Question_2

Pawan Kumar

2022-11-15

Problem 2

Q.1. Write a function in R which will compute the MLE of   = log( ) using

`*optim function in R. You can name it MyMLE*`

MyMLE <- function(pars,y){ alpha <- pars[1] sigma <- pars[2] logl <- sum(-log(gamma(alpha))-alpha*log(sigma)+(alpha-1)*log(y)-y/sigma) return(-logl) } optim(pars,MyMLE,y)

Q.2 Choose n=20, and alpha=1.5 and sigma=2.2

1. *(i). Simulate {X1 ,X2 , · · · ,Xn } from rgamma(n=20,shape=1.5,scale=2.2)*
2. *(ii). Apply the MyMLE to estimate   and append the value in a vector*
3. *(iii) Repeat the step (i) and (ii) 1000 times*
4. *(iv) Draw histogram of the estimated MLEs of .*
5. *(v) Draw a vertical line using abline function at the true value of*
6. *(vi) Check if the gap between 2.5 and 97.5-percentile points are shrinking as sample size n is increasing?*

```
rgamma(n=20,shape=1.5,scale=2.2)  #simulation of data for particular given
```

```
##  [1]  1.0009493  6.9211669  6.6162455  2.3872732  6.2294467  7.8117031
##  [7]  1.7515631  5.6169397  4.6252907  1.5288666  2.2722642 11.3661980
## [13]  0.4349043  2.3733008  1.3774733  4.1902459  0.7698479  4.3860904
## [19]  1.6450360  2.1173572
```

```
#values of respective parameters


MyMLE <- function(pars,y){
  alpha <- pars[1]
  sigma <- pars[2]
  logl <- sum(-log(gamma(alpha))-alpha*log(sigma)+(alpha-1)*log(y)-y/sigma)
  return(-logl)
}
optim(c(1.5,2.2),MyMLE,y = rgamma(n=20,shape=1.5,scale=2.2)) #MyMLE function
```

```
## $par
## [1] 1.524050 2.041544
##
```

```
## $value
## [1] 41.74499
##
## $counts
## function gradient
##       45       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

```
ne<- function(x){
  MyMLE <- function(pars,y){
    alpha <- pars[1]
    sigma <- pars[2]
    logl <- sum(-log(gamma(alpha))-alpha*log(sigma)+(alpha-1)*log(y)-y/sigma)
    return(-logl)
  }
  optim(c(1.5,2.2),MyMLE,y = rgamma(n=20,shape=1.5,scale=2.2))
}


n <- 1000
#give 1 to function new n times
dat <- lapply(rep(1,n),ne)
#replicate 1000 times
#re1000 <- replicate(n, ne(1), simplify=FALSE)
#head(re1000)
#making a dummy function
lap <- lapply(seq_len(n), function(x) ne(1))
head(lap)
```

```
## [[1]]
## [[1]]$par
## [1] 1.450453 1.658162
##
## [[1]]$value
## [1] 36.79052
##
## [[1]]$counts
## function gradient
##       55       NA
##
## [[1]]$convergence
## [1] 0
##
## [[1]]$message
## NULL
##
##
## [[2]]
## [[2]]$par
```

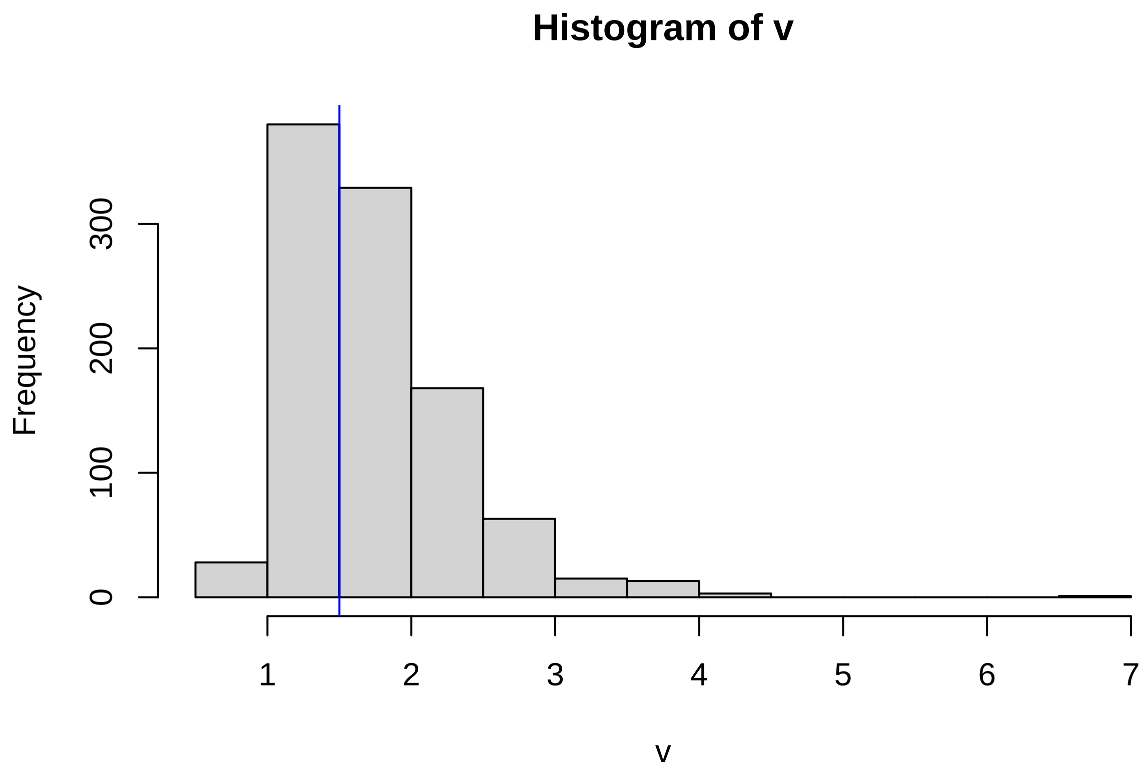
```
## [1] 2.491092 1.775981
##
## [[2]]$value
## [1] 46.04381
##
## [[2]]$counts
## function gradient
##       51       NA
##
## [[2]]$convergence
## [1] 0
##
## [[2]]$message
## NULL
##
##
## [[3]]
## [[3]]$par
## [1] 2.683453 1.313120
##
## [[3]]$value
## [1] 40.97622
##
## [[3]]$counts
## function gradient
##       55       NA
##
## [[3]]$convergence
## [1] 0
##
## [[3]]$message
## NULL
##
##
## [[4]]
## [[4]]$par
## [1] 1.237936 3.450946
##
## [[4]]$value
## [1] 48.77445
##
## [[4]]$counts
## function gradient
##       53       NA
##
## [[4]]$convergence
## [1] 0
##
## [[4]]$message
## NULL
##
##
## [[5]]
## [[5]]$par
```

```
## [1] 2.356291 1.516565
##
## [[5]]$value
## [1] 42.1357
##
## [[5]]$counts
## function gradient
##       55       NA
##
## [[5]]$convergence
## [1] 0
##
## [[5]]$message
## NULL
##
##
## [[6]]
## [[6]]$par
## [1] 2.752459 1.482182
##
## [[6]]$value
## [1] 43.72371
##
## [[6]]$counts
## function gradient
##       53       NA
##
## [[6]]$convergence
## [1] 0
##
## [[6]]$message
## NULL
```

```r
v <- c()
for(i in 1:1000){
  v <- append(v,dat[[i]][[1]][1])
}                                  #repetation of MyMLE 1000 times and
#appending vector for corresponding value of theta

head(v)    #pulling few data
```

```
## [1] 1.284671 2.169263 1.533335 1.759508 1.608820 1.450176
```

```r
hist(v)    #plotting histogram

abline(v = 1.5,col='blue') # for drawing vertical line using abline function
```




```
quant = quantile(v, probs = c(2.5/100,97.5/100))
diff = quant[2]-quant[1]
paste('2.5 percentile point is', quant[1])
```

```
## [1] "2.5 percentile point is 0.997404497969434"
```

```
paste("97.5 percentile point is ", quant[2])
```

```
## [1] "97.5 percentile point is  3.21668945833986"
```

```
paste(" The gap between 2.5 and 97.5-percentile for n=20, and alpha=1.5 and sigma=2.2 is ",diff)
```

```
## [1] " The gap between 2.5 and 97.5-percentile for n=20, and alpha=1.5 and sigma=2.2 is  2.2192849603
```

Q.3 Choose n=40, and alpha=1.5 and repeat the (2).

```
rgamma(n=40,shape=1.5,scale=2.2)  #simulation of data for particular given
```

```
##  [1] 3.9672123 3.6183375 9.9768570 1.2779829 0.5815070 7.4726024 2.1216578
##  [8] 1.3830538 5.2901240 7.2746198 0.8900509 4.6637345 1.2192899 0.4952831
## [15] 1.1166562 5.5153988 2.3400015 0.1596237 1.1203351 1.9764111 1.6967178
```

```
## [22] 8.4906469 2.6511609 4.1356515 0.5778183 3.1012390 7.5291264 1.5754941
## [29] 2.7278540 0.8282763 6.5258364 4.2447207 2.6293170 1.1791884 3.0757275
## [36] 4.8801808 1.0963045 6.0144341 2.6180060 0.8909645
```

```
#values ofrespective parameters

MyMLE <- function(pars,y){
  alpha <- pars[1]
  sigma <- pars[2]
  logl <- sum(-log(gamma(alpha))-alpha*log(sigma)+(alpha-1)*log(y)-y/sigma)
  return(-logl)
}
optim(c(1.5,2.2),MyMLE,y = rgamma(n=40,shape=1.5,scale=2.2)) #MyMLE function for
```

```
## $par
## [1] 1.242014 2.328148
##
## $value
## [1] 81.91984
##
## $counts
## function gradient
##       47       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

```
#particular given values of respective parameters


ne<- function(x){
  MyMLE <- function(pars,y){
    alpha <- pars[1]
    sigma <- pars[2]
    logl <- sum(-log(gamma(alpha))-alpha*log(sigma)+(alpha-1)*log(y)-y/sigma)
    return(-logl)
  }
  optim(c(1.5,2.2),MyMLE,y = rgamma(n=40,shape=1.5,scale=2.2))
}


n <- 1000
#give 1 to function new n times
dat <- lapply(rep(1,n),ne)
#replicate 1000 times
#replicate(n, ne(1), simplify=FALSE)
#making a dummy function
lap1 <- lapply(seq_len(n), function(x) ne(1))
head(lap1)
```

```
## [[1]]
```

```
## [[1]]$par
## [1] 1.430188 2.213254
##
## [[1]]$value
## [1] 84.67874
##
## [[1]]$counts
## function gradient
##       43       NA
##
## [[1]]$convergence
## [1] 0
##
## [[1]]$message
## NULL
##
##
## [[2]]
## [[2]]$par
## [1] 1.961946 1.999182
##
## [[2]]$value
## [1] 90.25281
##
## [[2]]$counts
## function gradient
##       43       NA
##
## [[2]]$convergence
## [1] 0
##
## [[2]]$message
## NULL
##
##
## [[3]]
## [[3]]$par
## [1] 1.415752 2.031455
##
## [[3]]$value
## [1] 80.92033
##
## [[3]]$counts
## function gradient
##       47       NA
##
## [[3]]$convergence
## [1] 0
##
## [[3]]$message
## NULL
##
##
## [[4]]
```

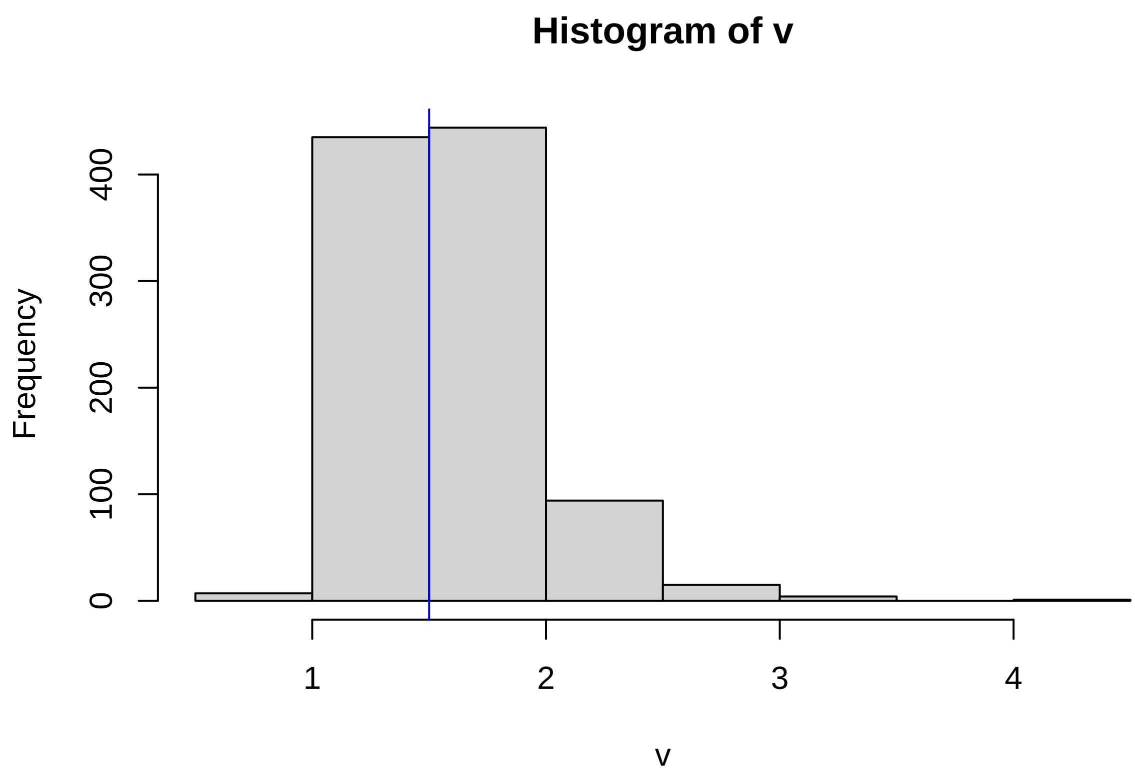
```
## [[4]]$par
## [1] 1.746163 1.842438
##
## [[4]]$value
## [1] 83.57627
##
## [[4]]$counts
## function gradient
##       49       NA
##
## [[4]]$convergence
## [1] 0
##
## [[4]]$message
## NULL
##
##
## [[5]]
## [[5]]$par
## [1] 1.312851 2.466615
##
## [[5]]$value
## [1] 86.15083
##
## [[5]]$counts
## function gradient
##       47       NA
##
## [[5]]$convergence
## [1] 0
##
## [[5]]$message
## NULL
##
##
## [[6]]
## [[6]]$par
## [1] 1.820741 2.157519
##
## [[6]]$value
## [1] 91.12969
##
## [[6]]$counts
## function gradient
##       43       NA
##
## [[6]]$convergence
## [1] 0
##
## [[6]]$message
## NULL
```

```
v <- c()
for(i in 1:1000){
```

```r
  v <- append(v,dat[[i]][[1]][1])
}                              #repetition of MyMLE 1000 times and
#appending vector for corresponding value of theta.


head(v) #pulling few data
```

```
## [1] 1.896284 2.062778 1.232529 1.886809 1.620025 1.614548
```

```r
hist(v)  #plotting histogram



abline(v = 1.5,col='blue')    # for drawing vertical line using abline function
```




```r
quant = quantile(v, probs = c(2.5/100,97.5/100))
diff = quant[2]-quant[1]
paste('2.5 percentile point is', quant[1])
```

```
## [1] "2.5 percentile point is 1.06883533557529"
```

```r
paste("97.5 percentile point is ", quant[2])
```

```
## [1] "97.5 percentile point is  2.43966997923378"
```

```r
paste(" The gap between 2.5 and 97.5-percentile for n=40, and alpha=1.5 and
      sigma=2.2 is ",diff)
```

```
## [1] " The gap between 2.5 and 97.5-percentile for n=40, and alpha=1.5 and \n      sigma=2.2 is  1.37
```

