Cpts_575: Data Science

Assignment VI

09/19/2019

Instructor: Assefaw Gebremedhin

Student: Yu-Chieh Wang 11641327

1. Auto.csv

a. A scatterplot matrix shows all the variables in the dataset.

```
import pandas as pd
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix
#import numpy as np

data = pd.read_csv("Auto.csv")
data1 = data.drop(columns=['origin', 'name'])

for i in range(len(data1)):
    a = data1['horsepower'][i]
    if a == '?':
        data1 = data1.drop([i])

data1['horsepower'] = pd.to_numeric(data1['horsepower'])

scatter_matrix(data1, alpha=0.7, figsize=(10,10))
#plt.show()

plt.savefig('a.png')
plt.close()
```

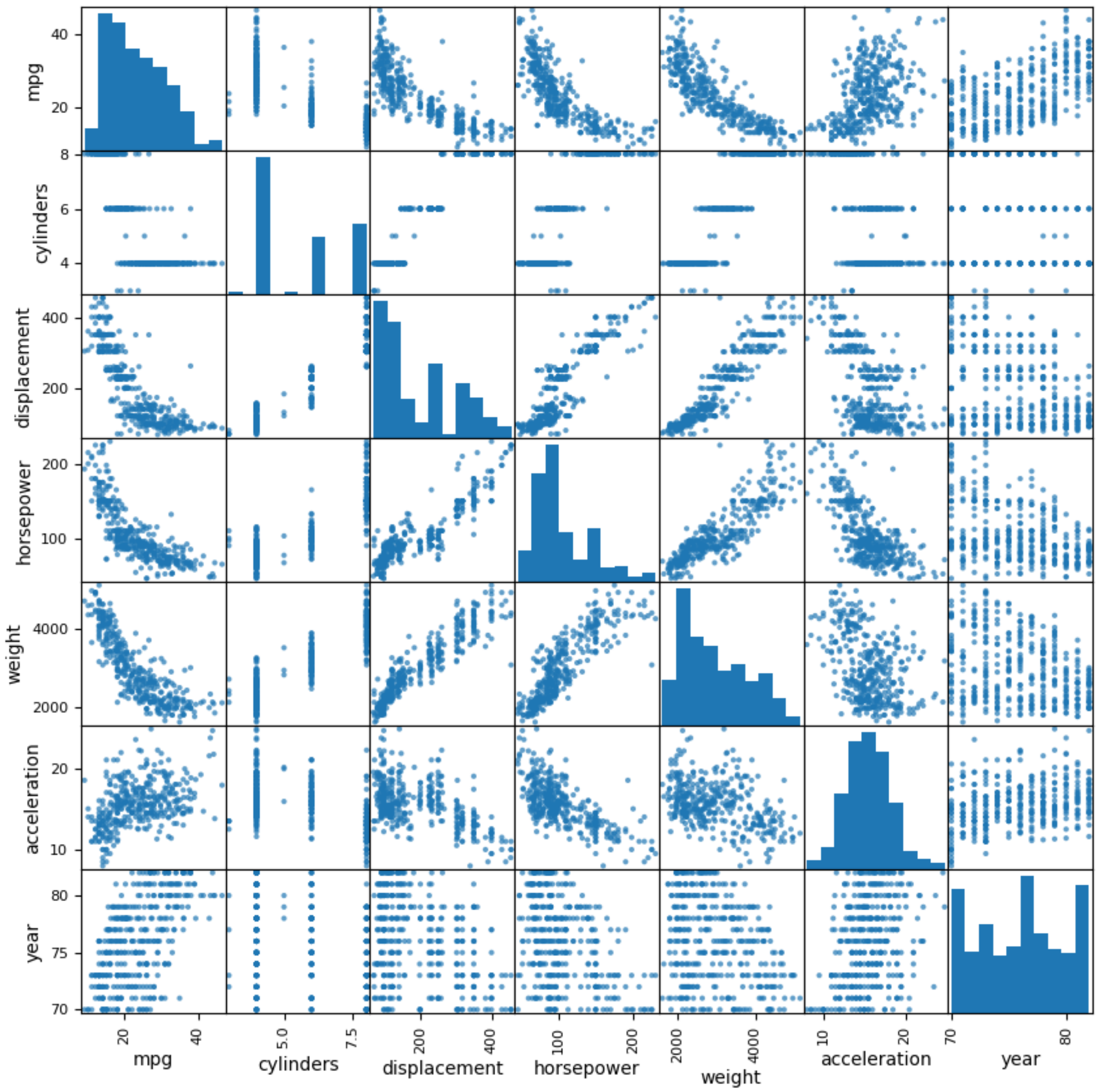
-Delete the rows with question mark.

