HW7 Report

인대영 (2022311154)
2022-04-28

1. (a) After saving 'amit.dat' in the dataframe 'amit', we define a function 'mv_reg_anal' which performs multivariate regression analysis and returns the matrices Y, Z, and $\hat{\beta}$ where

$$Y_{n\times m} = Z_{n\times(r+1)}\beta_{(r+1)\times m} + \varepsilon_{n\times m}$$

$$E(\underset{\sim}{\varepsilon}_{(i)}) = \underset{\sim}{0}, \quad Cov(\underset{\sim}{\varepsilon}_{(i)}, \underset{\sim}{\varepsilon}_{(k)}) = \sigma_{ik} I \quad (i, k = 1, 2, \cdots, m)$$

$$\hat{\beta}_{(r+1)\times m} = (Z'Z)^{-1}Z'Y$$

In [3]:
```python
#(a) Perform multivariate regression analysis. Analyze residuals to check multivariate normal assumption
def mv_reg_anal(data, m, r): #m: number of response var, r: number of predictor var
    y_colnames=data.columns[0:m]
    z_colnames=data.columns[m:(m+r)]
    Y=data[y_colnames]
    Z=data[z_colnames]
    ones=pd.DataFrame(np.ones(len(data),dtype=np.int8))
    ones.columns=['Z0']
    ones
    Z=pd.concat([ones,Z],axis=1)

    B_hat=pd.DataFrame((np.linalg.inv(Z.T.dot(Z))).dot(Z.T).dot(Y))

    return [Y, Z, B_hat]
```

In [4]:
```python
#B_hat matrix
#B0: intercept
#B1~B5: slope for Z1~Z5
Y, Z, B_hat=mv_reg_anal(amit, 2, 5)
B_hat.index = ['Z0','Z1','Z2','Z3','Z4','Z5']
B_hat.columns = ['Y1','Y2']
B_hat
```

Out[4]:

	Y1	Y2
Z0	-2879.478246	-2728.708544
Z1	675.650781	763.029762
Z2	0.284851	0.306373
Z3	10.272133	8.896198
Z4	7.251171	7.205560
Z5	7.598240	4.987051

After applying 'mv_reg_anal' on 'amit' data, we obtain $\hat{\beta}$ which is shown in the output of cell #4. The first row corresponds to the intercepts of the regression line for Y1 and Y2 respectively. The next five rows correspond to the slopes for Z1 ~ Z5 predictor variables.

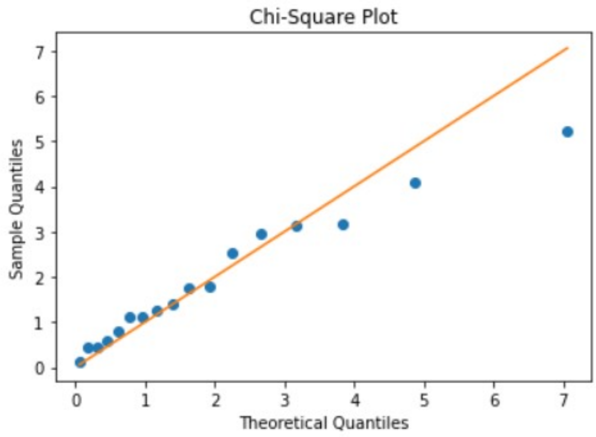
We define functions 'mean_vector', 'covariance_matrix', and 'chisq_plot' which we have used in previous homeworks. The residuals can be obtained as $\varepsilon = Y - Z\hat{\beta}$ and we save the residuals dataframe as 'eps'. By applying the chi-square plot function to the residuals, we can check the multivariate normal assumption. If the residuals are multivariate normal, the points in the chi-square plot should be close to the reference line.

In [8]:
```python
#residuals
eps=Y-Z.dot(B_hat)
eps
```

Out[8]:

	Y1	Y2
0	132.821721	161.527686
1	-72.003916	-264.353294
2	-399.247694	-373.852438
3	-382.847295	-247.294565
4	-152.391292	15.787769
5	366.786445	217.132056
6	4.499942	-83.742102
7	294.556802	462.724007
8	101.840674	223.035763
9	-180.052350	-251.053639
10	-182.639086	-103.054644

In [9]:
```python
#cqplot
chisq_plot(eps)
```



Observe that the points of the cqplot are close to the reference line. Thus, we can confirm the multivariate normal assumption.

