STAT 154/254 Homework 2

Raymond Tsao

TOTAL POINTS

63 / 71

QUESTION 1

1 14 pts

1.1 1.1 **4 / 4**

✓ **- 0 pts** *Entirely correct*

- 1 pts Correct work, but further simplification is needed for $$\hat{\beta_1}$$.

- 1.5 pts Correct solution, but no work or derivation is shown.

- 2 pts Partial work shown for Problem 1.1

- 4 pts No work was shown for Problem 1.1

1.2 1.2 **2 / 2**

✓ **- 0 pts** *Entirely correct*

- 1 pts Partial credit for Problem 1.2

- 2 pts No work shown for Problem 1.2

1.3 1.3 **8 / 8**

✓ **- 0 pts** *Entirely correct*

- 2 pts Correct results for expectation and variance, but no derivations are shown.

- 4 pts Partial credit for Problem 1.3

- 8 pts No work shown for Problem 1.3

QUESTION 2

2 12 pts

2.1 2.1 **3 / 3**

✓ **- 0 pts** *Entirely correct*

- 0.5 pts Correct, but needs justification as to why.

- 1.5 pts Partial credit for Q2.1

- 3 pts No work shown for Q2.1

2.2 2.2 **3 / 3**

✓ **- 0 pts** *Entirely correct*

- 0.5 pts Correct, but justification is needed.

- 1.5 pts Partial credit for Problem 2.2

- 3 pts No work shown for Problem 2.2

2.3 2.3 **3 / 3**

✓ **- 0 pts** *Entirely correct*

- 1.5 pts Partial credit for Problem 2.3

- 3 pts No work shown for Problem 2.3

2.4 2.4 **3 / 3**

✓ **- 0 pts** *Entirely correct*

- 1.5 pts Partial credit for Problem 2.4

- 3 pts No work shown for Problem 2.4

- 0.5 pts Correct, but more precise reasoning is needed.

QUESTION 3

3 5 pts

3.1 3.1 **2 / 2**

✓ **- 0 pts** *Entirely correct*

- 0.5 pts Correct, but no precise justification

provided

- 1 pts Partial credit for Problem 3.1

- 2 pts No work shown for Problem 3.1

3.2 3.2 **3 / 3**

✓ **- 0 pts** *Entirely correct*

- 0.5 pts Correct, but more precise justification is needed.

- 1.5 pts Partial credit for Problem 3.2

- 3 pts No work shown for Problem 3.2

QUESTION 4

4 26 pts

4.1 4.1 **3 / 3**

✓ **- 0 pts** *Entirely correct*

- 1.5 pts Partial credit for Problem 4.1

- 3 pts No work shown for Problem 4.1

4.2 4.2 **3 / 3**

✓ **- 0 pts** *Entirely correct*

- 1.5 pts Partial work for Problem 4.2

- 3 pts No work shown for Problem 4.2

4.3 4.3 **4 / 4**

✓ **- 0 pts** *Entirely correct*

- 2 pts Partial credit for Problem 4.3

- 4 pts No work shown for Problem 4.3

4.4 4.4 **4 / 4**

✓ **- 0 pts** *Entirely correct*

- 2 pts Partial credit for Problem 4.4

- 4 pts No work shown for Problem 4.4

4.5 4.5 **4 / 4**

✓ **- 0 pts** *Entirely correct*

- 2 pts Partial work for Problem 4.5

- 4 pts No work shown for Problem 4.5

4.6 4.6 (M) **0 / 8**

- 0 pts Entirely correct

- 2 pts Correct approach, but slight error in derivation

- 4 pts Partial credit for Problem 4.6

- 8 pts No work shown for Problem 4.6

✓ **- 8 pts** *Question is not intended for undergrads. Your total score is out of 63 points instead of 71 points.*

QUESTION 5

5 6 pts

5.1 3.7.11a **1 / 1**

✓ **- 0 pts** *Entirely correct*

- 0.5 pts Partial credit for Problem 5.1

- 1 pts No work shown for Problem 5.1

5.2 3.7.11b **1 / 1**

✓ **- 0 pts** *Entirely correct*

- 0.5 pts Partial credit for Problem 5.2

- 1 pts No work shown for Problem 5.2

5.3 3.7.11c **1 / 1**

✓ **- 0 pts** *Entirely correct*

- 0.5 pts Partial credit for Problem 5.3

- 1 pts No work shown for Problem 5.3

5.4 3.7.11d **1 / 1**

✓ **- 0 pts** *Entirely correct*

- 0.5 pts Partial credit for Problem 5.4

- **1 pts** No work shown for Problem 5.4

5.5 3.7.11e **1 / 1**

✓ **- 0 pts** *Entirely correct*

- **0.3 pts** Showed the t-statistics are equivalent, but didn't indicate why it was the case.
- **0.5 pts** Partial credit for Problem 5.5
- **1 pts** No work shown for Problem 5.5

5.6 3.7.11f **1 / 1**

✓ **- 0 pts** *Entirely correct*

- **0.5 pts** Partial credit shown for Problem 5.6
- **1 pts** No work shown for Problem 5.6

QUESTION 6

6 8 pts

6.1 6.1 **1 / 1**

✓ **- 0 pts** *Entirely correct*

- **0.5 pts** Partial credit for Problem 6.1
- **1 pts** No work shown for Problem 6.1

6.2 6.2 **1 / 1**

✓ **- 0 pts** *Entirely correct*

- **0.5 pts** Partial credit for Problem 6.2
- **1 pts** No work shown for Problem 6.2

6.3 6.3 **2 / 2**

✓ **- 0 pts** *Entirely correct*

- **1 pts** Partial credit for Problem 6.3
- **2 pts** No work shown for Problem 6.3

6.4 6.4 **2 / 2**

✓ **- 0 pts** *Entirely correct*

- **1 pts** Partial credit for Problem 6.4
- **2 pts** No work shown for Problem 6.4

6.5 6.5 **2 / 2**

✓ **- 0 pts** *Entirely correct*

- **1 pts** Partial credit for Problem 6.5
- **2 pts** No work shown for Problem 6.5

Problem 1

(1)

We minimize the mean square error

$$l = \frac{1}{n}\sum_{i=1}^{n}(\beta_0 + X_i\beta_1 - Y_i)^2$$

Taking partial derivatives we have

$$\begin{cases} \frac{\partial l}{\partial \beta_0} = \frac{2}{n}\sum_{i=1}^{n}(\beta_0 + X_i\beta_1 - Y_i) = 0 \\ \frac{\partial l}{\partial \beta_1} = \frac{2}{n}\sum_{i=1}^{n} x_i(\beta_0 + X_i\beta_1 - Y_i) = 0 \end{cases}$$

Rearranging, we have

$$\begin{cases} n\beta_0 + \beta_1\sum_{i=1}^{n} X_i - \sum_{i=1}^{n} Y_i = 0 \\ \beta_0\sum_{i=1}^{n} X_i + \beta_1\sum_{i=1}^{n} X_i^2 - \sum_{i=1}^{n} X_iY_i = 0 \end{cases}$$

From the first equation, we have

$$\hat{\beta}_0 = \frac{1}{n}\sum_{i=1}^{n} Y_i - \hat{\beta}_1\frac{1}{n}\sum_{i=1}^{n} X_i$$

Substituting this into the second line, we have

$$\frac{1}{n}\left(\sum_{i=1}^{n} Y_i\right)\left(\sum_{i=1}^{n} X_i\right) - \hat{\beta}_1\frac{1}{n}\left(\sum_{i=1}^{n} X_i\right)^2 + \hat{\beta}_1\sum_{i=1}^{n} X_i^2 - \sum_{i=1}^{n} X_iY_i$$

Which gives us

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n} X_iY_i - \frac{1}{n}\left(\sum_{i=1}^{n} Y_i\right)\left(\sum_{i=1}^{n} X_i\right)}{\sum_{i=1}^{n} X_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n} X_i\right)^2}$$

Hence, we have

$$\begin{cases} \hat{\beta}_0 = \overline{Y} - \hat{\beta}_1\overline{X} \\ \hat{\beta}_1 = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sum_{i=1}^{n}(X_i - \overline{X})^2} \end{cases}$$

Where $\overline{X}, \overline{Y}$ denote the mean of the data.

(2)

We have

$$\hat{\beta}_0 + \hat{\beta}_1\overline{X} = (\overline{Y} - \hat{\beta}_1\overline{X}) + \hat{\beta}_1\overline{X} = \overline{Y}$$

Hence, $(\overline{X}, \overline{Y})$ is on the least squares line.

1.1 1.1 **4 / 4**

✓ **- 0 pts** *Entirely correct*

- 1 pts Correct work, but further simplification is needed for $$\hat{\beta_1}$$.

- 1.5 pts Correct solution, but no work or derivation is shown.