-Use to_numeric function to change column ‘horsepower’ type from string to int.

-Use scatter_matrix function to make a scatterplot matrix.

-Use savefig function to save the plot.

-The result is as follow:



b. The matrix of correlations between the variables.

```
import pandas as pd

data = pd.read_csv("Auto.csv")
data1 = data.drop(columns=['name'])

for i in range(len(data1)):
    a = data1['horsepower'][i]
    if a == '?':
        data1 = data1.drop([i])

data1['horsepower'] = pd.to_numeric(data1['horsepower'])
print(data1.corr(method='pearson'))
```

-Use corr function to get the correlations.

-The method='pearson' presents standard correlation coefficient.

-The result is as follow:

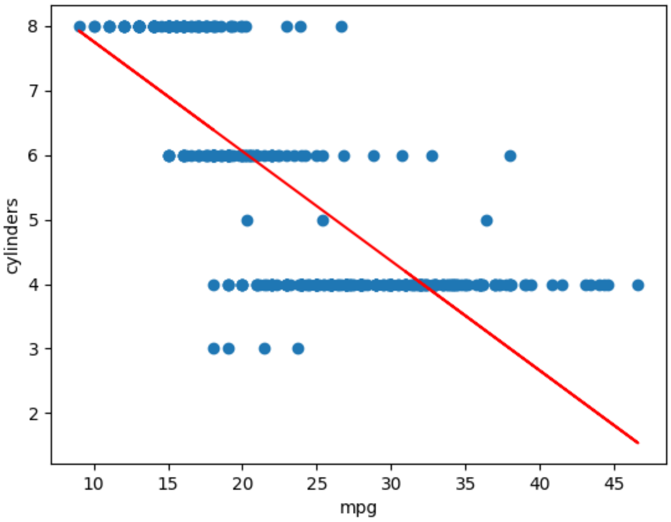
```
                   mpg  cylinders  displacement  horsepower    weight
mpg           1.000000  -0.777618     -0.805127   -0.778427 -0.832244
cylinders    -0.777618   1.000000      0.950823    0.842983  0.897527
displacement -0.805127   0.950823      1.000000    0.897257  0.932994
horsepower   -0.778427   0.842983      0.897257    1.000000  0.864538
weight       -0.832244   0.897527      0.932994    0.864538  1.000000
acceleration  0.423329  -0.504683     -0.543800   -0.689196 -0.416839
year          0.580541  -0.345647     -0.369855   -0.416361 -0.309120
origin        0.565209  -0.568932     -0.614535   -0.455171 -0.585005

              acceleration      year    origin
mpg               0.423329  0.580541  0.565209
cylinders        -0.504683 -0.345647 -0.568932
displacement     -0.543800 -0.369855 -0.614535
horsepower       -0.689196 -0.416361 -0.455171
weight           -0.416839 -0.309120 -0.585005
acceleration      1.000000  0.290316  0.212746
year              0.290316  1.000000  0.181528
origin            0.212746  0.181528  1.000000
```

