Homework 3

Name: Quan Yuan **UNI**: qy2205

2.2

No, we cannot make that conclusion. Since when $\beta_1 < 0$, there exists a negative correlation between X and Y.

2.23

(a) ANOVA Table

```
# linear regression model
lm0 = LinearRegression()
lm0.fit(X = data['ACT'].values.reshape(-1,1), y = data['GPA'])
y = data['GPA'].values
y_hat = lm0.predict(data['ACT'].values.reshape(-1,1))
y_mean = np.mean(y)
# slope and intercept
print('slope is {0}, intercept is {1}'.format(round(lm0.coef_[0],4), round(lm0.intercept_, 4)))
# SS regression
SS_reg = sum((y_hat - y_mean)**2)
# SS residual
SS_residual = sum((y - y_hat)**2)
# SS total
SS_total = SS_reg + SS_residual
print('SS_reg: {0}, SS_residual: {1}'.format(round(SS_reg,4), round(SS_residual, 4)))
# MSS
n = 120
MSS_reg = SS_reg/1
MSS_residual = SS_residual/(n-2)
print('MSS_reg: {0}, MSS_residual: {1}'.format(round(MSS_reg, 4), round(MSS_residual, 4)))
# F-stat
F = MSS_reg/MSS_residual
print('F statistics: {0}'.format(round(F, 4)))
# p-value
p_value = 1 - f.cdf(F, 1, 118)
print('P value: {0}'.format(round(p_value,4)))
```

```
slope is 0.0388, intercept is 2.114
SS_reg: 3.5878, SS_residual: 45.8176
MSS_reg: 3.5878, MSS_residual: 0.3883
F statistics: 9.2402
P value: 0.0029
```

```
mod = ols('GPA~ACT', data=data).fit()
aov_table = sm.stats.anova_lm(mod, typ = 2)
aov_table
```

	sum_sq	df	F	PR(>F)
ACT	3.587846	1.0	9.240243	0.002917
Residual	45.817608	118.0	NaN	NaN

Table 1: ANOVA Table

	df	SS	MSS	F-stat	p-value
Regression	1	3.5878	3.5878	9.2402	0.0029
Residual	118	45.8176	0.3883		
Total	119	49.4055			

(b)

$$SSR = \sum (\hat{Y}_i - \bar{Y})^2$$
$$SSE = \sum (\hat{Y}_i - Y_i)^2$$

MSR in ANOVA Table is 3.5878, MSE is 0.3883.

$$E(MSR) = E\left(\sum_{i=1}^{n} (\hat{Y}_i - \bar{Y})^2\right) = \sigma^2 + {\beta_1}^2 \sum_{i=1}^{n} (x_i - \bar{x})^2$$
$$\mathrm{E(MSE)} = \sigma^2$$

So, when $\beta_1 = 0$, $MSR = MSE$

(c)

Alternatives: $H_0: \beta_1 = 0, H_1: \beta_1 \neq 0$

F-test: $F - statistics: F = \frac{MSR}{MSE} = 9.2402$

Decision rule: $F - statistics > F(0.99; 1, 118) = 6.8546, reject\ H_0$

So $\beta_1 \neq 0$

(d)

The absolute magnitude of the reduction is $SS\_regression = 3.5878$
The relative reduction is

$$R^2 = \frac{SS\_regression}{SS\_total} = \frac{3.5878}{49.4054} = 0.0726$$

The name of latter is called coefficient of determination.

(e)

Since slope is 0.0388, r should be positive

$$r = \sqrt{R^2} = 0.2694$$

(f)

R^2 has more clear-cut operational interpretation. It shows the proportionate reduction of total variation associated with the use of predictor variable X.

