Mini Project 4 (Solution)

Mini Project Duo Group # 12
Contribution of each group member
Chetan Siddappareddy – 50%
Ankit Sahu – 50%
Both of us have contributed equally to the project. We learnt R through collaboration and then write the R scripts for the corresponding and report all the findings.

Section 1 for explanation (and R code snippets part wise) and Section 2 for R code (from local R Studio).

Section 1

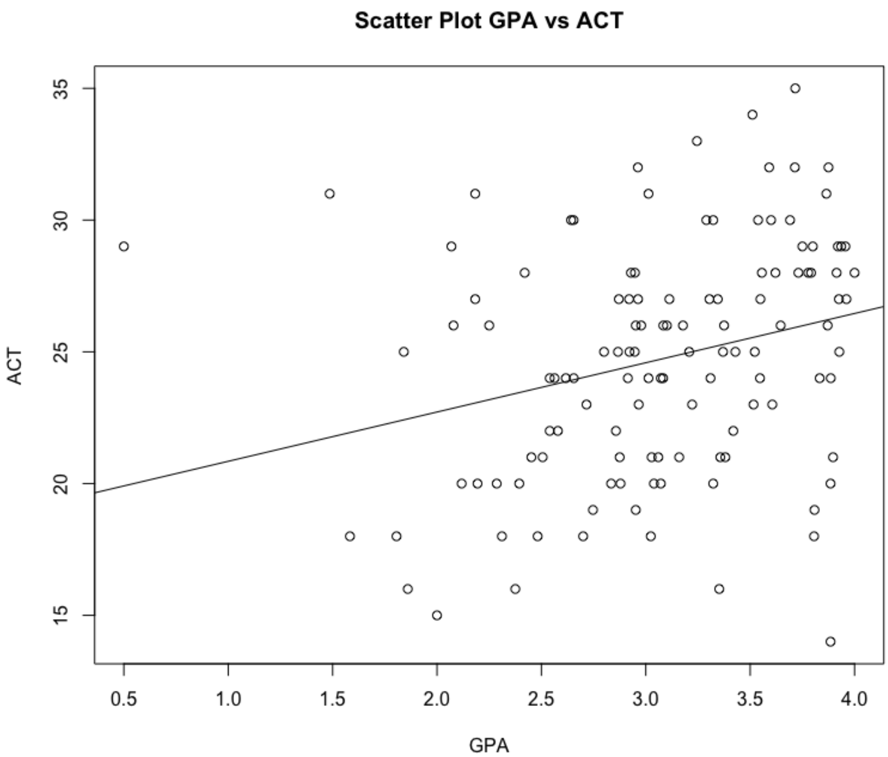
Problem 1

Reading the data given(gpa.csv)

```
data = read.csv("gpa.csv")
```

Storing the gpa and act in separate variable and then plotting the scatter plot.

```
# getting the gpa scores of the student in variable gpa
gpa = as.numeric(data$gpa)

# getting the act scores of the student in variable act
act = as.numeric(data$act)

plot(gpa, act, main = "Scatter Plot GPA vs ACT", xlab = "GPA", ylab="ACT");
```



Using Regression model

```
#Correlation
#abline() can be used to add vertical, horizontal or regression lines to a graph.
#lm() function -> is used to fit linear models -> regression
abline(lm(act~gpa))
```

From the plot generated, it can be infer that the value of GPA increases the value of the act score also. Now finding the correlation.

```
> cor(gpa, act)
[1] 0.2694818
```

Using some other sample for correlation:

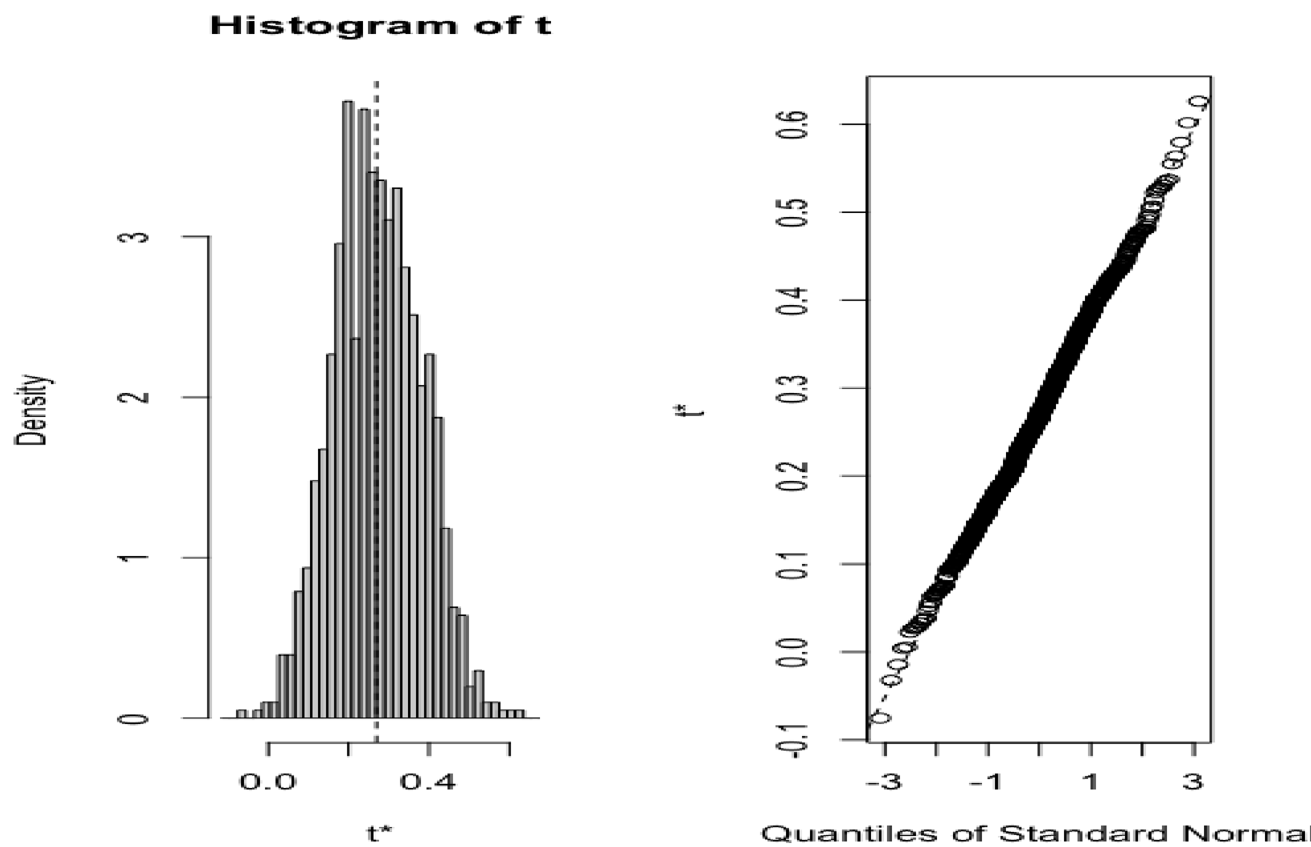
```
> library(boot)
>
> cov.npar = function(ank, iters){
+   gpa = ank$gpa[iters]
+   act = ank$act[iters]
+   result = cor(gpa, act)
+   return (result)
+ }
>
```

```
> cov.npar.boot = boot(data, cov.npar, R=999, sim="ordinary", stype="i")
>
> print(cov.npar.boot)
```

Calculating the correlation estimate :

```
ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = data, statistic = cov.npar, R = 999, sim = "ordinary",
    stype = "i")


Bootstrap Statistics :
     original      bias    std. error
t1* 0.2694818 0.003158357   0.1080157
```