c. Multiple linear regression

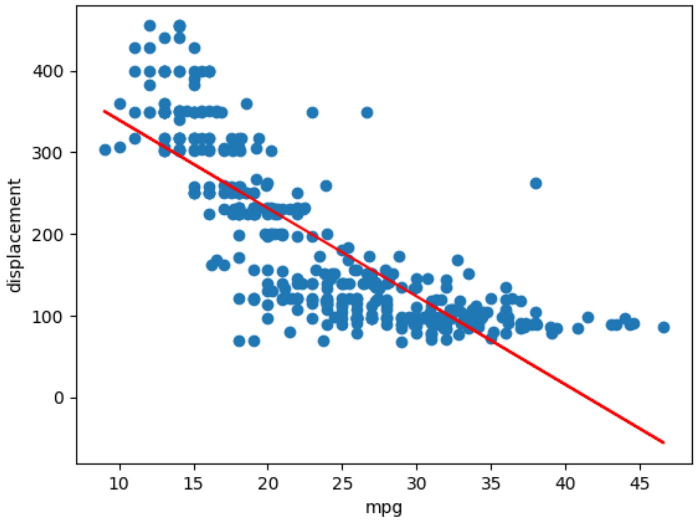
```
X = data1['mpg'].values.reshape(-1, 1)
#Y = data1['cylinders'].values.reshape(-1,1)
Y = data1['origin'].values.reshape(-1,1)
#X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.4, random_state=1)
regr = LinearRegression()
regr.fit(X, Y)
y_pred = regr.predict(X)
print("mpg_origin")
print('Coefficients:', regr.coef_)
print('Mean squared error:', mean_squared_error(Y, y_pred))
print('Variance score:', r2_score(Y, y_pred))
plt.scatter(X, Y)
plt.plot(X, y_pred, color='red')
plt.xlabel('mpg')
plt.ylabel('origin')

#plt.show()
plt.savefig('mpg_origin.png')
```

-Use all data as our training data. (I tried to separate all data to training and testing data, but it will be hard to draw the plots.)

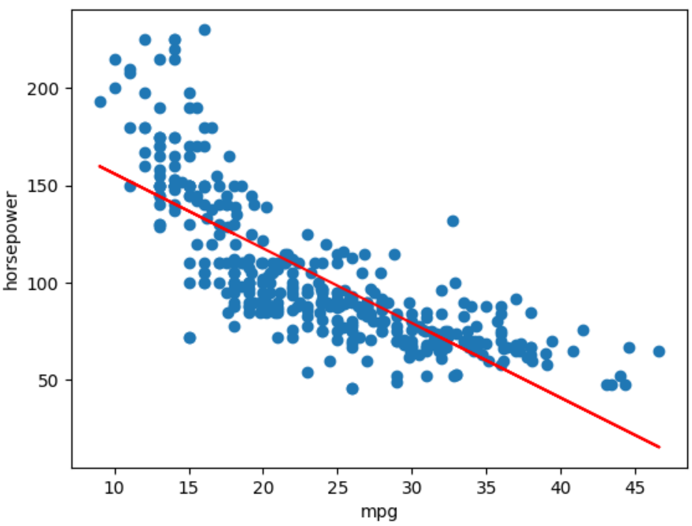
-The result is as follow:



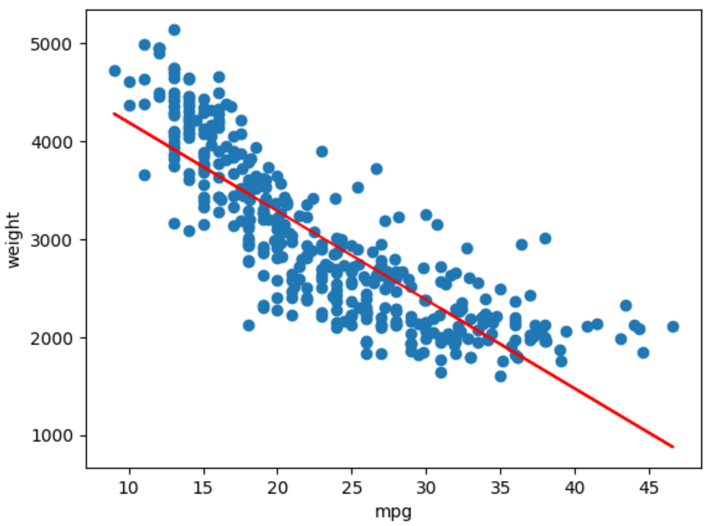

```
mpg_cylinders
Coefficients:  [[-0.16994819]]
Mean squared error: 1.1473007871436645
Variance score: 0.6046889889441245
```