2.26

(a) ANOVA table

```
anova(data, 'elapsed', 'hardness')
```

```
slope is 2.0344, intercept is 168.6
SS_reg: 5297.5125, SS_residual: 146.425
MSS_reg: 5297.5125, MSS_residual: 10.4589
F statistics: 506.5062
P value: 0.0
```

```python
def anova(data, x, y):
    # linear regression model
    lm0 = LinearRegression()
    lm0.fit(X = data[x].values.reshape(-1,1), y = data[y])
    y = data[y].values
    y_hat = lm0.predict(data[x].values.reshape(-1,1))
    y_mean = np.mean(y)
    # slope and intercept
    print('slope is {0}, intercept is {1}'.format(round(lm0.coef_[0],4), round(lm0.intercept_, 4)))
    # SS regression
    SS_reg = sum((y_hat - y_mean)**2)
    # SS residual
    SS_residual = sum((y - y_hat)**2)
    # SS total
    SS_total = SS_reg + SS_residual
    print('SS_reg: {0}, SS_residual: {1}'.format(round(SS_reg,4), round(SS_residual, 4)))
    # MSS
    n = len(data)
    MSS_reg = SS_reg/1
    MSS_residual = SS_residual/(n-2)
    print('MSS_reg: {0}, MSS_residual: {1}'.format(round(MSS_reg, 4), round(MSS_residual, 4)))
    # F-stat
    F = MSS_reg/MSS_residual
    print('F statistics: {0}'.format(round(F, 4)))
    # p-value
    p_value = 1 - f.cdf(F, 1, 118)
    print('P value: {0}'.format(round(p_value,4)))
```

Table 2: ANOVA Table

	df	SS	MSS	F-stat	p-value
Regression	1	5297.513	5297.513	506.506	0.0
Residual	14	146.425	10.459		
Total	15	5443.938			

(b)

Alternatives: $H_0: \beta_1 = 0, H_1: \beta_1 \neq 0$

F-test: $F - statistics: F = \frac{MSR}{MSE} = 9.2402$

Decision rule: $F - statistics > F(0.99; 1, 118) = 6.8546, reject\ H_0$

So $\beta_1 \neq 0$

(c)

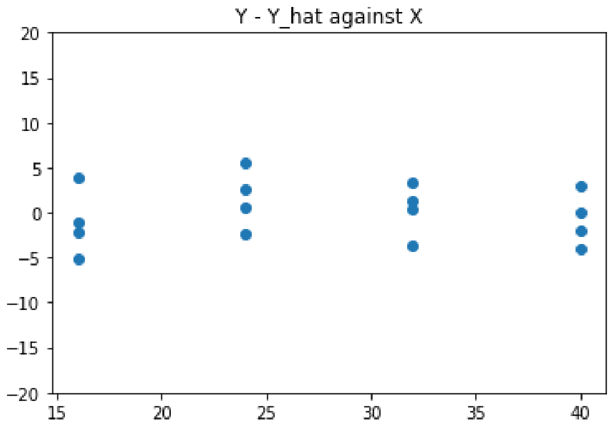


Figure 1: Y – Y_hat against X

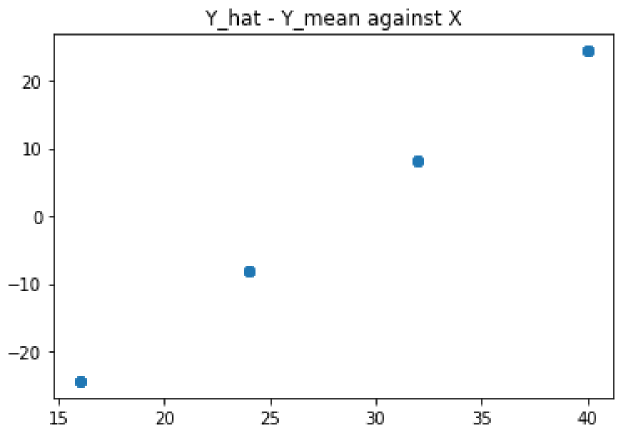


Figure 2: Y_hat – Y_mean against X

From the plot, we can see that SSR appear to be the larger proportion of SSTO, it implies that R^2 is close to 1 than to 0.