From above, we can conclude that correlation estimate is 0.2694818



Using boot.ci for 95% CI (confidence interval):

```
#Confidence Interval(with boot.ci)
> print(boot.ci(cov.npar.boot))
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 999 bootstrap replicates

CALL :
boot.ci(boot.out = cov.npar.boot)

Intervals :
Level      Normal              Basic
95%   ( 0.0546,  0.4780 )   ( 0.0597,  0.4693 )

Level     Percentile            BCa
95%   ( 0.0696,  0.4793 )   ( 0.0630,  0.4683 )
Calculations and Intervals on Original Scale
```

Now, calculating 95% CI using percentile bootstrap

```
> print(sort(cov.npar.boot$t)[c(25,975)])
[1] 0.06963114 0.47930259
```

So, 95% CI using percentile bootstrap is (0.06963, 0.4793)

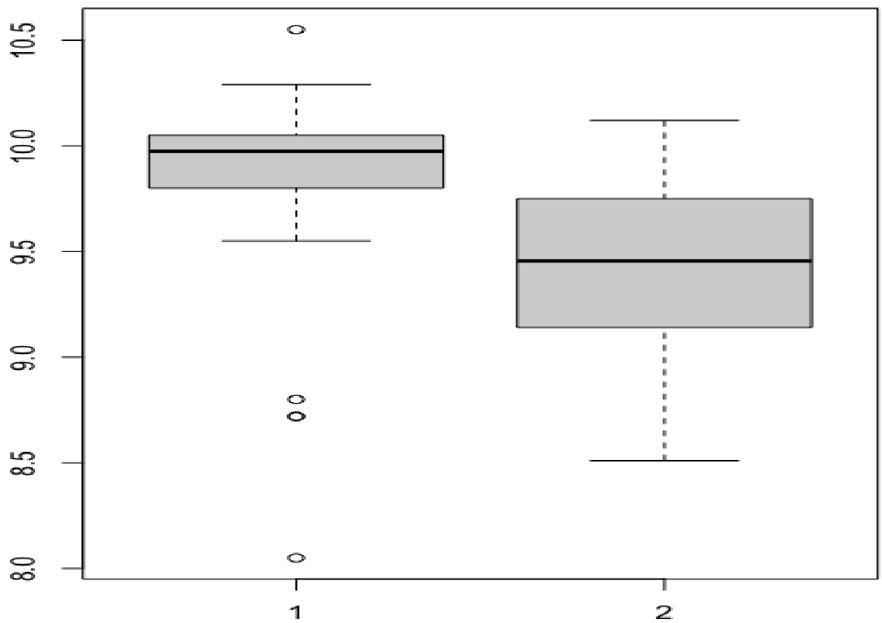
Problem 2

a) Exploration using the box plot (5 value analysis).
Filtering for remote and local also is done.

