Chapter 6

Multiple Linear Regression (solutions to exercises)

Contents

Import Python packages

```
# Import all needed python packages
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import scipy.stats as stats
import statsmodels.formula.api as smf
import statsmodels.api as sm
```

6.1 Nitrate concentration

|||| Exercise 6.1 Nitrate concentration

In order to analyze the effect of reducing nitrate loading in a Danish fjord, it was decided to formulate a linear model that describes the nitrate concentration in the fjord as a function of nitrate loading, it was further decided to correct for fresh water runoff. The resulting model was

$$Y_i = \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i} + \varepsilon_i, \quad \varepsilon_i \sim N(0, \sigma^2), \tag{6-1}$$

where Y_i is the natural logarithm of nitrate concentration, $x_{1,i}$ is the natural logarithm of nitrate loading, and $x_{2,i}$ is the natural logarithm of fresh water run off.

a) Which of the following statements are assumed fulfilled in the usual multiple linear regression model?

1) $\varepsilon_i = 0$ for all $i = 1, ..., n$, and β_j follows a normal distribution

2) $E[x_1] = E[x_2] = 0$ and $V[\varepsilon_i] = \beta_1^2$

3) $E[\varepsilon_i] = 0$ and $V[\varepsilon_i] = \beta_1^2$

4) ε_i is normally distributed with constant variance, and ε_i and ε_j are independent for $i \neq j$

5) $\varepsilon_i = 0$ for all $i = 1, ..., n$, and x_j follows a normal distribution for $j = \{1, 2\}$

|||| Solution

1) ε_i follows a normal distribution with expectation equal zero, but the realizations are not zero, and further β_j is deterministic and hence it does not follow a distribution ($\hat{\beta}_j$ does), hence 1) is not correct

2)- 3) There are no assumptions on the expectation of x_j and the variance of ε equal σ^2, not β_1^2 hence 2) and 3) are not correct

4) Is correct, this is the usual assumption about the errors

5) Is incorrect since ε_j follow a normal distribution, further the are no distributional assumptions on x_j. In fact we assume that x_j is known

The parameters in the model were estimated in Python and the following results are available (slightly modified output from summary):

```
                         OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.3438
Model:                            OLS   Adj. R-squared:                  0.3382
No. Observations:                 240   F-statistic:                      62.07
Covariance Type:            nonrobust   Prob (F-statistic):             2.2e-16
==============================================================================
             coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    -2.36500      0.222    -10.661     <2e-16           *           *
x1             0.4762      0.062      7.720   3.25e-13           *           *
x2             0.0827      0.070      1.185      0.273           *           *
==============================================================================
```

b) What are the parameter estimates for the model parameters ($\hat{\beta}_i$ and $\hat{\sigma}^2_{\beta_i}$) and how many degrees of freedom are there in the estimation?

|||| Solution

The number of degrees of freedom is equal $n - (p+1)$, and since the number of observations is $n = 240$ and $p = 2$, we get $df = 240 - (2+1) = 237$. The parameters are given in the first column of the the first column of the summary table, i.e.

$$\hat{\beta}_0 = -2.365 \tag{6-2}$$
$$\hat{\beta}_1 = 0.476 \tag{6-3}$$
$$\hat{\beta}_2 = 0.083 \tag{6-4}$$

and the estimated standard errors for the parameters are given in the second column:

$$\hat{\sigma}_{\beta_0} = 0.222 \tag{6-5}$$
$$\hat{\sigma}_{\beta_1} = 0.062 \tag{6-6}$$
$$\hat{\sigma}_{\beta_2} = 0.070 \tag{6-7}$$

c) Calculate the usual 95% confidence intervals for the parameters (β_0, β_1, and β_2).