- 2 pts Partial work shown for Problem 1.1

- 4 pts No work was shown for Problem 1.1

Problem 1

(1)

We minimize the mean square error

$$l = \frac{1}{n}\sum_{i=1}^{n}(\beta_0 + X_i\beta_1 - Y_i)^2$$

Taking partial derivatives we have

$$\begin{cases} \frac{\partial l}{\partial \beta_0} = \frac{2}{n}\sum_{i=1}^{n}(\beta_0 + X_i\beta_1 - Y_i) = 0 \\ \frac{\partial l}{\partial \beta_1} = \frac{2}{n}\sum_{i=1}^{n} x_i(\beta_0 + X_i\beta_1 - Y_i) = 0 \end{cases}$$

Rearranging, we have

$$\begin{cases} n\beta_0 + \beta_1\sum_{i=1}^{n} X_i - \sum_{i=1}^{n} Y_i = 0 \\ \beta_0\sum_{i=1}^{n} X_i + \beta_1\sum_{i=1}^{n} X_i^2 - \sum_{i=1}^{n} X_iY_i = 0 \end{cases}$$

From the first equation, we have

$$\hat{\beta}_0 = \frac{1}{n}\sum_{i=1}^{n} Y_i - \hat{\beta}_1\frac{1}{n}\sum_{i=1}^{n} X_i$$

Substituting this into the second line, we have

$$\frac{1}{n}\left(\sum_{i=1}^{n} Y_i\right)\left(\sum_{i=1}^{n} X_i\right) - \hat{\beta}_1\frac{1}{n}\left(\sum_{i=1}^{n} X_i\right)^2 + \hat{\beta}_1\sum_{i=1}^{n} X_i^2 - \sum_{i=1}^{n} X_iY_i$$

Which gives us

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n} X_iY_i - \frac{1}{n}\left(\sum_{i=1}^{n} Y_i\right)\left(\sum_{i=1}^{n} X_i\right)}{\sum_{i=1}^{n} X_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n} X_i\right)^2}$$

Hence, we have

$$\begin{cases} \hat{\beta}_0 = \overline{Y} - \hat{\beta}_1\overline{X} \\ \hat{\beta}_1 = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sum_{i=1}^{n}(X_i - \overline{X})^2} \end{cases}$$

Where $\overline{X}, \overline{Y}$ denote the mean of the data.

(2)

We have

$$\hat{\beta}_0 + \hat{\beta}_1\overline{X} = (\overline{Y} - \hat{\beta}_1\overline{X}) + \hat{\beta}_1\overline{X} = \overline{Y}$$

Hence, $(\overline{X}, \overline{Y})$ is on the least squares line.

(3)

First note that $\hat{\beta}_0, \hat{\beta}_1$ are both Gaussian random variables since they are both linear combinations of Y_i (which is a Gaussian distribution).
Hence, it remains finding the expectation and variance of $\hat{\beta}_0$ and $\hat{\beta}_1$.
Note that

$$\mathbb{E}[\hat{\beta}_1] = \frac{1}{\sum_{i=1}^n (X_i - \overline{X})^2} \sum_{i=1}^n (X_i - \overline{X}) \mathbb{E}[Y_i - \overline{Y}]$$

Since

$$\begin{aligned} \mathbb{E}[Y_i - \overline{Y}] &= \mathbb{E}[\hat{\beta}_0 + \hat{\beta}_1 X_i + \epsilon_i] - \frac{1}{n} \sum_{i=1}^n \mathbb{E}[\hat{\beta}_0 + \hat{\beta}_1 X_i + \epsilon_i] \\ &= \hat{\beta}_1 (X_i - \overline{X}) \end{aligned}$$

We have

$$\begin{aligned} \mathbb{E}[\hat{\beta}_1] &= \frac{1}{\sum_{i=1}^n (X_i - \overline{X})^2} \sum_{i=1}^n (X_i - \overline{X}) \hat{\beta}_1 (X_i - \overline{X}) \\ &= \hat{\beta}_1 \end{aligned}$$

With this, we have

$$\begin{aligned} \mathbb{E}[\hat{\beta}_0] &= \mathbb{E}[\overline{Y} - \hat{\beta}_1 \overline{X}] \\ &= \mathbb{E}[\overline{Y}] - \overline{X} \mathbb{E}[\hat{\beta}_1] \\ &= \hat{\beta}_0 \end{aligned}$$

We now compute the variance. Note that

$$\mathrm{Var}(\hat{\beta}_1) = \frac{1}{(\sum_{i=1}^n (X_i - \overline{X})^2)^2} \left[\mathrm{Var}(\sum_{i=1}^n (X_i - \overline{X}) Y_i) - \mathrm{Var}(\sum_{i=1}^n (X_i - \overline{X}) \overline{Y}) \right]$$

Note that the second term vanishes, this leaves us

$$\begin{aligned} \mathrm{Var}(\hat{\beta}_1) &= \frac{1}{(\sum_{i=1}^n (X_i - \overline{X})^2)^2} \mathrm{Var}(\sum_{i=1}^n (X_i - \overline{X}) Y_i) \\ &= \frac{1}{(\sum_{i=1}^n (X_i - \overline{X})^2)^2} \mathrm{Var}(\sum_{i=1}^n (X_i - \overline{X}) \epsilon_i) \\ &= \frac{1}{(\sum_{i=1}^n (X_i - \overline{X})^2)^2} \sum_{i=1}^n (X_i - \overline{X})^2 \mathrm{Var}(\epsilon_i) \\ &= \frac{\sigma^2}{\sum_{i=1}^n (X_i - \overline{X})^2} \end{aligned}$$

1.2 1.2 **2 / 2**

✓ **- 0 pts** *Entirely correct*

- 1 pts Partial credit for Problem 1.2

- 2 pts No work shown for Problem 1.2

(3)

First note that $\hat{\beta}_0, \hat{\beta}_1$ are both Gaussian random variables since they are both linear combinations of Y_i (which is a Gaussian distribution).
Hence, it remains finding the expectation and variance of $\hat{\beta}_0$ and $\hat{\beta}_1$.
Note that

$$\mathbb{E}[\hat{\beta}_1] = \frac{1}{\sum_{i=1}^n (X_i - \overline{X})^2} \sum_{i=1}^n (X_i - \overline{X}) \mathbb{E}[Y_i - \overline{Y}]$$

Since

$$\begin{aligned} \mathbb{E}[Y_i - \overline{Y}] &= \mathbb{E}[\hat{\beta}_0 + \hat{\beta}_1 X_i + \epsilon_i] - \frac{1}{n} \sum_{i=1}^n \mathbb{E}[\hat{\beta}_0 + \hat{\beta}_1 X_i + \epsilon_i] \\ &= \hat{\beta}_1 (X_i - \overline{X}) \end{aligned}$$

We have

$$\begin{aligned} \mathbb{E}[\hat{\beta}_1] &= \frac{1}{\sum_{i=1}^n (X_i - \overline{X})^2} \sum_{i=1}^n (X_i - \overline{X}) \hat{\beta}_1 (X_i - \overline{X}) \\ &= \hat{\beta}_1 \end{aligned}$$

With this, we have

$$\begin{aligned} \mathbb{E}[\hat{\beta}_0] &= \mathbb{E}[\overline{Y} - \hat{\beta}_1 \overline{X}] \\ &= \mathbb{E}[\overline{Y}] - \overline{X} \mathbb{E}[\hat{\beta}_1] \\ &= \hat{\beta}_0 \end{aligned}$$

We now compute the variance. Note that

$$\mathrm{Var}(\hat{\beta}_1) = \frac{1}{(\sum_{i=1}^n (X_i - \overline{X})^2)^2} \left[\mathrm{Var}(\sum_{i=1}^n (X_i - \overline{X}) Y_i) - \mathrm{Var}(\sum_{i=1}^n (X_i - \overline{X}) \overline{Y}) \right]$$

Note that the second term vanishes, this leaves us

$$\begin{aligned} \mathrm{Var}(\hat{\beta}_1) &= \frac{1}{(\sum_{i=1}^n (X_i - \overline{X})^2)^2} \mathrm{Var}(\sum_{i=1}^n (X_i - \overline{X}) Y_i) \\ &= \frac{1}{(\sum_{i=1}^n (X_i - \overline{X})^2)^2} \mathrm{Var}(\sum_{i=1}^n (X_i - \overline{X}) \epsilon_i) \\ &= \frac{1}{(\sum_{i=1}^n (X_i - \overline{X})^2)^2} \sum_{i=1}^n (X_i - \overline{X})^2 \mathrm{Var}(\epsilon_i) \\ &= \frac{\sigma^2}{\sum_{i=1}^n (X_i - \overline{X})^2} \end{aligned}$$

Next, we have

$$\begin{aligned}
\mathrm{Var}(\hat{\beta}_0) &= \mathrm{Var}(\overline{Y} - \hat{\beta}_1\overline{X}) \\
&= \mathrm{Var}(\overline{Y}) + \overline{X}^2\mathrm{Var}(\hat{\beta}_1) - 2\overline{X}\mathrm{Cov}(\overline{Y}, \hat{\beta}_1) \\
&= \frac{\sigma^2}{n} + \frac{\sigma^2}{\sum_{i=1}^n (X_i - \overline{X})^2} - \frac{2\overline{X}}{n}\frac{\sum_{i=1}^n (X_i - \overline{X})\mathrm{Cov}(Y_i, X_i)}{\sum_{i=1}^n (X_i - \overline{X})^2} \\
&= \frac{\sigma^2 \sum_{i=1}^n X_i^2}{n\sum_{i=1}^n (X_i - \overline{X})^2}
\end{aligned}$$

We conclude that $\hat{\beta}_0$ and $\hat{\beta}_1$ are Gaussian random variable with the above mean and variance.