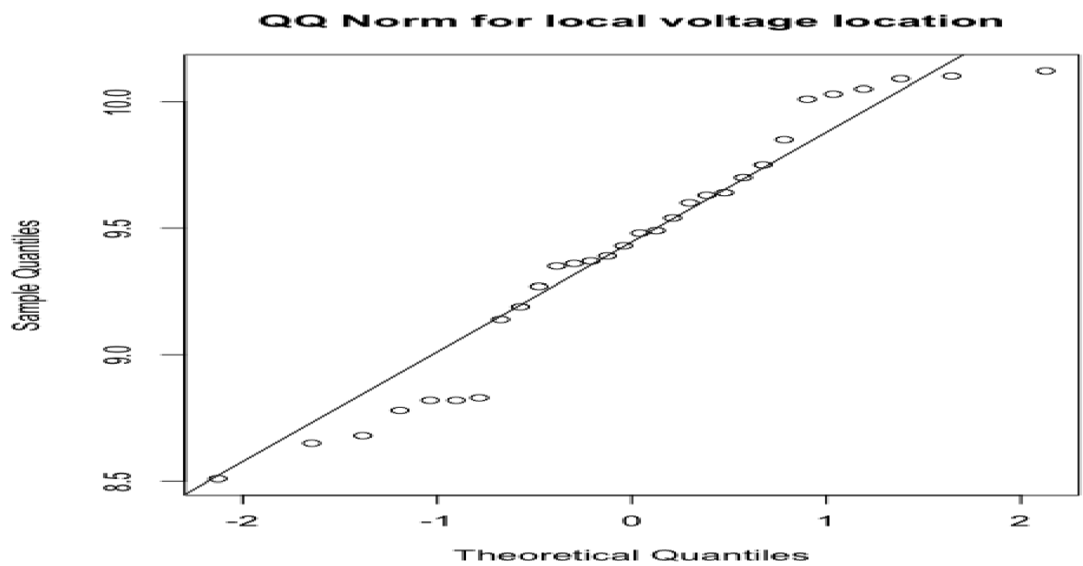
```
voltages = read.csv("VOLTAGE.csv")
print(voltages)

#for remote location 0 and local location 1

> remote = voltages$voltage[voltages$location==0]
> local = voltages$voltage[voltages$location==1]
> print(remote)
 [1]  9.98 10.26 10.05 10.29 10.03  8.05 10.55 10.26  9.97  9.87 10.12 10.05  9.80 10.15
10.00
[16]  9.87  9.55  9.95  9.70  8.72  9.84 10.15 10.02  9.80  9.73 10.01  9.98  8.72  8.80
9.84
> print(local)
 [1]  9.19  9.63 10.10  9.70 10.09  9.60 10.05 10.12  9.49  9.37 10.01  8.82  9.43 10.03
9.85
[16]  9.27  8.83  9.39  9.48  9.64  8.82  8.65  8.51  9.14  9.75  8.78  9.35  9.54  9.36  8.68
```

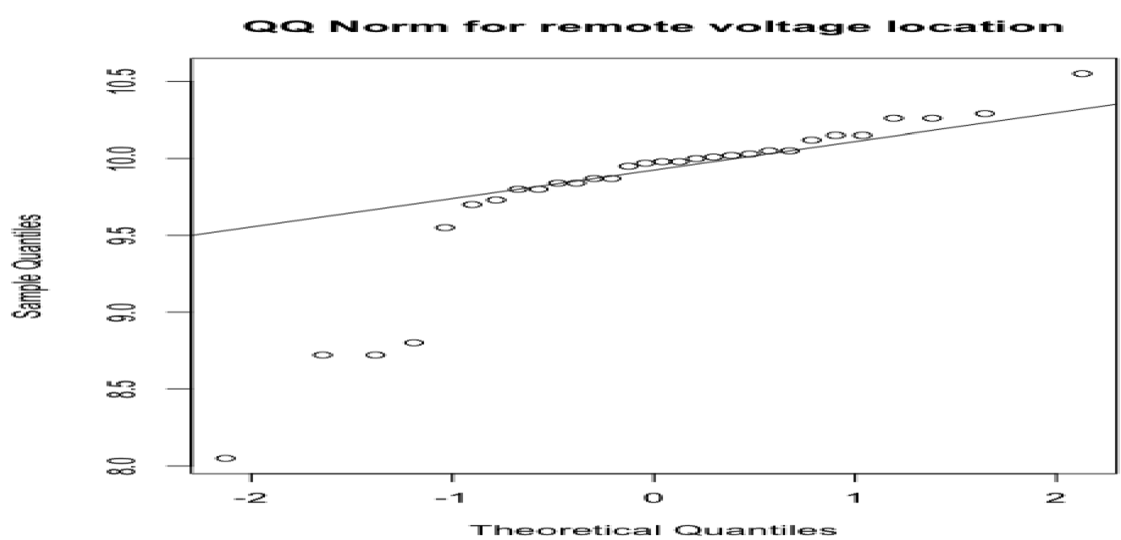
The box plot below shows that the voltage is higher for local (right) than the remote(left). Also, the remote location voltage is left skewed.
The QQ plots for the remote and the local locations also confirms the dissimilarity.

```
qqnorm(remote, main="QQ Norm for remote voltage location")
> qqline(remote)
> qqnorm(local, main="QQ Norm for local voltage location")
> qqline(local)
```



10.5
10.0
9.5
9.0
8.5
8.0
1
2





b)

We need to find variance to show dissimilarity between them.