|||| Solution

From Theorem 6.5 we know that the confidence intervals can be calculated by

$$\hat{\beta}_i \pm t_{1-\alpha/2}\, \hat{\sigma}_{\beta_i},$$

where $t_{1-\alpha/2}$ is based on 237 degrees of freedom, and with $\alpha = 0.05$, we get $t_{0.975} = 1.97$. The standard errors for the estimates is the second column of the summary, and the confidence intervals become

$$\hat{\beta}_0 = \quad -2.365 \pm 1.97 \cdot 0.222 \tag{6-8}$$

$$\hat{\beta}_1 = \quad 0.467 \pm 1.97 \cdot 0.062 \tag{6-9}$$

$$\hat{\beta}_2 = \quad 0.083 \pm 1.97 \cdot 0.070 \tag{6-10}$$

d) On level $\alpha = 0.05$ which of the parameters are significantly different from 0, also find the p-values for the tests used for each of the parameters?

|||| Solution

We can see directly from the confidence intervals above that β_0 and β_1 are significantly different from zero (the confidence intervals does not cover zero), while we cannot reject that $\beta_2 = 0$ (the confidence interval cover zero). The p-values we can see directly in the Python output: for β_0 is less than 10^{-16} and the p-value for β_1 is $3.25 \cdot 10^{-13}$, i.e. very strong evidence against the null hypothesis in both cases.

6.2 Multiple linear regression model

Exercise 6.2 Multiple linear regression model

The following measurements have been obtained in a study:

No.	1	2	3	4	5	6	7	8	9	10	11	12	13
y	1.45	1.93	0.81	0.61	1.55	0.95	0.45	1.14	0.74	0.98	1.41	0.81	0.89
x_1	0.58	0.86	0.29	0.20	0.56	0.28	0.08	0.41	0.22	0.35	0.59	0.22	0.26
x_2	0.71	0.13	0.79	0.20	0.56	0.92	0.01	0.60	0.70	0.73	0.13	0.96	0.27

No.	14	15	16	17	18	19	20	21	22	23	24	25
y	0.68	1.39	1.53	0.91	1.49	1.38	1.73	1.11	1.68	0.66	0.69	1.98
x_1	0.12	0.65	0.70	0.30	0.70	0.39	0.72	0.45	0.81	0.04	0.20	0.95
x_2	0.21	0.88	0.30	0.15	0.09	0.17	0.25	0.30	0.32	0.82	0.98	0.00

It is expected that the response variable y can be described by the independent variables x_1 and x_2. This imply that the parameters of the following model should be estimated and tested

$$Y_i = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \varepsilon_i, \quad \varepsilon_i \sim N(0, \sigma^2).$$

a) Calculate the parameter estimates ($\hat{\beta}_0$, $\hat{\beta}_1$, $\hat{\beta}_2$, and $\hat{\sigma}^2$), in addition find the usual 95% confidence intervals for β_0, β_1, and β_2.
You can copy the following lines to Python to load the data:

```
df = pd.DataFrame({
    'x1': [0.58, 0.86, 0.29, 0.20, 0.56, 0.28, 0.08, 0.41, 0.22,
        0.35, 0.59, 0.22, 0.26, 0.12, 0.65, 0.70, 0.30, 0.70,
        0.39, 0.72, 0.45, 0.81, 0.04, 0.20, 0.95],
    'x2': [0.71, 0.13, 0.79, 0.20, 0.56, 0.92, 0.01, 0.60, 0.70,
        0.73, 0.13, 0.96, 0.27, 0.21, 0.88, 0.30, 0.15, 0.09,
        0.17, 0.25, 0.30, 0.32, 0.82, 0.98, 0.00],
    'y': [1.45, 1.93, 0.81, 0.61, 1.55, 0.95, 0.45, 1.14, 0.74,
        0.98, 1.41, 0.81, 0.89, 0.68, 1.39, 1.53, 0.91, 1.49,
        1.38, 1.73, 1.11, 1.68, 0.66, 0.69, 1.98]
})
```

|||| Solution

The question is answered by Python. Start by loading data into and and estimating the parameters in Python:

```
fit = smf.ols('y ~ x1 + x2', data=df).fit()
print(fit.summary(slim=True))

                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.940
Model:                            OLS   Adj. R-squared:                  0.934
No. Observations:                  25   F-statistic:                     172.0
Covariance Type:            nonrobust   Prob (F-statistic):           3.70e-14
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.4335      0.066      6.571      0.000       0.297       0.570
x1             1.6530      0.095     17.355      0.000       1.455       1.851
x2             0.0039      0.075      0.053      0.958      -0.151       0.159
==============================================================================



# Residual standard deviation (error standard deviation estimate)
sigma = np.sqrt(fit.mse_resid) # fit.mse_resid or fit.scale
print(sigma)

0.11271746832914663
```

|||| Solution

The parameter estimates are given in the first column, i.e.