Comparing the variance formula obtained using matrix form, we have

$$\mathrm{Var}(\hat{\beta}) = \begin{bmatrix} \frac{\sigma^2 \sum_{i=1}^n X_i^2}{n\sum_{i=1}^n (X_i - \overline{X})^2} & \frac{-\sigma^2 \sum_{i=1}^n X_i}{n\sum_{i=1}^n (X_i - \overline{X})^2} \\ \frac{-\sigma^2 \sum_{i=1}^n X_i}{n\sum_{i=1}^n (X_i - \overline{X})^2} & \frac{\sigma^2}{\sum_{i=1}^n (X_i - \overline{X})^2} \end{bmatrix}$$

Which is the same as the formula we derived.

By the same approach, one can also show that

$$\mathrm{Cov}(\hat{\beta}_0, \hat{\beta}_1) = \frac{-\sigma^2 \sum_{i=1}^n X_i}{n\sum_{i=1}^n (X_i - \overline{X})^2}$$

1.3 1.3 **8 / 8**

✓ **- 0 pts** *Entirely correct*

- 2 pts Correct results for expectation and variance, but no derivations are shown.

- 4 pts Partial credit for Problem 1.3

- 8 pts No work shown for Problem 1.3

Problem 2

(a)

We would expected the training RSS for cubic regression to be smaller.
Since it is given that the underdyling relationship between X and Y is linear, using cubic regression is likely to overfit the data, which results in a lower RSS during train time (but most likely higher in test time due to poor generalization)

(b)

We would expected the testing RSS for cubic regression to be higher.
Since cubic regression is likely to overfit the data, it is more likely that the model will perform poorly on test data, leading to a higher testing RSS.

(c)

We would expected the training RSS for cubic regression to be smaller.
By the same argument, we expect the cubic regression model to fit the training data better since it has more parameters.

(d)

There is not enough information to tell.
If the true relationship between X and Y is roughly linear, then linear regression model would more likely have a lower test RSS. However, if the true relationship between X and Y is nonlinear, then cubic regression might give a lower RSS.

2.1 2.1 **3 / 3**

✓ **- 0 pts** *Entirely correct*

- 0.5 pts Correct, but needs justification as to why.

- 1.5 pts Partial credit for Q2.1

- 3 pts No work shown for Q2.1

Problem 2

(a)

We would expected the training RSS for cubic regression to be smaller.
Since it is given that the underdyling relationship between X and Y is linear, using cubic regression is likely to overfit the data, which results in a lower RSS during train time (but most likely higher in test time due to poor generalization)

(b)

We would expected the testing RSS for cubic regression to be higher.
Since cubic regression is likely to overfit the data, it is more likely that the model will perform poorly on test data, leading to a higher testing RSS.

(c)

We would expected the training RSS for cubic regression to be smaller.
By the same argument, we expect the cubic regression model to fit the training data better since it has more parameters.

(d)

There is not enough information to tell.
If the true relationship between X and Y is roughly linear, then linear regression model would more likely have a lower test RSS. However, if the true relationship between X and Y is nonlinear, then cubic regression might give a lower RSS.

2.2 2.2 **3 / 3**

✓ **- 0 pts** *Entirely correct*

- 0.5 pts Correct, but justification is needed.

- 1.5 pts Partial credit for Problem 2.2

- 3 pts No work shown for Problem 2.2

Problem 2

(a)

We would expected the training RSS for cubic regression to be smaller.
Since it is given that the underdyling relationship between X and Y is linear, using cubic regression is likely to overfit the data, which results in a lower RSS during train time (but most likely higher in test time due to poor generalization)

(b)

We would expected the testing RSS for cubic regression to be higher.
Since cubic regression is likely to overfit the data, it is more likely that the model will perform poorly on test data, leading to a higher testing RSS.

(c)

We would expected the training RSS for cubic regression to be smaller.
By the same argument, we expect the cubic regression model to fit the training data better since it has more parameters.

(d)

There is not enough information to tell.
If the true relationship between X and Y is roughly linear, then linear regression model would more likely have a lower test RSS. However, if the true relationship between X and Y is nonlinear, then cubic regression might give a lower RSS.

2.3 2.3 **3 / 3**

✓ **- 0 pts** *Entirely correct*

- 1.5 pts Partial credit for Problem 2.3

- 3 pts No work shown for Problem 2.3

Problem 2

(a)

We would expected the training RSS for cubic regression to be smaller.
Since it is given that the underdyling relationship between X and Y is linear, using cubic regression is likely to overfit the data, which results in a lower RSS during train time (but most likely higher in test time due to poor generalization)

(b)

We would expected the testing RSS for cubic regression to be higher.
Since cubic regression is likely to overfit the data, it is more likely that the model will perform poorly on test data, leading to a higher testing RSS.

(c)

We would expected the training RSS for cubic regression to be smaller.
By the same argument, we expect the cubic regression model to fit the training data better since it has more parameters.

(d)

There is not enough information to tell.
If the true relationship between X and Y is roughly linear, then linear regression model would more likely have a lower test RSS. However, if the true relationship between X and Y is nonlinear, then cubic regression might give a lower RSS.

2.4 2.4 **3 / 3**

✓ **- 0 pts** *Entirely correct*

- 1.5 pts Partial credit for Problem 2.4

- 3 pts No work shown for Problem 2.4

- 0.5 pts Correct, but more precise reasoning is needed.

Problem 3

(a)

Let X be the number of type I errors, then $X \sim \text{Bin}(m, \alpha)$. Therefore

$$\mathbb{E}[X] = m\alpha$$

(b)

We want to find the probability of making one or more false discoveries.
Let E_i be the event of making an error on ith hypothesis, then we are interested in

$$\mathbb{P}\{\cup_{i=1}^{m} E_i\} = 1 - \mathbb{P}\{\cap_{i=1}^{m} E_i^c\} = 1 - (1-\alpha)^m$$

Hence, the family wise error rate is given by $1 - (1-\alpha)^m$

3.1 3.1 **2 / 2**

✓ **- 0 pts** *Entirely correct*

- 0.5 pts Correct, but no precise justification provided

- 1 pts Partial credit for Problem 3.1

- 2 pts No work shown for Problem 3.1

Problem 3

(a)

Let X be the number of type I errors, then $X \sim \text{Bin}(m, \alpha)$. Therefore

$$\mathbb{E}[X] = m\alpha$$

(b)

We want to find the probability of making one or more false discoveries.
Let E_i be the event of making an error on ith hypothesis, then we are interested in

$$\mathbb{P}\{\cup_{i=1}^{m} E_i\} = 1 - \mathbb{P}\{\cap_{i=1}^{m} E_i^c\} = 1 - (1-\alpha)^m$$

Hence, the family wise error rate is given by $1 - (1-\alpha)^m$

3.2 3.2 **3 / 3**

✓ **- 0 pts** *Entirely correct*

- 0.5 pts Correct, but more precise justification is needed.

- 1.5 pts Partial credit for Problem 3.2

- 3 pts No work shown for Problem 3.2

Problem 4

(a)

Note that

$$\begin{aligned} \text{RSS} &= ||(I-P)Y||_2^2 \\ &= Y^T(I-P)Y \end{aligned}$$

Where

$$P = X(X^TX)^{-1}X^T$$

We can further decompose the above expression to get

$$\begin{aligned} \text{RSS} &= Y^T(I-P)Y \\ &= (X\beta+\epsilon)^T(I-P)(X\beta+\epsilon) \end{aligned}$$

Since

$$\begin{aligned} X^T(I-P) &= X^T - X^TXX(X^TX)^{-1}X^T \\ &= X^T - X^T = 0 \end{aligned}$$

And

$$\begin{aligned} (I-P)X &= X - XX(X^TX)^{-1}X^TX \\ &= X - X = 0 \end{aligned}$$

We have

$$\begin{aligned} \text{RSS} &= (X\beta+\epsilon)^T(I-P)(X\beta+\epsilon) \\ &= \epsilon^T(I-P)\epsilon \end{aligned}$$

Dividing both sides by σ^2 gives us

$$\frac{\text{RSS}}{\sigma^2} = \left(\frac{\epsilon}{\sigma}\right)^T(I-P)\left(\frac{\epsilon}{\sigma}\right) \qquad (*)$$

With that $\epsilon/\sigma \sim N(0,1)$.

Since $I-P$ is symmetric and idempotent, it can be diagonalized and hence under a suitable orthonormal basis, $(*)$ can be written as the linear combination of squared normal distribution (chi squared).

