Machine Learning-I (CS/DS 706)

Assignment 1

Chetna Jain (MT2016044)
Kanika Narang (MT2016069)
Tehreem Ansari (MT2016145)

Question 1: Sampling (R): Pick any dataset with a numeric column C and a couple of statistical aggregates on the column value (mean, median, variance, etc.) — denoted as say τ . Carry out the following experiment on the dataset. Compute τ on C across the entire dataset — call it τ c∗ . Let n denote the number of rows in the dataset. For this experiment to
work well, you would need a fairly large dataset.

a). Pick a fraction in the range ρ ∈ (0, 1). Randomly (uniform) sample nρ elements from the dataset and compute the sample statistic τci for C in the sample, where i is the sample index. Repeat this m times, each time with a different sample. Compute the average of the sample statistic.

$$\tau'_{c*} = \frac{\sum_{i=1}^{m} \tau_{ci}}{m}$$

Generate plots to illustrate how τ c∗ converges to τ c∗ for different values of ρ and m.

Solution :

Step 1 Pick the DataSet: The dataset picked is a salary data. It contains two columns 'Id' and 'TotalPay' and number of rows is 148654.

```
> # salary is a table with two columns Id and TotalPay and 148654 rows
> salary <- data.frame(Salaries[,c("Id","TotalPay")])
```

In the above R code the data set is loaded in the form of data from into the salary object.

	Id	TotalPay
1	1	567595.4
2	2	538909.3
3	3	335279.9
4	4	332343.6
5	5	326373.2
6	6	316285.7
7	7	315981.0
8	8	307899.5

$n = 148654$

Step 2 Pick the aggregate Function: The aggregate function we are using is mean.

$\tau = mean$

Now calculating the mean on the entire column TotalPay :

```
> # calculating mean on the column TotalPay
> meanTotalPay <- mean(salary$TotalPay)
> meanTotalPay
[1] 74768.32
```

$\tau = 74768.32$

Step 3 Pick a fraction in the range ρ ∈ (0, 1) : Let ρ =0.4

```
> p= 0.4
> n= 148654
> np= n * p
> # np = 59462
```

Step 4 Randomly (uniform) sample nρ elements from the dataset:

```
> # To select 59462 random rows from the total 148654 rows
> listofrows <- sample(1:148654, np, replace=FALSE)
> listofrows
   [1] 126248  63626  43518  39320 117308  29941  88021  90868  71047
  [10]  25601  59574  36585  80845  67294  63727 100691  71088  99071
  [19]  50352 144892 140021  48490  75797  56570 104001 119695 111454
  [28]  65480 127177  62306 103879  44864 116263 107416  18363  91006
  [37]  73574 100447 118167 106158  89638  43843  38232  79610   3106
  [46]  10769 137377  51843  27261 129338 137321  48001  85589 118718
```

Here we have used the Sample function of R to generate np elements (Here we picked the row numbers randomly using sample function) .

sample takes a sample of the specified size from the elements of x using either with or without replacement.

Syntax of sample function is as follows:

`sample(x, size, replace = FALSE, prob = NULL)`

```
> # select the TotalPay from the sampled rows
> sampleTotalpay <- salary[listofrows,"TotalPay"]
> sampleTotalpay
   [1]  81061.52  35737.20 112641.67 138391.76 123858.47  16706.01
   [7]  84610.54  76387.78   3140.23  46948.89  56033.03 206378.55
  [13] 116623.02  14603.19  40032.41  38319.29   3039.97  51615.24
  [19]  82138.50   7615.77  22364.46  89042.05 159467.60  64783.18
  [25]  19822.92 109078.05 198259.49  25830.83  78229.14  42482.45
```

In the above code we have selected the values of the column ‘TotalPay’ using the sampled row numbers.

Step 5 Compute the sample statistic тci:

```
> # calculating mean on sampled data
> samplemean <- mean(sampleTotalpay)
> samplemean
[1] 74689.14
```

Tci = 74689.14

Step 6 Repeating step 5 m times :

Let m= 100

```
> # we need to do this m times let m=100
>
> x <- replicate(100, {
+   listofrows <- sample(1:148654, 59462 , replace=FALSE)
+   sampleTotalpay <- salary[listofrows,"TotalPay"]
+   samplemean <- mean(sampleTotalpay)
+ })
> x
  [1] 74886.64 74649.74 74676.90 74693.43 75070.29 75038.51 74876.49
  [8] 74721.07 74655.37 75005.89 74814.11 74829.63 74831.21 74908.21
 [15] 74550.41 74960.11 74831.65 74740.64 74611.33 74679.20 75064.52
 [22] 74907.12 74619.68 74886.85 74827.65 74644.19 74756.26 74814.37
 [29] 74828.54 74905.21 74351.74 74764.10 74629.34 75057.37 74605.71
```

In the above code x is the list with stores the mean value of each list of samples picked from the dataset in each iteration. In the above code 59462 is the value of np.

Step 7 Calculating the average mean:

$$\tau'_{c*} = \frac{\sum_{i=1}^{m} \tau_{ci}}{m}$$

```
> # to compute the average of mean of m samples
> avg <- mean(x)
> avg
[1] 74783.23
```

So average mean = 74783.23

Step 8 Generate plots to illustrate how τc∗ converges to τc∗ for different values of ρ and m:

```
> # we need to do the same for different values of p
> # generating 100 values of p at a gap of 0.01
> pvector <- seq(0.01,1, by=0.01)
> pvector
  [1] 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.10 0.11 0.12 0.13 0.14 0.15 0.16 0.17 0.18 0.19
 [20] 0.20 0.21 0.22 0.23 0.24 0.25 0.26 0.27 0.28 0.29 0.30 0.31 0.32 0.33 0.34 0.35 0.36 0.37 0.38
 [39] 0.39 0.40 0.41 0.42 0.43 0.44 0.45 0.46 0.47 0.48 0.49 0.50 0.51 0.52 0.53 0.54 0.55 0.56 0.57
 [58] 0.58 0.59 0.60 0.61 0.62 0.63 0.64 0.65 0.66 0.67 0.68 0.69 0.70 0.71 0.72 0.73 0.74 0.75 0.76
 [77] 0.77 0.78 0.79 0.80 0.81 0.82 0.83 0.84 0.85 0.86 0.87 0.88 0.89 0.90 0.91 0.92 0.93 0.94 0.95
 [96] 0.96 0.97 0.98 0.99 1.00
```

