MA500 - Assignment One

1 Question 1.1

1. Determine the values for $b_0,\ b_1,\ b_2$ for which $y = b_0 + b_1x + b_2x^2$ is the least squares estimator for the model $y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \epsilon_i$.

Firstly, convert the dataset to matrix form in order to be used with the Ordinary Least Squares fitting algorithm. Add all values y_i as entries in the matrix Y. Add a column of 1s to the matrix X to allow for an intercept term in the regression fit. x_{i1} are the x values from the data set. x_{i2} are the squared x values from the data set since we want to fit a polynomial. B are the coefficients. After doing this, we end up with the following:

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix},\ X = \begin{bmatrix} 1 & x_{11} & x_{12} \\ 1 & x_{21} & x_{22} \\ \vdots & \vdots & \vdots \\ 1 & x_{n1} & x_{n2} \end{bmatrix},\ B = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix}$$

Expressing the normal equations in matrix form, we can derive the equation for finding B.

$$B = (X^T X)^{-1} X^T Y$$

Solving this in Python results in $b_0 = 12.34270521,\ b_1 = 5.00610241,\ b_2 = -0.10068145$.

2. Exhibit a single plot of the data points (in say blue) and the curve $y = b_0 + b_1x + b_2x^2$ (in say red).

Plotted in python using the pyplot interface to matplotlib (Figure 1).

3. Determine the coefficient of determination $r^2 = 1 - (SSE/SSTO)$ for this least squares fit.

$$SSTO = Y^T Y - n\bar{y}^2 = 398889.6736$$
$$SSE = Y^T Y - B^T X^T Y = 5725.8343$$

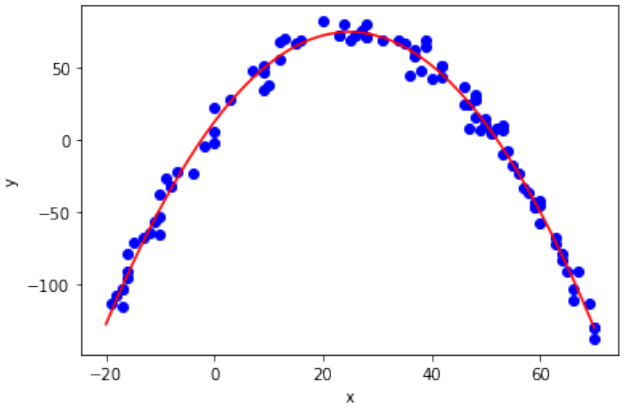


Figure 1

Hence $r^2 = 0.9856$. This is close to 1 and hence is a good fit to the data.

2 Question 1.2

1. (a) Determine the least squares estimator $y = b_0 + b_1x_1 + b_2x_2$.

Using the same method as in Question 1.1, we get $b_0 = 3.36508318$, $b_1 = 4.06510383$, $b_2 = 4.72071287$.

1. (b) Test whether there is a regression equation, using a level of significance of 0.05.

We have a theorem that states that if $\beta_1 = \beta_2 = \cdots = \beta_n = 0$ then $F^* = \frac{MSR}{MSE}$ follows an F distribution with $p-1$ and $n-p$ degrees of freedom. Hence if $F^* \leq F(1-\alpha,\ p-1,\ n-p)$ then we can conclude that $\beta_1 = \beta_2 = \cdots = \beta_n = 0$, that is, there is no regression equation.
If $F^* \geq F(1-\alpha,\ p-1,\ n-p)$ then we can conclude that $\beta_i \neq 0$ for at least one i, that is, that there is a regression equation. Here p is the number of β parameters, n is the number of data points, and α is the significance level.

Calculate F score with Scipy to get $F(1-0.05,\ 2-1,\ 10-2) = 2624.8399$. Here $F^* = \frac{25720.1101}{9.7987}$. $F > F^*$ so conclude that man-minutes **are** related to the number of drums and weights.

