```
x=c(83.9,83.6,81.3,85.4,83.9,81.1,84.9,81.1,81.9,79.6,77.7,76.4,77.2,80.1,76.9,81.5,80.9,83.1
,84.2,80.3,83.4,79.1,86.1,78.6,80.1,81.6,78.6,82.5,84.1,87.1)

sam.median=median(x) #median of the data set
sam.median
```

```
[1] 81.4
```

```
jk.estimator=rep(0,30)
for(i in 2:31){
  jk.estimator[i-1]=median(x[-i+1]) #jk estimator for ith sample
}
jk.estimator
```

```
 [1] 81.3 81.3 81.5 81.3 81.3 81.5 81.3 81.5 81.3 81.5 81.5 81.5 81.5 81.5 81.5
[16] 81.3 81.5 81.3 81.3 81.5 81.3 81.5 81.3 81.5 81.5 81.3 81.5 81.3 81.3 81.3
```

```
JK=mean(jk.estimator) #the jackknife estimate
JK
```

```
[1] 81.4
```

```
bias=29*(JK-sam.median) #the bias of estimator, median
bias
```

```
[1] 0
```

```
s=sum((jk.estimator-JK)^2) #standard error of the estimator
se=sqrt((29/30)*s)
se
```

```
[1] 0.5385165
```

The Resampling Method : BOOTSTRAP

What is a Bootstrap?

A Bootstrap is a very useful **resampling technique** when we have too small sample size to draw inference. Also it is surprisingly helpful for constructing confidence intervals and calculating standard errors for difficult statistics.
In Bootstrapping, samples are drawn with replacement. It consists in taking multiples samples out of our original sample and study the resulting distribution. It allows us to see how much variation there would have been and gives further understanding of the sample that we took in the first place.

Why should we use Bootstrap?

- It is easy to implement.

- It doesn't rely in any kind of assumption of population distribution.
- Enables us to draw inference even when we have sample size like 10.

Our Illustration:

Here we have considered two data sets, consisting the test scores of 10 students on a certain subject,before and after a teaching method was introduced.
Object: We want to see if the teaching method was useful at all.
Procedure:
* We consider Bootstrapping from the sets of data.
* Defining the absolute difference of means of two groups, our *test statistics*.
* Setting our null and alternative hypothesis, H_0: The method is effective and H_1:The method is not effective.

```
library(readxl)
dt1=read_excel("C:/Users/Hp/Desktop/scores.xlsx",sheet="scores")
View(dt1)

obs.diff=abs(mean(dt1$method1)-mean(dt1$method2)) #the crical value
obs.diff
```

```
[1] 2.5
```