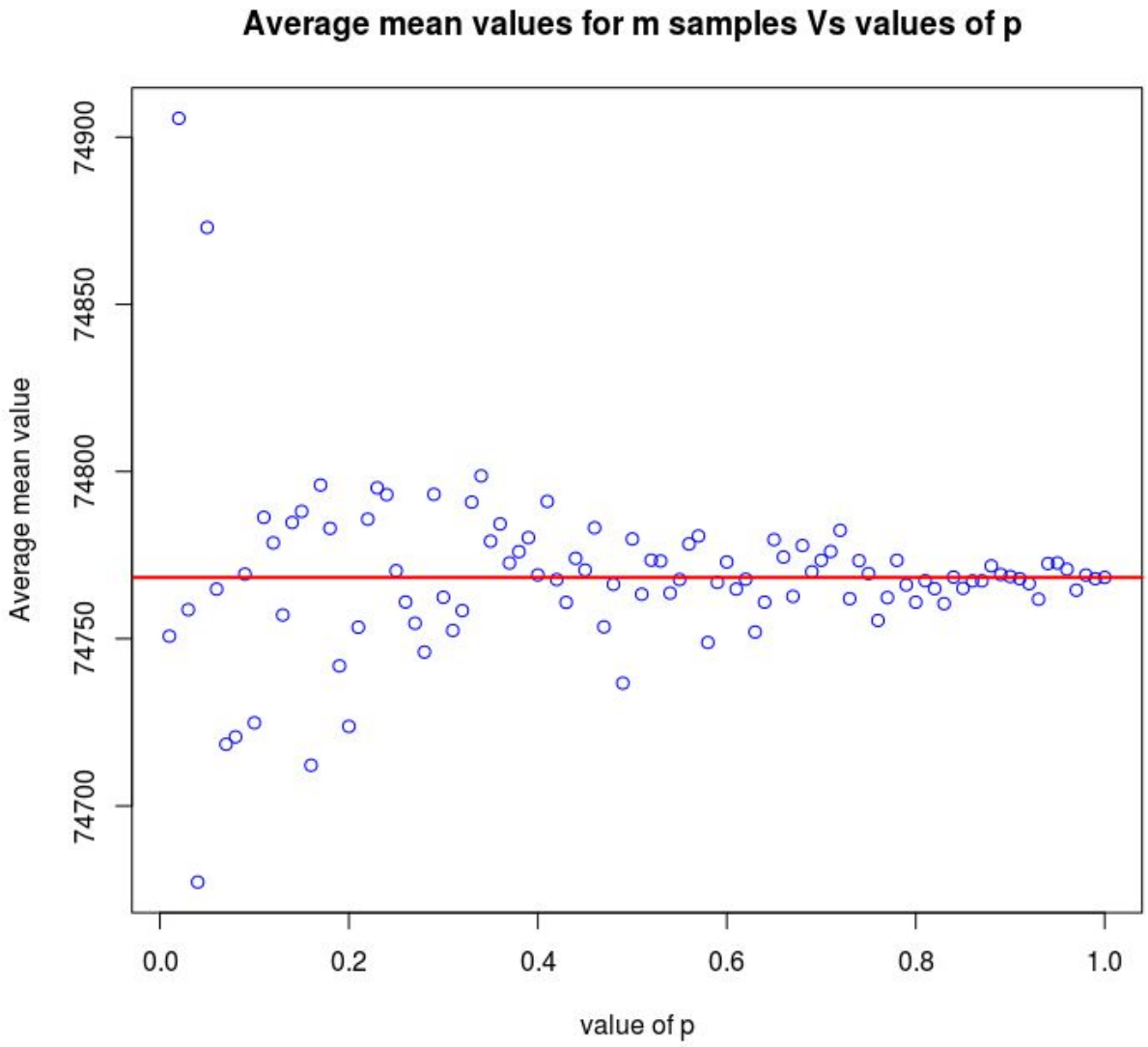
In the above code we are generating 100 different values of ρ using the seq function in a gap of 0.01 .

```
> n= 148654
> # y vector will store the 100 average mean generated for different p values
> y= c()
> for (p in pvector)
+ {
+   np <- n*p
+   x <- replicate(100, {
+     listofrows <- sample(1:148654, np , replace=FALSE)
+     sampleTotalpay <- salary[listofrows,"TotalPay"]
+     samplemean <- mean(sampleTotalpay)
+   })
+   meanx <- mean(x)
+   y <- c(y, meanx)
+ }
> y
  [1] 74750.75 74905.62 74758.68 74677.20 74872.97 74764.82 74718.44 74720.64 74769.31 74724.83
 [11] 74786.29 74778.67 74757.09 74784.74 74788.09 74712.14 74795.97 74783.00 74741.86 74723.77
 [21] 74753.43 74785.75 74795.12 74793.08 74770.32 74760.98 74754.62 74746.01 74793.18 74762.39
 [31] 74752.46 74758.41 74790.83 74798.68 74779.15 74784.33 74772.62 74776.02 74780.21 74769.07
 [41] 74791.05 74767.68 74760.85 74774.03 74770.52 74783.17 74753.54 74766.26 74736.71 74779.81
 [51] 74763.30 74773.49 74773.24 74763.63 74767.70 74778.33 74780.71 74748.87 74766.83 74772.94
 [61] 74764.83 74767.79 74752.00 74760.93 74779.61 74774.41 74762.65 74777.86 74770.01 74773.45
 [71] 74776.08 74782.41 74761.95 74773.34 74769.47 74755.47 74762.30 74773.44 74766.08 74760.91
 [81] 74767.33 74764.97 74760.49 74768.43 74765.05 74767.29 74767.30 74771.77 74769.18 74768.60
 [91] 74767.95 74766.46 74761.80 74772.46 74772.62 74770.75 74764.51 74769.03 74767.95 74768.32
```

Here in the above code we have got a list ‘y’ which contains the average mean of the m iterations of the experiment where we calculated mean of np samples for different values of p.

Now to plot the values we used the plot function on the y-axis the average mean values are plotted and x-axis the p values.