$$\hat{\beta}_0 = 0.434,$$
$$\hat{\beta}_1 = 1.653,$$
$$\hat{\beta}_2 = 0.0039,$$

and the error variance estimate is $\hat{\sigma}^2 = 0.11^2$.The confidence intervals can be seen in the last two columns or calculated again in Python:

```
print(fit.conf_int(alpha=0.05))

                  0         1
Intercept  0.296707  0.570387
x1         1.455467  1.850520
x2        -0.151292  0.159182
```

b) Still using confidence level $\alpha = 0.05$ reduce the model if appropriate.

Solution

Since the confidence interval for β_2 cover zero (and the p-value is much larger than 0.05), the parameter should be removed from the model to get the simpler model

$$y_i = \beta_0 + \beta_1 x_1 + \varepsilon_i, \quad \varepsilon_i \sim N(0, \sigma^2),$$

the parameter estimates in the simpler model are

```
fit = smf.ols('y ~ x1', data=df).fit()
print(fit.summary(slim=True))

                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.940
Model:                            OLS   Adj. R-squared:                  0.937
No. Observations:                  25   F-statistic:                     359.6
Covariance Type:            nonrobust   Prob (F-statistic):           1.54e-15
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.4361      0.044      9.913      0.000       0.345       0.527
x1             1.6512      0.087     18.963      0.000       1.471       1.831
==============================================================================
```

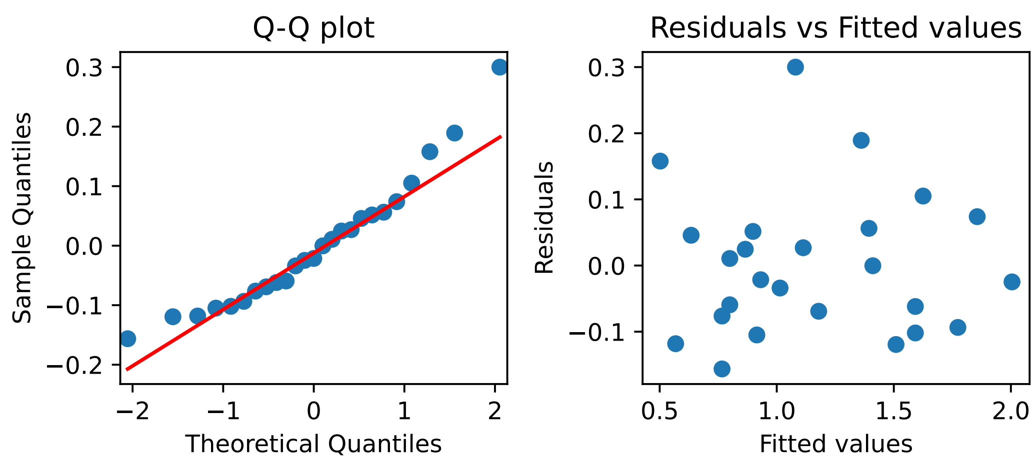
and both parameters are now significant.

c) Carry out a residual analysis to check that the model assumptions are fulfilled.

|||| Solution

We are interested in inspecting a q-q plot of the residuals and a plot of the residuals as a function of the fitted values

```
# Predicted values for the data
ypred = fit.predict(df)
# Prepare plot
fig, ax = plt.subplots(1,2)
# Q-Q plot
sm.qqplot(df['y']-ypred, line="q",a=1/2, ax=ax[0])
ax[0].set_title("Q-Q plot")
# Scatter plot of residuals vs. fitted values
ax[1].scatter(fit.fittedvalues, df['y']-ypred)
ax[1].set_xlabel("Fitted values")
ax[1].set_ylabel("Residuals")
ax[1].set_title("Residuals vs Fitted values")
plt.tight_layout()
plt.show()
```



there are no strong evidence against the assumptions, the qq-plot is are a straight line and the are no obvious dependence between the residuals and the fitted values, and we conclude that the assumptions are fulfilled.