Therefore, we see that $(*)$ is a chi-squared distribution with parameter being the rank of

$I - P$.

$$\begin{aligned}\text{rank}(I-P) &= \text{trace}(I-P)\\ &= n - \text{trace}(P)\\ &= n - \text{rank}(P)\\ &= n - p - 1\end{aligned}$$

Hence,

$$\frac{\text{RSS}}{\sigma^2} \sim \chi^2_{n-p-1}$$

(b)

Note that

$$\begin{aligned}\sum_{i=1}^{N}(y_i - \beta^T x_i)^2 &= (Y - X\beta)^T(Y - X\beta)\\ &= (X\beta + \epsilon - X\beta)^T(X\beta + \epsilon - X\beta)\\ &= \epsilon^T\epsilon\end{aligned}$$

Hence

$$\Delta = \epsilon^T P \epsilon \implies \frac{\Delta}{\sigma^2} = \left(\frac{\epsilon}{\sigma}\right)^T P \left(\frac{\epsilon}{\sigma}\right)$$

By the same argument, Δ/σ^2 is a chi square distribution with parameter being the rank of P. Since $\text{rank}(P) = p + 1$, we have

$$\frac{\Delta}{\sigma^2} \sim \chi^2_{p+1}$$

(c)

Since A is symmetric, it can be diagonalized as $A = V\Lambda V^T$.

Letting Λ^* be Λ with non-zero eigenvalues replaced as its inverse, we get a new matrix $A^* = V\Lambda V^*$ satisfying the property

$$A = A^T A^* A$$

This means that we can decompose Q into

$$QX^T AX = C^T A^T A^* AX = (AX)^T A^*(AX)$$

Hence, it suffice proving that the AX and BX are independent.

4.1 4.1 **3 / 3**

✓ **- 0 pts** *Entirely correct*

- 1.5 pts Partial credit for Problem 4.1

- 3 pts No work shown for Problem 4.1

$I - P$.

$$\begin{aligned}\text{rank}(I-P) &= \text{trace}(I-P)\\ &= n - \text{trace}(P)\\ &= n - \text{rank}(P)\\ &= n - p - 1\end{aligned}$$

Hence,

$$\frac{\text{RSS}}{\sigma^2} \sim \chi^2_{n-p-1}$$

(b)

Note that

$$\begin{aligned}\sum_{i=1}^{N}(y_i - \beta^T x_i)^2 &= (Y - X\beta)^T(Y - X\beta)\\ &= (X\beta + \epsilon - X\beta)^T(X\beta + \epsilon - X\beta)\\ &= \epsilon^T\epsilon\end{aligned}$$

Hence

$$\Delta = \epsilon^T P \epsilon \implies \frac{\Delta}{\sigma^2} = \left(\frac{\epsilon}{\sigma}\right)^T P \left(\frac{\epsilon}{\sigma}\right)$$

By the same argument, Δ/σ^2 is a chi square distribution with parameter being the rank of P. Since $\text{rank}(P) = p + 1$, we have

$$\frac{\Delta}{\sigma^2} \sim \chi^2_{p+1}$$

(c)

Since A is symmetric, it can be diagonalized as $A = V\Lambda V^T$.

Letting Λ^* be Λ with non-zero eigenvalues replaced as its inverse, we get a new matrix $A^* = V\Lambda V^*$ satisfying the property

$$A = A^T A^* A$$

This means that we can decompose Q into

$$QX^T AX = C^T A^T A^* AX = (AX)^T A^*(AX)$$

Hence, it suffice proving that the AX and BX are independent.

4.2 4.2 **3 / 3**

✓ **- 0 pts** *Entirely correct*

- 1.5 pts Partial work for Problem 4.2

- 3 pts No work shown for Problem 4.2

$I - P$.

$$\begin{aligned}\text{rank}(I-P) &= \text{trace}(I-P)\\ &= n - \text{trace}(P)\\ &= n - \text{rank}(P)\\ &= n - p - 1\end{aligned}$$

Hence,

$$\frac{\text{RSS}}{\sigma^2} \sim \chi^2_{n-p-1}$$

(b)

Note that

$$\begin{aligned}\sum_{i=1}^{N}(y_i - \beta^T x_i)^2 &= (Y - X\beta)^T(Y - X\beta)\\ &= (X\beta + \epsilon - X\beta)^T(X\beta + \epsilon - X\beta)\\ &= \epsilon^T\epsilon\end{aligned}$$

Hence

$$\Delta = \epsilon^T P \epsilon \implies \frac{\Delta}{\sigma^2} = \left(\frac{\epsilon}{\sigma}\right)^T P \left(\frac{\epsilon}{\sigma}\right)$$

By the same argument, Δ/σ^2 is a chi square distribution with parameter being the rank of P. Since $\text{rank}(P) = p + 1$, we have

$$\frac{\Delta}{\sigma^2} \sim \chi^2_{p+1}$$

(c)

Since A is symmetric, it can be diagonalized as $A = V\Lambda V^T$.

Letting Λ^* be Λ with non-zero eigenvalues replaced as its inverse, we get a new matrix $A^* = V\Lambda V^*$ satisfying the property

$$A = A^T A^* A$$

This means that we can decompose Q into

$$QX^T AX = C^T A^T A^* AX = (AX)^T A^*(AX)$$

Hence, it suffice proving that the AX and BX are independent.

Since $BA = 0$, we have

$$\begin{aligned}\text{Cov}(AX, BX) &= \mathbb{E}[(AX - \mathbb{E}[AX])(BX - \mathbb{E}[BX])^T] \\ &= A\mathbb{E}[(X - \mathbb{E}[X])(X - \mathbb{E}[X])]B^T \\ &= AB^T = (BA^T)^T = (BA)^T = 0\end{aligned}$$

This proves that AX, BX are independent. Hence, Q and T is independent.

(d)

1. $\hat{\beta}$ and RSS are independent:
Note that

$$\text{RSS} = \epsilon^T(I - P)\epsilon$$

And

$$\begin{aligned}\hat{\beta} &= (X^TX)^{-1}X^TY \\ &= (X^TX)^{-1}X^T(X\beta + \epsilon) \\ &= \beta + (X^TX)^{-1}X^T\epsilon\end{aligned}$$

Letting $A = (I - P), B = (X^TX)^{-1}X^T$, since

$$\begin{aligned}BA &= (X^TX)^{-1}X^T(I - P) \\ &= (X^TX)^{-1}X^T - (X^TX)^{-1}X^TX(X^TX)^{-1}X^T \\ &= 0\end{aligned}$$

We see that RSS and $\hat{\beta} - \beta$ are independent. Since independence is invariant under translation, we see that $\hat{\beta}$ and RSS are independent.

2. RSS and Δ are independent:
Note that

$$\Delta = \epsilon^T P\epsilon$$

Since $(I - P)P = 0$, it follows they are orthogonal.
Since RSS and Δ are both quadratic forms of orthogonal matrix, it follows that they are independent.

(e)

Since $\hat{\beta}_i \sim N(\beta_i, \sigma^2(X^TX)^{-1}_{ii})$, by standardizing, we have

$$\frac{\hat{\beta}_i - \beta_i}{\sqrt{\sigma^2(X^TX)^{-1}_{ii}}} \sim N(0, 1)$$

4.3 4.3 **4 / 4**

✓ **- 0 pts** *Entirely correct*

- 2 pts Partial credit for Problem 4.3

- 4 pts No work shown for Problem 4.3

Since $BA = 0$, we have

$$\begin{aligned}\text{Cov}(AX, BX) &= \mathbb{E}[(AX - \mathbb{E}[AX])(BX - \mathbb{E}[BX])^T] \\ &= A\mathbb{E}[(X - \mathbb{E}[X])(X - \mathbb{E}[X])]B^T \\ &= AB^T = (BA^T)^T = (BA)^T = 0\end{aligned}$$

This proves that AX, BX are independent. Hence, Q and T is independent.

(d)

1. $\hat{\beta}$ and RSS are independent:
Note that

$$\text{RSS} = \epsilon^T(I - P)\epsilon$$

And

$$\begin{aligned}\hat{\beta} &= (X^TX)^{-1}X^TY \\ &= (X^TX)^{-1}X^T(X\beta + \epsilon) \\ &= \beta + (X^TX)^{-1}X^T\epsilon\end{aligned}$$

Letting $A = (I - P), B = (X^TX)^{-1}X^T$, since

$$\begin{aligned}BA &= (X^TX)^{-1}X^T(I - P) \\ &= (X^TX)^{-1}X^T - (X^TX)^{-1}X^TX(X^TX)^{-1}X^T \\ &= 0\end{aligned}$$

We see that RSS and $\hat{\beta} - \beta$ are independent. Since independence is invariant under translation, we see that $\hat{\beta}$ and RSS are independent.

2. RSS and Δ are independent:
Note that

$$\Delta = \epsilon^T P\epsilon$$

Since $(I - P)P = 0$, it follows they are orthogonal.
Since RSS and Δ are both quadratic forms of orthogonal matrix, it follows that they are independent.