(d)

$$R^2 = \frac{SS_{Regression}}{SS_{Total}} = \frac{5297.512}{5443.938} = 0.9731$$

Since $\beta_1 = 2.034 > 0$, r should be positive.

$$r = \sqrt{R^2} = \sqrt{0.9731} = 0.9865$$

2.56

(a)

$$E(MSR) = E\left(\sum_{i=1}^{n}\left(\hat{Y}_i - \bar{Y}\right)^2\right) = \sigma^2 + {\beta_1}^2 \sum_{i=1}^{n}(x_i - \bar{x})^2 = 1026.36$$

$$E(MSE) = \sigma^2 = 0.36$$

(b)

It would be worse since the distance between x_i are small, which makes MSR small and F-statistics equal to 1.

No. Since the confidence interval is

$$Y \pm qMSE\sqrt{\frac{1}{n} + \frac{(x - \bar{x})^2}{\sum(x_i - \bar{x})^2}}$$

Since the mean of X = 6, 7, 8, 9, 10 and the mean of X = 1, 4, 10, 11, 14 are the same. So the confidence intervals for two observations are of the same.

2.61

According to 1.10 (a)

$$b_1 = \frac{\sum(X_i - \bar{X})(Y_i - \bar{Y})}{\sum(X_i - \bar{X})^2}$$

According to 2.51

$$SSR = {b_1}^2 \sum (X_i - \bar{X})^2$$

$$SSTO = \sum (Y_i - \bar{Y})^2$$

$$\frac{SSR}{SSTO} = \frac{b_1{}^2 \sum(X_i - \bar{X})^2}{\sum(Y_i - \bar{Y})^2} = \frac{(\sum(X_i - \bar{X})(Y_i - \bar{Y}))^2}{\sum(Y_i - \bar{Y})^2 \sum(X_i - \bar{X})^2}$$

So based on the formula above, we could see whether Y_1 is regressed on Y_2 or Y_2 is regressed on Y_1, the results are the same.

2.66

(a)

Confidence interval for $E(Y_h)$, $1 - \alpha$ confidence limits are:

$$\hat{Y}_h \pm t\left(1 - \frac{\alpha}{2}; n - 2\right) s\{\hat{Y}_h\}$$

$$s\{\hat{Y}_h\} = \sqrt{\frac{\sum(Y_i - \hat{Y}_i)^2}{n-2}\left[\frac{1}{n} + \frac{(x_h - \bar{x})^2}{\sum(x_i - \bar{x})^2}\right]}$$

```
# random error
error = np.random.normal(0, 25, 5)
# five x
x = np.array([4, 8, 12, 16, 20])
# five y
y = np.zeros(5)
for i, x_error in enumerate(zip(x, error)):
    y[i] = 20 + 4*x_error[0] + x_error[1]
print('y: {0}'.format(y))
# run linear regression
lm0 = LinearRegression()
lm0.fit(X = x.reshape(-1,1), y = y)
b1 = lm0.coef_[0]
b0 = lm0.intercept_
# slope and intercept
print('slope is {0}, intercept is {1}'.format(round(b1, 4), round(b0, 4)))
# interval
yh = lm0.predict(10)[0]
t_stat = t.ppf(1-0.025, 3)
y_hat = lm0.predict(x.reshape(-1,1))
mse = sum((y-y_hat)**2)/(5-2)
s_yh = np.sqrt(mse*(1/5 + (10-np.mean(x))**2/sum((x-np.mean(x))**2)))
upper = yh + t_stat*s_yh
lower = yh - t_stat*s_yh
print('upper: {0}; lower: {1}'.format(upper, lower))
```

```
y: [ 31.74943652  31.06275428  64.11925164  66.49482808 140.13238469]
slope is 6.3049, intercept is -8.9477
upper: 88.19477186238319; lower: 20.00889321413991
```

(b) & (c)

$$E(b_1) = \beta_1 = 4$$

$$\sigma(b_1) = \frac{\sigma}{\sqrt{\sum(x_i - \bar{x})^2}} = 0.3953$$

```python
def run():
    # random error
    error = np.random.normal(0, 5, 5)
    # five x
    x = np.array([4, 8, 12, 16, 20])
    # five y
    y = np.zeros(5)
    for i, x_error in enumerate(zip(x, error)):
        y[i] = 20 + 4*x_error[0] + x_error[1]
    # run linear regression
    lm0 = LinearRegression()
    lm0.fit(X = x.reshape(-1,1), y = y)
    b1 = lm0.coef_[0]
    b0 = lm0.intercept_
    # interval
    yh = lm0.predict(10)[0]
    t_stat = t.ppf(1-0.025, 3)
    y_hat = lm0.predict(x.reshape(-1,1))
    mse = sum((y-y_hat)**2)/(5-2)
    s_yh = np.sqrt(mse*(1/5 + (10-np.mean(x))**2/sum((x-np.mean(x))**2)))
    upper = yh + t_stat*s_yh
    lower = yh - t_stat*s_yh
    return {'b0': b0, 'b1': b1, 'upper': upper, 'lower': lower}
```

```python
# run 200 times
n = 200
b1 = np.zeros(200)
for i in range(200):
    result = run()
    b1[i] = result['b1']
print('mean of b1: {0}, standard deviation of b1: {1}'.format(np.mean(b1), np.std(b1)))
```

```
mean of b1: 3.9679822178014774, standard deviation of b1: 0.3769201133860657
```

Based on the program output, the results consistent with theoretical expectations.

(d)

The proportion is 96% and the result consistent with theoretical expectations.

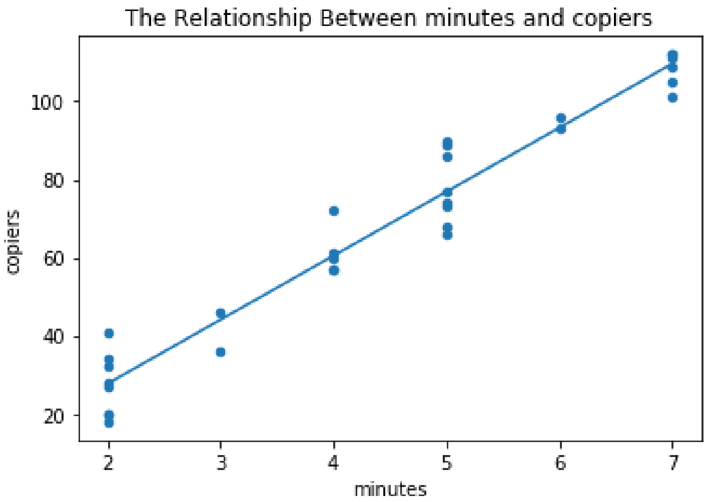
```python
# run 200 times
n = 200
E_yh = 60
count = 0
for i in range(200):
    result = run()
    upper = result['upper']
    lower = result['lower']
    if E_yh <= upper and E_yh >= lower:
        count += 1
print('The proportion is:', count/200)
```

```
The proportion is: 0.96
```

2.68

(a)

With the class we developed in homework 1, the result is showing below.



(b)