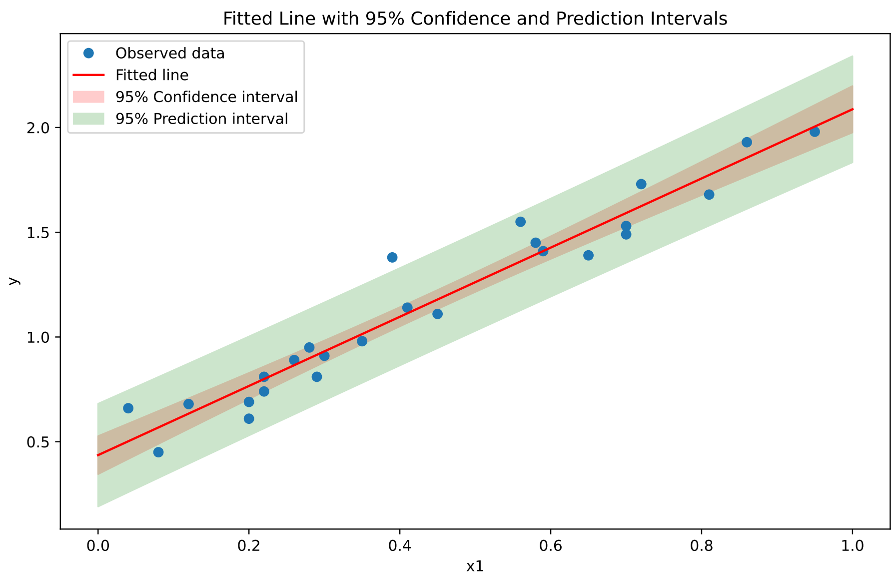
d) Make a plot of the fitted line and 95% confidence and prediction intervals of the line for $x_1 \in [0,1]$ (it is assumed that the model was reduced above).

Solution

```
x1_new = pd.DataFrame({'x1': np.linspace(0, 1, 100)})

prediction_summary = fit.get_prediction(x1_new).summary_frame(alpha=0.05)

plt.figure(figsize=(10, 6))
plt.plot(df['x1'], df['y'], 'o', label='Observed data')
plt.plot(x1_new, prediction_summary['mean'], 'r-', label='Fitted line')
plt.fill_between(x1_new['x1'],
                 prediction_summary['mean_ci_lower'],
                 prediction_summary['mean_ci_upper'],
                 color='red', alpha=0.2, label='95% Confidence
interval')
plt.fill_between(x1_new['x1'],
                 prediction_summary['obs_ci_lower'],
                 prediction_summary['obs_ci_upper'],
                 color='green', alpha=0.2, label='95% Prediction
interval')
plt.xlabel('x1')
plt.ylabel('y')
plt.legend()
plt.title('Fitted Line with 95% Confidence and Prediction Intervals')
plt.show()
```



6.3 MLR simulation exercise

Exercise 6.3 MLR simulation exercise

The following measurements have been obtained in a study:

Nr.	1	2	3	4	5	6	7	8
y	9.29	12.67	12.42	0.38	20.77	9.52	2.38	7.46
x_1	1.00	2.00	3.00	4.00	5.00	6.00	7.00	8.00
x_2	4.00	12.00	16.00	8.00	32.00	24.00	20.00	28.00

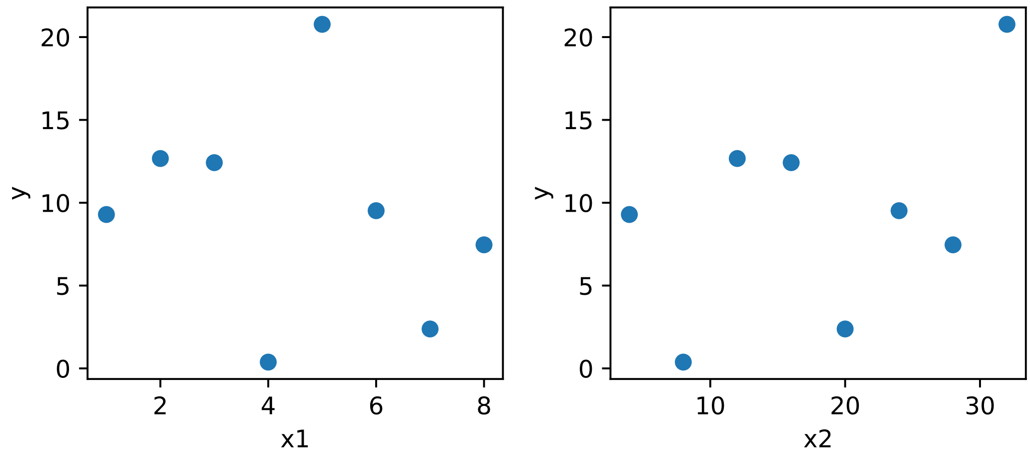
a) Plot the observed values of y as a function of x_1 and x_2. Does it seem reasonable that either x_1 or x_2 can describe the variation in y?
You may copy the following lines into Python to load the data

```
df = pd.DataFrame({
    'y': [9.29,12.67,12.42,0.38,20.77,9.52,2.38,7.46],
    'x1': [1.00,2.00,3.00,4.00,5.00,6.00,7.00,8.00],
    'x2': [4.00,12.00,16.00,8.00,32.00,24.00,20.00,28.00]
})
```

Solution

The data is plotted with

```
fig, ax = plt.subplots(1, 2)
ax[0].scatter(df['x1'], df['y'])
ax[0].set_xlabel('x1')
ax[0].set_ylabel('y')
ax[1].scatter(df['x2'], df['y'])
ax[1].set_xlabel('x2')
ax[1].set_ylabel('y')
plt.tight_layout()
plt.show()
```



There does not seem to be a strong relation between y and x_1 or x_2.

b) Estimate the parameters for the two models

$$Y_i = \beta_0 + \beta_1 x_{1,i} + \varepsilon_i, \quad \varepsilon_i \sim N(0, \sigma^2),$$

and

$$Y_i = \beta_0 + \beta_1 x_{2,i} + \varepsilon_i, \quad \varepsilon_i \sim N(0, \sigma^2),$$

and report the 95% confidence intervals for the parameters. Are any of the parameters significantly different from zero on a 5% confidence level?

|||| Solution

The models are fitted with

```
fit1 = smf.ols('y ~ x1', data=df).fit()
fit2 = smf.ols('y ~ x2', data=df).fit()
print(fit1.conf_int())

                  0          1
Intercept -0.542637  24.897637
x1        -3.144796   1.893129

print(fit2.conf_int())

                  0          1
Intercept -7.558092  15.965949
x2        -0.295789   0.868825
```

since all confidence intervals cover zero we cannot reject that the parameters are in fact zero, and we would conclude neither x_1 nor x_2 explain the variations in y.

c) Estimate the parameters for the model

$$Y_i = \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i} + \varepsilon_i, \quad \varepsilon_i \sim (N(0, \sigma^2), \tag{6-11}$$

and go through the steps of Method 6.16 (use confidence level 0.05 in all tests).