(e)

Since $\hat{\beta}_i \sim N(\beta_i, \sigma^2(X^TX)^{-1}_{ii})$, by standardizing, we have

$$\frac{\hat{\beta}_i - \beta_i}{\sqrt{\sigma^2(X^TX)^{-1}_{ii}}} \sim N(0, 1)$$

4.4 4.4 **4 / 4**

✓ **- 0 pts** *Entirely correct*

- 2 pts Partial credit for Problem 4.4

- 4 pts No work shown for Problem 4.4

Since $BA = 0$, we have

$$\begin{aligned}\text{Cov}(AX, BX) &= \mathbb{E}[(AX - \mathbb{E}[AX])(BX - \mathbb{E}[BX])^T] \\ &= A\mathbb{E}[(X - \mathbb{E}[X])(X - \mathbb{E}[X])]B^T \\ &= AB^T = (BA^T)^T = (BA)^T = 0\end{aligned}$$

This proves that AX, BX are independent. Hence, Q and T is independent.

(d)

1. $\hat{\beta}$ and RSS are independent:
Note that

$$\text{RSS} = \epsilon^T(I - P)\epsilon$$

And

$$\begin{aligned}\hat{\beta} &= (X^TX)^{-1}X^TY \\ &= (X^TX)^{-1}X^T(X\beta + \epsilon) \\ &= \beta + (X^TX)^{-1}X^T\epsilon\end{aligned}$$

Letting $A = (I - P), B = (X^TX)^{-1}X^T$, since

$$\begin{aligned}BA &= (X^TX)^{-1}X^T(I - P) \\ &= (X^TX)^{-1}X^T - (X^TX)^{-1}X^TX(X^TX)^{-1}X^T \\ &= 0\end{aligned}$$

We see that RSS and $\hat{\beta} - \beta$ are independent. Since independence is invariant under translation, we see that $\hat{\beta}$ and RSS are independent.

2. RSS and Δ are independent:
Note that

$$\Delta = \epsilon^T P\epsilon$$

Since $(I - P)P = 0$, it follows they are orthogonal.
Since RSS and Δ are both quadratic forms of orthogonal matrix, it follows that they are independent.

(e)

Since $\hat{\beta}_i \sim N(\beta_i, \sigma^2(X^TX)^{-1}_{ii})$, by standardizing, we have

$$\frac{\hat{\beta}_i - \beta_i}{\sqrt{\sigma^2(X^TX)^{-1}_{ii}}} \sim N(0, 1)$$

By the definition of standard error

$$s^2 = \frac{\text{RSS}}{n-p-1} \implies \frac{s^2(n-p-1)}{\sigma^2} \sim \chi^2_{n-p-1}$$

Hence,

$$\begin{aligned}\frac{\hat{\beta}_i - \beta_i}{\sqrt{s^2(X^TX)^{-1}_{ii}}} &= \frac{N(0,1)}{\sqrt{\frac{s^2}{\sigma^2}}} \\ &= \frac{N(0,1)}{\sqrt{\frac{\chi^2_{n-p-1}}{n-p-1}}} = t_{n-p-1}\end{aligned}$$

4.5 4.5 **4 / 4**

✓ **- 0 pts** *Entirely correct*

- 2 pts Partial work for Problem 4.5

- 4 pts No work shown for Problem 4.5

By the definition of standard error

$$s^2 = \frac{\text{RSS}}{n-p-1} \implies \frac{s^2(n-p-1)}{\sigma^2} \sim \chi^2_{n-p-1}$$

Hence,

$$\begin{aligned}\frac{\hat{\beta}_i - \beta_i}{\sqrt{s^2(X^TX)^{-1}_{ii}}} &= \frac{N(0,1)}{\sqrt{\frac{s^2}{\sigma^2}}} \\ &= \frac{N(0,1)}{\sqrt{\frac{\chi^2_{n-p-1}}{n-p-1}}} = t_{n-p-1}\end{aligned}$$

4.6 4.6 (M) **0 / 8**

- **0 pts** Entirely correct
- **2 pts** Correct approach, but slight error in derivation
- **4 pts** Partial credit for Problem 4.6
- **8 pts** No work shown for Problem 4.6

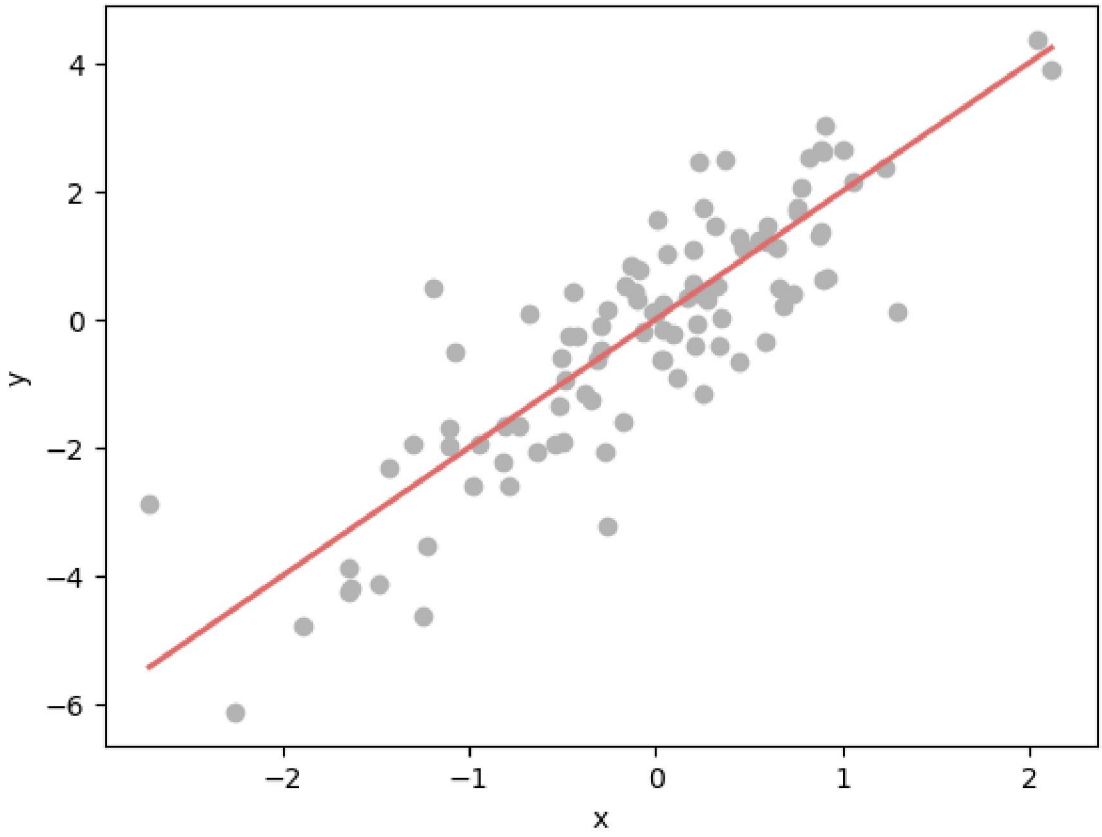
✓ **- 8 pts** *Question is not intended for undergrads. Your total score is out of 63 points instead of 71 points.*

In [1]:
```
import math
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
```

Problem 5

In [2]:
```
# Generating random data
rng = np.random.default_rng(1)
x = rng.normal(size=100)
y = 2 * x + rng.normal(size=100)
```

In [3]:
```
# Plot a scatter plot of x and y
plt.scatter(x, y, color="#B3B3B4")
x_arr = np.linspace(min(x), max(x), 100)
y_true = 2 * x_arr
plt.plot(x_arr, y_true, color='#E16A6A', lw=2)
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



Part a.

In [4]:
```
# Fit ordinary least squares model without intercept of x onto y
model = sm.OLS(endog=y, exog=x)
result = model.fit()
```

In [5]:
```
# Summary of the results
print(result.summary())
```

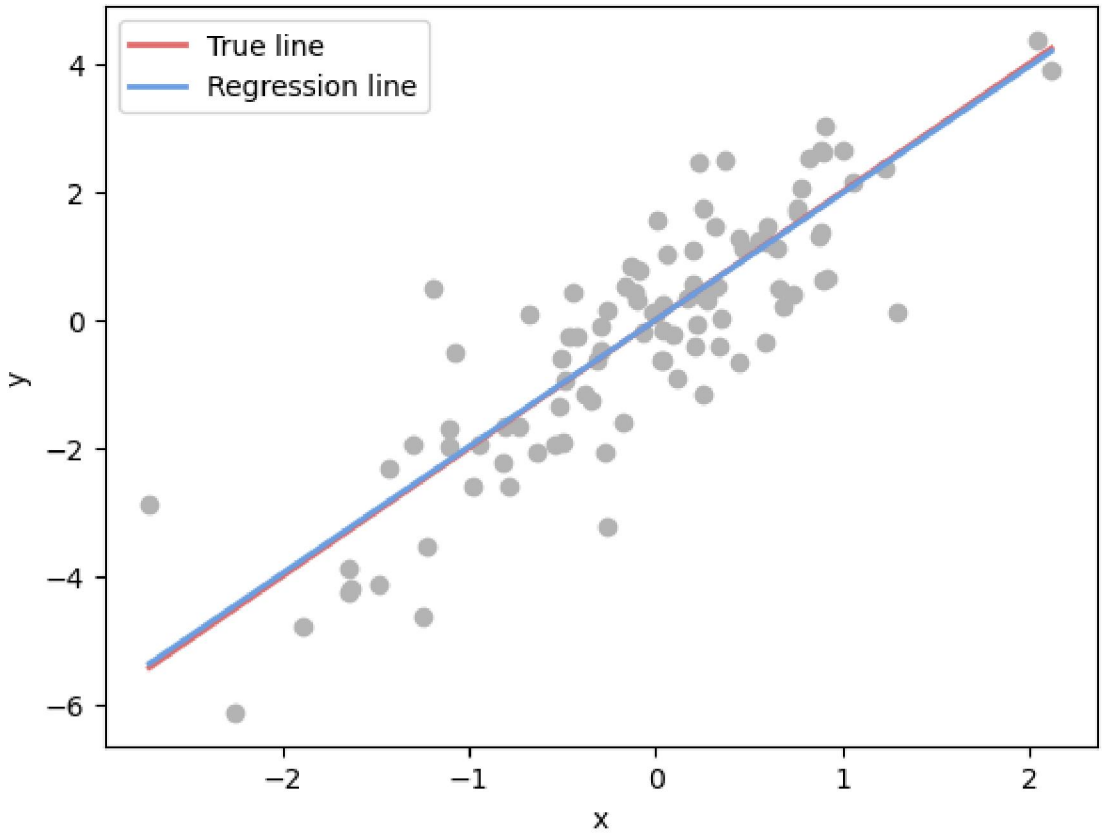
```
                                 OLS Regression Results
```