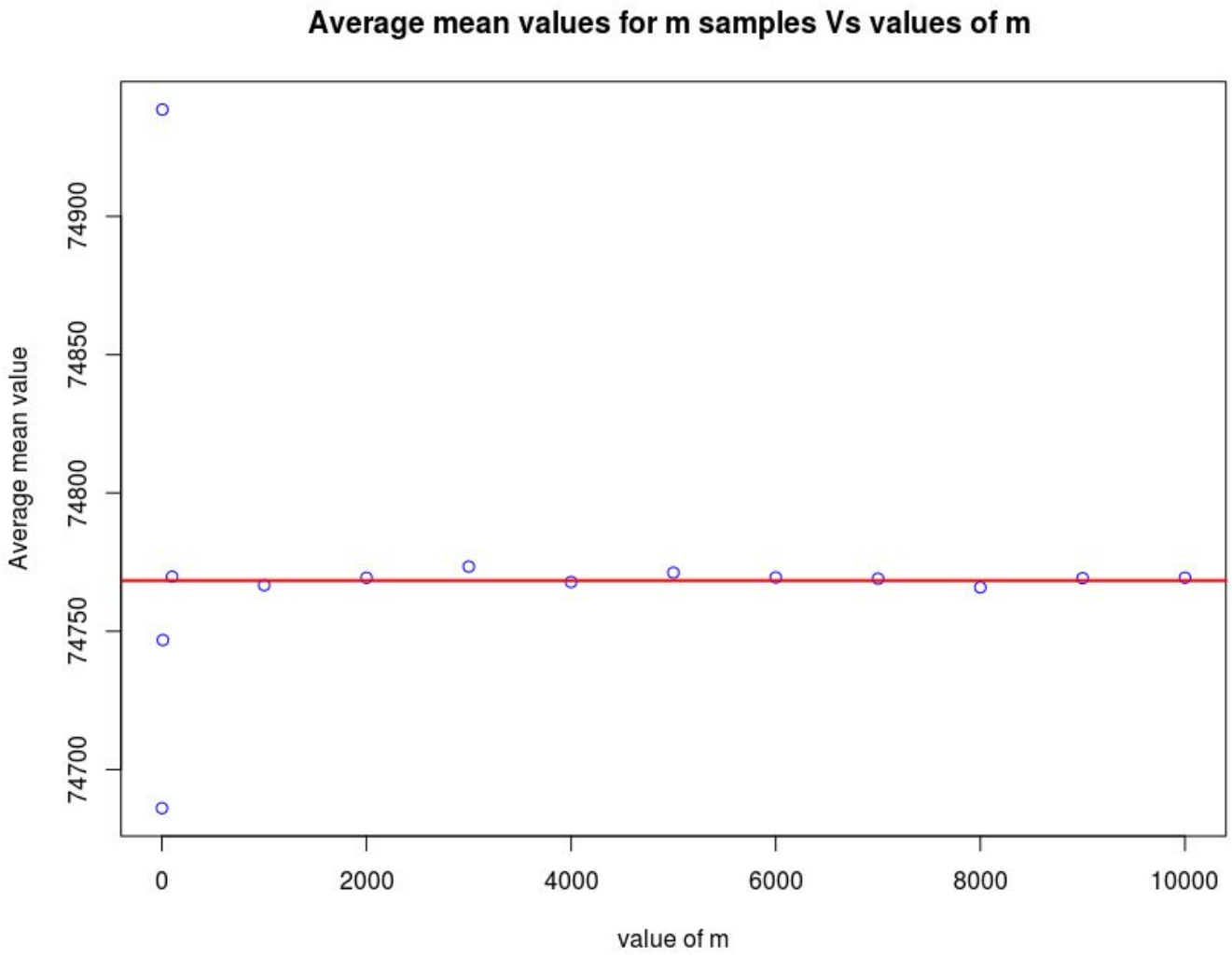
```
> #ploting the the mean values of
> plot(pvector ,y, main="Average mean values for m samples Vs values of p ",ylim = c(min(y),max(y)),pch = 1, col = "blue" ,xlab="value of p"
, ylab="Average mean value")
>
> #adding the line to show average value of entire data
> abline(h=74768.32, col="red", lwd=2)
>
```



We can see in the plot that as the value of p increases and gets close to 1 the value of average mean also get close to the value of mean of the entire column.
In the above plot the red line shows the value of mean of the entire column.($\tau = 74768.32$)

Now I am plotting a graph with different values of m with fixed p=0.4

```
> # Doing Sampling for different values of m
> p=0.4
> m_vector=c(1,5,10,100,1000,2000,3000,4000,5000,6000,7000,8000,9000,10000
> y= c()
> for (m in m_vector)
+ {
+   np <- n*p
+   x <- replicate(m, {
+     listofrows <- sample(1:148654, np , replace=FALSE)
+     sampleTotalpay <- salary[listofrows,"TotalPay"]
+     samplemean <- mean(sampleTotalpay)
+   })
+   meanx <- mean(x)
+   y <- c(y, meanx)
+ }
> y
 [1] 74686.05 74938.59 74746.82 74769.72 74766.65 74769.34 74773.37
 [8] 74767.81 74771.18 74769.46 74769.01 74765.86 74769.19 74769.35
> #ploting the the mean values
> plot(m_vector ,y, main="Average mean values for m samples Vs values of m
",pch = 1, col = "blue" ,xlab="value of m", ylab="Average mean value")
>
> #adding the line to show average value of entire data
> abline(h=74768.32, col="red", lwd=2)
```

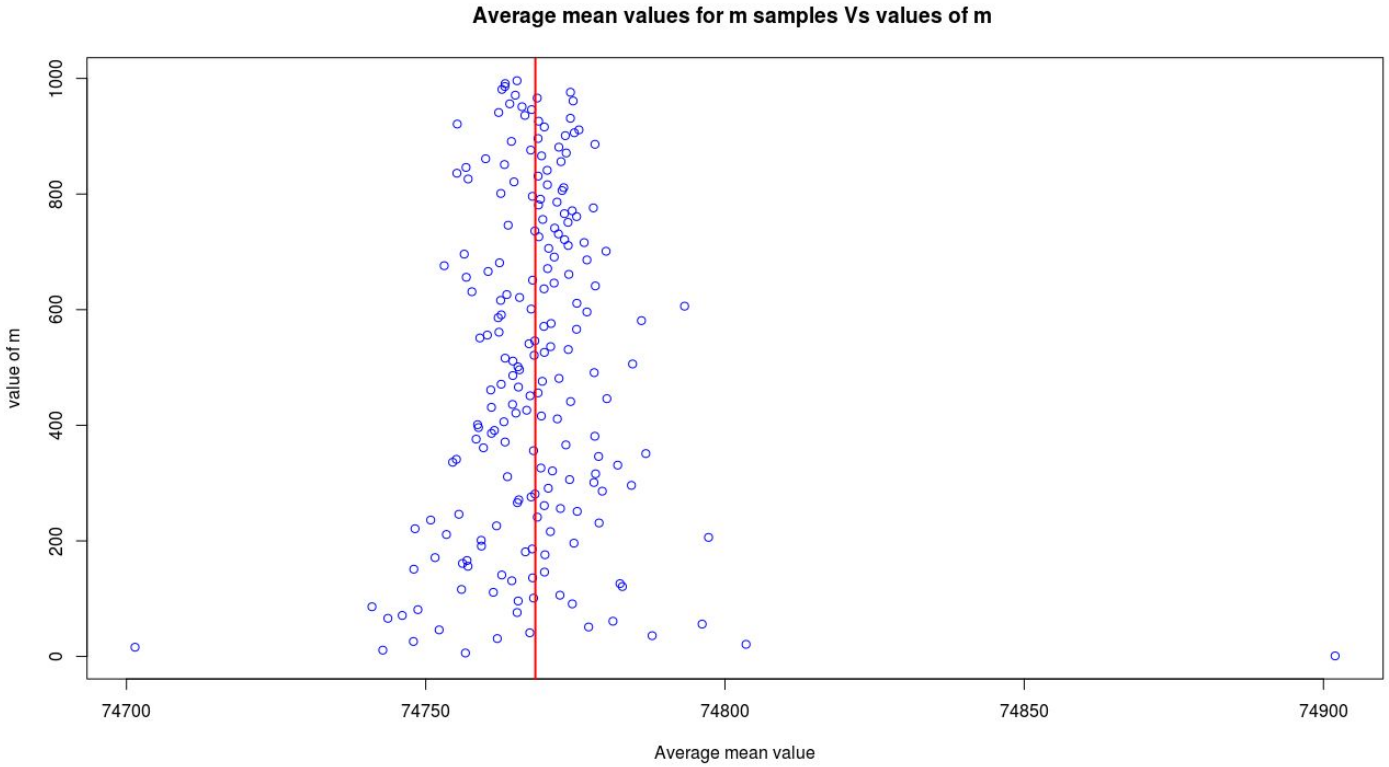


We see here as the value of m increases the value of average mean gets close to the mean of entire column.