Solution

The model is fitted with

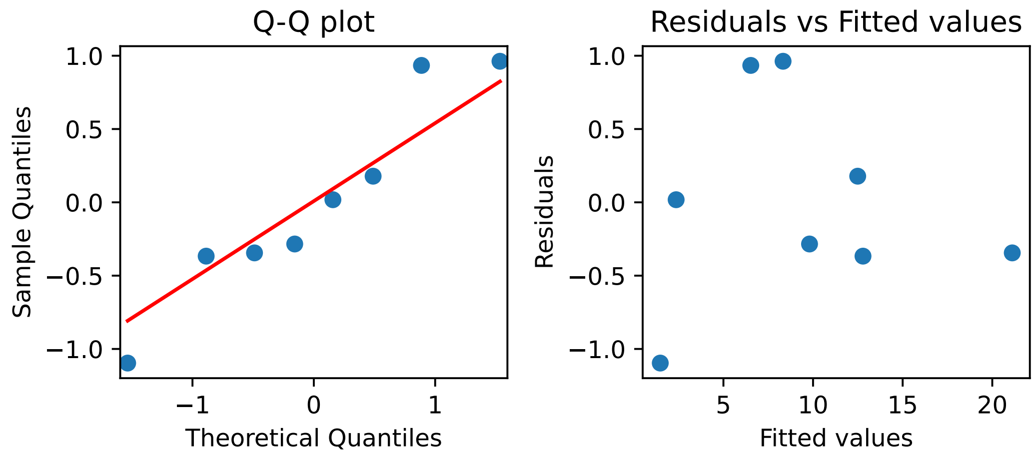
```
fit = smf.ols('y ~ x1 + x2', data=df).fit()
print(fit.summary(slim=True))

                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.988
Model:                            OLS   Adj. R-squared:                  0.983
No. Observations:                   8   F-statistic:                     208.0
Covariance Type:            nonrobust   Prob (F-statistic):           1.54e-05
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      8.0325      0.673     11.939      0.000       6.303       9.762
x1            -3.5734      0.195    -18.280      0.000      -4.076      -3.071
x2             0.9672      0.049     19.790      0.000       0.842       1.093
==============================================================================
```

Solution

Before discussing the parameter let's have a look at the residuals:

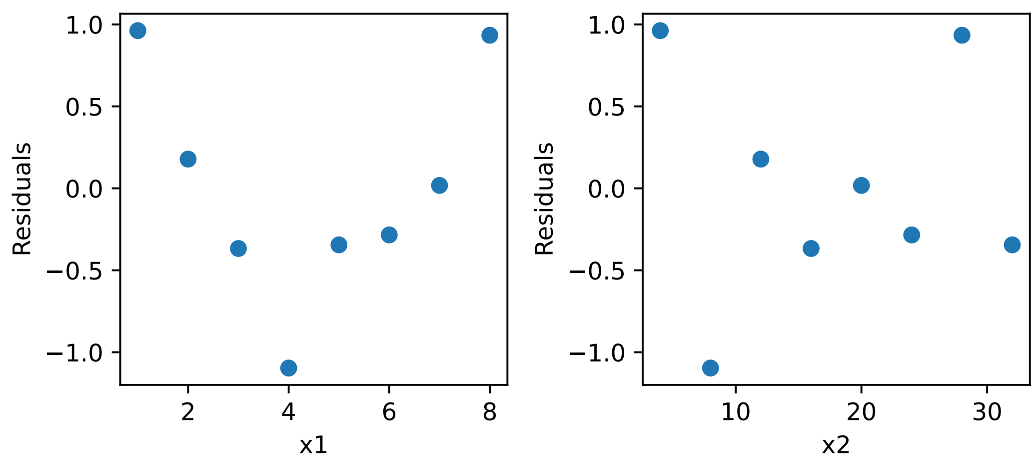
```
fig, ax = plt.subplots(1, 2)
sm.qqplot(fit.resid.values, line="q",a=1/2,ax=ax[0])
ax[0].set_title("Q-Q plot")
ax[1].scatter(fit.fittedvalues, fit.resid)
ax[1].set_xlabel("Fitted values")
ax[1].set_ylabel("Residuals")
ax[1].set_title("Residuals vs Fitted values")
plt.tight_layout()
plt.show()
```



The are no obvious structures in the residuals as a function of the fitted values and also there does not seem be be serious departure from normality, but lets try to look at the residuals as a function of the independent variables anyway

|||| Solution

```
fig, ax = plt.subplots(1, 2)
ax[0].scatter(df['x1'], fit.resid)
ax[0].set_xlabel('x1')
ax[0].set_ylabel('Residuals')
ax[1].scatter(df['x2'], fit.resid)
ax[1].set_xlabel('x2')
ax[1].set_ylabel('Residuals')
plt.tight_layout()
plt.show()
```



the plot of the residuals as a function of x_1 suggest that there could be a quadratic dependence.

|||| Solution

Now include the quadratic dependence of x_1

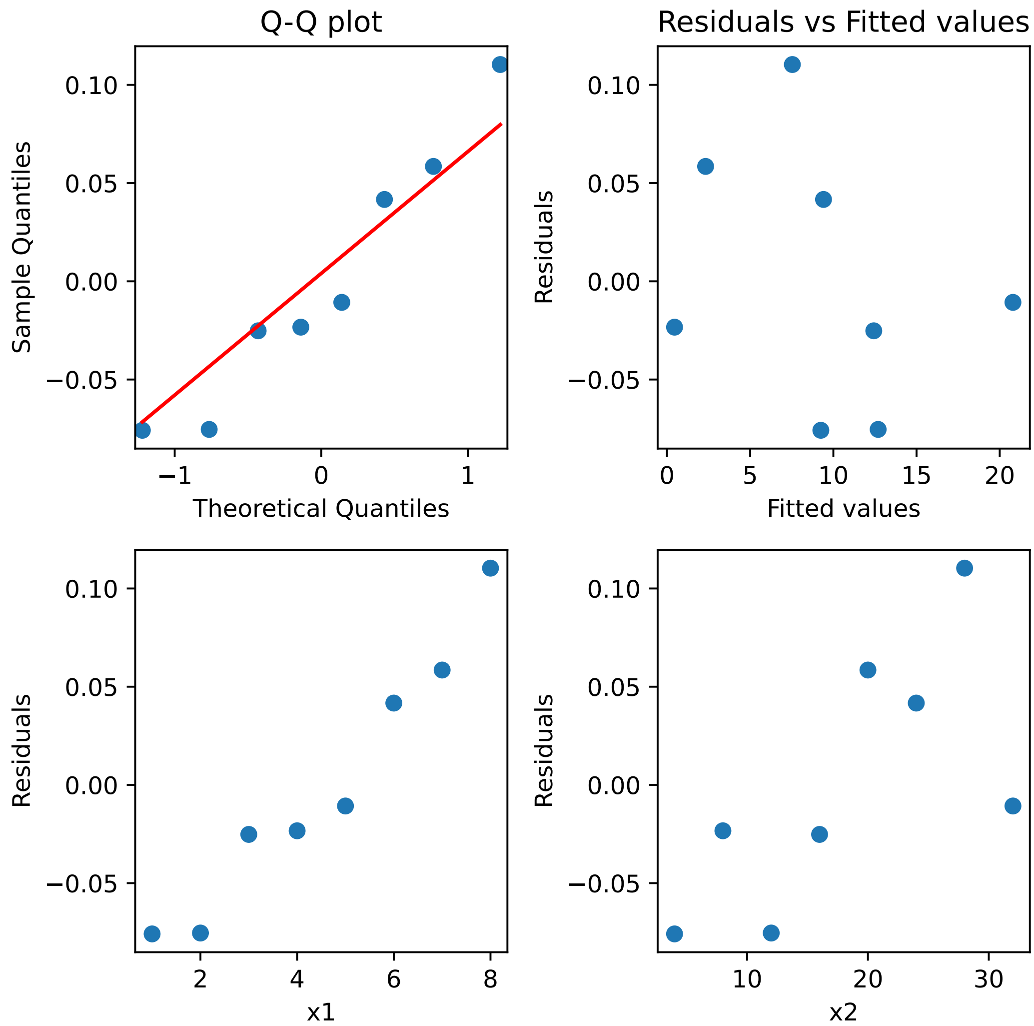
```
df['x3'] = df['x1']**2
fit3 = smf.ols('y ~ x1 + x2 + x3', data=df).fit()
print(fit3.summary(slim=True))

                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       1.000
Model:                            OLS   Adj. R-squared:                  1.000
No. Observations:                   8   F-statistic:                 1.257e+04
Covariance Type:            nonrobust   Prob (F-statistic):           2.11e-08
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     10.1007      0.121     83.334      0.000       9.764      10.437
x1            -5.0024      0.071    -70.549      0.000      -5.199      -4.806
x2             1.0006      0.005    185.205      0.000       0.986       1.016
x3             0.1474      0.007     21.067      0.000       0.128       0.167
==============================================================================
```

we can see that all parameters are still significant, and we can do the residual analysis of the resulting model.