```
================================================================================
Dep. Variable:                      y   R-squared (uncentered):                   0.743
Model:                            OLS   Adj. R-squared (uncentered):              0.740
Method:                 Least Squares   F-statistic:                              285.6
Date:                Tue, 26 Sep 2023   Prob (F-statistic):                    6.23e-31
Time:                        16:01:33   Log-Likelihood:                         -141.35
No. Observations:                 100   AIC:                                      284.7
Df Residuals:                      99   BIC:                                      287.3
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1             1.9762      0.117     16.898      0.000       1.744       2.208
==============================================================================
Omnibus:                        1.376   Durbin-Watson:                   2.184
Prob(Omnibus):                  0.503   Jarque-Bera (JB):                0.847
Skew:                           0.121   Prob(JB):                        0.655
Kurtosis:                       3.381   Cond. No.                         1.00
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not contain a con
stant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specifi
ed.
```

The coefficient estimate is $\beta = 1.9762$, with standard error of 0.117.

The t-statistics is given by 16.898, and the p-value is given by 0.000, which is smaller than any reasonable significance level. Hence, we reject the null hypothesis.

In [6]:
```python
# Plot a scatter plot of x and y with the regression line
plt.scatter(x, y, color="#B3B3B4")
x_arr = np.linspace(min(x), max(x), 100)
y_true, y_reg = 2 * x_arr, 1.9762 * x_arr
plt.plot(x_arr, y_true, color='#E16A6A', lw=2, label='True line')
plt.plot(x_arr, y_reg, color='#6A9EE1', lw=2, label='Regression line')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.show()
```



From the above graph, we see that the slope is clearly non-zero. Therefore, intuitively we should expect the p-value to be small.

Part b.

In [7]:
```
# Fit ordinary least squares model without intercept of y onto x
model = sm.OLS(endog=x, exog=y)
result = model.fit()
```

In [8]:
```
# Summary of the results
print(result.summary())
```

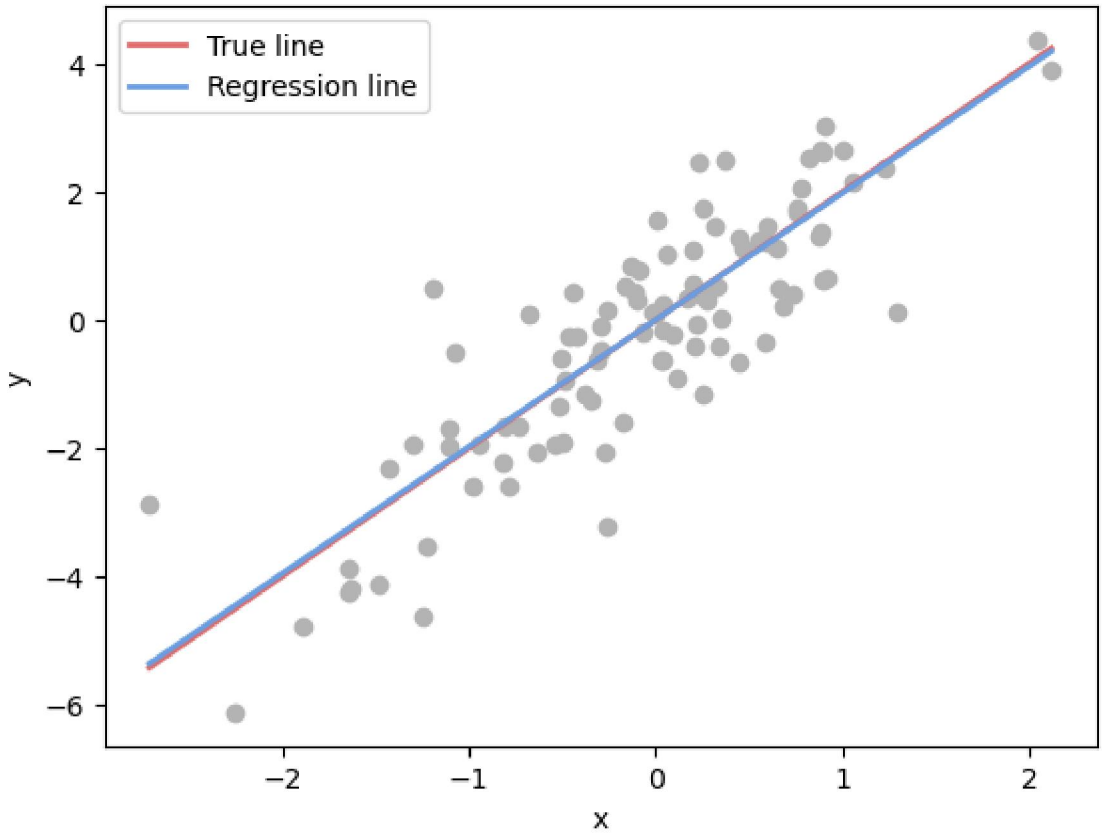
```
                                 OLS Regression Results
=======================================================================================
Dep. Variable:                          y   R-squared (uncentered):                   0.743
Model:                                OLS   Adj. R-squared (uncentered):              0.740
Method:                     Least Squares   F-statistic:                              285.6
Date:                    Tue, 26 Sep 2023   Prob (F-statistic):                    6.23e-31
Time:                            16:01:36   Log-Likelihood:                         -58.349
No. Observations:                     100   AIC:                                      118.7
Df Residuals:                          99   BIC:                                      121.3
Df Model:                               1
Covariance Type:                nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1             0.3757      0.022     16.898      0.000       0.332       0.420
==============================================================================
Omnibus:                       13.156   Durbin-Watson:                   2.034
Prob(Omnibus):                  0.001   Jarque-Bera (JB):               22.596
Skew:                          -0.528   Prob(JB):                     1.24e-05
Kurtosis:                       5.075   Cond. No.                         1.00
==============================================================================