1. (c) Estimate β_1 and β_2 jointly, using a 95% family confidence coefficient. We can use the formula

$$b_k - T \cdot s(b_k) \leq \beta_k \leq b_k + T \cdot s(b_k)$$

where $T = t(1-\frac{\alpha}{2q}, n-p)$ and $q = 2$ is the number of coefficients to be estimated. Find $s(b_k)$ as

$$MSE \cdot (X^T X)^{-1} = \begin{bmatrix} s^2(b_0) & s(b_0 b_1) & s(b_0 b_2) \\ s(b_1 b_0) & s^2(b_1) & s(b_1 b_2) \\ s(b_2 b_0) & s(b_2 b_1) & s^2(b_2) \end{bmatrix}$$

We get $s(b_1) = 0.2144,\ s(b_1) = 0.2255$. Then perform t-test in Scipy to get the following confidence intervals:

$$3.4752790766212787 < \beta_1 < 4.654928592254059$$

$$4.100304098832011 < \beta_2 < 5.34112164440534$$

1. (d) Management desires simultaneous interval estimates of the mean handling times for five typical shipments specified in table. Obtain the family of estimates, using a 90 family confidence coefficient.

Put new x values into $y = b_0 + b_1 x_1 + b_2 x_2$.
$\hat{Y} = \pm t_{n-2} \cdot MSE(\frac{1}{n} + SSTO \cdot (X^T X)^{-1})$
...

2. Obtain the residuals and make appropriate residual plots to ascertain whether the model with normal error terms is appropriate. Summarize your findings.

Plots are all centred around 0 with no distinguishable patterns, therefore we can conclude that this model is appropriate.

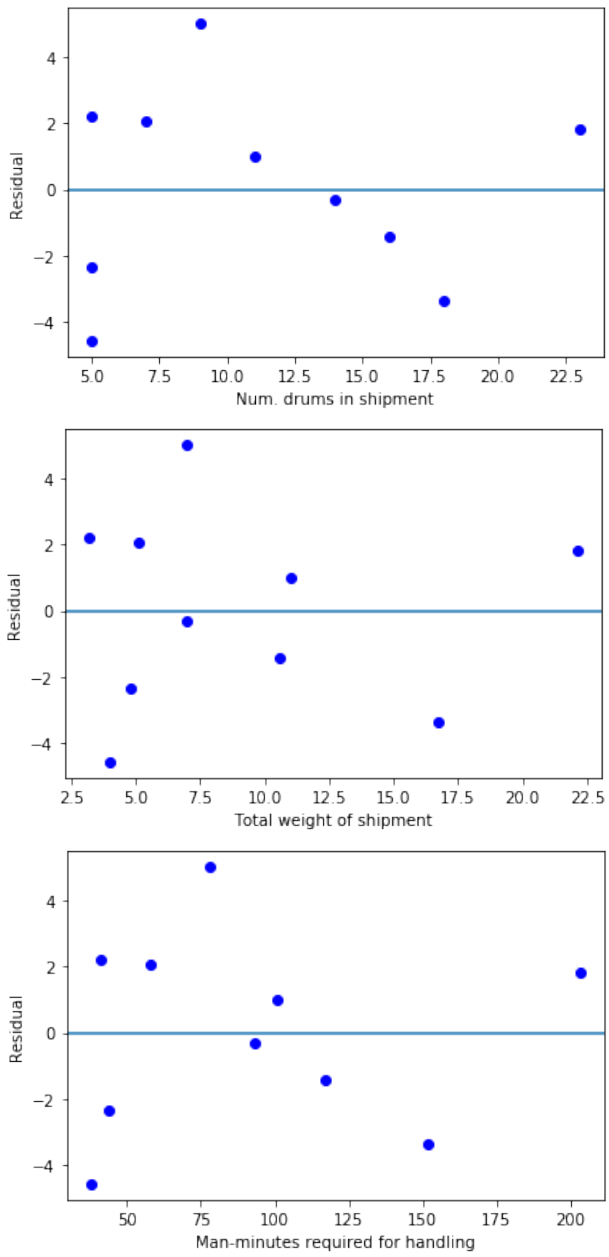

Residual
Num. drums in shipment
Residual
Total weight of shipment
Residual
Man-minutes required for handling

Figure 2

hw1-1

February 6, 2020

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Load text file into pandas dataframe using comma separation.
Split at '=' sign, keeping RHS and converting to float.