Now to get more insight i am taking another set of values of m (total 200 different values of m)

```
> p=0.4
> m_vector2=seq(1,1000,5)
> length(m_vector2)
[1] 200
> y2= c()
> for (m in m_vector2)
+ {
+   np <- n*p
+   x <- replicate(m, {
+     listofrows <- sample(1:148654, np , replace=FALSE)
+     sampleTotalpay <- salary[listofrows,"TotalPay"]
+     samplemean <- mean(sampleTotalpay)
+   })
+   meanx <- mean(x)
+   y2 <- c(y2, meanx)
+ }
> y2
  [1] 74901.95 74756.63 74742.83 74701.43 74803.52 74747.95 74761.96 74787.84
  [9] 74767.39 74752.24 74777.22 74796.19 74781.29 74743.68 74746.08 74765.27
 [17] 74748.69 74741.02 74774.49 74765.45 74768.02 74772.43 74761.30 74755.98
 [25] 74782.86 74782.46 74764.38 74767.86 74762.70 74769.84 74748.01 74757.05
 [33] 74756.15 74756.90 74751.56 74769.92 74766.66 74767.80 74759.31 74774.76
 [41] 74759.27 74797.26 74753.46 74770.83 74748.22 74761.84 74778.96 74750.83
 [49] 74768.65 74755.55 74775.31 74772.51 74769.81 74765.31 74765.54 74767.62
 [57] 74768.27 74779.50 74770.48 74784.37 74778.10 74774.04 74763.66 74778.38
 [65] 74771.17 74769.24 74782.08 74754.50 74755.13 74778.88 74786.78 74768.02
 [73] 74759.64 74773.42 74763.26 74758.43 74778.25 74760.99 74761.47 74758.82
 [81] 74758.67 74763.06 74772.00 74769.32 74765.07 74766.87 74760.98 74764.51
```

```
> #ploting the the mean values
> plot( y2,m_vector2, main="Average mean values for m samples Vs values of m ",pch = 1, col = "blue" ,ylab="value of m", xlab="Average mean value")
>
> #adding the line to show average value of entire data
> abline(v=74768.32, col="red", lwd=2)
>
```



The above plots shows that the values of sample mean generated using different values of mean follows a distribution where values a close to the mean of the population is mean of the entire column.
It signifies the central limit theorem which says that the distribution of the average of a large number of independent , identically distributed variables will be approximately normal, regardless of the underlying distribution.

Observation:

1. As the value of ρ increases and gets close to 1, the value of the average mean from the m iterations of experiment generated as τ'_{c*} gets close to value of the mean of the entire column 'TotalPay' . (**Law of Large Number**)

2. If we increase the value of m then the average mean value calculated using $\tau'_{c*} = \frac{\sum_{i=1}^{m} \tau_{ci}}{m}$ will get close to the mean of entire column. The sample mean will be close to the mean of population, it follows the normal distribution.
(Central limit Theorem)

m	τ'_{c*}
1	74689.14
100	74783.23
1000	74767.31
10000	74767.31
100000	74768.89
Mean of entire column	74768.32

b). Repeat the above experiment but for varying sample sizes in the same 'run'. Specifically, pick m samples as above but the sample sizes are in turn randomly sampled from a Gaussian. Generate plots showing how convergence changes with μ, σ and m where μ, σ^2 are the mean and variance of the Gaussian from which the sample sizes are picked.

Solution:
It means now the sample size is not defined by the value of ρ. It will depend on the value of μ, σ of the Gaussian Distribution from which we will randomly pick the sample size.

Step1 and Step2 are same as above in question 1 a).

Step 3 Pick sample size randomly sampled from a Gaussian.

We will be using rnorm function with μ=1000, σ =100

```
> S=rnorm(n = 1, mean = 1000, sd = 100)
> S = ceiling(S)
> S
[1] 1082
```

arguments:

- n = sample size
- mean = mean of sample
- sd = standard deviation of sample

We took ceiling so that we could get rounded number for sample size.
Let S= Sample size

Step 4 Randomly (uniform) sample S elements from the dataset:

```
> # To select S random rows from the total 148654 rows
> listofrows <- sample(1:148654, S, replace=FALSE)
> listofrows
   [1]  44683  45273  96668 102017 132541  87941 121146 117650 145195  99129 129526  44071
  [13]  50521 128503  99562  53785  65787  63325  18620   1214  83029  32151  18332 128663
  [25]  23537  83114 101022 104238  52065  86466 144604 130674 108790 136152  24254  66557
  [37]  53035  73116 121165 122683  28228  19355  83445  98971 128803  70173  31500  54756
  [49]  70074  80096  99957  11689  22960  58554  63380  82178 141708 143747  77294  24323
  [61]  78879  85677 100467  31051  11521  89523  20984 100054  78045  39164  17569 144722
  [73]  12426 110385  61357 137432  77970  69847 112948  10561 122361  97289 115441 115194
```

Here we have used the Sample function of R to generate S elements (Here we picked the row numbers randomly using sample function) .

sample takes a sample of the specified size from the elements of x using either with or without replacement.

Syntax of sample function is as follows:

```
sample(x, size, replace = FALSE, prob = NULL)
```

```
> # select the TotalPay from the sampled rows
> sampleTotalpay <- salary[listofrows,"TotalPay"]
> sampleTotalpay
   [1] 130713.52 102789.62  56717.82  32870.00  62365.78  85364.73 100525.32 120848.74   6705.70
  [10]  50323.60  73974.89 117221.48  80237.03  77888.42  45979.44  72358.70  16965.98  47860.91
  [19]  67040.21 164895.30 104250.54   8371.09  67685.70  74201.03  53448.13 108104.58  36745.34
  [28]  18226.36  77143.73  91594.54   8376.00  70185.93   2930.65  50404.79  51436.72  17655.76
  [37]  75124.74 225896.50 100719.64  94494.52  28059.26  65141.63 103430.42  49831.41  82579.00
  [46]   4479.80  10328.68  70222.96   5774.67 116437.07  41124.41  90322.64  54706.91  58209.68
  [55]  39265.07 109347.66  16929.60  11011.85 134058.62  51234.90 130197.79  93639.18  38918.52
  [64]  11994.00  90867.50  79742.50  60134.96  50956.28 143445.57 152116.90  69642.01   7004.32
```

In the above code we have selected the values of the column ‘TotalPay’ using the sampled row numbers.

