# Bangladesh Data Analysis

Gustavo Cepparo and Milica Cudina

As before, first we import the data.

```
bangladesh=read.csv("bangladesh-data.csv", header=TRUE)
names(bangladesh)
```

```
## [1] "Arsenic"  "Chlorine" "Cobalt"
```

```
#accessing single columns
#bangladesh$Arsenic
attach(bangladesh)
Arsenic
```

```
##   [1] 2400    6  904  321 1280  151  141 1050  511  688   81    8   37    6   22
##  [16]   43   39   92  253  200  255 1150 1180    9  107    6  149    6   46   13
##  [31]    6  150    6  189  364   42  390    6  270  248  139    6   82   82  256
##  [46]  165    6  180   86    6   38  262  404    8   85   98    6   22    6    6
##  [61]    6   15  103   86    6   46   62   43    6    6   55    6  107   65  276
##  [76]  114    6    6    6   65  142  194    6   54  702    6  986  153   84   16
##  [91] 1460  306   49   36  106    6   41   84  278   41
##  [ reached 'max' / getOption("max.print") -- omitted 171 entries ]
```

```
mean(Arsenic)
```

```
## [1] 125.3199
```

```
var(Arsenic)
```

```
## [1] 88789.39
```

```
#hist(Arsenic)
#what are the dimensions of `bangladesh`
dim(bangladesh)
```

```
## [1] 271   3
```

```
#see if there are missing data
bangladesh=na.omit(bangladesh)
#what are the "new" dimensions of `bangladesh`
dim(bangladesh)
```

```
## [1] 268   3
```

```
attach(bangladesh)
```

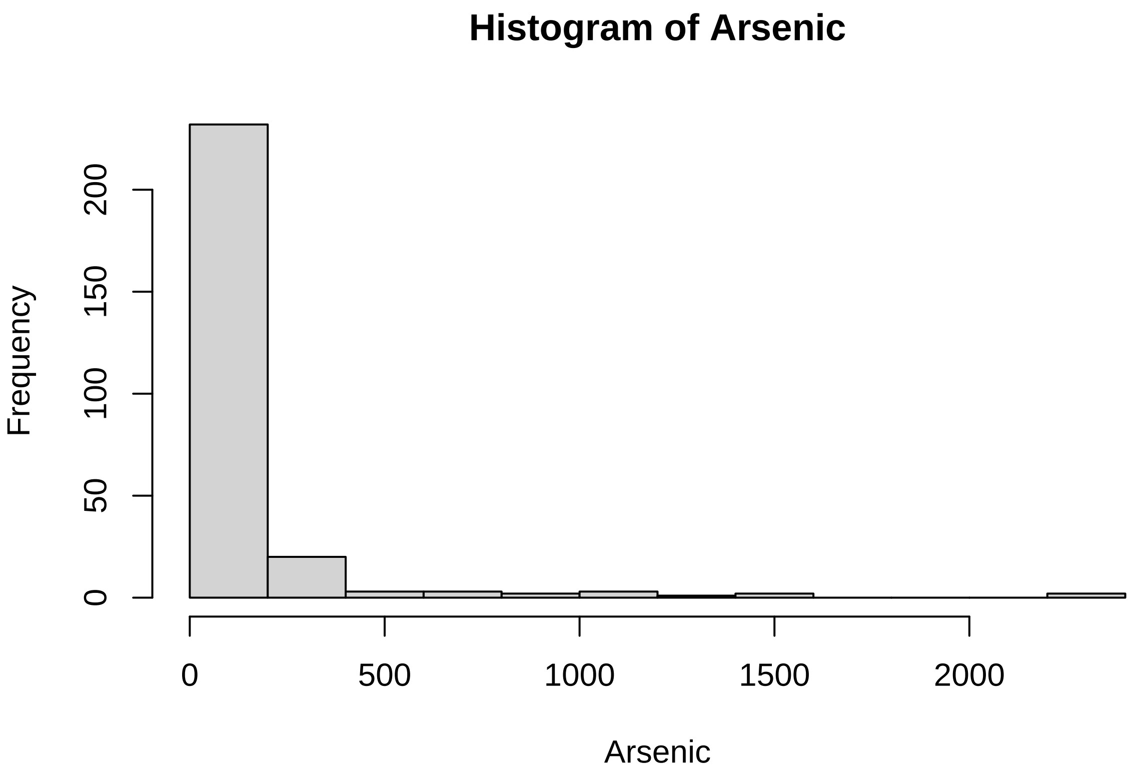
```
## The following objects are masked from bangladesh (pos = 3):
##
##     Arsenic, Chlorine, Cobalt
```

```
n=length(Arsenic)
n
```