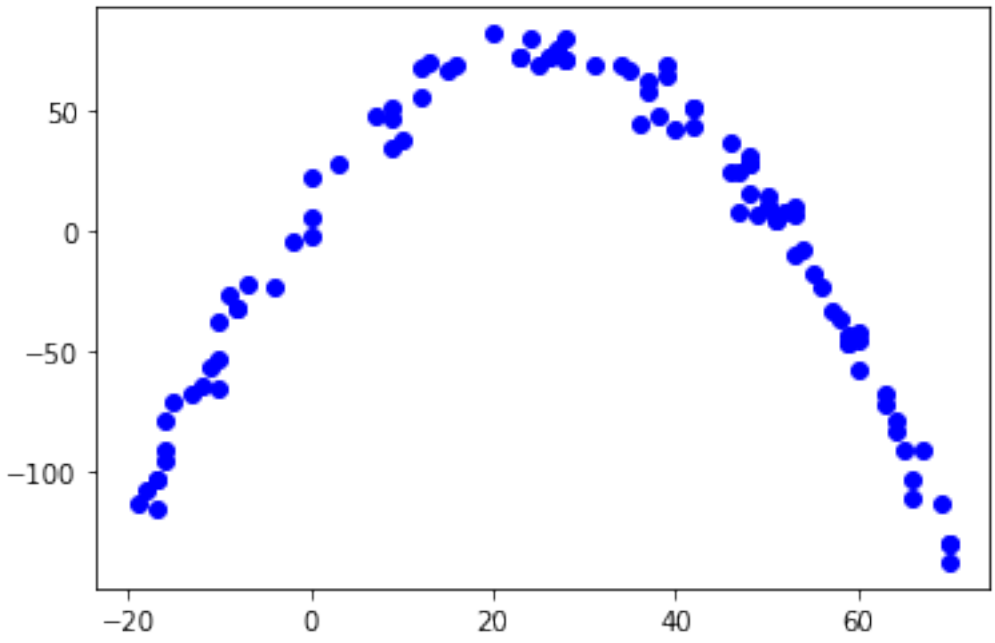
[2]:
```
data = pd.read_csv('data.txt', header=None)  # load data from txt file

x_col = data[0].str.split('=', expand=True)[1].str.strip().astype('float')  #␣
 ↪split at '=' sign, keep RHS, convert to float
y_col = data[1].str.split('=', expand=True)[1].str.strip().astype('float')

data = pd.concat([x_col, y_col], axis=1)  # combine into dataframe
data.columns = ['x', 'y']
```

[3]:
```
plt.plot(data['x'], data['y'], 'bo')
```

[3]:
```
[<matplotlib.lines.Line2D at 0x7f3acefc18d0>]
```



Put data into matrices as required.

[4]:
```
X = np.vstack((np.ones(data['x'].shape), data['x'].values, np.power(data['x'].
 ↪values, 2))).T
Y = data['y'].values
```

Now, find coefficient vector B:

$B = (X^T X)^{-1} X^T Y$

[5]:
```
B = np.linalg.inv(X.T @ X) @ X.T @ Y
Y_hat = X @ B
B
```

[5]:
```
array([12.34270521,  5.00610241, -0.10068145])
```

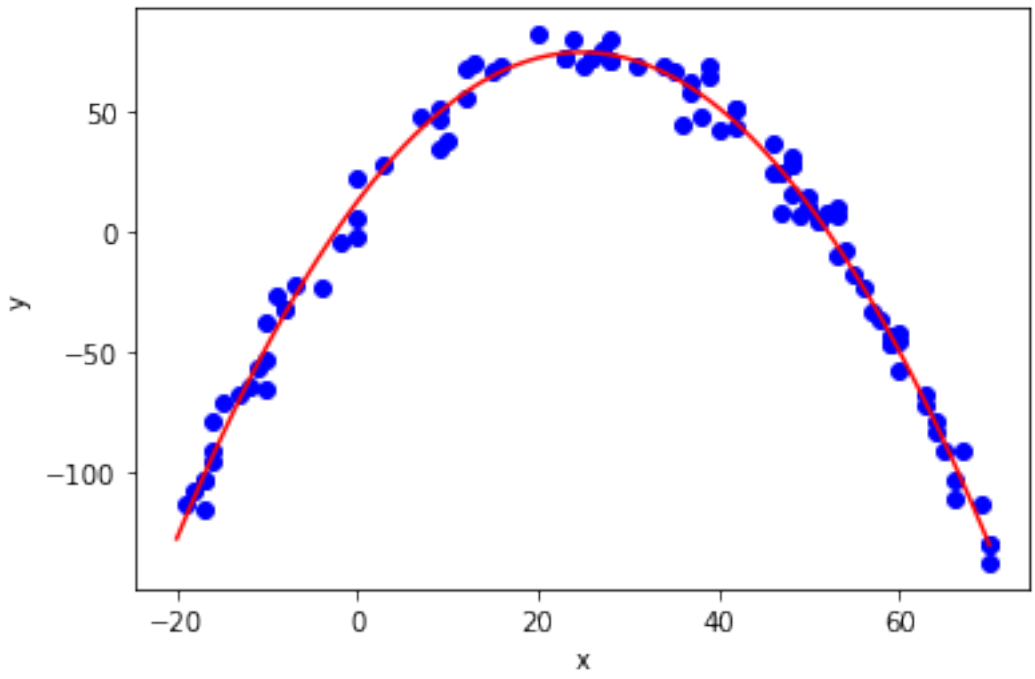
Plot $\hat{Y} = XB$

[6]:
```
x_linsp = np.linspace(-20, 70, num=100)
X_linsp = np.vstack((np.ones(x_linsp.shape), x_linsp, np.power(x_linsp, 2))).T

Y_hat_linsp = X_linsp @ B
```

[7]:
```
plt.plot(data['x'], data['y'], 'bo')
plt.plot(x_linsp, Y_hat_linsp, 'r-')
plt.xlabel('x')
plt.ylabel('y')
```

[7]:
```
Text(0, 0.5, 'y')
```



Calculate $r^2 = 1-$ (SSE/SSTO)

SSE $= \sum (y_i - \hat{y}_i)^2$

SSTO $= \sum (y_i - \bar{y})^2$

[8]:
```
# SSE = 0
# for i, _ in enumerate(Y):
#     SSE += (Y[i] - Y_hat[i])**2

SSE = Y.T @ Y - B.T @ X.T @ Y
SSE
```

[8]: 5725.834315827582

[9]:
```
Y_bar = (1/len(Y)) * np.sum(Y)
# SSTO = 0
# for i, _ in enumerate(Y):
#     SSTO += (Y[i] - Y_bar)**2

SSTO = Y.T @ Y - len(Y)*Y_bar**2
SSTO
```

[9]: 398889.6736

[10]:
```
r2 = 1 - SSE/SSTO
r2
```

[10]: 0.9856455689510545

[ ]:
```

```

hw1-2

February 6, 2020

[24]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
import math
```

[2]:
```
x1 = np.array([7, 18, 5, 14, 11, 5, 23, 9, 16, 5])
x2 = np.array([5.11, 16.70, 3.20, 7.00, 11.00, 4.00, 22.10, 7.00, 10.60, 4.80])
y = np.array([58, 152, 41, 93, 101, 38, 203, 78, 117, 44])
```

[3]:
```
X = np.vstack((np.ones(10), x1, x2)).T
B = np.linalg.inv(X.T @ X) @ X.T @ y
y_hat = X @ B
B
```

[3]:
```
array([3.36508318, 4.06510383, 4.72071287])
```

[5]:
```
# SSE = 0
# for i, _ in enumerate(y):
#     SSE += (y[i] - y_hat[i])**2

# y_bar = (1/len(y)) * np.sum(y)
# SSTO = 0
# for i, _ in enumerate(y):
#     SSTO += (y[i] - y_bar)**2

# SSR = 0
# for i, _ in enumerate(y):
#     SSR += (y_hat[i] - y_bar)**2

y_bar = (1/len(y)) * np.sum(y)

SSE = y.T @ y - B.T @ X.T @ y
SSTO = y.T @ y - len(y)*y_bar**2
SSR = B.T @ X.T @ y - len(y)*y_bar**2

print(SSE, SSTO, SSR)
```

```
78.38987976062344 25798.5 25720.110120239377
```

[6]:
```
r2 = 1 - SSE/SSTO
r2
```

[6]:
```
0.9969614559078774
```

[7]:
```
from scipy import stats

n = len(y)
p = 2

MSE = SSE/(n-p)
print('MSE is: ', MSE)

MSR = SSR/(p-1)
print('MSR is: ', MSR)