Q.4 Choose n=100, and alpha=1.5 and repeat the (2).

```r
rgamma(n=100,shape=1.5,scale=2.2)#simulation of data for particular given values
```

```
##   [1] 10.02511145  0.87211217  5.72728208  0.28567662  2.67655929  0.70922781
##   [7]  2.26104675  0.28030791  0.17070588  4.40724341  4.29824326  1.06921938
##  [13]  1.67618129  1.96952285  7.73705869  1.10071207  1.30602365  2.81879854
##  [19]  1.72852898  1.17957863  6.89984089  4.85078239  2.15135955  8.01408440
##  [25]  1.79131262  3.95151999  1.58344974  0.70948020  2.62814249  0.03520425
##  [31]  7.43021296  1.10270684  3.29730837  0.21500607  1.91094840  5.10908684
##  [37]  1.03617468  3.96148055  3.56519774  4.72748105  3.66035778  0.70791470
##  [43]  4.77079633  0.96445925  6.66888605  0.80329143  4.01959918  2.28191278
##  [49]  1.88334014  1.56383363  2.61871592  2.75468450  0.14772126  1.57941900
##  [55]  2.14942378  3.63822987  0.54284176  2.37818082  4.41421925  7.72129412
##  [61]  1.74129089  0.38496310  0.28634581  5.01011209  2.81025062  0.86137371
##  [67]  1.27072176  4.46189819  6.26596067  4.45793011  2.89494488  2.33086052
##  [73]  0.46885199  1.26543169  5.75633183  1.04294153  4.75763761  4.28624265
##  [79]  2.58197194  0.44965034 10.34983648  7.01122397  3.41390028  0.69818844
##  [85]  0.53741874  2.80132895  6.12834999  1.76014086  3.59366121  2.82747441
##  [91]  1.32162412  0.94341167  1.13341407  6.10889201  1.19585043  1.43582432
##  [97]  4.15487864  0.96768597  2.25014493  3.26566895
```

```r
#of respective parameters

MyMLE <- function(pars,y){
  alpha <- pars[1]
  sigma <- pars[2]
  logl <- sum(-log(gamma(alpha))-alpha*log(sigma)+(alpha-1)*log(y)-y/sigma)
  return(-logl)
}
optim(c(1.5,2.2),MyMLE,y = rgamma(n=100,shape=1.5,scale=2.2))  # MyMLE function
```

```
## $par
## [1] 2.181193 1.448566
##
## $value
## [1] 200.8383
##
## $counts
## function gradient
##       49       NA
##
## $convergence
## [1] 0
##
```

```
## $message
## NULL
```

```r
#for particular given values of parameters




ne<- function(x){
  MyMLE <- function(pars,y){
    alpha <- pars[1]
    sigma <- pars[2]
    logl <- sum(-log(gamma(alpha))-alpha*log(sigma)+(alpha-1)*log(y)-y/sigma)
    return(-logl)
  }
  optim(c(1.5,2.2),MyMLE,y = rgamma(n=100,shape=1.5,scale=2.2))
}


n <- 1000
#give 1 to function new n times
dat <- lapply(rep(1,n),ne)
#replicate 1000 times
#replicate(n, ne(1), simplify=FALSE)
#making a dummy function
lap2 <- lapply(seq_len(n), function(x) ne(1))
head(lap2)
```

```
## [[1]]
## [[1]]$par
## [1] 1.709344 1.738785
##
## [[1]]$value
## [1] 201.5578
##
## [[1]]$counts
## function gradient
##       59       NA
##
## [[1]]$convergence
## [1] 0
##
## [[1]]$message
## NULL
##
##
## [[2]]
## [[2]]$par
## [1] 1.855750 1.737703
##
## [[2]]$value
## [1] 207.5815
##
## [[2]]$counts
```

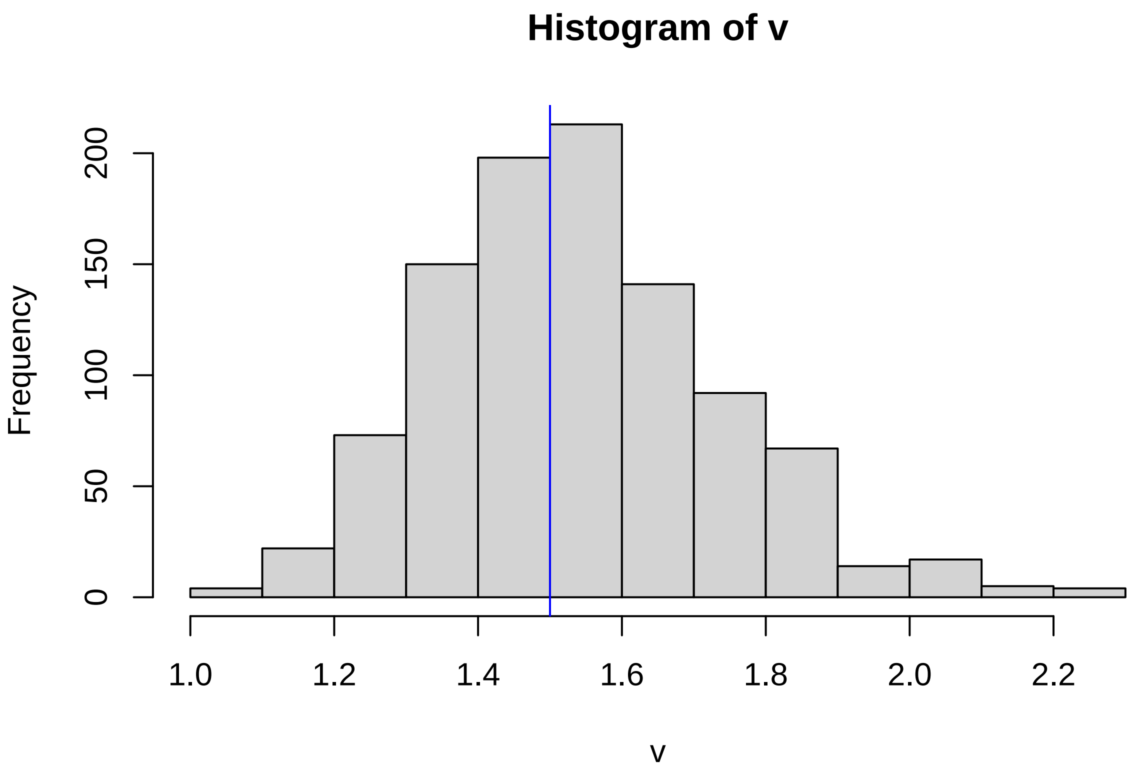
```
## function gradient
##       45       NA
##
## [[2]]$convergence
## [1] 0
##
## [[2]]$message
## NULL
##
##
## [[3]]
## [[3]]$par
## [1] 1.763906 1.717601
##
## [[3]]$value
## [1] 202.6893
##
## [[3]]$counts
## function gradient
##       49       NA
##
## [[3]]$convergence
## [1] 0
##
## [[3]]$message
## NULL
##
##
## [[4]]
## [[4]]$par
## [1] 1.819739 1.935359
##
## [[4]]$value
## [1] 216.9236
##
## [[4]]$counts
## function gradient
##       43       NA
##
## [[4]]$convergence
## [1] 0
##
## [[4]]$message
## NULL
##
##
## [[5]]
## [[5]]$par
## [1] 1.921821 1.591569
##
## [[5]]$value
## [1] 201.3433
##
## [[5]]$counts
```

```
## function gradient
##       91       NA
##
## [[5]]$convergence
## [1] 0
##
## [[5]]$message
## NULL
##
##
## [[6]]
## [[6]]$par
## [1] 1.599834 2.269361
##
## [[6]]$value
## [1] 223.1251
##
## [[6]]$counts
## function gradient
##       39       NA
##
## [[6]]$convergence
## [1] 0
##
## [[6]]$message
## NULL
```

```r
v <- c()
for(i in 1:1000){
  v <- append(v,dat[[i]][[1]][1])
}                                  #repetation of MyMLE 1000 times and
#appending vector for corresponding value of theta.


head(v) #pulling only few data
```

```
## [1] 1.746884 1.443561 1.363733 1.212774 1.566346 1.637655
```

```r
hist(v)  #plotting histogram



abline(v = 1.5,col='blue')  #drawing vertical line using abline function
```




```
quant = quantile(v, probs = c(2.5/100,97.5/100))
diff = quant[2]-quant[1]
paste('2.5 percentile point is', quant[1])
```

```
## [1] "2.5 percentile point is 1.19848148464983"
```

```
paste("97.5 percentile point is ", quant[2])
```

```
## [1] "97.5 percentile point is  2.00460992156331"
```

```
paste(" The gap between 2.5 and 97.5-percentile for n=100, and alpha=1.5 and sigma=2.2 is ",diff)
```

```
## [1] " The gap between 2.5 and 97.5-percentile for n=100, and alpha=1.5 and sigma=2.2 is  0.80612843
```

Q.5 Check if the gap between 2.5 and 97.5-percentile points are shrinking

as sample size n is increasing?

- For n =20, the gap is approx. ~2.2154 for n = 40, the gap is approx. ~1.2814 for n = 100, the gap is approx. ~0.8006 it turns out the gap between 2.5 and 97.5 percentile points is shrinking as the sample size increases.