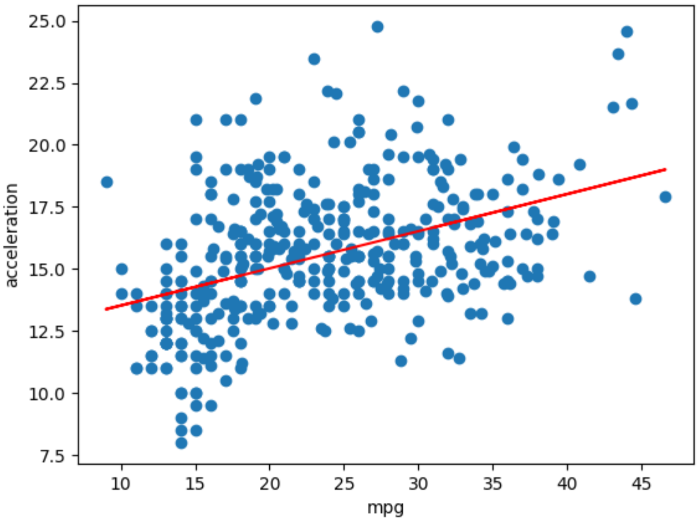

```
mpg_displacement
Coefficients:  [[-10.79457099]]
Mean squared error: 3842.190786324079
Variance score: 0.6482294003193045
```



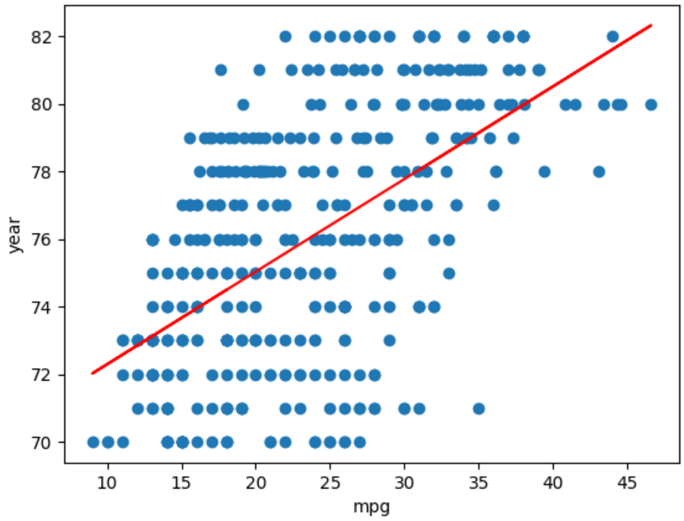

```
mpg_horsepower
Coefficients:  [[-3.83888803]]
Mean squared error: 582.3256763788177
Variance score: 0.6059482578894348
```



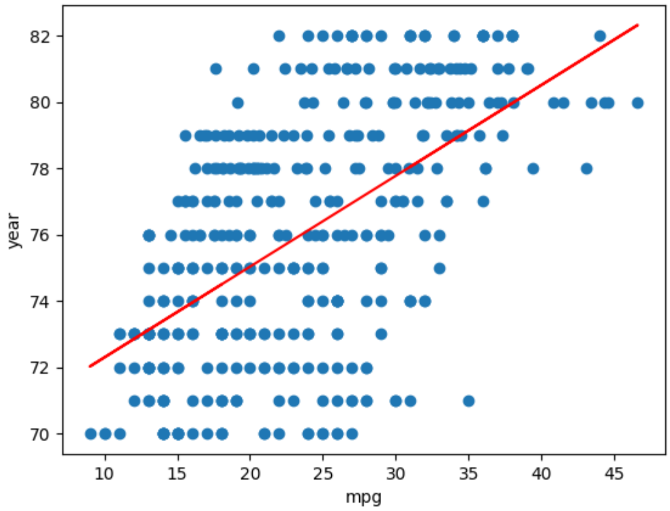

```
mpg_weight
Coefficients:  [[-90.57138867]]
Mean squared error: 221196.72200110494
Variance score: 0.6926304331206254
```




```
mpg_acceleration
Coefficients:  [[0.14963546]]
Mean squared error: 6.231389952284171
Variance score: 0.1792070501562545
```




```
mpg_year
Coefficients:  [[0.27399845]]
Mean squared error: 8.973525975056463
Variance score: 0.3370278133096223
```




```
mpg_origin
Coefficients:  [[0.05833254]]
Mean squared error: 0.4404477999238199
Variance score: 0.3194609386689674
```

-Red line is the result of predictions.