F_star = MSR/MSE
print('F* is: ', F_star)

f = stats.f.ppf(1-0.05, p-1, n-p)
print('F is: ', f)
```

```
MSE is:  9.79873497007793
MSR is:  25720.110120239377
F* is:  2624.8398593063307
F is:  5.317655071578714
```

[8]:
```
S2 = MSE*np.linalg.inv(X.T @ X)
S2
```

[8]:
```
array([[ 4.80188237, -0.54479138,  0.25506871],
       [-0.54479138,  0.21436297, -0.20516994],
       [ 0.25506871, -0.20516994,  0.22547827]])
```

[9]:
```
t = stats.t.ppf(1 - 0.05/4, 10-2)
t
```

[9]:
```
2.7515235937129474
```

[26]:
```
print(B[1] - t*math.sqrt(S2[1, 1]), ' < b1 < ', B[1] + t*math.sqrt(S2[1, 1]))
print(B[2] - t*S2[2, 2], ' < b2 < ', B[2] + t*S2[2, 2])
```

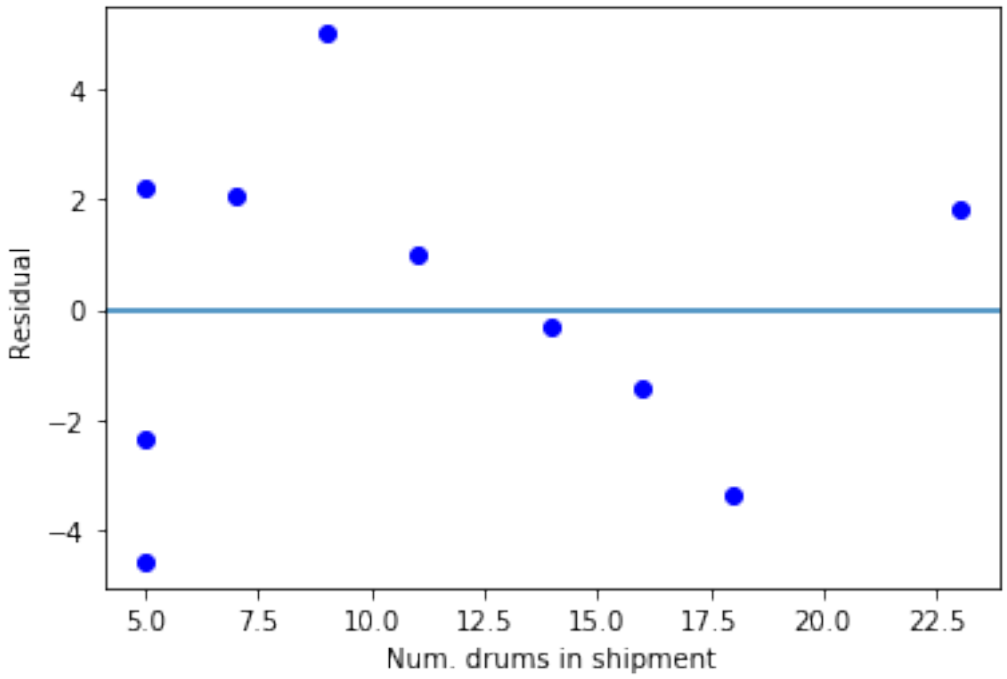
```
2.791166340960135  < b1 <  5.3390413279152025
4.100304098832011  < b2 <  5.34112164440534
```

[11]:
```
# t = stats.t.ppf(1 - 0.10/4, 10-2)