(b) Assuming fixed values $\underset{\sim}{z}_{0\,(r+1)\times 1}$ of the predictor variables,

$$\hat{\beta}_{m\times(r+1)}\, \underset{\sim}{z}_0 \sim N_m\left(\beta' \underset{\sim}{z}_0,\ \underset{\sim}{z}_0'(Z'Z)^{-1}\underset{\sim}{z}_0 \Sigma\right)$$

We can obtain 95% simultaneous confidence interval for $E(Y1) = \underset{\sim}{z}_0' \underset{\sim}{\beta}_{(1)}$ as

$$\underset{\sim}{z}_0' \underset{\sim}{\beta}_{(1)} \pm \sqrt{\frac{m(n-r-1)}{n-r-m} F_{m,\,n-r-m}(0.05)} \sqrt{\underset{\sim}{z}_0'(Z'Z)^{-1}\underset{\sim}{z}_0 \left(\frac{n}{n-r-1}\hat{\sigma}_{11}\right)}$$

where $\underset{\sim}{\beta}_{(1)}$ is the first column of $\hat{\beta}$ and $\hat{\sigma}_{11}$ is the first diagonal element of $\hat{\Sigma}$.

We want to estimate $E(Y1)$ at $z1=1$, $z2=1200$, $z3=140$, $z4=70$, $z5=85$.

So we define $\underset{\sim}{z}_0 = [1, 1, 1200, 140, 70, 85]'$ which is denoted as 'Z_fixed' in the code. We calculate the formula for the 95% simultaneous confidence interval in a function we define as 'simul_conf_int'.

In [11]:
```python
def simul_conf_int(data, m, r, fixed): #m: number of response var, r: number of predictor var
    Y, Z, B_hat=mv_reg_anal(data, m, r)
    n=len(data)

    #calculate term1
    B_hat.index = ['Z0','Z1','Z2','Z3','Z4','Z5']
    B_hat.columns = ['Y1','Y2']
    fixed.index = ['Z0','Z1','Z2','Z3','Z4','Z5']
    term1 = fixed.T.dot(B_hat).iloc[0,0]

    #calculate term2
    eps=Y-Z.dot(B_hat)
    sig_hat = 1/n*eps.T.dot(eps)
    fixed.reset_index(inplace = True, drop = True)
    B_hat.reset_index(inplace = True, drop = True)
    B_hat.index = ['Z0','Z1','Z2','Z3','Z4','Z5']
    term2 = np.sqrt(m*(n-r-1)/(n-r-m)*f.isf(0.05,m,n-r-m))*np.sqrt(fixed.T.dot(np.linalg.inv(Z.T.dot(Z))).dot(fixed).iloc[0,0]*(r

    return [term1-term2, term1+term2]
```

In [12]:
```python
simul_conf_int(amit,2,5,Z_fixed)
```

Out[12]:
```
[319.0202417848874, 1140.0293024396533]
```

We obtain a confidence interval of [319.020, 1140.029].

(c) The 95% simultaneous confidence interval for individual Y2 can be obtained with a code similar to that of #1(b). However, the formula is slightly different.

$$\underset{\sim}{z}_0'\underset{\sim}{\beta}_{(2)} \pm \sqrt{\frac{m(n-r-1)}{n-r-m}F_{m,n-r-m}(0.05)}\sqrt{1+\underset{\sim}{z}_0'(Z'Z)^{-1}\underset{\sim}{z}_0\left(\frac{n}{n-r-1}\hat{\sigma}_{11}\right)}$$

The function corresponding to this formula is defined as 'pred_int' as shown below.

In [13]:
```
#(c) 95% simultaneous confidence interval for individual Y2
def pred_int(data, m, r, fixed): #m: number of response var, r: number of predictor var
    Y, Z, B_hat=mv_reg_anal(data, m, r)
    n=len(data)

    #calculate term1
    B_hat.index = ['Z0','Z1','Z2','Z3','Z4','Z5']
    B_hat.columns = ['Y1','Y2']
    fixed.index = ['Z0','Z1','Z2','Z3','Z4','Z5']
    term1 = fixed.T.dot(B_hat).iloc[0,1]

    #calculate term2
    eps=Y-Z.dot(B_hat)
    sig_hat = 1/n*eps.T.dot(eps)
    fixed.reset_index(inplace = True, drop = True)
    B_hat.reset_index(inplace = True, drop = True)
    B_hat.index = ['Z0','Z1','Z2','Z3','Z4','Z5']
    term2 = np.sqrt(m*(n-r-1)/(n-r-m)*f.isf(0.05,m,n-r-m))*np.sqrt((1+fixed.T.dot(np.linalg.inv(Z.T.dot(Z))).dot(fixed).iloc[0,0]

    return [term1-term2, term1+term2]
```

