Mini Project 4 Report

Jiadao Zou, jxz172230

Q1

In [62]:
```
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
from scipy import stats
```

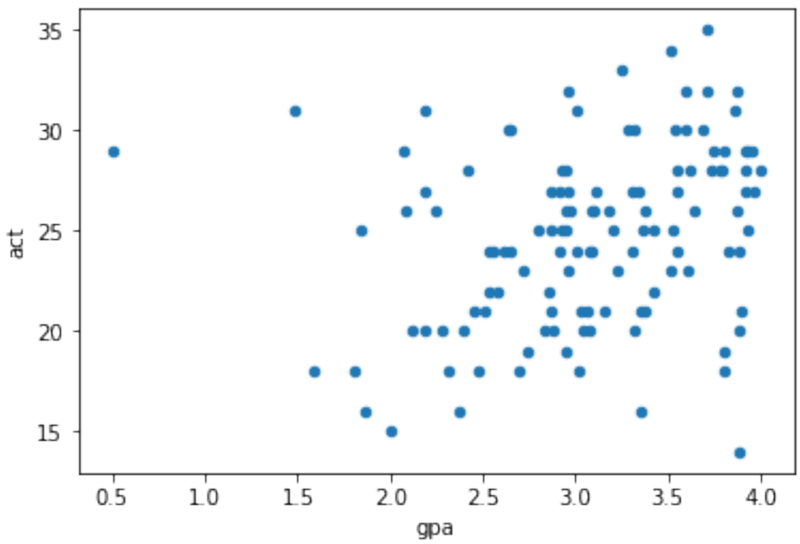
In [64]:
```
gpa = pd.read_csv("gpa.csv")
```

In [65]:
```
gpa.head(3)
#gpa.describe()
```

Out[65]:

	gpa	act
0	3.897	21
1	3.885	14
2	3.778	28

In [66]:
```
fig1_1 = gpa.plot.scatter(x='gpa', y='act')
```



Answer:

From the plot, we could see that: in general, while GPA is going up, ACT would goes up as well. We could say that there is a positive relationship between those two data. Since the data point is not so close to each other, the strength is not very strong.

In [67]:
```
cor = gpa['gpa'].corr(gpa['act'])
cor
```

Out[67]:
```
0.2694818032662637
```

Answer:

Furthermore, we compute correlation coefficient as well. Since the result is population correlation 0.269, since we already know that:

```
1. 0 indicates no linear relationship.
2. +1 indicates a perfect positive linear relationship.
3. -1 indicates a perfect negative linear relationship.
4. Values between 0 and 0.3 (0 and -0.3) indicate a weak positive (nega
tive) linear relationship via a shaky linear rule.
5. Values between 0.3 and 0.7 (-0.3 and -0.7) indicate a moderate posit
ive (negative) linear relationship via a fuzzy-firm linear rule.
6. Values between 0.7 and 1.0 (-0.7 and -1.0) indicate a strong positiv
e (negative) linear relationship via a firm linear rule.
```

Thus, there is a weak positive linear relationship between "GPA" and "ACT".

R code in the following ¶

```
my <- read.csv(file = "/home/jiadao/Code/Github/CS6313-Stat/Pj4/gpa.csv
", header = TRUE, sep = ",")
df <- data.frame(my)
n = nrow(df)
Brep = 10000

# Bootstrap the correlation coefficient
cc <- function(d,i=c(1:n)){
        d2 <- d[i,]
        return(cor(d2$gpa,d2$act))
     }
bootcorr <- boot(data=df,statistic=cc,R=Brep)
bootcorr
        ORDINARY NONPARAMETRIC BOOTSTRAP

        Call:
        boot(data = df, statistic = cc, R = Brep)

        Bootstrap Statistics :
            original      bias    std. error
        t1* 0.2694818 0.003035951   0.1060727
boot.ci(bootcorr,conf=.95)
        BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
        Based on 10000 bootstrap replicates

        CALL :
        boot.ci(boot.out = bootcorr, conf = 0.95)

        Intervals :
        Level      Normal              Basic
        95%   ( 0.0585,  0.4743 )   ( 0.0581,  0.4738 )

        Level     Percentile            BCa
        95%   ( 0.0652,  0.4809 )   ( 0.0414,  0.4604 )
        Calculations and Intervals on Original Scale
```

Answer:

From above, we could see that Normal and Percentile CI at the confidence level of 95% which is the point estimation of μ. The mean, bias and standard error could be seen from the Bootstrap Statistics.
Comparing its with the basic method result: $CI \in [\bar{M} - Z_{\alpha/2} * \frac{S}{\sqrt{N}}, \bar{M} + Z_{\alpha/2} * \frac{S}{\sqrt{N}}]$. Also, we could see Percentile bootstrap has a larger difference with its Basic approximation comparing to normal one.