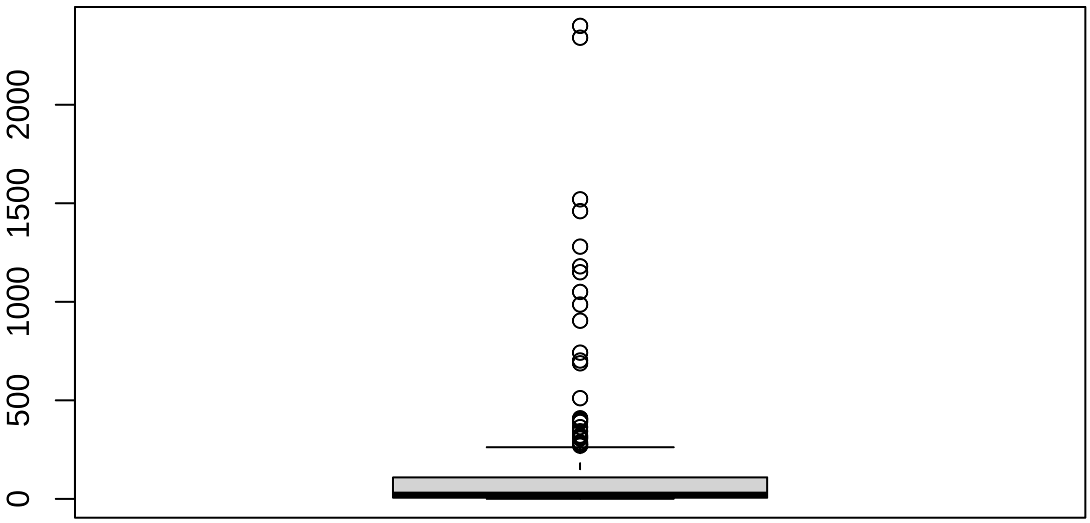
```
## [1] 268
```

Again, we undertake a rudimentary exploratory data analysis.

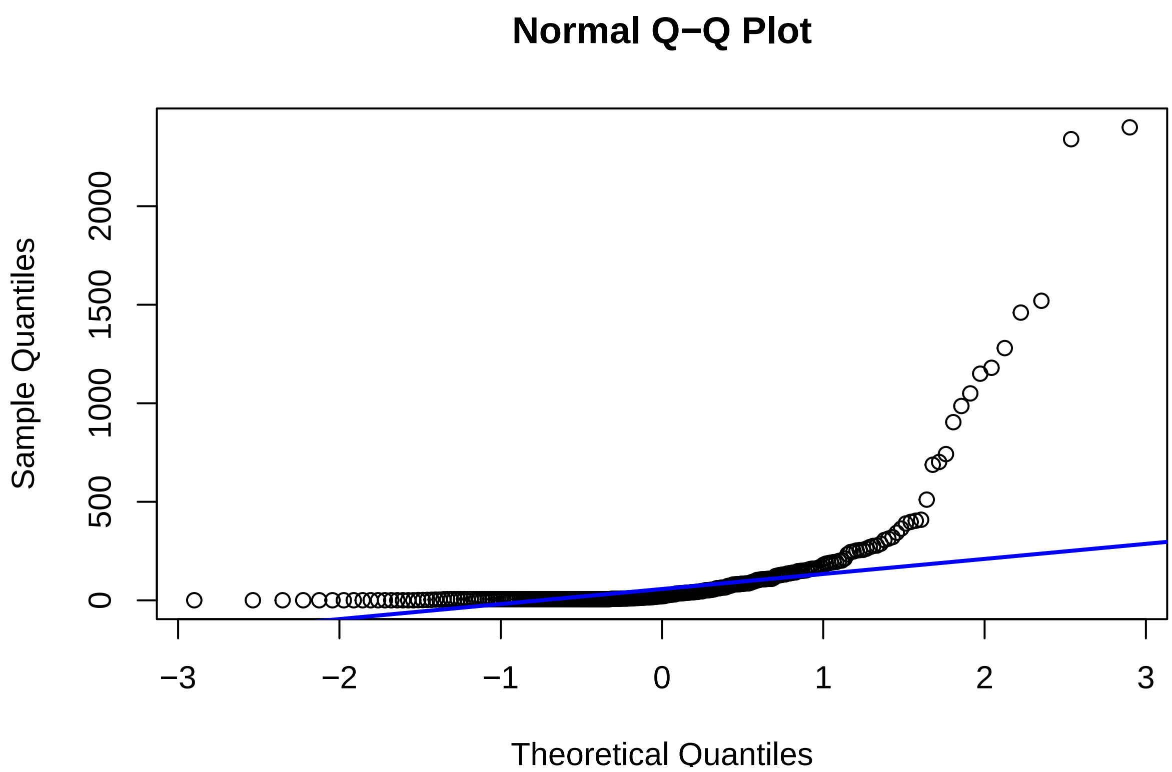
```r
hist(Arsenic)
```




```r
boxplot(Arsenic)
```




```r
qqnorm(Arsenic)
qqline(Arsenic, col="blue", lwd=2)
```




What does the test of normality tell us?

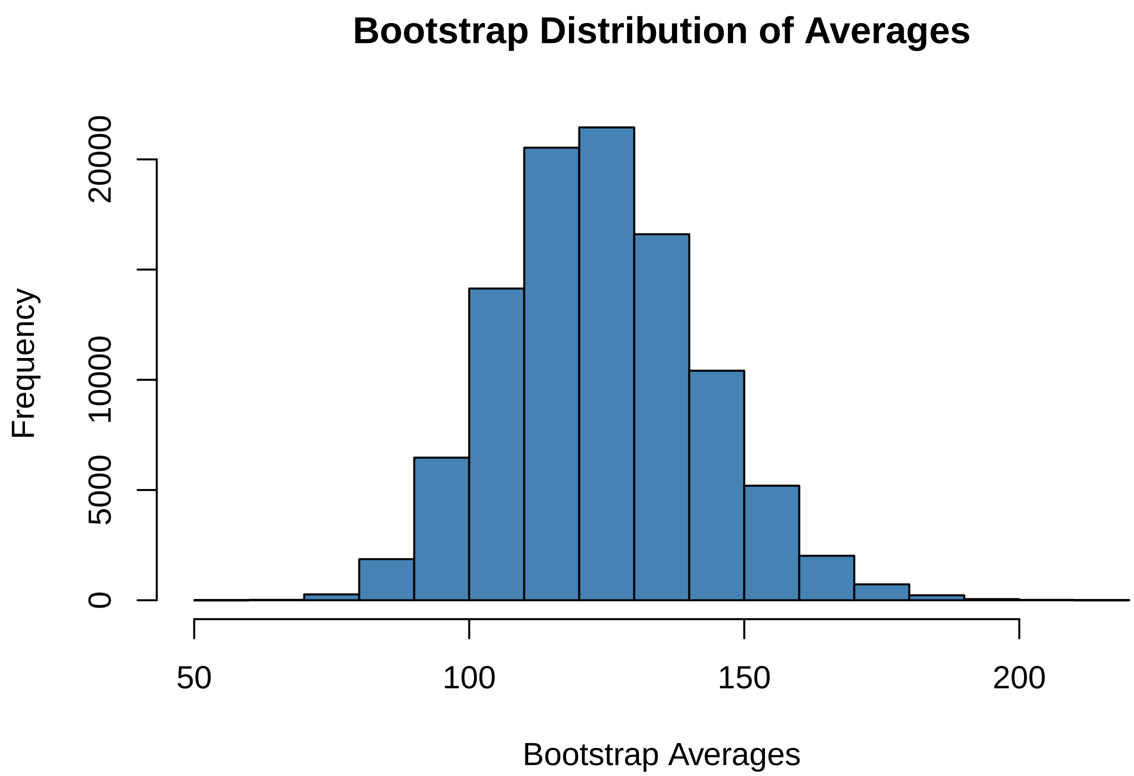
```
shapiro.test(Arsenic)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  Arsenic
## W = 0.42284, p-value < 2.2e-16
```

**Even though we have ample evidence against the normality of the data, due to the large sample size, we could use the classical approach to the confidence interval.**

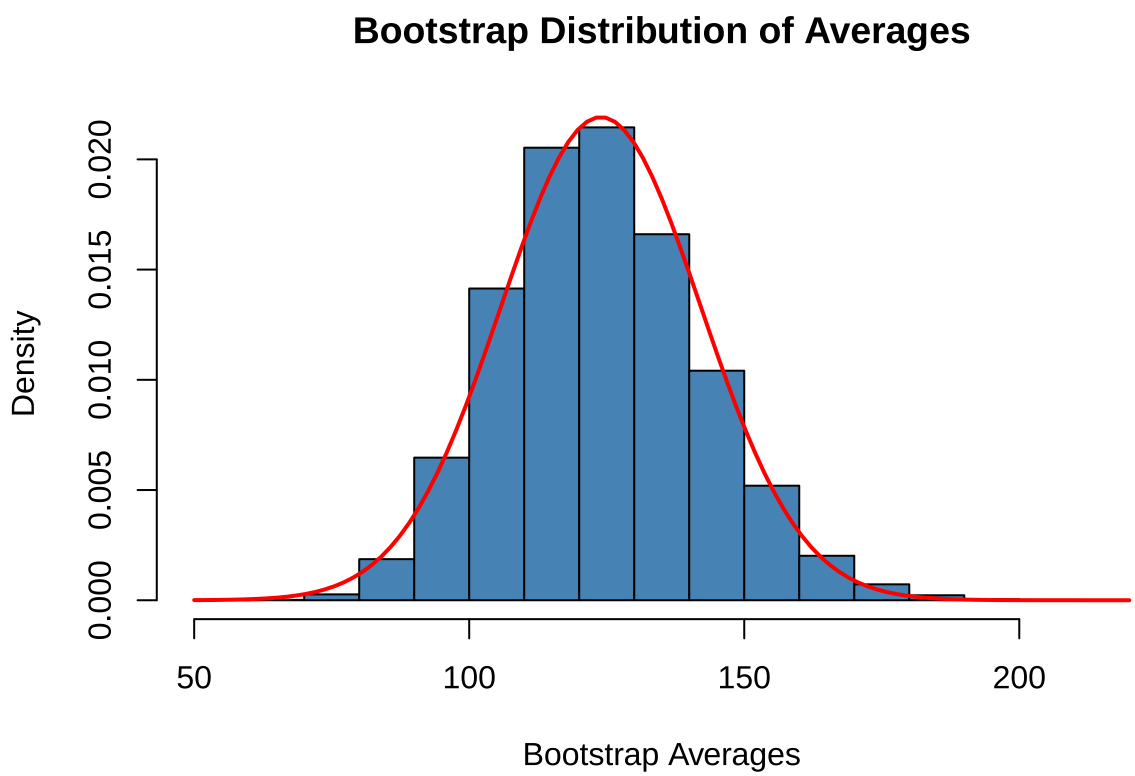
Now, what about bootstrap?

```
n.boot=10^5
arsenic.mean=replicate(n.boot, mean(sample(Arsenic, n, replace=TRUE)))
hist(arsenic.mean,
     main="Bootstrap Distribution of Averages",
     xlab="Bootstrap Averages",
     col="steelblue")
```




**Superimposing the normal bell curve.**

```
hist(arsenic.mean,
     main="Bootstrap Distribution of Averages",
     xlab="Bootstrap Averages",
     col="steelblue",
     prob=TRUE)
curve(dnorm(x, mean=mean(arsenic.mean), sd=sd(arsenic.mean)), col="red", lwd=2, add=TRUE)
```




We could now construct a 2SE bootstrap confidence interval.

```
#bootstrap mean
mu.boot=mean(arsenic.mean)
mu.boot
```

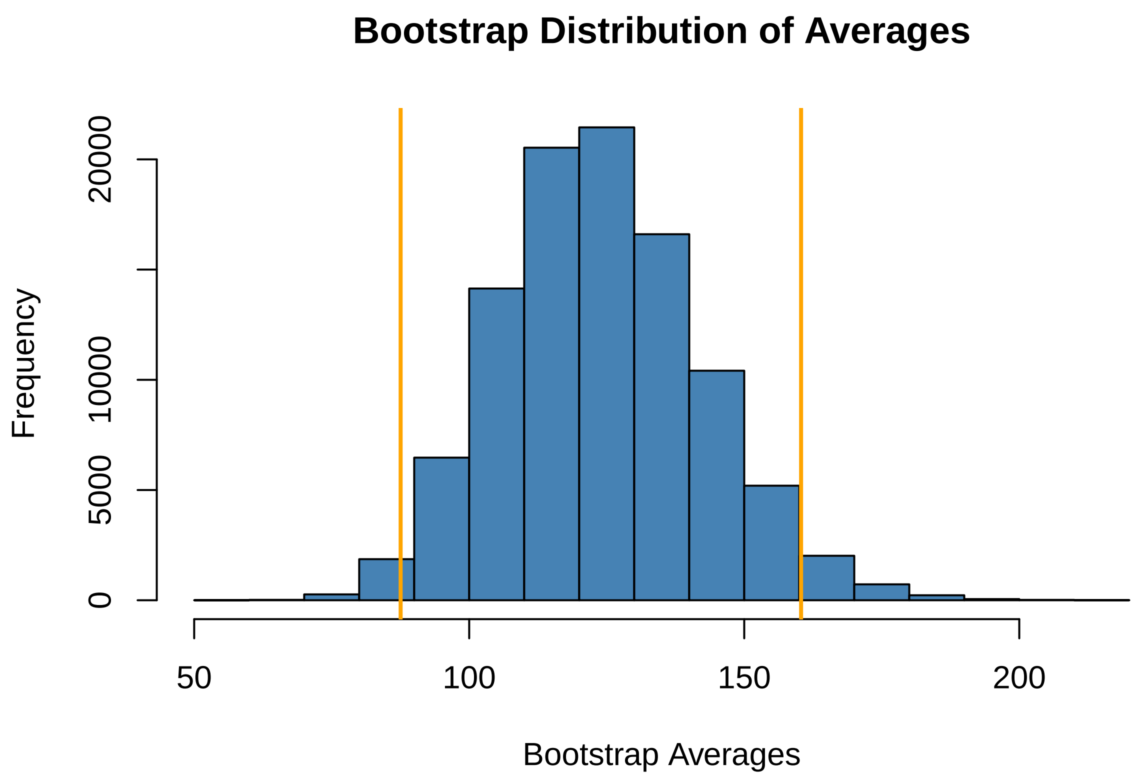
```
## [1] 123.9344
```

```
#bootstrap SE
se.boot=sd(arsenic.mean)

#lower bound
l.bd=mu.boot-2*se.boot
#upper bound
u.bd=mu.boot+2*se.boot

print(c(l.bd, u.bd))
```

```
## [1]  87.53369 160.33506
```

It might be interesting to superimpose it on the histogram.

```
hist(arsenic.mean,
     main="Bootstrap Distribution of Averages",
     xlab="Bootstrap Averages",
     col="steelblue")
abline(v=l.bd, col="orange", lwd=2)
abline(v=u.bd, col="orange", lwd=2)
```




We can also construct a bootstrap 95%-percentile confidence interval.

```
median(arsenic.mean)
```

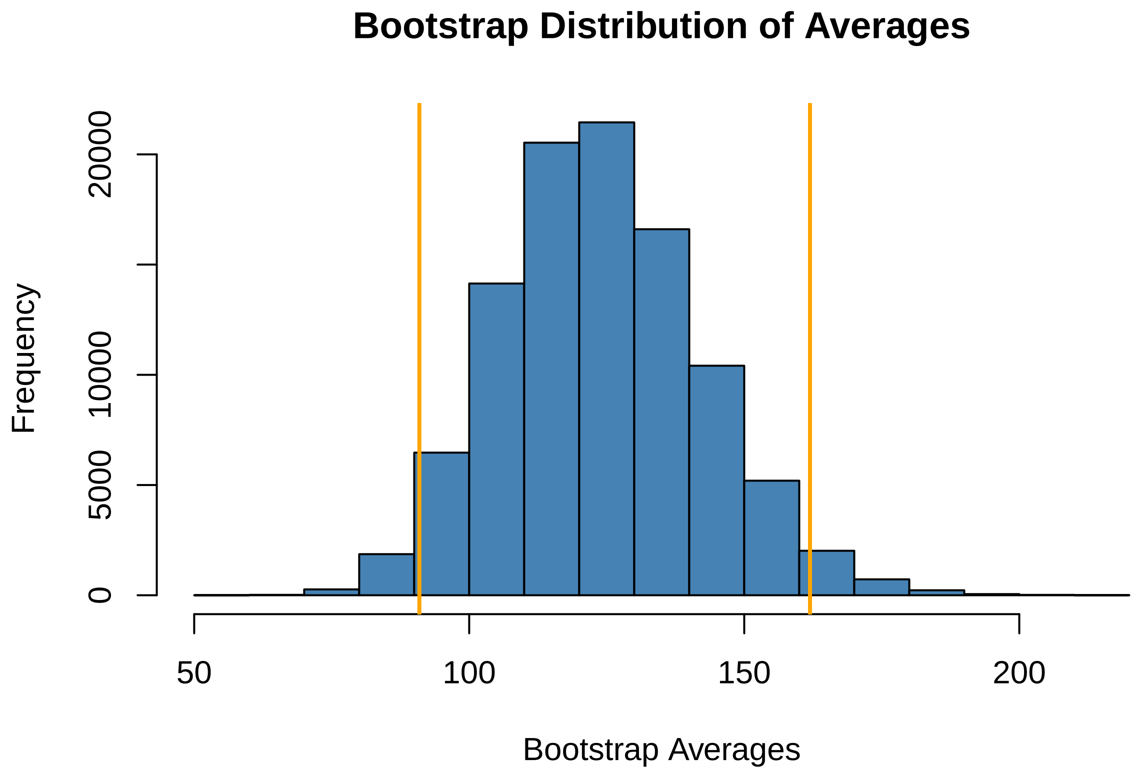
```
## [1] 122.9991
```

```
bds=quantile(arsenic.mean, c(0.025, 0.975))
bds
```

```
##      2.5%     97.5%
##  90.94432 161.95598
```

**An analogous plot.**

```
hist(arsenic.mean,
     main="Bootstrap Distribution of Averages",
     xlab="Bootstrap Averages",
     col="steelblue")
abline(v=bds, col="orange", lwd=2)
```


Bootstrap Distribution of Averages
Frequency
0
5000
10000
20000
50
100
150
200
Bootstrap Averages