In [14]:
```
pred_int(amit,2,5,Z_fixed)
```

Out[14]:
```
[-401.0652893243076, 1552.5162238632938]
```

We obtain a confidence interval of [-401.065, 1552.516].

(d) Using the formula $\hat{\Sigma} = \frac{1}{n}\hat{\varepsilon}'\hat{\varepsilon} = \frac{1}{n}(Y-Z\hat{B})'(Y-Z\hat{B})$, we obtain $\hat{\Sigma}$ which is denoted as 'sig_hat' in the code below.

In [15]:
```
#(d)
n=17
m=2
r=5

sig_hat = 1/n*eps.T.dot(eps)
sig_hat
```

Out[15]:

	Y1	Y2
Y1	51176.959440	45039.792706
Y2	45039.792706	55335.817611

(e) To calculate Wilk's Lambda value, we need to find the following matrices:

$$E = n\hat{\Sigma} = (Y-Z\hat{B})'(Y-Z\hat{B}) \qquad H = n(\hat{\Sigma}_1 - \hat{\Sigma})$$

$$E_1 = n\hat{\Sigma}_1 = (Y-Z\hat{B}_{(1)})'(Y-Z\hat{B}_{(1)})$$

We also need the nonzero eigenvalues λ_1, λ_2 of HE^{-1}

This is under the null hypothesis

$$H_0: \beta_{(2)} = 0 \quad \text{where} \quad \beta = \begin{bmatrix} \beta_{(1)}\ (q+1)\times m \\ \cdots\cdots \\ \beta_{(2)}\ (r-q)\times m \end{bmatrix}$$

Then Wilk's Lambda $= \prod_{i=1}^{2} \frac{1}{1+\lambda_i}$ can be used to test the hypothesis.

The necessary calculations are shown in the code below. $\Lambda^* = 0.0050818$

In [16]:
```
#(e) Calculate Wilk's lambda value
Z.columns=range(0,6)
E=n*sig_hat
E.index=range(0,2)
E.columns=range(0,2)

#select columns for (Z0, Z1, Z2)
Z_r = Z.iloc[:,:3]
Z_r.columns = ['Z0','Z1','Z2']

#select rows for (Z0, Z1, Z2)
B_hat_r = B_hat.iloc[:3,:]

eps_r=Y-Z_r.dot(B_hat_r)
sig_hat_r = 1/n*eps_r.T.dot(eps_r)

H = n*(sig_hat_r - sig_hat)
HE_inv = H.dot(np.linalg.inv(E))
HE_inv
```

Out[16]:

	0	1
Y1	191.408443	-27.709528
Y2	158.571261	-22.813777

In [17]:
```
eigval, eigvec = np.linalg.eig(HE_inv)
eigval
```

Out[17]: `array([1.68433265e+02, 1.61401095e-01])`

In [18]:
```
val=1
for i in range(2):
    val=val*1/(1+eigval[i])

wilk_ld = val
wilk_ld
```

Out[18]: `0.005081817790983487`

(f) Under H_0, $-\left[n-r-1-\frac{1}{2}(m-r+q+1)\right]\ln(\Lambda^*) \sim \chi^2_{m(r-q)}$

The test statistic is computed as 60.7439. Since this value is greater than $\chi^2_{m(r-q)}(0.05) = 15.5073$, we reject the null hypothesis. Thus, we can say that $(Y1, Y2)$ are dependent on $(Z3, Z4, Z5)$ at 0.05 significance level.

In [19]:
```
#(f) Test hypothesis
q=1
test_stat = -(n-r-1-(m-r+q+1)/2)*np.log(wilk_ld)
test_stat
```

Out[19]: `60.74399185807079`

In [20]:
```
chi2.isf(0.05,m*(r-q))
#test_stat = 60.7439 > 15.5073
#reject null hypothesis
```

Out[20]: `15.507313055865454`