-Lately, I found the answer is not correct because sklearn is for machine learning which is not for statistics, so it doesn't have standard error, t-statistic, and P-value. Therefore, I use other package 'statsmodels' to get the correct answer.

```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm
from scipy import stats

data = pd.read_csv("Auto.csv")
data1 = data.drop(columns=['name'])

for i in range(len(data1)):
    a = data1['horsepower'][i]
    if a == '?':
        data1 = data1.drop([i])

data1['horsepower'] = pd.to_numeric(data1['horsepower'])

X = data1.drop(columns=['mpg'])
Y = data1['mpg']
x2 = sm.add_constant(X)
est = sm.OLS(Y, x2)
est2 = est.fit()
print(est2.summary())
```

-The result is as follow:

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    mpg   R-squared:                       0.821
Model:                            OLS   Adj. R-squared:                  0.818
Method:                 Least Squares   F-statistic:                     252.4
Date:                Wed, 09 Oct 2019   Prob (F-statistic):          2.04e-139
Time:                        15:47:16   Log-Likelihood:                -1023.5
No. Observations:                 392   AIC:                             2063.
Df Residuals:                     384   BIC:                             2095.
Df Model:                           7
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const          -17.2184      4.644     -3.707      0.000     -26.350      -8.087
cylinders       -0.4934      0.323     -1.526      0.128      -1.129       0.142
displacement     0.0199      0.008      2.647      0.008       0.005       0.035
horsepower      -0.0170      0.014     -1.230      0.220      -0.044       0.010
weight          -0.0065      0.001     -9.929      0.000      -0.008      -0.005
acceleration     0.0806      0.099      0.815      0.415      -0.114       0.275
year             0.7508      0.051     14.729      0.000       0.651       0.851
origin           1.4261      0.278      5.127      0.000       0.879       1.973
================================================================================
Omnibus:                       31.906   Durbin-Watson:                   1.309
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               53.100
Skew:                           0.529   Prob(JB):                     2.95e-12
Kurtosis:                       4.460   Cond. No.                     8.59e+04
==============================================================================
```

-i: According to the output result, there is a relationship between t-statistic and P-value because when the value of P>|t| close to 0, the blue dots will be concentrated near the red line.

-ii: The distance between the blue point and the red prediction line.

d. The residual plots does suggest unusually large outliers. However, there is a problem in previous question. According to the plots and the output result, horsepower is related to ‘mpg’ on the prediction plot, but it’s not on the output result. As the result, it is not correct at all.

e. Some of prediction results are meaning for, such as ‘mpg' with ‘displacement’, ‘horsepower’, and ‘weight’.

f. I try to make all variables become X^3 and $log(X)$, but it doesn’t change their relationship. Therefore, I let some predictors which are related to ‘mpg’ become $log(X)$ and the others become X^3. Finally, those predictors which were not related to ‘mpg’ start have relationship with it.

2. Boston.csv

```
Call:
lm(formula = crim ~ nox, data = Boston)

Residuals:
    Min      1Q  Median      3Q     Max 
-12.371  -2.738  -0.974   0.559  81.728 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  -13.720      1.699  -8.073 5.08e-15 ***
nox           31.249      2.999  10.419  < 2e-16 ***
```

```
Call:
lm(formula = crim ~ chas, data = Boston)

Residuals:
   Min     1Q Median     3Q    Max 
-3.738 -3.661 -3.435  0.018 85.232 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   3.7444     0.3961   9.453   <2e-16 ***
chas         -1.8928     1.5061  -1.257    0.209
```

a. Use R studio to represent the relationships between 'crim', 'nox', 'chas', 'medv', and 'dis'.

```
Call:
lm(formula = crim ~ medv, data = Boston)

Residuals:
   Min     1Q Median     3Q    Max 
-9.071 -4.022 -2.343  1.298 80.957 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 11.79654    0.93419   12.63   <2e-16 ***
medv        -0.36316    0.03839   -9.46   <2e-16 ***
```

```
Call:
lm(formula = crim ~ dis, data = Boston)

Residuals:
   Min     1Q Median     3Q    Max 
-6.708 -4.134 -1.527  1.516 81.674 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   9.4993     0.7304  13.006   <2e-16 ***
dis          -1.5509     0.1683  -9.213   <2e-16 ***
```

-The number of starts '*' indicates the relationship between vectors. The more stars between them, the closer they are. According to the result, predictors 'nox', 'medv', 'dis' are related to the predictor 'crim', but the predictor 'chas' has less relationship with 'crim'.

b. Fit a multiple regression model to predict the response.