Notes:
```

5.1 3.7.11a **1 / 1**

✓ **- 0 pts** *Entirely correct*

- 0.5 pts Partial credit for Problem 5.1

- 1 pts No work shown for Problem 5.1



From the above graph, we see that the slope is clearly non-zero. Therefore, intuitively we should expect the p-value to be small.

Part b.

In [7]:
```
# Fit ordinary least squares model without intercept of y onto x
model = sm.OLS(endog=x, exog=y)
result = model.fit()
```

In [8]:
```
# Summary of the results
print(result.summary())
```

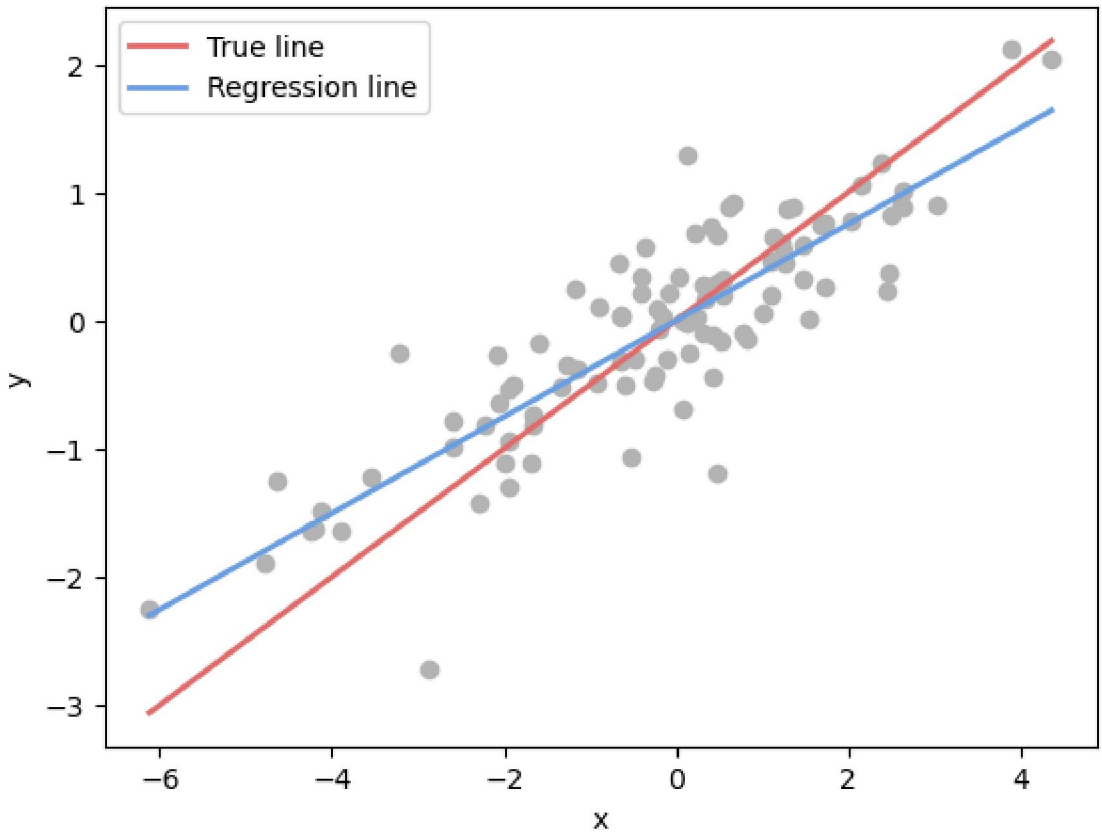
```
                                 OLS Regression Results
=======================================================================================
Dep. Variable:                          y   R-squared (uncentered):                   0.743
Model:                                OLS   Adj. R-squared (uncentered):              0.740
Method:                     Least Squares   F-statistic:                              285.6
Date:                    Tue, 26 Sep 2023   Prob (F-statistic):                    6.23e-31
Time:                            16:01:36   Log-Likelihood:                         -58.349
No. Observations:                     100   AIC:                                      118.7
Df Residuals:                          99   BIC:                                      121.3
Df Model:                               1
Covariance Type:                nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1             0.3757      0.022     16.898      0.000       0.332       0.420
==============================================================================
Omnibus:                       13.156   Durbin-Watson:                   2.034
Prob(Omnibus):                  0.001   Jarque-Bera (JB):               22.596
Skew:                          -0.528   Prob(JB):                     1.24e-05
Kurtosis:                       5.075   Cond. No.                         1.00
==============================================================================

Notes:
```

```
[1] R² is computed without centering (uncentered) since the model does not contain a con
stant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specifi
ed.
```

The coefficient estimate is $\beta = 0.3757$, with standard error of 0.022.

The t-statistics is given by 16.898, and the p-value is given by 0.000, which is smaller than any reasonable significance level. Hence, we reject the null hypothesis.

In [9]:
```
# Plot a scatter plot of y and x with the regression line
plt.scatter(y, x, color="#B3B3B4")
y_arr = np.linspace(min(y), max(y), 100)
x_true, x_reg = 0.5 * y_arr, 0.3757 * y_arr
plt.plot(y_arr, x_true, color='#E16A6A', lw=2, label='True line')
plt.plot(y_arr, x_reg, color='#6A9EE1', lw=2, label='Regression line')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.show()
```



From the above graph, we see that the slope is clearly non-zero. Therefore, intuitively we should expect the p-value to be small. However, note that the OLS estimate is far from the true value, even though visually it seems to be a good fit

Part c.

Theoretically, if β is the slope estimated from part (a), then the slope estimated from part (b) should be around $1/\beta$. However, this is not true in our case since the OLS estimate differs significantly in part (b). </br> </br> Also, note that the t-statistics and p-value for both problem happen to be the same, we will justify this in part (d) and (e).

5.2 3.7.11b **1 / 1**

✓ **- 0 pts** *Entirely correct*

- 0.5 pts Partial credit for Problem 5.2

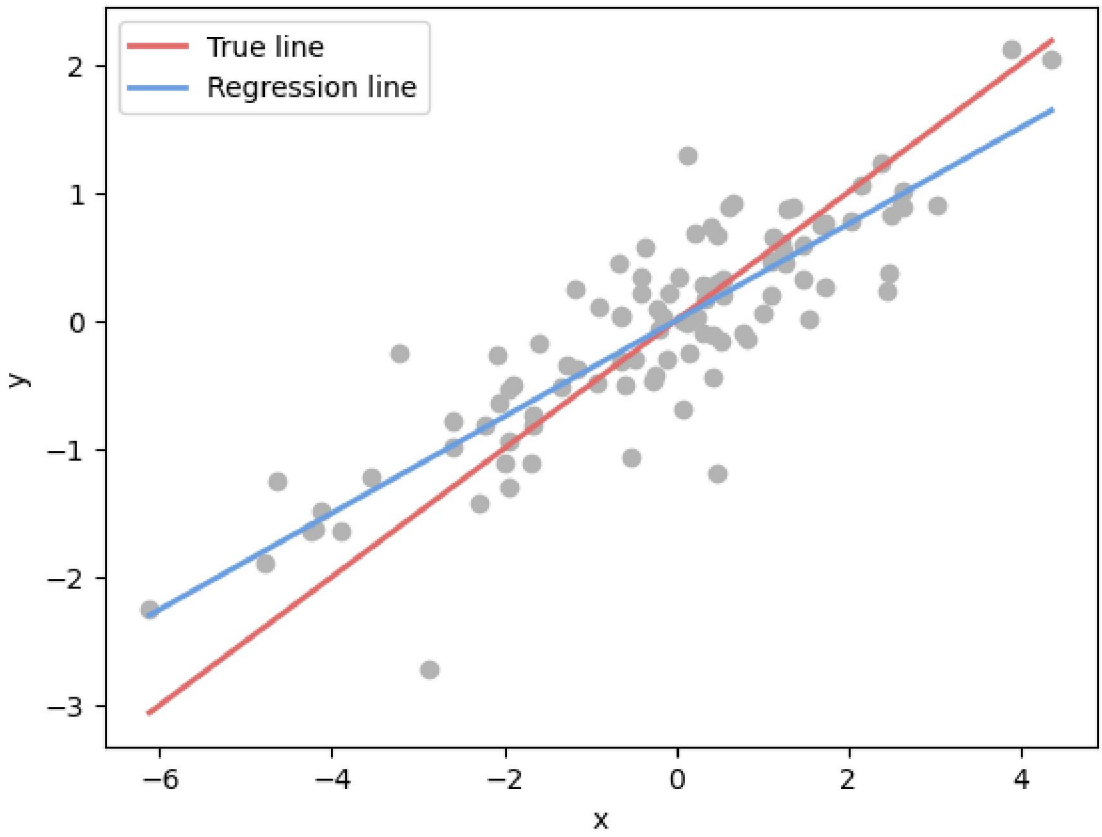
- 1 pts No work shown for Problem 5.2

