a = NaN)"
```

```
> bsigma = sqrt(sum((blist - bsigma_hat)^2)/(B-1))
> print(sprintf('The bootstrap estimate of standard error for b = %s)' , bsigma))
```

```
[1] "The bootstrap estimate of standard error for b = NaN)"
```

12 Make a scatter plot of the estimates of shape and scale for each of the individual bootstrap samples. (How strong is the dependence between the shape and scale estimates?)

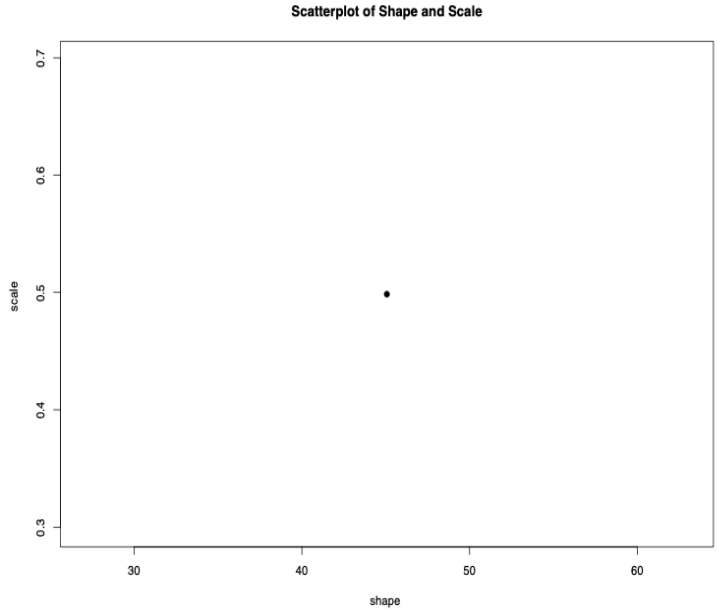
```
> # Simple Scatterplot for shape and scale
> pdf("figure2.pdf",width=11,height=8.5) ;
> plot(shapelist, scalelist, main="Scatterplot of Shape and Scale",
+      xlab="shape ", ylab="scale ", pch=19)
> cors_s = cor(shapelist, scalelist, method = c("pearson", "kendall", "spearman"))
> print(sprintf('The correlation between shape and scale is cor = %s)' , cors_s))
```

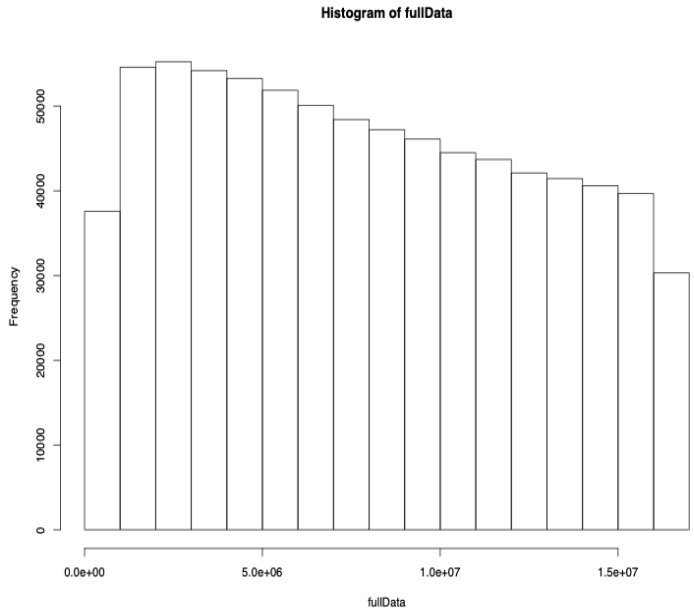
```
[1] "The correlation between shape and scale is cor = NA)"
```

```
> # We see that the correlation is quite significant
```

13 Plot of histogram of the raw height data and overlay the estimated density (scaled correctly), as a curve.

```
> pdf("figure3.pdf",width=11,height=8.5) ;
> library(ggplot2)
> fullData = Tel[,]
> fun.13 <- function(h) {(1/h)*(pdfforGammaWithTruncate(log(h),shape=shape,scale=scale
> hpic <- hist(fullData,breaks=15)
> xhist<-c(min(hpic$breaks),hpic$breaks)
> yhist<-c(0,hpic$density,0)
> xfit<-seq(min(fullData),max(fullData),length=1e6)
> yfit<-fun.13(xfit)
> plot(xhist,yhist,type="s",ylim=c(0,max(yhist,yfit)), main='Normal pdf and histogram'
> lines(xfit,yfit, col='red')
```


Scatterplot of Shape and Scale
scale
0.3
0.4
0.5
0.6
0.7
30
40
50
60
shape


Histogram of fullData
Frequency
0
10000
20000
30000
40000
50000
0.0e+00
5.0e+06
1.0e+07
1.5e+07
fullData

14 Given the estimated parameters, what portion of the total collection of rank 9 elliptic curves are you seeing? That is, what is the probability that a Gamma random variable with those parameters is $<= log(2^(24))$?

```
> per = pgamma(log(2^(24)), shape =  shape,  scale = scale,lower.tail = TRUE)
> print(sprintf('The portion of the total collection of rank 9 elliptic curves are you

[1] "The portion of the total collection of rank 9 elliptic curves are you seeing is p =
```