Step 5 Compute the sample statistic τ_{ci}:

```
> # calculating mean on sampled data
> samplemean <- mean(sampleTotalpay)
> samplemean
[1] 72887.22
```

T_{ci} = 72887.22

Step 6 Repeating step 6 m times :

Let m= 100 (Same as above in question 1 a.)

```
> # we need to do this m times let m=100
> x <- replicate(100, {
+   listofrows <- sample(1:148654, S, replace=FALSE)
+   sampleTotalpay <- salary[listofrows,"TotalPay"]
+   samplemean <- mean(sampleTotalpay)
+ })
> x
  [1] 76236.18 74345.00 74021.03 72650.08 76534.99 77529.69 73834.40 74985.10 75616.94 75175.16
 [11] 74584.05 75686.40 74221.64 77047.79 78106.69 77244.43 75231.73 74540.53 74958.43 73429.68
 [21] 76049.10 74705.36 73617.05 76247.02 74008.55 74669.15 75384.87 74087.49 78597.41 75015.76
 [31] 76217.69 74724.79 74082.43 74062.19 77915.81 73664.08 72989.69 73234.30 76178.36 77244.57
 [41] 76005.69 75118.38 74973.63 72503.62 73584.42 75925.42 77064.62 75057.37 73662.47 73828.26
```

In the above code x is the list with stores the mean value of each list of samples picked from the dataset in each iteration.

Step 7 Calculating the average mean:

$$\tau'_{c*} = \frac{\sum_{i=1}^{m} \tau_{ci}}{m}$$

```
> # to compute the average of mean of m samples
> avg <- mean(x)
> avg
[1] 75031.08
```

So average mean = 75031.08

Step 8 Generate plots showing how convergence changes with μ

First we are generating different values of μ. We are generating different values of

```
> # Generating different mean values from 101 to 148554
> # with a gap of 1485 so that we get 100 differnt values
>
> mean_vector <- seq(101,148554, by=1485)
> mean_vector
  [1]    101   1586   3071   4556   6041   7526   9011  10496  11981  13466  14951  16436  17921
 [14]  19406  20891  22376  23861  25346  26831  28316  29801  31286  32771  34256  35741  37226
 [27]  38711  40196  41681  43166  44651  46136  47621  49106  50591  52076  53561  55046  56531
 [40]  58016  59501  60986  62471  63956  65441  66926  68411  69896  71381  72866  74351  75836
 [53]  77321  78806  80291  81776  83261  84746  86231  87716  89201  90686  92171  93656  95141
 [66]  96626  98111  99596 101081 102566 104051 105536 107021 108506 109991 111476 112961 114446
 [79] 115931 117416 118901 120386 121871 123356 124841 126326 127811 129296 130781 132266 133751
 [92] 135236 136721 138206 139691 141176 142661 144146 145631 147116
```

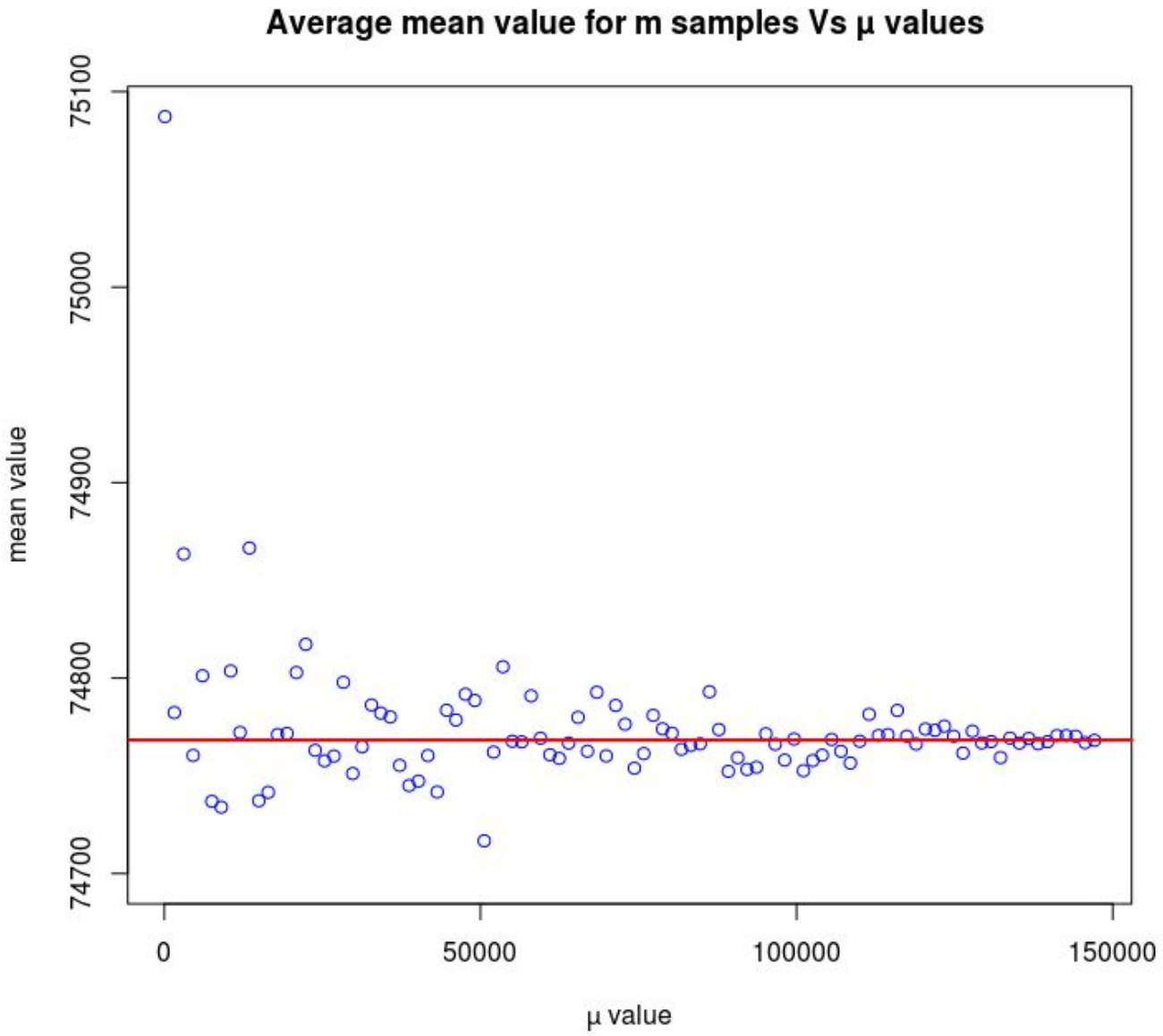
Now we will iterate over this list of mean_vector which contains all the values of μ and perform the experiment of getting average mean of m samples. (here we have kept the σ and m same throughout)

```
> n= 148654
> # y vector will store the 100 average mean generated for different μ values
> y= c()
> for (mu in mean_vector)
+ {
+   S= (rnorm(n = 1, mean = mu, sd = 100))
+   S = ceiling(S)
+   if(S> n)
+   {
+     S=n
+   }
+   else if(S<0)
+   {
+     S=0
+   }
+   x <- replicate(100, {
+     listofrows <- sample(1:148654, S , replace=FALSE)
+     sampleTotalpay <- salary[listofrows,"TotalPay"]
+     samplemean <- mean(sampleTotalpay)
+   })
+   meanx <- mean(x)
+   y <- c(y, meanx)
+ }
> y
  [1] 75087.22 74782.36 74863.45 74760.42 74801.16 74736.92 74733.97 74803.64 74772.20 74866.53
 [11] 74737.25 74741.55 74771.11 74771.66 74802.82 74817.26 74763.07 74757.64 74760.09 74797.84
 [21] 74751.16 74764.84 74786.25 74782.04 74780.09 74755.33 74745.01 74747.28 74760.35 74741.70
 [31] 74783.54 74778.56 74791.73 74788.54 74716.75 74762.12 74805.79 74767.68 74767.35 74790.86
 [41] 74769.20 74760.78 74758.89 74766.74 74779.88 74762.58 74792.74 74760.17 74785.99 74776.39
 [51] 74753.86 74761.36 74780.92 74774.00 74771.80 74763.56 74765.52 74766.37 74793.00 74773.64
 [61] 74752.27 74759.24 74753.25 74754.36 74771.47 74766.15 74758.04 74768.78 74752.60 74757.80
 [71] 74760.57 74768.59 74762.55 74756.54 74767.70 74781.45 74770.60 74771.07 74783.45 74770.22
 [81] 74766.26 74773.91 74773.27 74775.36 74770.21 74761.63 74772.84 74766.82 74767.52 74759.33
 [91] 74769.27 74766.74 74769.07 74766.41 74767.42 74770.56 74770.56 74770.21 74766.97 74768.15
>
```