|||| Solution

```
fig, ax = plt.subplots(2, 2)
sm.qqplot(fit3.resid, line="q", ax=ax[0, 0])
ax[0, 0].set_title("Q-Q plot")
ax[0, 1].scatter(fit3.fittedvalues, fit3.resid)
ax[0, 1].set_xlabel("Fitted values")
ax[0, 1].set_ylabel("Residuals")
ax[0, 1].set_title("Residuals vs Fitted values")
ax[1, 0].scatter(df['x1'], fit3.resid)
ax[1, 0].set_xlabel('x1')
ax[1, 0].set_ylabel('Residuals')
ax[1, 1].scatter(df['x2'], fit3.resid)
ax[1, 1].set_xlabel('x2')
ax[1, 1].set_ylabel('Residuals')
plt.tight_layout()
plt.show()
```



There are no obvious structures left and there is no departure from normality, and we can report the finally selected model as

$$Y_i = \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i} + \beta_3 x_{1,i}^2 + \varepsilon_i, \quad \varepsilon_i \sim (N(0, \sigma^2),$$

with the parameters estimates given above.

d) Find the standard error for the line, and the confidence and prediction intervals for the line for the points $(\min(x_1), \min(x_2))$, $(\bar{x}_1, \bar{x}_2)$, $(\max(x_1), \max(x_2))$.

Solution

The question is solved by

```
## New data
Dnew = pd.DataFrame({
    'x1': [np.min(df['x1']), np.mean(df['x1']), np.max(df['x1'])],
    'x2': [np.min(df['x2']), np.mean(df['x2']), np.max(df['x2'])],
    'x3': [np.min(df['x1'])**2, np.mean(df['x1'])**2,
np.max(df['x1'])**2]
})

# Summary frame
pred = fit3.get_prediction(Dnew).summary_frame(alpha=0.05)
print(round(pred,3))

     mean  mean_se  mean_ci_lower  mean_ci_upper  obs_ci_lower  obs_ci_upper
0   9.248    0.073          9.045          9.451         8.934         9.563
1   8.587    0.048          8.454          8.720         8.312         8.862
2  11.538    0.080         11.317         11.760        11.211        11.866
```

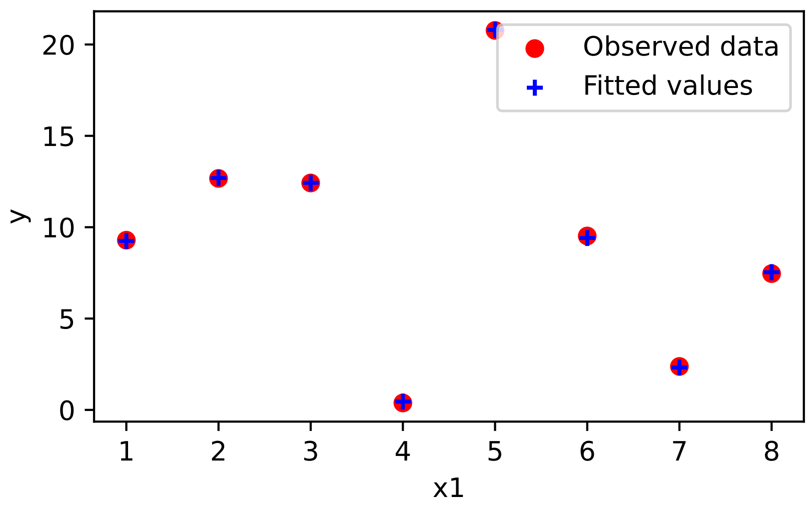
The standard error for line for the new points can be seen in "mean_se" followed by the lower and upper bounds for the confidence and prediction intervals, respectively.

e) Plot the observed values together with the fitted values (e.g. as a function of x_1).

Solution

The question is solved by

```
plt.figure()
plt.scatter(df['x1'], df['y'], marker='o', color='red', label='Observed
data')
plt.scatter(df['x1'], fit3.fittedvalues, marker='+', color='blue',
label='Fitted values')
plt.xlabel('x1')
plt.ylabel('y')
plt.legend()
plt.tight_layout()
plt.show()
```



Notice that we have an almost perfect fit when including x_1, x_2 and x_1^2 in the model, while neither x_1 nor x_2 alone could predict the outcomes.