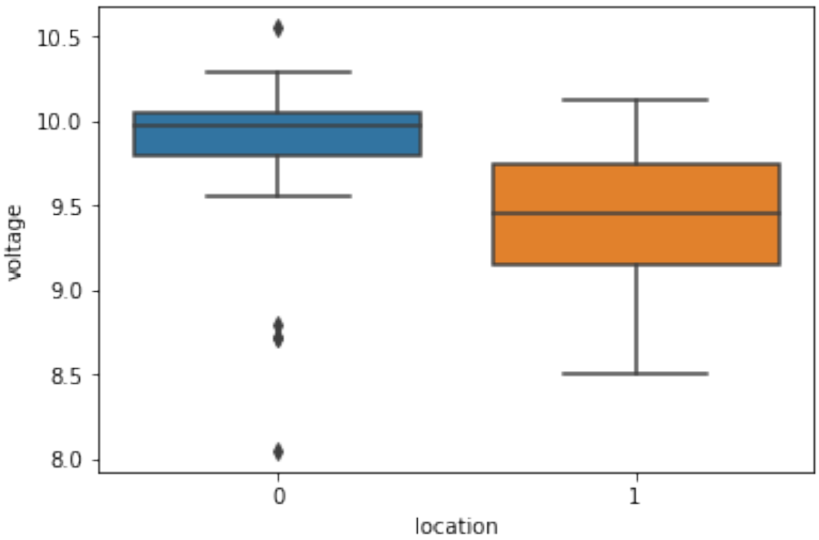
Q2

In [71]:
```
volt = pd.read_csv("VOLTAGE.csv")
```

In [72]:
```
volt_loc = volt.loc[volt["location"] == 1]
volt_rem = volt.loc[volt["location"] == 0]
print("sample number of LOCAL is %d and sample number of REMOTE is %d" %(len(volt_loc), len(volt_rem)))
```

```
sample number of LOCAL is 30 and sample number of REMOTE is 30
```

In [73]:
```
import seaborn as sns
fig2 = sns.boxplot(data=volt, x='location', y='voltage')
```



Answer:

From the boxplot, we could these two sample data are different, either in mean or Interquartile Range. Also, there are some outliers in remote dataset. So, we can't simply approximate the remote result by local situation.

In [74]:
```
def normCI(data, confidence=0.95):
    a = 1.0 * np.array(data)
    n = len(a)
    m, se = np.mean(a), stats.sem(a)
    h = se * stats.norm.ppf((1 + confidence) / 2.)
    #     return m, m-h, m+h
    print("From {:0.2%}".format(confidence),"confidence interval analysis of normal distribution, mean is %f, lower bound is %f, upper bound is %f" %(m, m-h, m+h))
```

In [75]:
```
normCI(volt_rem['voltage'])
```

```
From 95.00% confidence interval analysis of normal distribution, m
ean is 9.803667, lower bound is 9.610106, upper bound is 9.997227
```

In [76]:
```
normCI(volt_loc['voltage'])
```

```
From 95.00% confidence interval analysis of normal distribution, m
ean is 9.422333, lower bound is 9.250973, upper bound is 9.593694
```

Answer:

By doing normal confidence analysis (since the size of each class is 30 > 25, and we could applying CLT). Since the CI of two class are not overlapped with each other. There is some difference between those two locations

Answer:

By comparing two results above, we could see they agree with each other in that we can't expect remote working value is the same as local working.

Q3

In [78]:
```
vapor = pd.read_csv("VAPOR.csv")
```

In [79]:
```
vapor['diff'] = vapor['theoretical'] - vapor['experimental']
```

In [80]:
```
pd.DataFrame(vapor['diff']).describe()
```

Out[80]:

	diff
count	16.000000
mean	0.000688
std	0.014216
min	-0.026000
25%	-0.010000
50%	0.004000
75%	0.008500
max	0.029000

above is the statistics information of the sample (difference of theoretical and experimental)

In [ ]:

```
def tCI(data, confidence=0.95):
    a = 1.0 * np.array(data)
    n = len(a)
    m, se = np.mean(a), stats.sem(a)
    h = se * stats.t.ppf((1 + confidence) / 2., n-1)
#     return m, m-h, m+h
    print("From {:0.2%}".format(confidence),"confidence interval analysis of t-student distribution, mean is %f, lower bound is %f, upper bound is %f" %(m, m-h, m+h))
```

In [81]:

```
tCI(vapor['diff'])
```

```
From 95.00% confidence interval analysis of t-student distribution
, mean is 0.000688, lower bound is -0.006888, upper bound is 0.008
263
```

Answer:

By using t-distribution to simulate the population, we could see that though mean is not 0, but it is still pretty close to it, along with that 0 lay inside the CI's lower bound and upper bound. So we could make the conculsion that the theoretical model for vapor pressure is a good model of reality.