# se = MSE*(1/len(y) + SSTO*np.linalg.inv(X.T@X))
# se
```

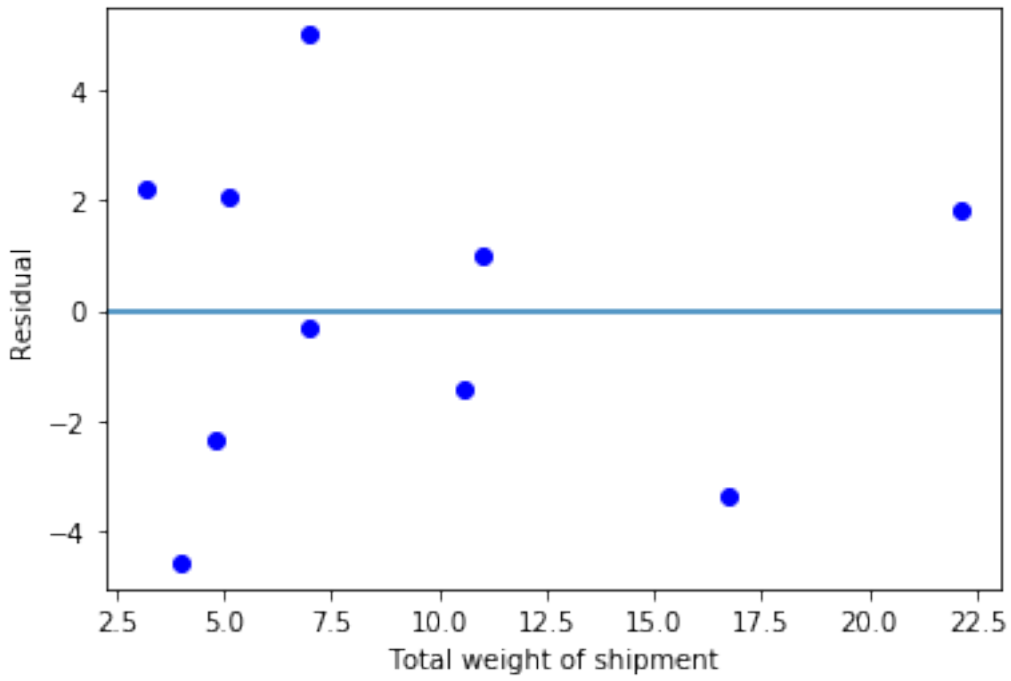
[20]:
```
errors = y - y_hat
plt.plot(x1, errors, 'bo')
plt.axhline(0)
plt.xlabel('Num. drums in shipment')
plt.ylabel('Residual')
```

[20]:
```
Text(0, 0.5, 'Residual')
```



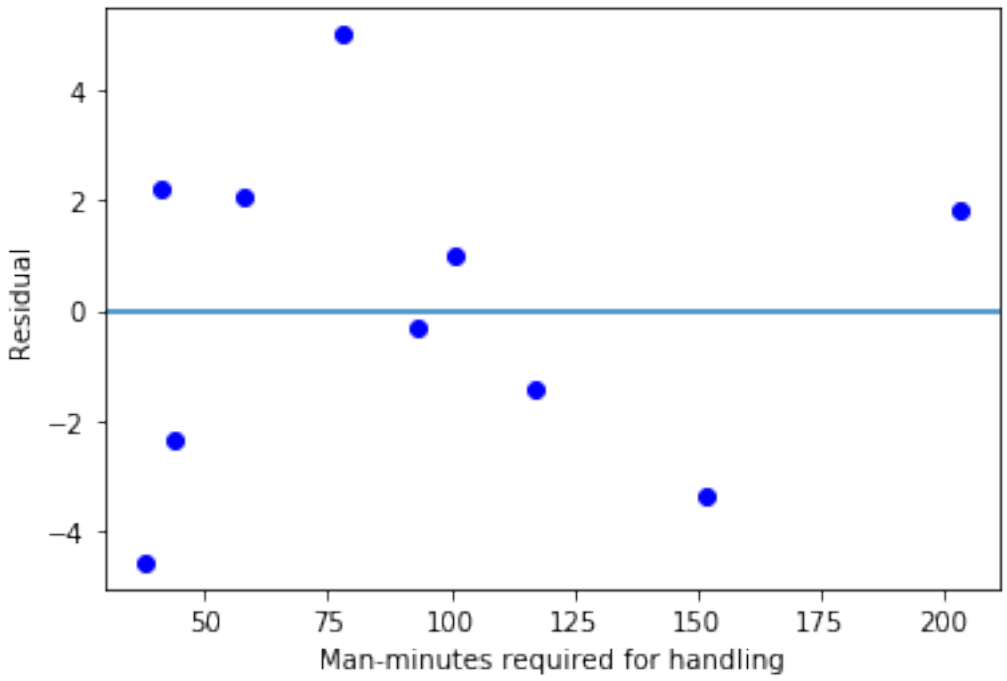
[21]:
```
plt.plot(x2, errors, 'bo')
plt.axhline(0)
plt.xlabel('Total weight of shipment')
plt.ylabel('Residual')
```

[21]:
```
Text(0, 0.5, 'Residual')
```



[22]:
```
plt.plot(y, errors, 'bo')
plt.axhline(0)
plt.xlabel('Man-minutes required for handling')
plt.ylabel('Residual')
```

[22]: Text(0, 0.5, 'Residual')


Residual
4
2
0
−2
−4
50
75
100
125
150
175
200
Man-minutes required for handling

[ ]: