Problem 2

In [1]:
```
import pandas as pd, numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv('APPENC01.txt', sep = '\s+', header =None)
df.head()
```

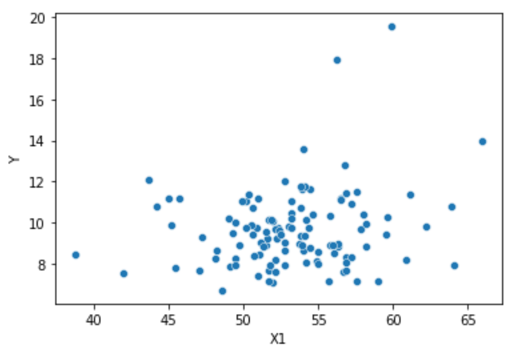
Out[2]:

	0	1	2	3	4	5	6	7	8	9	10	11
0	1	7.13	55.7	4.1	9.0	39.6	279	2	4	207	241	60.0
1	2	8.82	58.2	1.6	3.8	51.7	80	2	2	51	52	40.0
2	3	8.34	56.9	2.7	8.1	74.0	107	2	3	82	54	20.0
3	4	8.95	53.7	5.6	18.9	122.8	147	2	4	53	148	40.0
4	5	11.20	56.5	5.7	34.5	88.9	180	2	1	134	151	40.0

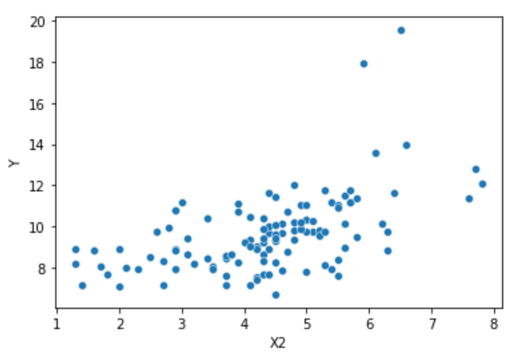
a. Prepare a dot plot for each of the predictor variables. What information do these plots provide?

In [3]:
```
#Model I
Y = df[1]
X1 = df[2]
X2 = df[3]
X3 = df[11]
df1 = pd.DataFrame({'Y':Y,'X1':X1,'X2':X2,'X3':X3})
df1.head()
x1 = df1['X1']
x2 = df1['X2']
x3 = df1['X3']
y = df1['Y']
```

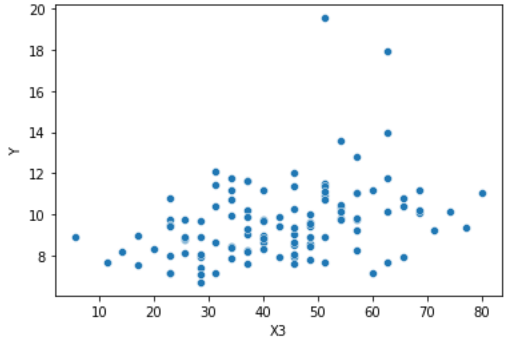
In [4]:
```
sns.scatterplot(x='X1', y='Y', data=df1);
```



In [5]:
```
sns.scatterplot(x='X2', y='Y', data=df1);
```

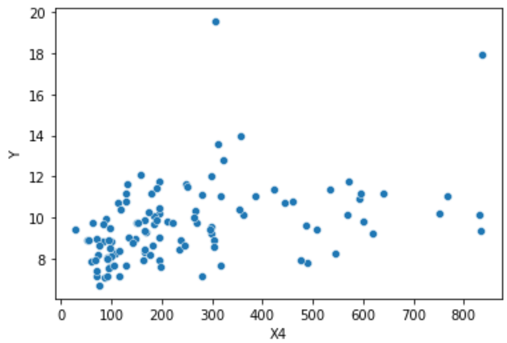


In [6]:
```
sns.scatterplot(x='X3', y='Y', data=df1);
```

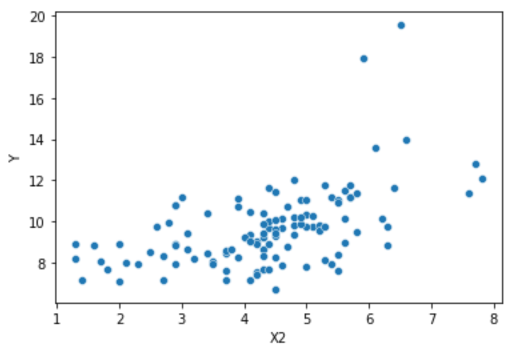


In [7]:
```
#Model II
Y = df[1]
X4 = df[6]
X2 = df[3]
X3 = df[11]
df2 = pd.DataFrame({'Y':Y,'X4':X4,'X2':X2,'X3':X3})
df2.head()
x4 = df2['X4']
x2 = df2['X2']
x3 = df2['X3']
y = df2['Y']
```

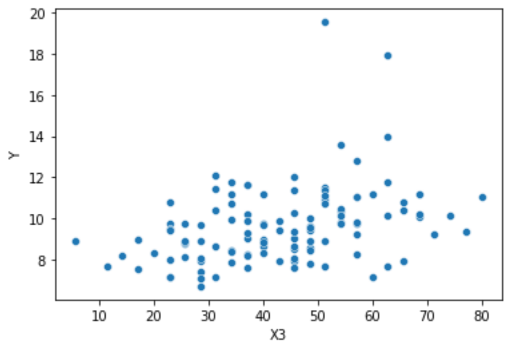
In [8]:
```
sns.scatterplot(x='X4', y='Y', data=df2);
```



In [9]:
```
sns.scatterplot(x='X2', y='Y', data=df2);
```



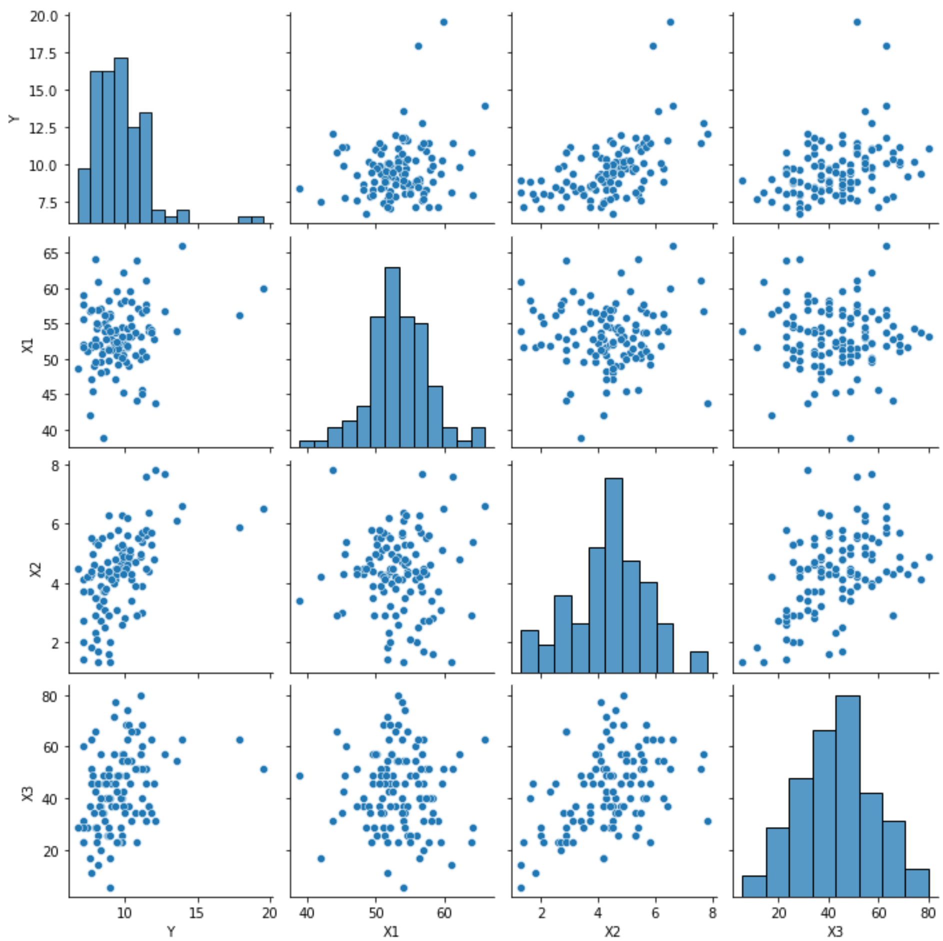
In [10]:
```
sns.scatterplot(x='X3', y='Y', data=df2);
```



There are some outliners in this data

b. Obtain the scatter plot matrix and the correlation matrix for each proposed model. Interpret these and state your principal findings.

In [11]:
```
#the scatter plot matrix model I
sns.pairplot(data=df1);
```

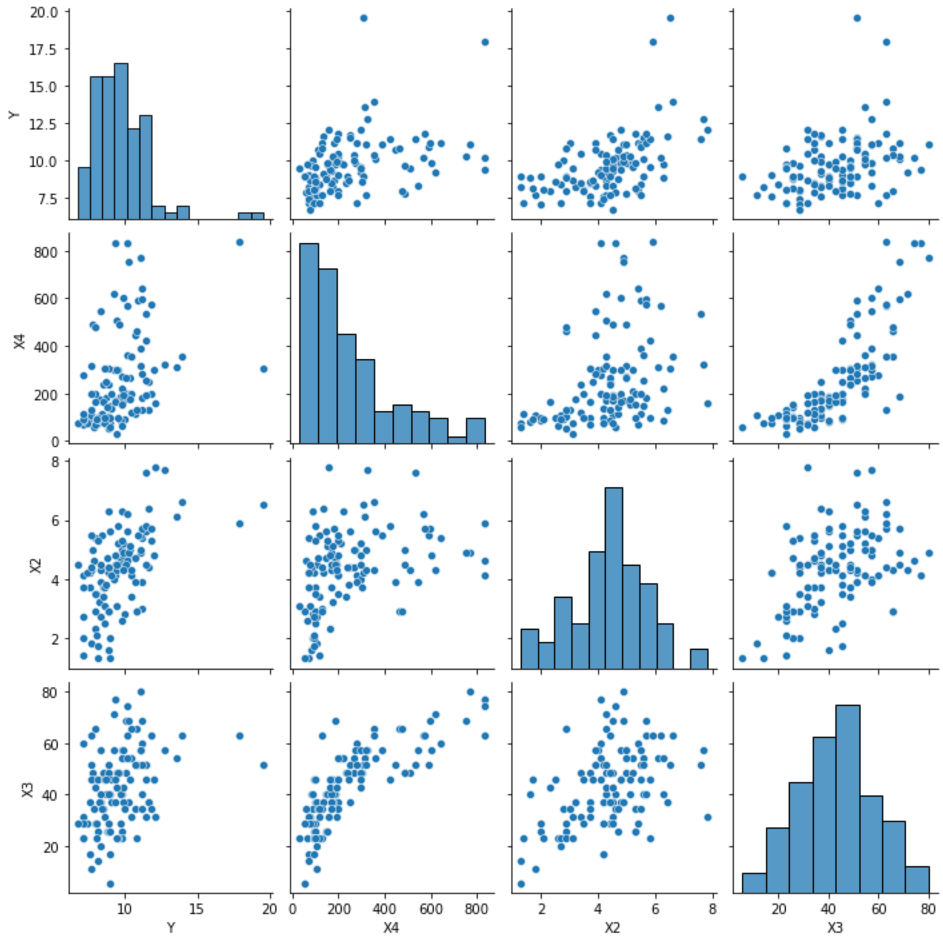


In [12]:
```
# the correlation matrix model I
matrix = df1.corr()
print(matrix)
```

```
           Y        X1        X2        X3
Y   1.000000  0.188914  0.533444  0.355538
X1  0.188914  1.000000  0.001093 -0.040451
X2  0.533444  0.001093  1.000000  0.412601
X3  0.355538 -0.040451  0.412601  1.000000
```

From the correlation matrix, we can conclude that Y is positively correlated to X1 , X2 and X3. Moreover, The correlation is stronger between Y and X2 (0.53) than between Y and X1 (0.19) and between Y and X3 (0.35)

In [13]:
```python
#the scatter plot matrix model II
sns.pairplot(data=df2);
```



In [14]:
```python
# the correlation matrix model II
matrix = df2.corr()
print(matrix)
```

```
           Y        X4        X2        X3
Y   1.000000  0.409265  0.533444  0.355538
X4  0.409265  1.000000  0.359770  0.794524
X2  0.533444  0.359770  1.000000  0.412601
X3  0.355538  0.794524  0.412601  1.000000
```

From the correlation matrix, we can conclude that Y is positively correlated to X4 , X2 and X3. Moreover, The correlation is stronger between Y and X2 (0.53) than between Y and X4 (0.4) and between Y and X3 (0.35)

c. For each of the two models, fit first-order regression model (6.5) with three predictor variables.

In [16]:
```
import statsmodels.api as sm
import statsmodels.formula.api as smf
model1 = smf.ols('y ~ x1+x2+x3', data=df1)
results1 = model1.fit()
results1.summary()
```

Out[16]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.345
Model:	OLS	**Adj. R-squared:**	0.327
Method:	Least Squares	**F-statistic:**	19.12
Date:	Sat, 12 Nov 2022	**Prob (F-statistic):**	4.93e-10
Time:	23:27:16	**Log-Likelihood:**	-209.16
No. Observations:	113	**AIC:**	426.3
Df Residuals:	109	**BIC:**	437.2
Df Model:	3		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	1.3865	1.866	0.743	0.459	-2.312	5.085
x1	0.0837	0.033	2.518	0.013	0.018	0.150
x2	0.6584	0.121	5.426	0.000	0.418	0.899
x3	0.0217	0.011	2.029	0.045	0.001	0.043

Omnibus:	57.943	**Durbin-Watson:**	2.181
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	273.953
Skew:	1.680	**Prob(JB):**	3.25e-60
Kurtosis:	9.848	**Cond. No.**	878.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Coefficients:

- b0 = 1.38
- b1 = 0.08
- b2 = 0.65
- b3 = 0.02

Regression function: Y_hat = beta[0] + beta[1]*X1 + beta[2]*X2 + beta[3]*X3

Y_hat = 1.38 + (0.08)X1 + (0.65)X2 + (0.02)X3

In [18]:
```
import statsmodels.api as sm
import statsmodels.formula.api as smf
model2 = smf.ols('y ~ x4+x2+x3', data=df2)
results2 = model2.fit()
results2.summary()
```

Out[18]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.341
Model:	OLS	**Adj. R-squared:**	0.323
Method:	Least Squares	**F-statistic:**	18.78
Date:	Sat, 12 Nov 2022	**Prob (F-statistic):**	6.85e-10
Time:	23:27:24	**Log-Likelihood:**	-209.51
No. Observations:	113	**AIC:**	427.0
Df Residuals:	109	**BIC:**	437.9
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	6.4674	0.615	10.513	0.000	5.248	7.687
x4	0.0030	0.001	2.373	0.019	0.000	0.006
x2	0.6477	0.122	5.313	0.000	0.406	0.889
x3	-0.0093	0.017	-0.562	0.575	-0.042	0.023

Omnibus:	63.365	**Durbin-Watson:**	2.109
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	329.279
Skew:	1.835	**Prob(JB):**	3.15e-72
Kurtosis:	10.515	**Cond. No.**	1.34e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.34e+03. This might indicate that there are strong multicollinearity or other numerical problems.

Coefficients:

- b0 = 6.467
- b1 = 0.003
- b2 = 0.647
- b3 = -0.009

Regression function: Y_hat = beta[0] + beta[1]*X1 + beta[2]*X2 + beta[3]*X3

Y_hat = 6.467 + (0.003)X1 + (0.647)X2 + (-0.009)X3

d. Calculate R-squared, adjusted R-squared for each model. What do they indicate here? Is one model clearly preferable in terms of this measure?

In [39]:
```
# Model I
sse1 = np.sum((results1.fittedvalues - df1.Y)**2)
ssr1 = np.sum((results1.fittedvalues - df1.Y.mean())**2)
ssto1 = ssr1 + sse1
R_square1 = 1 - (sse1/ssto1)
print(R_square1)
```

```
0.3447741199761676
```

In [40]:
```
# Model II
sse2 = np.sum((results2.fittedvalues - df2.Y)**2)
ssr2 = np.sum((results2.fittedvalues - df2.Y.mean())**2)
ssto2 = ssr2 + sse2
R_square2 = 1 - (sse2/ssto2)
print(R_square2)
```

```
0.3407360016904716
```

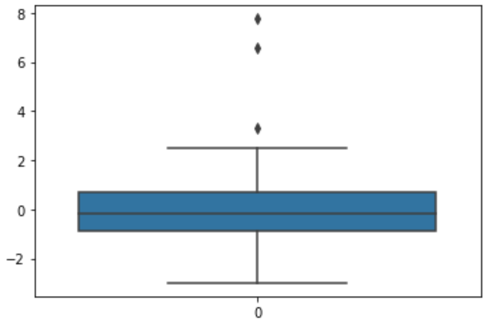
Adding more X variables to the regression model can only increase R^ 2 , because SSE is smaller with more X variables added to the model and SSTO is always the same for a given set of responses.

model 1 is clearly preferable in terms of this measure

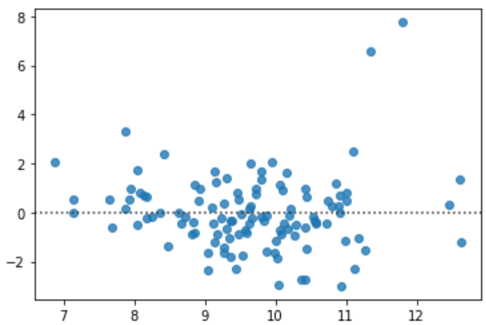
e. For each model, obtain the residuals and plot them against fitted values, each of the three predictor variables, and each of the two-factor interaction terms. Also prepare a normal probability plot of the residuals for each of the two fitted models. Interpret your plots and state your findings. Is one model clearly more appropriate than the other?

 # Model I

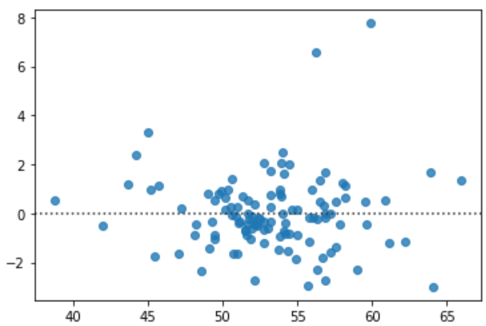
In [21]:
```
resid1 = results1.resid
sns.boxplot(data=resid1);
```



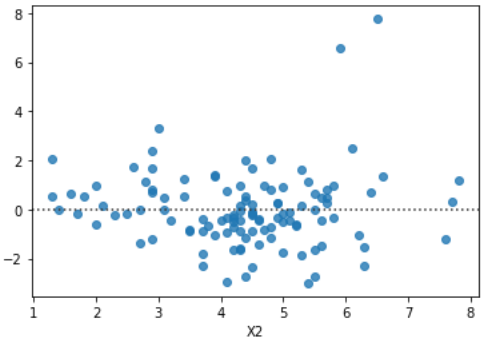
In [22]:
```
#Y_hat
Y_hat1 = results1.predict()
sns.residplot(x=Y_hat1, y=resid1, data=df1);
```



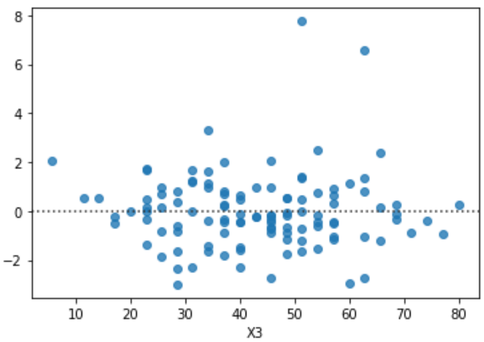
In [23]:
```
# x1
sns.residplot(x=x1, y=resid1, data=df1);
```



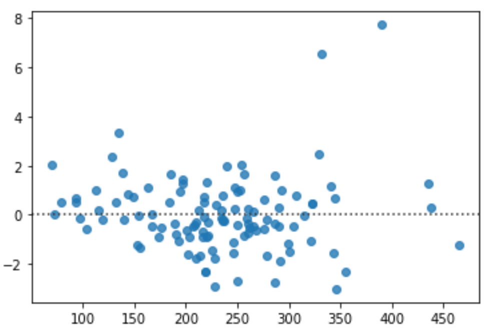
In [24]:
```
# x2
sns.residplot(x=x2, y=resid1, data=df1);
```



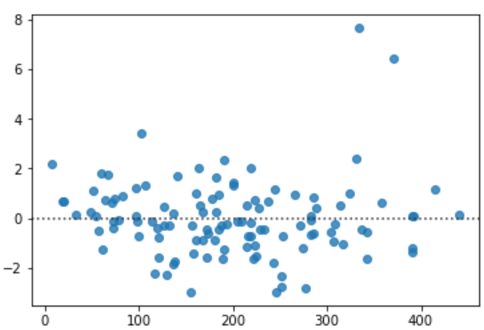
In [25]:
```
# x3
sns.residplot(x=x3, y=resid1, data=df1);
```



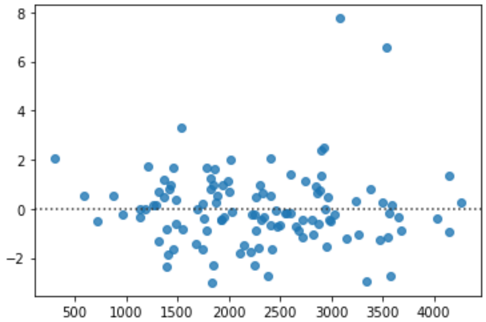
In [26]:
```
# X1X2
X1X2 = x1*x2
sns.residplot(x=X1X2, y=resid1, data=df1);
```



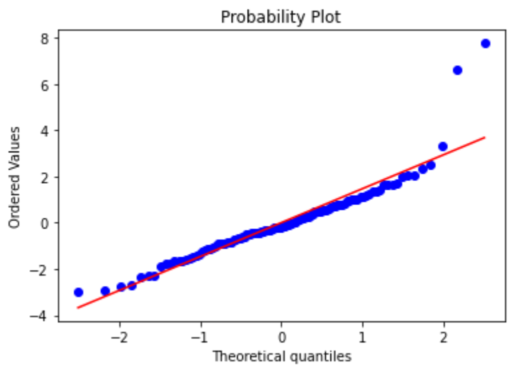
In [27]:
```
#X2X3
X2X3 = x2*x3
sns.residplot(x=X2X3, y=resid1, data=df1);
```



In [28]:
```
# X1X3
X1X3 = x1*x3
sns.residplot(x=X1X3, y=resid1, data=df1);
```



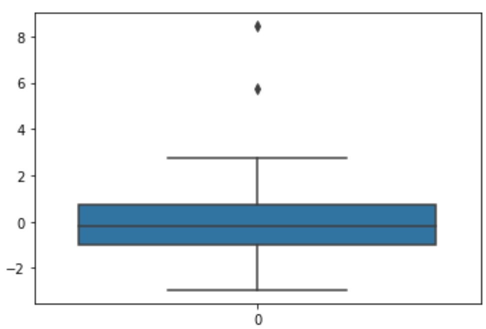
In [29]:
```
import scipy.stats as stats
stats.probplot(resid1, dist="norm", plot = plt);
plt.show();
```



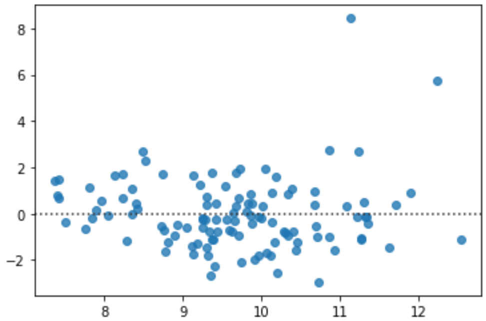
Model I: There is a slight increase in variability for ei vs. Y_hat, but overall it looks okay. The normal probability plot of the residuals looks fine (relatively straight). The plots of the residuals vs. each predictor and each two-way interaction all look appropriately “random.”

Model II

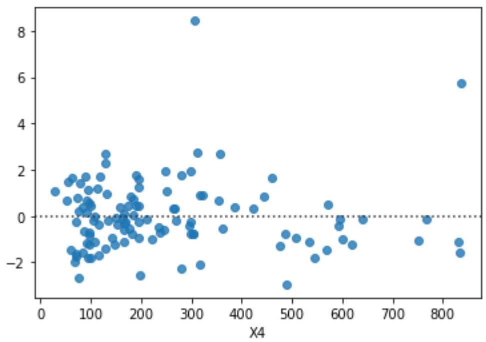
In [30]:
```
resid2 = results2.resid
sns.boxplot(data=resid2);
```



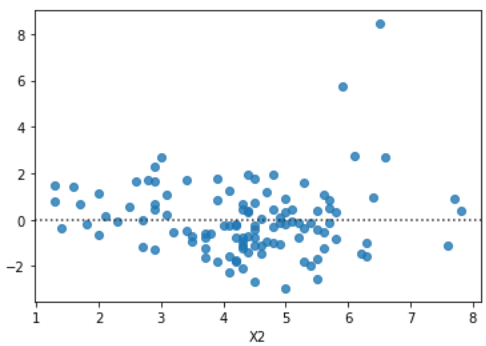
In [31]:
```
#Y_hat
Y_hat2 = results2.predict()
sns.residplot(x=Y_hat2, y=resid2, data=df2);
```



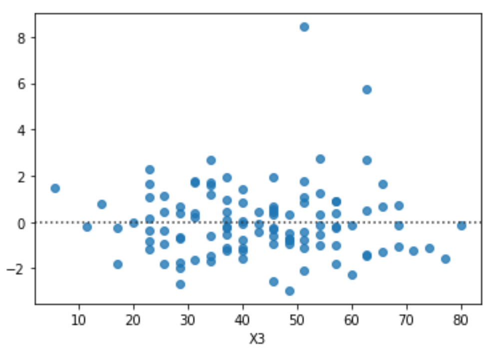
In [32]:
```
# x4
sns.residplot(x=x4, y=resid2, data=df2);
```



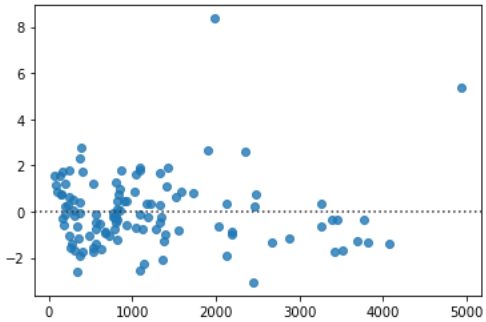
In [33]:
```
# x2
sns.residplot(x=x2, y=resid2, data=df2);
```



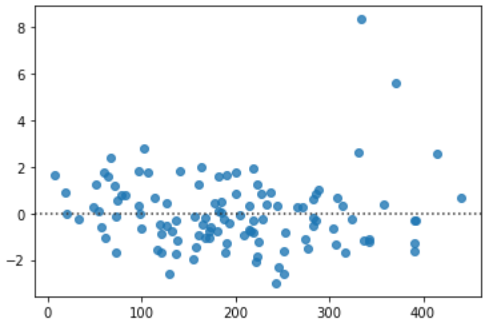
In [34]:
```
# x3
sns.residplot(x=x3, y=resid2, data=df2);
```



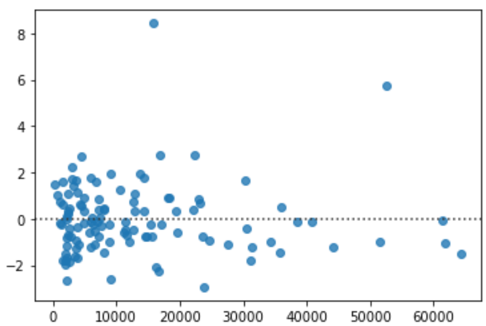
In [35]:
```
# X4X2
X4X2 = x4*x2
sns.residplot(x=X4X2, y=resid2, data=df2);
```



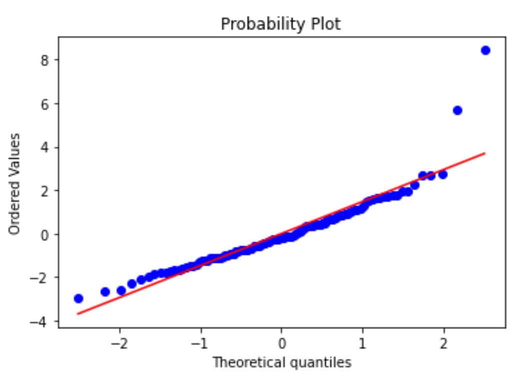
In [36]:
```
#X2X3
X2X3 = x2*x3
sns.residplot(x=X2X3, y=resid2, data=df2);
```



In [37]:
```
# X4X3
X4X3 = x4*x3
sns.residplot(x=X4X3, y=resid2, data=df2);
```



In [38]:
```
import scipy.stats as stats
stats.probplot(resid2, dist="norm", plot = plt);
plt.show();
```



Model II: There is a slight increase in variability for ei vs. Y_hat, but overall it looks okay. The normal probability plot of the residuals looks fine (relatively straight). The plots of the residuals vs. each predictor and each two-way interaction all look appropriately “random.”

Model 2 ís clearly more appropriate than the model 1