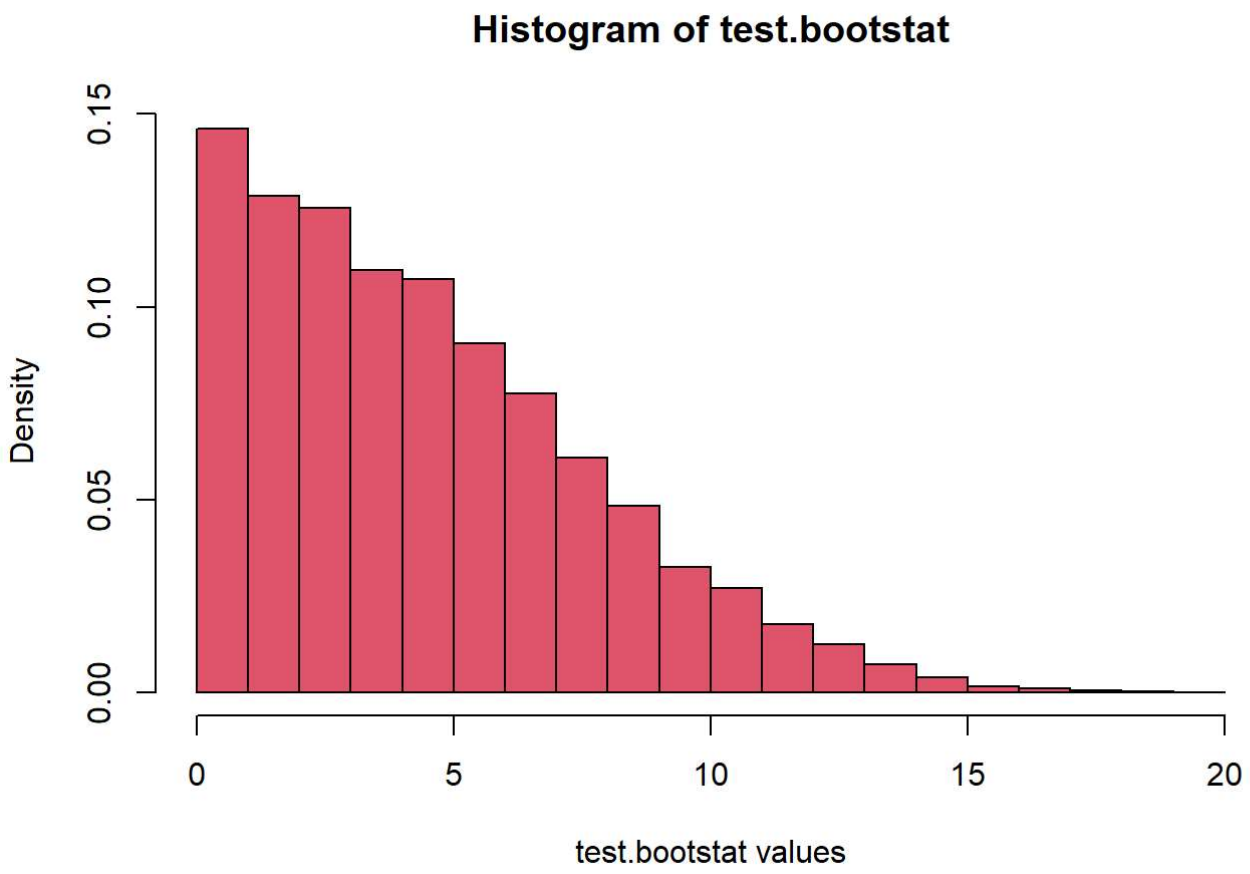
R code and Output of Bootstrapping:

```
boot.num=10000 #number of times we are generating bootstrap samples
test.bootstat=rep(0,boot.num) #observed values of test statistic
set.seed(11)
for(i in 1:boot.num){
  x=sample(dt1$method1,10,replace=TRUE) #sampling from group 1
  y=sample(dt1$method2,10,replace=TRUE) #sampling from group 2
  boot.sam=c(x,y)                        #bootstrap samples
  test.bootstat[i]=abs((mean(x)-mean(y)))
}

head(test.bootstat,100)
```

```
  [1]  0.5  0.5 12.3  1.9  0.5  2.4  4.2  4.5  5.8  2.1  0.7  3.7  9.6  6.1  6.1
 [16]  2.6  5.7  5.0  6.5  0.2  4.3  8.1  0.5  2.0  9.0  2.0  5.9  2.8  4.3  0.3
 [31]  1.9  6.3  1.1  0.5  0.2  4.3  1.4  4.0  0.4  7.5  5.1  3.3  6.9  5.4  1.7
 [46]  2.6  1.3  4.3  1.3  0.9  8.9  7.3  7.5  2.2  2.0  2.5  2.5  0.5  3.6  3.7
 [61]  3.9  4.5  5.0  6.4  3.6  3.3 10.9  2.8  0.5  3.6  1.1  8.4  0.5  5.6  6.3
 [76]  5.6  2.0  7.9 11.3  2.9  3.1  0.4  1.8  2.1  0.8  7.9  3.2  2.5 10.7  4.3
 [91]  2.5  5.5  8.9  3.9  4.4  9.9  4.9  4.8  1.1  2.1
```

```
hist(test.bootstat,xlab="test.bootstat values",probability=T,col=2) #histogram of the test st
atistic
```



Observing the histogram, we assume that the distribution of the test statistic is a **Folded Normal.**

Obtaing the P-value:

```
n=sum(test.bootstat>obs.diff)
p.value=(n/boot.num)
p.value
```

```
[1] 0.664
```

```
if(p.value>0.05){
  print("We accept H0")   #our conclusion
} else {
  print("We reject H0")
}
```

```
[1] "We accept H0"
```

Our Conclusion:

We accept H_0, i.e. the new teaching method is effective.

Monte Carlo Method

What is Monte Carlo?

Monte Carlo simulation is named after the popular gambling destination of Monaco. This technique was developed by Stanislaw Ulan who worked on Manhattan Project.This method is a burning example of how we can actually bypass the complications in Mathematical calculations and step forward. Monte Carlo simulation is used for modelling the probability of different outcomes in a process that can not be predicted due to the presence of random variables which are not easy to deal with.

Monte Carlo method is very flexible in handling complicated integrals which benefits us. We can still calculate expectation and variance (which are basically integrals) even when the tractable form of the probability function is unavailable.

So why Monte Carlo when we have Numerical Methods?

Indeed we have numerical methods to solve complex integrals. But those methods generally cannot be extended to higher dimension. Even if we could extend them, they need very intense algebraic calculations. Here is where Monte Carlo surpasses them.

If it is not possible to obtain the exact analytic solution often Monte Carlo method can be used to provide a very good approximate solution to,

- Calculation the area below a curve.
- Calculation multidimensional integration.
- Optimization.
- Analyzing any complicated stochastic system.

Monte Carlo Integration

Methodology

Let $X \sim f$ be a random variable (could be multidimensional!) and consider a real valued function g and the corresponding random variable $g(X)$. The expectation of $g(X)$ is given by, $E[g(x)] = \int_{\chi} g(x)f(x)dx$

where χ is the support of X.Two common examples of g are $g(x) = x$ and $g(z) = (z - E[X])$. In these cases $E[g(X)$ is nothing but mean and variance of X.

Let $\theta = E[g(X)]$,the Monte Carlo algorithm to compute an approximation to θ is denoted by $\hat{\theta_{MC}}$ is as follows:

* Generate n samples from $f(x)$: $X_1, X_2, \ldots X_n$, then the Monte Carlo approximation is given by, $\hat{\theta_{MC}} = \frac{1}{n}\sum_{i=1}^{n} g(X_i)$

An Illustration:

Suppose we have to compute $I = \int_2^{10} log(x)^{log(x)}dx$

We could write, $I = \int_2^{10} log(x)^{log(x)}dx$

or, $I = 8\int_2^{10} \frac{1}{10-2} log(x)^{log(x)}dx$

so we have $g(x) = log(x)^{log(x)}$ and $f(x) = \frac{1}{10-2}$

So we generate n samples from $U(2, 10)$ and use the Monte Carlo approximation.

```
# the evaluated value of the integral
f=function(x){
  (log(x)^log(x))
}
integrate(f,2,10)
```

```
25.26641 with absolute error < 1.7e-08
```

```
#The Monte Carlo approximation
set.seed(11)
u=runif(1000,2,10)
x=(log(u)^log(u))
8*mean(x) #pretty close!
```

```
[1] 24.96239
```

```
#lets increase the sample size,
set.seed(11)
u=runif(5000,2,10)
x=(log(u)^log(u))
8*mean(x) #Hallelujah! 25.261!
```

```
[1] 25.26181
```

Hence we have successfully obtained our value.

But then every coin has two sides.
If we come across a rare event and are to sample that then we have to hit a very large size of samples. Which could be time consuming or might not serve our purpose. Then we opt for Importance Sampling.