```
> total<-lm(formula = crim~.,data = Boston)
> summary(total)
Call:
lm(formula = crim ~ ., data = Boston)

Residuals:
   Min     1Q Median     3Q    Max 
-9.924 -2.120 -0.353  1.019 75.051 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  17.033228   7.234903   2.354 0.018949 *  
zn            0.044855   0.018734   2.394 0.017025 *  
indus        -0.063855   0.083407  -0.766 0.444294    
chas         -0.749134   1.180147  -0.635 0.525867    
nox         -10.313535   5.275536  -1.955 0.051152 .  
rm            0.430131   0.612830   0.702 0.483089    
age           0.001452   0.017925   0.081 0.935488    
dis          -0.987176   0.281817  -3.503 0.000502 ***
rad           0.588209   0.088049   6.680 6.46e-11 ***
tax          -0.003780   0.005156  -0.733 0.463793    
ptratio      -0.271081   0.186450  -1.454 0.146611    
black        -0.007538   0.003673  -2.052 0.040702 *  
lstat         0.126211   0.075725   1.667 0.096208 .  
medv         -0.198887   0.060516  -3.287 0.001087 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

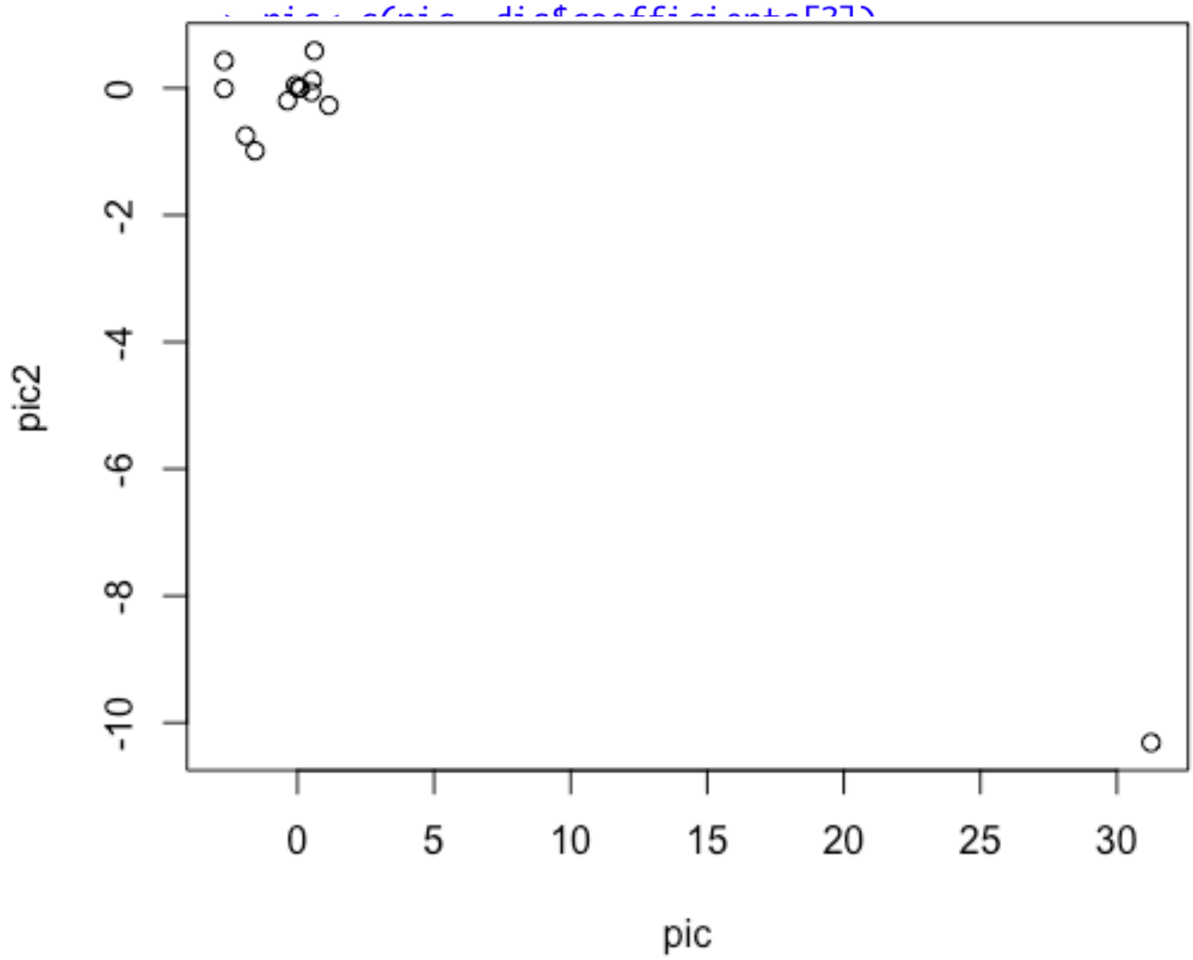
-According to the result, we can reject predictor ‘zn’, ‘indus’, ‘dis’, ‘rad’, ‘black’, and ‘medv’ because they are related to the predictor ‘crim’.

c. Use a plot to show the various predictors.

```
> zn<-lm(formula = crim~zn, data=Boston)
> indus<-lm(formula = crim~indus, data=Boston)
> chas<-lm(formula = crim~chas, data=Boston)
> nox<-lm(formula = crim~nox, data=Boston)
> rm<-lm(formula = crim~rm, data=Boston)
> age<-lm(formula = crim~age, data=Boston)
> dis<-lm(formula = crim~dis, data=Boston)
> rad<-lm(formula = crim~rad, data=Boston)
> tax<-lm(formula = crim~tax, data=Boston)
> ptratio<-lm(formula = crim~ptratio, data=Boston)
> lstat<-lm(formula = crim~lstat, data=Boston)
> medv<-lm(formula = crim~medv, data=Boston)
> summary(nox)
```

-The result is as follow:

```
> pic<-vector("numeric",0)
> pic<-c(pic, zn$coefficients[2])
> pic<-c(pic, indus$coefficients[2])
> pic<-c(pic, chas$coefficients[2])
> pic<-c(pic, nox$coefficients[2])
> pic<-c(pic, rm$coefficients[2])
> pic<-c(pic, age$coefficients[2])
```



-According to the plot, we can observe that most of predictors' coefficients change, so we get that the result considering only one predictor and considering all predictors are different.

d. $Y = B_0 + B_1X + B_2X^2 + B_3X^3 + \varepsilon$

-The instructions are as follows:

Dzn = lm(crim~poly(zn, 3), data = Boston)

Dindus = lm(crim~poly(indus, 3), data = Boston)

```
Dchas = lm(crim~poly(chas, 1), data = Boston)

Dnox = lm(crim~poly(nox, 3), data = Boston)

Drm = lm(crim~poly(rm, 3), data = Boston)

Dage = lm(crim~poly(age, 3), data = Boston)

Ddis = lm(crim~poly(dis, 3), data = Boston)

Drad = lm(crim~poly(rad, 3), data = Boston)

Dtax = lm(crim~poly(tax, 3), data = Boston)

Dptratio = lm(crim~poly(ptratio, 3), data = Boston)

Dblack = lm(crim~poly(black, 3), data = Boston)

Dlstat = lm(crim~poly(lstat, 3), data = Boston)

Dmedv = lm(crim~poly(medv, 3), data = Boston)
```

-The output results show that:

Predictor ‘zn’ supports for X and X^2.

Predictor ‘indus’ supports for X.

Predictor ‘chas’ doesn’t support anything.

Predictor ‘nox’ supports for X, X^2, and X^3.

Predictor ‘rm’ supports for X and X^2.

Predictor ‘age’ supports for X, X^2, and X^3.

Predictor ‘dis’ supports for X, X^2, and X^3.

Predictor ‘rad’ supports for X and X^2.

Predictor ‘tax’ supports for X and X^2.

Predictor ‘ptratio’ supports for X, X^2, and X^3.

Predictor ‘black’ supports for X.

Predictor ‘lstat’ supports for X and X^2.

Predictor 'medv' supports for X, X^2, and X^3.

3. The error term in linear regression

a. If the error terms are correlated, the regression coefficients may be affected because the error terms and predictors are related. Then, after the coefficient change, it also make the standard error change. Finally, the confidence intervals are related to a function which contains predictors so that it will be affected, too.

b. Ridge Regression can help us solve the problem.