Here in the above code we have got a list ‘y’ which contains the average mean of the m iterations of the experiment where we calculated mean of S samples for different values of μ.

Now to plot the values we used the plot function on the y-axis the average mean values are plotted and x-axis the different values of μ.

```
> #ploting the the mean values of 100 experiments
> plot( mean_vector,y , main="Average mean value for m samples Vs μ values " , ylim = c(74700,max(y)) ,
+       pch = 1, col = "blue" ,xlab="μ value", ylab="mean value")
>
> #adding the line to show average value of entire data
> abline(h=74768.32, col="red", lwd=2)
```



We can see in the plot that as the value of μ increases and gets close to the total number of rows, the value of average mean also get close to the value of mean of the entire column. This is happening because as the value of μ increases the sample size increases thus due to **law of large numbers** the value of sample mean gets close to the mean of the entire population.
In the above plot the red line shows the value of mean of the entire column.($\tau = 74768.32$)

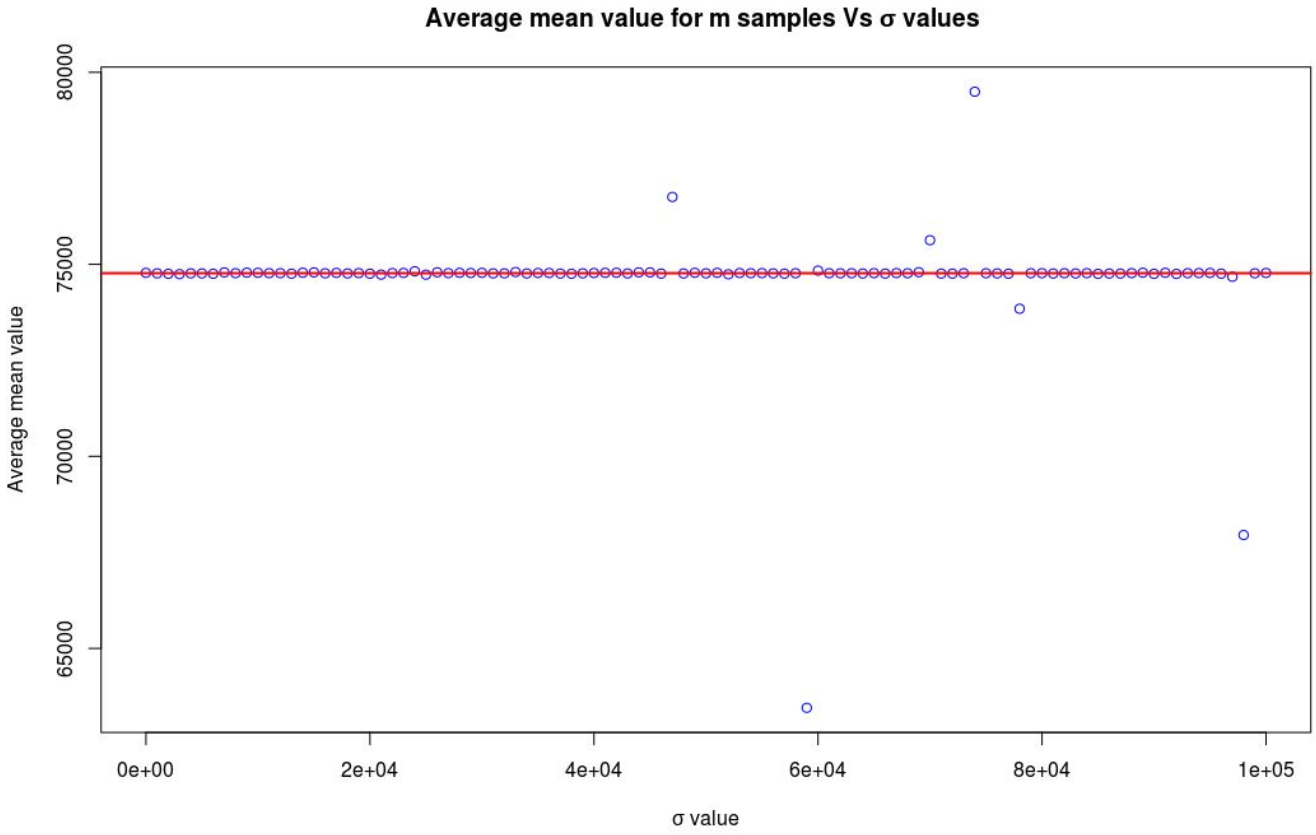
Step 9 Generate plots showing how convergence changes with σ
First we will generate the different values of σ
Here we are generating σ using the seq function between 0 and 100000 with a gap of 1000
Thus we got total 101 different values of σ .

Now we do the sampling experiment for the above generated values of σ and keep the μ and m same.

```
> n= 148654
> # y vector will store the 101 average mean generated for different σ values
> y= c()
>
> for (sd in sd_vector)
+ {
+   S= (rnorm(n = 1, mean = (n/2), sd = sd))
+   S = ceiling(S)
+   if(S> n)
+   {
+     S=n
+   }
+   else if(S<0)
+   {
+     S=1
+   }
+   x <- replicate(100, {
+     listofrows <- sample(1:148654, S , replace=FALSE)
+     sampleTotalpay <- salary[listofrows,"TotalPay"]
+     samplemean <- mean(sampleTotalpay)
+   })
+   meanx <- mean(x)
+   y <- c(y, meanx)
+ }
> y
  [1] 74778.00 74764.44 74753.59 74744.36 74764.38 74763.09 74754.61 74790.20 74769.33
 [10] 74783.43 74779.14 74770.34 74769.14 74756.98 74781.12 74790.54 74765.93 74777.07
 [19] 74763.27 74771.81 74757.17 74731.86 74771.99 74775.68 74819.11 74730.84 74792.63
 [28] 74770.19 74779.15 74774.26 74778.58 74764.00 74764.40 74800.02 74762.36 74773.61
 [37] 74773.38 74760.23 74754.48 74765.76 74773.27 74778.65 74781.14 74762.13 74791.02
 [46] 74790.11 74758.22 76754.21 74760.97 74779.87 74765.91 74779.10 74740.54 74773.41
 [55] 74768.32 74771.13 74764.38 74757.18 74768.32 63453.00 74836.24 74768.32 74766.18
 [64] 74768.32 74760.58 74770.01 74758.83 74771.75 74768.32 74794.69 75627.82 74759.19
 [73] 74758.26 74768.32 79494.19 74768.25 74763.25 74756.35 73843.12 74768.32 74768.32
 [82] 74760.45 74768.17 74762.88 74769.36 74752.00 74760.10 74758.75 74770.13 74781.36
 [91] 74753.13 74779.69 74750.53 74769.67 74771.71 74775.89 74756.99 74676.41 67951.80
[100] 74766.97 74775.42
>
```

Now plotting the above generated average mean values for different values of σ.

```
> #ploting the the mean values of 100 experiments
> plot( sd_vector,y , main="Average mean value for m samples Vs σ values " , ylim = c(min(y),max(y)) ,
+       pch = 1, col = "blue" ,xlab="σ value", ylab="Average mean value")
>
> #adding the line to show average value of entire data
> abline(h=74768.32, col="red", lwd=2)
```



The values which are not on mean line (red line depicting the mean value of entire column) are those values which were not valid number of rows(that is the sample size generated from the gaussian came out to be less than 0 so in this case i assigned sample size as 1.)
The above plot also shows that since mean is fixed to (total number of rows)/2 and the sigma varies will not lead to much deviation in the average mean value. But we can see it will depend on the sample size getting randomly picked from the gaussian.

Step 10 Generate plots showing how convergence changes with m
First we need to generate the different values of m keeping the sigma and mean of gaussian same.

We are generating values of m between 1 to 1000 with a gap of 10 using seq function.

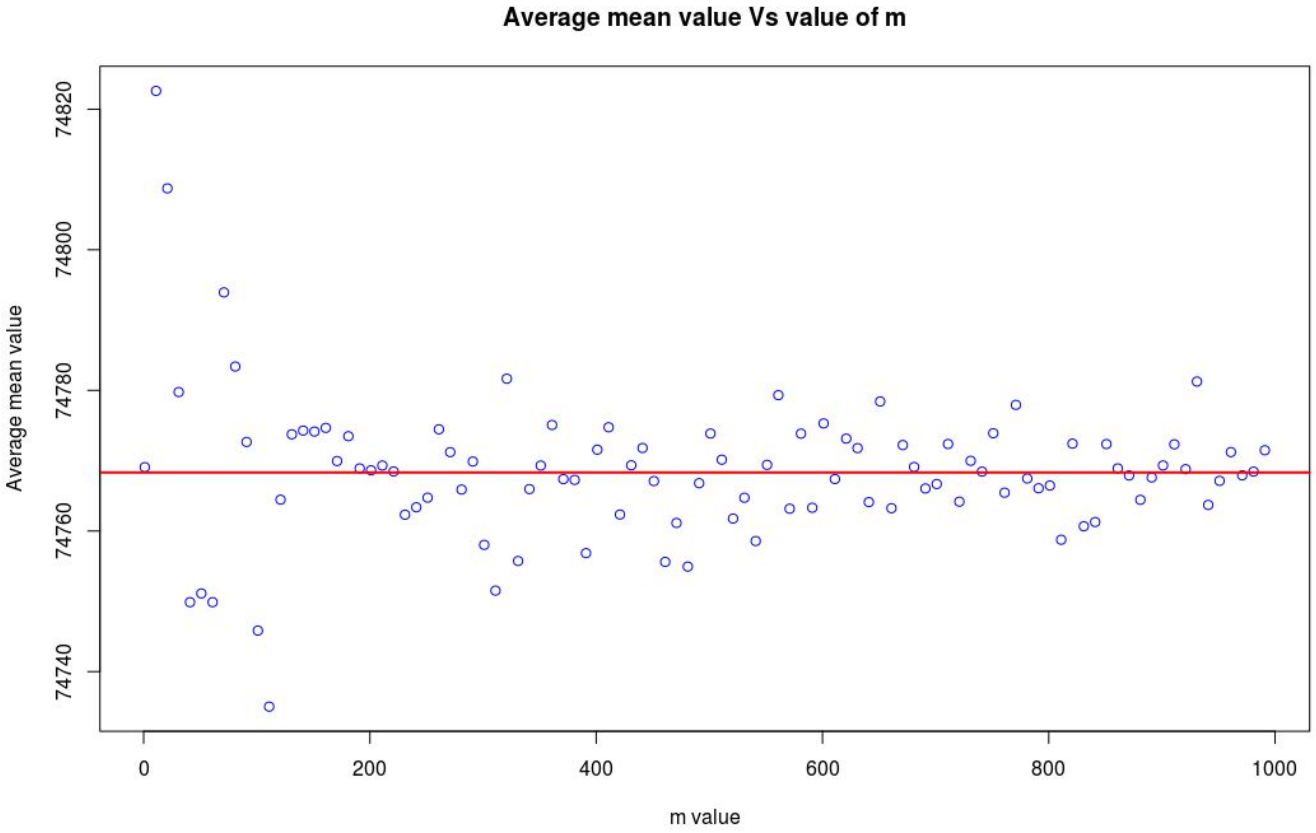
```
> #.....For different values of m
> # Generating different values of m
> m_vector <- seq(1, 1000, by =10)
> m_vector
  [1]   1  11  21  31  41  51  61  71  81  91 101 111 121 131 141 151 161 171 181 191 201 211 221 231 241 251 261 271
 [29] 281 291 301 311 321 331 341 351 361 371 381 391 401 411 421 431 441 451 461 471 481 491 501 511 521 531 541 551
 [57] 561 571 581 591 601 611 621 631 641 651 661 671 681 691 701 711 721 731 741 751 761 771 781 791 801 811 821 831
 [85] 841 851 861 871 881 891 901 911 921 931 941 951 961 971 981 991
```

Now we will do the sampling for the above generated m values.

```
> # y vector will store the 100 average mean generated for different m values with μ=n/2 and σ=100
> y= c()
>
> for (m in m_vector)
+ {
+   S= (rnorm(n = 1, mean = (n/2), sd = 100))
+   S = ceiling(S)
+   if(S> n)
+   {
+     S=n
+   }
+   else if(S<0)
+   {
+     S=1
+   }
+   x <- replicate(m, {
+     listofrows <- sample(1:148654, S , replace=FALSE)
+     sampleTotalpay <- salary[listofrows,"TotalPay"]
+     samplemean <- mean(sampleTotalpay)
+   })
+   meanx <- mean(x)
+   y <- c(y, meanx)
+ }
> y
  [1] 74836.86 74845.91 74808.29 74778.19 74757.22 74721.70 74777.87 74763.85 74753.52 74792.81 74765.27 74783.71
 [13] 74754.00 74754.89 74777.06 74777.31 74759.84 74795.51 74772.49 74755.54 74780.24 74758.65 74742.78 74773.92
 [25] 74766.25 74767.58 74764.61 74764.73 74759.48 74763.18 74763.53 74775.74 74760.65 74783.23 74763.87 74777.88
 [37] 74768.18 74764.77 74761.79 74773.83 74769.56 74767.86 74762.19 74763.16 74761.64 74775.70 74760.37 74760.23
 [49] 74769.67 74763.45 74763.89 74770.65 74760.63 74766.35 74765.62 74768.69 74768.83 74771.85 74764.65 74768.39
 [61] 74773.48 74773.60 74768.62 74774.29 74763.04 74764.96 74764.85 74768.03 74770.03 74778.83 74765.39 74775.06
 [73] 74772.28 74768.93 74767.62 74765.38 74774.25 74765.71 74766.31 74771.64 74767.54 74775.46 74767.12 74762.45
 [85] 74771.92 74766.76 74769.05 74766.87 74762.80 74769.25 74771.93 74762.52 74766.18 74764.32 74773.13 74769.15
 [97] 74765.60 74769.09 74777.22 74768.43
>
```

Now plotting these values

```
> #ploting the the mean values of 100 different values of m
> plot( m_vector,y , main="Average mean value for m samples Vs ?? values " , ylim = c(min(y),max(y)) ,
+       pch = 1, col = "blue" ,xlab="m value", ylab="Average mean value")
>
> #adding the line to show average value of entire data
> abline(h=74768.32, col="red", lwd=2)
```



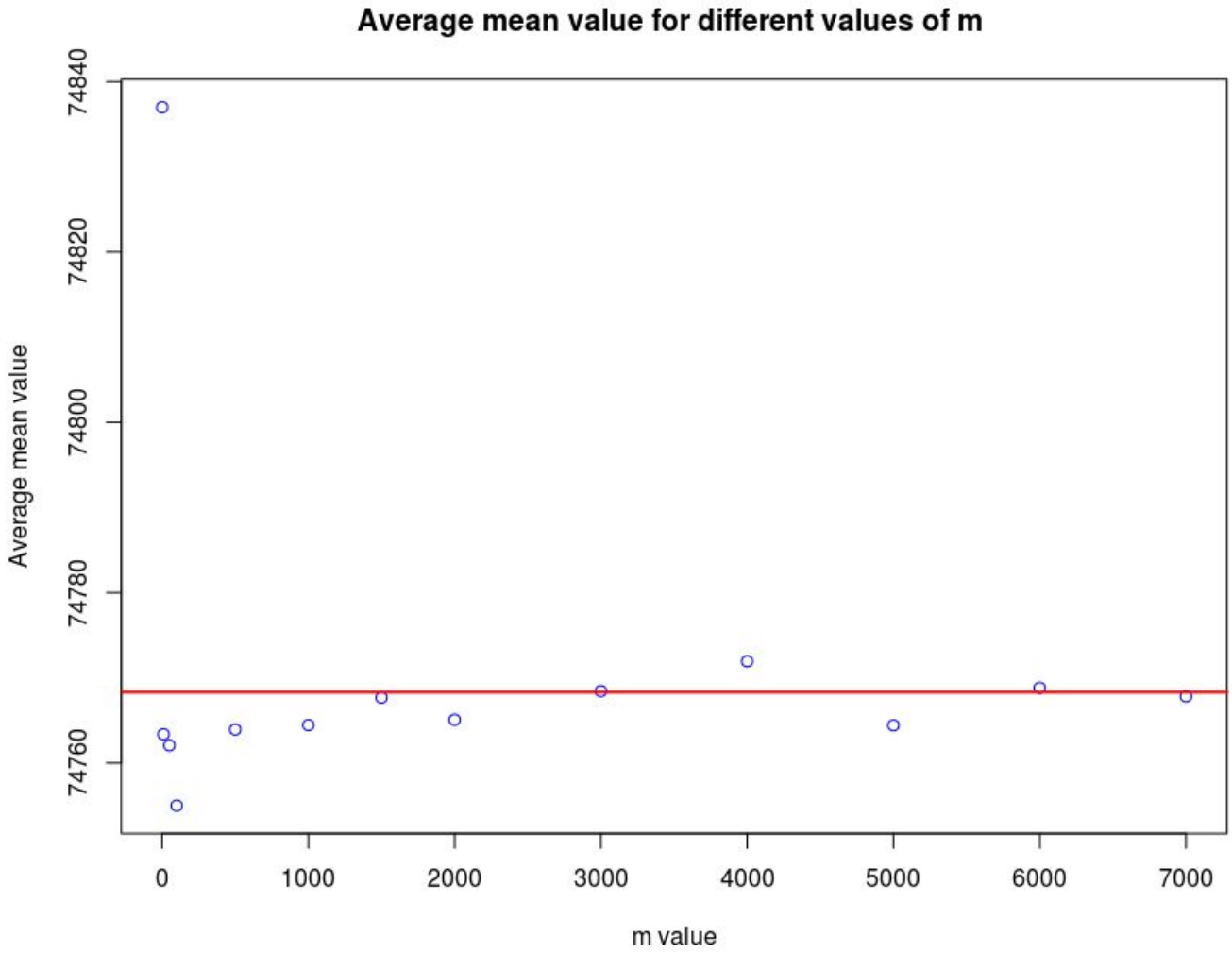
Here we can observe as the value of m increases the average mean value gets close to the mean of the entire column. Red line shows the mean of the entire column. The above plot also shows that sample mean values are following a normal distribution. That is what central limit theorem says, the distribution of the average of a large number of independent, identically distributed variables will be approximately normal, regardless of the underlying distribution.

With another different set of values of m

```
> m_vector =c(1,10,50,100,500,1000,1500,2000,3000,4000,5000,6000,7000)
>
> n= 148654
> # y vector will store the 100 average mean generated for different m values with mean=n/2 and sigma=100
> y= c()
>
> for (m in m_vector)
+ {
+   S= (rnorm(n = 1, mean = (n/2), sd = 100))
+   S = ceiling(S)
+   if(S> n)
+   {
+     S=n
+   }
+   else if(S<0)
+   {
+     S=1
+   }
+   x <- replicate(m, {
+     listofrows <- sample(1:148654, S , replace=FALSE)
+     sampleTotalpay <- salary[listofrows,"TotalPay"]
+     samplemean <- mean(sampleTotalpay)
+   })
+   meanx <- mean(x)
+   y <- c(y, meanx)
+ }
> y
 [1] 74836.99 74763.37 74762.06 74754.99 74763.93 74764.45 74767.67
 [8] 74765.07 74768.44 74771.93 74764.42 74768.82 74767.81
```

Now plotting

```
> #ploting
> plot( m_vector,y , main="Average mean value for different values of m " , ylim = c(min(y),max(y)) ,
+       pch = 1, col = "blue" ,xlab="m value", ylab="Average mean value")
>
> #adding the line to show average value of entire data
> abline(h=74768.32, col="red", lwd=2)
>
```



This plot also shows that as the value of m increases the average mean values generating from the above done sampling experiment will get close to the mean value of the entire column.

Observation:

1. As the value of μ increases and gets close to the total number of rows, the value of average mean also get close to the value of mean of the entire column. (**Law of Large number**)

2. As the value of m increases the average mean value gets close to the mean of the entire column. (**Central limit theorem**)