> local_variance = var(local)

```
> remote_variance = var(remote)
> print(local_variance)
[1] 0.229322
> print(remote_variance)
[1] 0.2925895
```

As we can see from the R output, both population variances are different. So, we find the confidence interval for the population means of the voltages at the two locations. I am assuming that the populations are normally distributed and hence we can use the Welch's two sample t-test for finding the CI.

```
> t.test(remote,local, alternative = "two.sided", conf.level = 0.95,var.equal = FALSE)

Welch Two Sample t-test

data:  remote and local
t = 2.8911, df = 57.16, p-value = 0.005419
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.1172284 0.6454382
sample estimates:
mean of x mean of y
 9.803667  9.422333
```

Since, the CI does not contain zero value then we can conclude that the difference in population means of local and remote voltages cannot be zero. Therefore, we cannot establish the manufacturing process locally. We can perform manual calculations of the CI for the difference of population means to verify the assumptions.

```
> mean_remote = mean(remote)
> mean_local = mean(local)
> print(mean_local)
[1] 9.422333
> print(mean_remote)
[1] 9.803667
> ci = (mean_remote - mean_local) +c(-1,1)*qt(0.025,58)*sqrt((var(local) + var(remote))/30)
> print(ci)
[1] 0.6453556 0.1173110
```

c) We showed in the (a) that the two distributions of the voltages, remote and local are dissimilar, which led to the conclusion that the population means would be different. From the (b) we concluded the same. So we can say that the manufacturing cannot be established locally.

Problem 3

We will load the "VAPOR.csv" file and then analyze the data provided on the theoretical and experimental values. As per the problem, I am interested in calculating the difference between the experimental values and the theoretical values at the given set of temperatures. Thus, we calculate the difference from the given data as a paired sample.

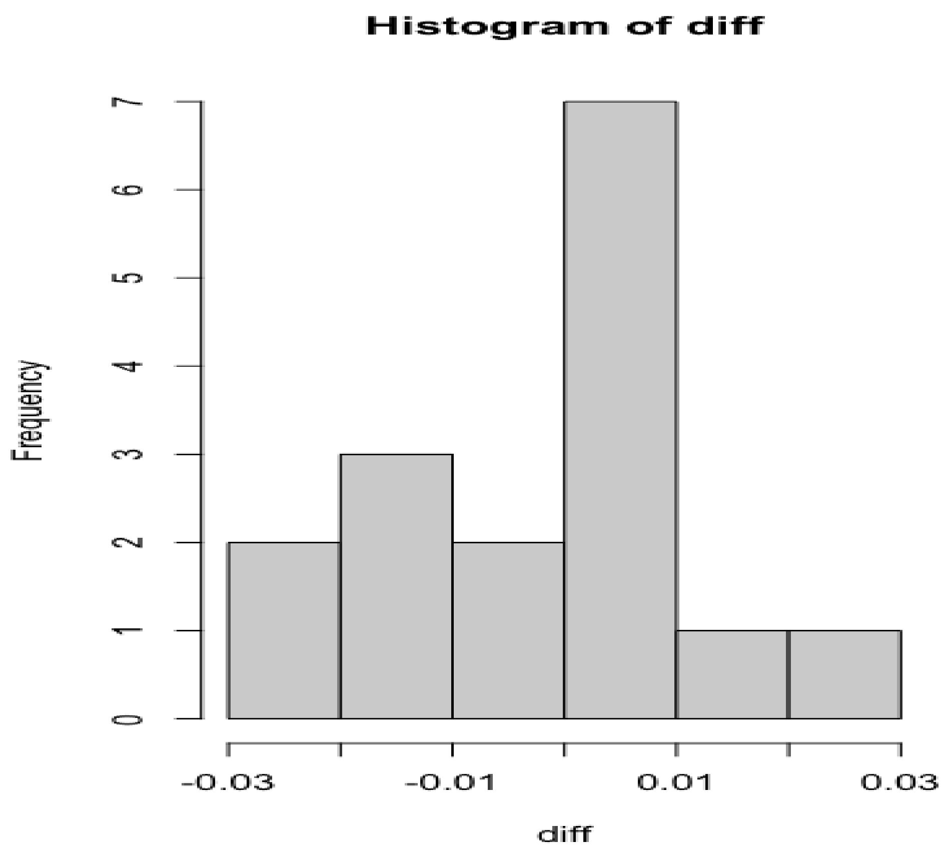
```
> vapor = read.csv("VAPOR.csv")


> theoretical = vapor$theoretical
> experimental = vapor$experimental
>
> diff = theoretical-experimental
>
> print(diff)
 [1]  0.006  0.007 -0.015  0.014 -0.022  0.008  0.000  0.002 -0.026  0.029  0.008  0.000 -0.010
0.010
[15] -0.010  0.010
```