The confidence band for regression line is

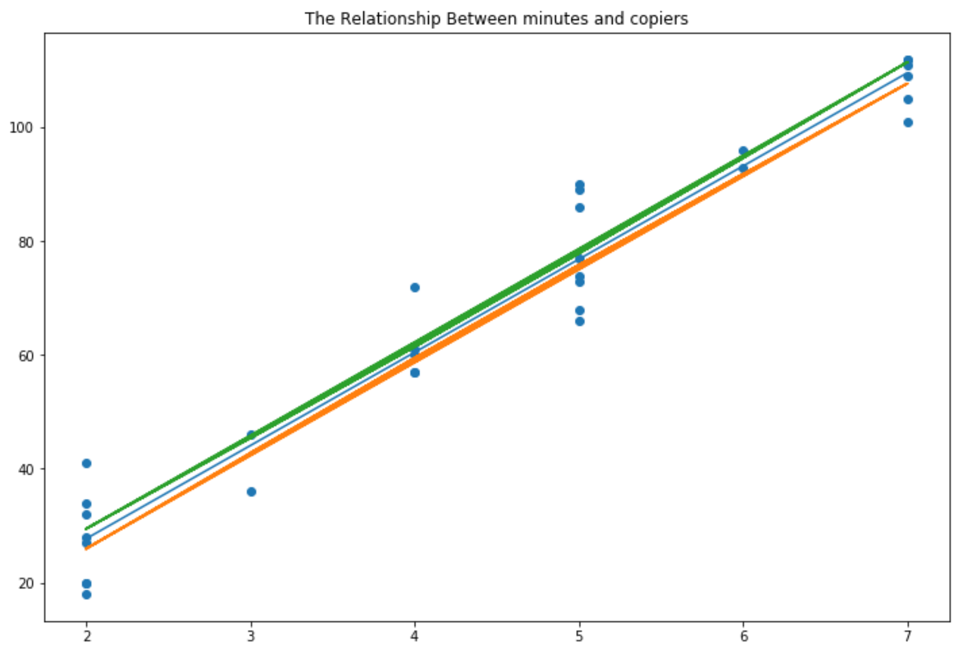
$$\hat{Y}_h \pm w \cdot s\{\hat{Y}_h\}$$

Where:

$$w^2 = 2F(1-\alpha; 2, n-2)$$

$$\hat{Y}_h = b_0 + b_1 X_h$$

$$s^2\{\hat{Y}_h\} = MSE\left[\frac{1}{n} + \frac{(x_h - \bar{x})^2}{\sum(x_i - \bar{x})^2}\right]$$



Yes, the confidence band suggest that the true regression relation has been precisely estimated since all three lines are close to each other.

```
class LinearReg:
    def __init__(self, data):
        '''data: type: pandas dataframe'''
        self.data = data
        self.length = len(data)
    def ols(self, x, y):
        '''x: column name y: column name'''
        X = np.matrix(np.vstack([np.ones(self.length), self.data[x].values]).T)
        y = np.matrix(self.data[y].values).T
        beta = np.linalg.inv(X.T*X)*X.T*y
        return beta
    def visual(self, x, y, step = 0.01):
        para = self.ols(x, y)
        X = self.data[x]
        Y = self.data[y]
        min_x, max_x = min(X), max(X)
        # x is also matrix
        func = lambda x: x*para
        x_sim = np.arange(min_x, max_x, step)
        xm = np.vstack([np.ones(len(x_sim)), x_sim]).T
        y_sim = func(xm)
        self.data.plot.scatter(x,y)
        plt.plot(x_sim, y_sim)
        plt.title('The Relationship Between {0} and {1}'.format(x, y))
```

```
def visual_ci(self, x, y, step = 0.01):
    para = self.ols(x, y)
    X = self.data[x]
    Y = self.data[y]
    min_x, max_x = min(X), max(X)
    # x is also matrix
    func = lambda x: x*para
    x_sim = np.arange(min_x, max_x, step)
    xm = np.vstack([np.ones(len(x_sim)), x_sim]).T
    y_sim = func(xm)
    y_hat = func(np.vstack([np.ones(len(X)), X]).T)
    # confidence interval
    w = np.sqrt(2*f.cdf(1 - 0.1, 2, 45-2))
    mse = sum((Y.values - np.array(y_hat).reshape(1,-1)[0])**2)/(45-2)
    upper = []
    lower = []
    for each, each_y_hat in zip(X.values, np.array(y_hat).reshape(1,-1)[0]):
        s = np.sqrt(mse*(1/45 + (each - np.mean(X))**2/sum((X - np.mean(X))**2)))
        lower.append(each_y_hat - w*s)
        upper.append(each_y_hat + w*s)
    # visualization
    plt.figure(figsize = (12, 8))
    plt.scatter(X,Y)
    plt.plot(x_sim, y_sim)
    plt.title('The Relationship Between {0} and {1}'.format(x, y))
    plt.plot(X, lower)
    plt.plot(X, upper)
```