Using histogram for looking at the difference.

```
> hist(diff)
```

By looking at the histogram, I can infer that the data is normally distributed. Now I will calculate the confidence interval of the mean of the difference in the data observation in both of the cases and if the confidence interval that we calculated includes a zero, we can say that the theoretical model for vapor pressure is a good model of reality.

```
> ci = mean(diff) + c(1,-1)*qt(0.975, 15)* (sd(diff)/sqrt(16))
> print(ci)
[1]  0.008262694 -0.006887694
```



Section 2

######################################
R CODE FOR PROBLEM 1:
######################################
 # Solution for Problem 1

```
setwd("/Users/sahuankit010/Desktop/Repo/CS-6313-Stats/Mini Projects/MP4")
getwd()
data = read.csv("gpa.csv")

print(data)

# getting the gpa scores of the student in variable gpa
gpa = as.numeric(data$gpa)
```

```
# getting the act scores of the student in variable act
act = as.numeric(data$act)


plot(gpa, act, main = "Scatter Plot GPA vs ACT", xlab = "GPA", ylab="ACT");

#Correlation
#abline() can be used to add vertical, horizontal or regression lines to a graph.
#lm() function -> is used to fit linear models -> regression
abline(lm(act~gpa))

cor(gpa, act)

#Estimates::

library(boot)

cov.npar = function(ank, iters){
  gpa = ank$gpa[iters]
  act = ank$act[iters]
  result = cor(gpa, act)
  return (result)
}

cov.npar.boot = boot(data, cov.npar, R=999, sim="ordinary", stype="i")

print(cov.npar.boot)

plot(cov.npar.boot)

# Now doing the Point Estimation of bootstrap

#mean
print(mean(cov.npar.boot$t))

#Confidence Interval(with boot.ci)
print(boot.ci(cov.npar.boot))

#Percentile CI
```

```
print(sort(cov.npar.boot$t)[c(25,975)])
```

```
> data = read.csv("gpa.csv")
>
> print(data)
     gpa act
1   3.897  21
2   3.885  14
3   3.778  28
4   2.540  22
5   3.028  21
6   3.865  31
7   2.962  32
8   3.961  27
9   0.500  29
10  3.178  26
11  3.310  24
12  3.538  30
13  3.083  24
14  3.013  24
15  3.245  33
16  2.963  27
17  3.522  25
18  3.013  31
19  2.947  25
20  2.118  20
21  2.563  24
22  3.357  21
23  3.731  28
24  3.925  27
25  3.556  28
26  3.101  26
27  2.420  28
28  2.579  22
29  3.871  26
30  3.060  21
```

```
113 2.394  20
114 2.286  20
115 1.486  31
116 3.885  20
117 3.800  29
118 3.914  28
119 1.860  16
120 2.948  28
>
> # getting the gpa scores of the student in variable gpa
> gpa = as.numeric(data$gpa)
>
> # getting the act scores of the student in variable act
> act = as.numeric(data$act)
>
>
> plot(gpa, act, main = "Scatter Plot GPA vs ACT", xlab = "GPA", ylab="ACT");
>
> #Correlation
> #abline() can be used to add vertical, horizontal or regression lines to a graph.
> #lm() function -> is used to fit linear models -> regression
> abline(lm(act~gpa))
>
> cor(gpa, act)
[1] 0.2694818
>
> #Estimates::
>
> library(boot)
>
> cov.npar = function(ank, iters){
+   gpa = ank$gpa[iters]
+   act = ank$act[iters]
+   result = cor(gpa, act)
+   return (result)
+ }
```

```
>
> cov.npar.boot = boot(data, cov.npar, R=999, sim="ordinary", stype="i")
>
> print(cov.npar.boot)

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = data, statistic = cov.npar, R = 999, sim = "ordinary",
    stype = "i")


Bootstrap Statistics :
     original       bias    std. error
t1* 0.2694818 0.004284193   0.1089403
>
> plot(cov.npar.boot)
>
> # Now doing the Point Estimation of bootstrap
>
> #mean
> print(mean(cov.npar.boot$t))
[1] 0.273766
>
> #Confidence Interval(with boot.ci)
> print(boot.ci(cov.npar.boot))
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 999 bootstrap replicates
```

```
CALL :
boot.ci(boot.out = cov.npar.boot)

Intervals :
Level      Normal              Basic
95%   ( 0.0517,  0.4787 )   ( 0.0527,  0.4831 )

Level     Percentile            BCa
95%   ( 0.0559,  0.4863 )   ( 0.0280,  0.4746 )
Calculations and Intervals on Original Scale
Warning message:
In boot.ci(cov.npar.boot) :
  bootstrap variances needed for studentized intervals
>
> #Percentile CI
>
> print(sort(cov.npar.boot$t)[c(25,975)])
[1] 0.05585446 0.48630454
>
```

R CODE FOR PROBLEM 2:

#######################################

 # Solution for Problem 2

a)

```
setwd("/Users/sahuankit010/Desktop/Repo/CS-6313-Stats/Mini Projects/MP4")
getwd()

voltages = read.csv("VOLTAGE.csv")
print(voltages)

#for remote location 0 and local location 1

remote = voltages$voltage[voltages$location==0]
local = voltages$voltage[voltages$location==1]

print(remote)
print(local)

boxplot(remote, local)

qqnorm(remote, main="QQ Norm for remote voltage location")
qqline(remote)

qqnorm(local, main="QQ Norm for local voltage location")
```

```
qqline(local)

local_variance = var(local)
remote_variance = var(remote)

print(local_variance)
print(remote_variance)

t.test(remote,local, alternative = "two.sided", conf.level = 0.95,var.equal = FALSE)

mean_remote = mean(remote)
mean_local = mean(local)

print(mean_local)
print(mean_remote)



ci = (mean_remote - mean_local) +c(-1,1)*qt(0.025,58)*sqrt((var(local) + var(remote))/30)
print(ci)
```

```
> voltages = read.csv("VOLTAGE.csv")
> print(voltages)
   location voltage
1         0    9.98
2         0   10.26
3         0   10.05
4         0   10.29
5         0   10.03
6         0    8.05
7         0   10.55
8         0   10.26
9         0    9.97
10        0    9.87
11        0   10.12
12        0   10.05
13        0    9.80
14        0   10.15
15        0   10.00
16        0    9.87
17        0    9.55
18        0    9.95
19        0    9.70
20        0    8.72
21        0    9.84
22        0   10.15
23        0   10.02
24        0    9.80
25        0    9.73
26        0   10.01
27        0    9.98
28        0    8.72
29        0    8.80
30        0    9.84
```

```
44        1   10.03
45        1    9.85
46        1    9.27
47        1    8.83
48        1    9.39
49        1    9.48
50        1    9.64
51        1    8.82
52        1    8.65
53        1    8.51
54        1    9.14
55        1    9.75
56        1    8.78
57        1    9.35
58        1    9.54
59        1    9.36
60        1    8.68
> remote = voltages$voltage[voltages$location==0]
> local = voltages$voltage[voltages$location==1]
> print(remote)
 [1]  9.98 10.26 10.05 10.29 10.03  8.05 10.55 10.26  9.97  9.87 10.12 10.05  9.80 10.15 10.00
[16]  9.87  9.55  9.95  9.70  8.72  9.84 10.15 10.02  9.80  9.73 10.01  9.98  8.72  8.80  9.84
> print(local)
 [1]  9.19  9.63 10.10  9.70 10.09  9.60 10.05 10.12  9.49  9.37 10.01  8.82  9.43 10.03  9.85
[16]  9.27  8.83  9.39  9.48  9.64  8.82  8.65  8.51  9.14  9.75  8.78  9.35  9.54  9.36  8.68
> boxplot(remote, local)
> qqnorm(remote, main="QQ Norm for remote voltage location")
> qqline(remote)
> qqnorm(local, main="QQ Norm for local voltage location")
> qqline(local)
> qqnorm(remote, main="QQ Norm for remote voltage location")
> qqline(remote)
> qqnorm(local, main="QQ Norm for local voltage location")
> qqline(local)
```

```
> qqline(local)
> qqnorm(remote, main="QQ Norm for remote voltage location")
> qqline(remote)
> qqnorm(local, main="QQ Norm for local voltage location")
> qqline(local)
> local_variance = var(local)
> remote_variance = var(remote)
> print(local_variance)
[1] 0.229322
> print(remote_variance)
[1] 0.2925895
> t.test(remote,local, alternative = "two.sided", conf.level = 0.95,var.equal = FALSE)

        Welch Two Sample t-test

data:  remote and local
t = 2.8911, df = 57.16, p-value = 0.005419
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.1172284 0.6454382
sample estimates:
mean of x mean of y
 9.803667  9.422333

> mean_remote = mean(remote)
> mean_local = mean(local)
> print(mean_local)
[1] 9.422333
> print(mean_remote)
[1] 9.803667
> ci = (mean_remote - mean_local) +c(-1,1)*qt(0.025,58)*sqrt((var(local) + var(remote))/30)
> print(ci)
[1] 0.6453556 0.1173110
>
```

R CODE FOR PROBLEM 3:

```
######################################

# Solution for Problem 3

vapor = read.csv("VAPOR.csv")
print(vapor)

theoretical = vapor$theoretical
experimental = vapor$experimental

diff = theoretical-experimental

print(diff)

hist(diff)

ci = mean(diff) + c(1,-1)*qt(0.975, 15)* (sd(diff)/sqrt(16))
print(ci)
```

Console Terminal

R 4.2.1 · ~/Desktop/Repo/CS-6313-Stats/Mini Projects/MP4/

```
> vapor = read.csv("VAPOR.csv")
> print(vapor)
   temperature theoretical experimental
1       100.60       0.282        0.276
2       101.36       0.314        0.307
3       104.60       0.335        0.350
4       106.44       0.404        0.390
5       108.70       0.422        0.444
6       110.96       0.513        0.505
7       112.62       0.554        0.554
8       115.21       0.642        0.640
9       116.69       0.669        0.695
10      119.38       0.834        0.805
11      121.08       0.890        0.882
12      123.61       1.010        1.010
13      124.90       1.070        1.080
14      127.74       1.260        1.250
15      130.24       1.420        1.430
16      131.75       1.550        1.540
>
> theoretical = vapor$theoretical
> experimental = vapor$experimental
>
> diff = theoretical-experimental
>
> print(diff)
 [1]  0.006  0.007 -0.015  0.014 -0.022  0.008  0.000  0.002 -0.026  0.029  0.008  0.000 -0.010  0.010
[15] -0.010  0.010
>
> hist(diff)
>
> ci = mean(diff) + c(1,-1)*qt(0.975, 15)* (sd(diff)/sqrt(16))
> print(ci)
[1]  0.008262694 -0.006887694
>
```