```
[1] R² is computed without centering (uncentered) since the model does not contain a con
stant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specifi
ed.
```

The coefficient estimate is $\beta = 0.3757$, with standard error of 0.022.

The t-statistics is given by 16.898, and the p-value is given by 0.000, which is smaller than any reasonable significance level. Hence, we reject the null hypothesis.

In [9]:
```
# Plot a scatter plot of y and x with the regression line
plt.scatter(y, x, color="#B3B3B4")
y_arr = np.linspace(min(y), max(y), 100)
x_true, x_reg = 0.5 * y_arr, 0.3757 * y_arr
plt.plot(y_arr, x_true, color='#E16A6A', lw=2, label='True line')
plt.plot(y_arr, x_reg, color='#6A9EE1', lw=2, label='Regression line')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.show()
```



From the above graph, we see that the slope is clearly non-zero. Therefore, intuitively we should expect the p-value to be small. However, note that the OLS estimate is far from the true value, even though visually it seems to be a good fit

Part c.

Theoretically, if β is the slope estimated from part (a), then the slope estimated from part (b) should be around $1/\beta$. However, this is not true in our case since the OLS estimate differs significantly in part (b). </br> </br> Also, note that the t-statistics and p-value for both problem happen to be the same, we will justify this in part (d) and (e).

Part d.

We have

$$
\begin{aligned}
t &= \frac{\sum_i x_i y_i / \sum_i x_i^2}{\sqrt{\sum_i (y_i - x_i \hat{\beta}^2)^2 / (n-1) \sum_i x_i^2}} \\
&= \frac{\sqrt{n-1} \sum_i x_i y_i}{\sqrt{\sum_i x_i^2} \sqrt{\sum_i (y_i - \beta x_i)^2}} \\
&= \frac{\sqrt{n-1} \sum_i x_i y_i}{\sqrt{\sum_i x_i^2} \sqrt{\sum_i (y_i - \frac{\sum_j x_j y_j}{\sum_j x_j^2} x_i)^2}} \\
&= \frac{\sqrt{n-1} \sum_i x_i y_i}{\sqrt{\sum_i x_i^2} \sqrt{\sum_i y_i^2 - \frac{(\sum_i x_i y_i)^2}{\sum_i x_i^2}}} \\
&= \frac{\sqrt{n-1} \sum_i x_i y_i}{\sqrt{\sum_i x_i^2 \sum_i y_i^2 - (\sum_i x_i y_i)^2}}
\end{aligned}
$$

We now show this numerically.

In [10]:
```
# Verifying the formula for t-value numerically
n = len(x)
xx, xy, yy = np.dot(x, x), np.dot(x, y), np.dot(y, y)
t = (n - 1) ** 0.5 * xy / (xx * yy - xy ** 2) ** 0.5
t
```

Out[10]:
```
16.8984170630351
```

Which is the same t-value we computed earlier.

Part e.

Since the formula in part (d) is symmetric, it follows that the t-statistics would be the same if we reverse x and y.

Part f.

In [11]:
```
# Perform linear regression of y onto x with intercept
x_withIntercept = sm.add_constant(x)
model = sm.OLS(y, x_withIntercept)
result = model.fit()
print(result.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.741
Model:                            OLS   Adj. R-squared:                  0.738
Method:                 Least Squares   F-statistic:                     280.0
Date:                Tue, 26 Sep 2023   Prob (F-statistic):           1.74e-30
Time:                        16:01:39   Log-Likelihood:                -141.06
No. Observations:                 100   AIC:                             286.1
Df Residuals:                      98   BIC:                             291.3
```

5.3 3.7.11c **1 / 1**

✓ **- 0 pts** *Entirely correct*

- 0.5 pts Partial credit for Problem 5.3

- 1 pts No work shown for Problem 5.3

Part d.

We have

$$
\begin{aligned}
t &= \frac{\sum_i x_i y_i / \sum_i x_i^2}{\sqrt{\sum_i (y_i - x_i \hat{\beta}^2)^2 / (n-1) \sum_i x_i^2}} \\
&= \frac{\sqrt{n-1} \sum_i x_i y_i}{\sqrt{\sum_i x_i^2} \sqrt{\sum_i (y_i - \beta x_i)^2}} \\
&= \frac{\sqrt{n-1} \sum_i x_i y_i}{\sqrt{\sum_i x_i^2} \sqrt{\sum_i (y_i - \frac{\sum_j x_j y_j}{\sum_j x_j^2} x_i)^2}} \\
&= \frac{\sqrt{n-1} \sum_i x_i y_i}{\sqrt{\sum_i x_i^2} \sqrt{\sum_i y_i^2 - \frac{(\sum_i x_i y_i)^2}{\sum_i x_i^2}}} \\
&= \frac{\sqrt{n-1} \sum_i x_i y_i}{\sqrt{\sum_i x_i^2 \sum_i y_i^2 - (\sum_i x_i y_i)^2}}
\end{aligned}
$$

We now show this numerically.

In [10]:
```
# Verifying the formula for t-value numerically
n = len(x)
xx, xy, yy = np.dot(x, x), np.dot(x, y), np.dot(y, y)
t = (n - 1) ** 0.5 * xy / (xx * yy - xy ** 2) ** 0.5
t
```

Out[10]:
```
16.8984170630351
```

Which is the same t-value we computed earlier.

Part e.

Since the formula in part (d) is symmetric, it follows that the t-statistics would be the same if we reverse x and y.

Part f.

In [11]:
```
# Perform linear regression of y onto x with intercept
x_withIntercept = sm.add_constant(x)
model = sm.OLS(y, x_withIntercept)
result = model.fit()
print(result.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.741
Model:                            OLS   Adj. R-squared:                  0.738
Method:                 Least Squares   F-statistic:                     280.0
Date:                Tue, 26 Sep 2023   Prob (F-statistic):           1.74e-30
Time:                        16:01:39   Log-Likelihood:                -141.06
No. Observations:                 100   AIC:                             286.1
Df Residuals:                      98   BIC:                             291.3
```

5.4 3.7.11d **1 / 1**

✓ **- 0 pts** *Entirely correct*

- 0.5 pts Partial credit for Problem 5.4

- 1 pts No work shown for Problem 5.4

Part d.

We have

$$
\begin{aligned}
t &= \frac{\sum_i x_i y_i / \sum_i x_i^2}{\sqrt{\sum_i (y_i - x_i \hat{\beta}^2)^2 / (n-1) \sum_i x_i^2}} \\
&= \frac{\sqrt{n-1} \sum_i x_i y_i}{\sqrt{\sum_i x_i^2} \sqrt{\sum_i (y_i - \beta x_i)^2}} \\
&= \frac{\sqrt{n-1} \sum_i x_i y_i}{\sqrt{\sum_i x_i^2} \sqrt{\sum_i (y_i - \frac{\sum_j x_j y_j}{\sum_j x_j^2} x_i)^2}} \\
&= \frac{\sqrt{n-1} \sum_i x_i y_i}{\sqrt{\sum_i x_i^2} \sqrt{\sum_i y_i^2 - \frac{(\sum_i x_i y_i)^2}{\sum_i x_i^2}}} \\
&= \frac{\sqrt{n-1} \sum_i x_i y_i}{\sqrt{\sum_i x_i^2 \sum_i y_i^2 - (\sum_i x_i y_i)^2}}
\end{aligned}
$$

We now show this numerically.

In [10]:
```
# Verifying the formula for t-value numerically
n = len(x)
xx, xy, yy = np.dot(x, x), np.dot(x, y), np.dot(y, y)
t = (n - 1) ** 0.5 * xy / (xx * yy - xy ** 2) ** 0.5
t
```

Out[10]:
```
16.8984170630351
```

Which is the same t-value we computed earlier.

Part e.

Since the formula in part (d) is symmetric, it follows that the t-statistics would be the same if we reverse x and y.

Part f.

In [11]:
```
# Perform linear regression of y onto x with intercept
x_withIntercept = sm.add_constant(x)
model = sm.OLS(y, x_withIntercept)
result = model.fit()
print(result.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.741
Model:                            OLS   Adj. R-squared:                  0.738
Method:                 Least Squares   F-statistic:                     280.0
Date:                Tue, 26 Sep 2023   Prob (F-statistic):           1.74e-30
Time:                        16:01:39   Log-Likelihood:                -141.06
No. Observations:                 100   AIC:                             286.1
Df Residuals:                      98   BIC:                             291.3
```

5.5 3.7.11e **1 / 1**

✓ **- 0 pts** *Entirely correct*

- 0.3 pts Showed the t-statistics are equivalent, but didn't indicate why it was the case.

- 0.5 pts Partial credit for Problem 5.5

- 1 pts No work shown for Problem 5.5

Part d.

We have

$$
\begin{aligned}
t &= \frac{\sum_i x_i y_i / \sum_i x_i^2}{\sqrt{\sum_i (y_i - x_i \hat{\beta}^2)^2 / (n-1) \sum_i x_i^2}} \\
&= \frac{\sqrt{n-1} \sum_i x_i y_i}{\sqrt{\sum_i x_i^2} \sqrt{\sum_i (y_i - \beta x_i)^2}} \\
&= \frac{\sqrt{n-1} \sum_i x_i y_i}{\sqrt{\sum_i x_i^2} \sqrt{\sum_i (y_i - \frac{\sum_j x_j y_j}{\sum_j x_j^2} x_i)^2}} \\
&= \frac{\sqrt{n-1} \sum_i x_i y_i}{\sqrt{\sum_i x_i^2} \sqrt{\sum_i y_i^2 - \frac{(\sum_i x_i y_i)^2}{\sum_i x_i^2}}} \\
&= \frac{\sqrt{n-1} \sum_i x_i y_i}{\sqrt{\sum_i x_i^2 \sum_i y_i^2 - (\sum_i x_i y_i)^2}}
\end{aligned}
$$

We now show this numerically.

In [10]:
```
# Verifying the formula for t-value numerically
n = len(x)
xx, xy, yy = np.dot(x, x), np.dot(x, y), np.dot(y, y)
t = (n - 1) ** 0.5 * xy / (xx * yy - xy ** 2) ** 0.5
t
```

Out[10]:
```
16.8984170630351
```

Which is the same t-value we computed earlier.

Part e.

Since the formula in part (d) is symmetric, it follows that the t-statistics would be the same if we reverse x and y.

Part f.

In [11]:
```
# Perform linear regression of y onto x with intercept
x_withIntercept = sm.add_constant(x)
model = sm.OLS(y, x_withIntercept)
result = model.fit()
print(result.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.741
Model:                            OLS   Adj. R-squared:                  0.738
Method:                 Least Squares   F-statistic:                     280.0
Date:                Tue, 26 Sep 2023   Prob (F-statistic):           1.74e-30
Time:                        16:01:39   Log-Likelihood:                -141.06
No. Observations:                 100   AIC:                             286.1
Df Residuals:                      98   BIC:                             291.3
```

```
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0760      0.101     -0.756      0.451      -0.276       0.124
x1             1.9686      0.118     16.734      0.000       1.735       2.202
==============================================================================
Omnibus:                        1.277   Durbin-Watson:                   2.198
Prob(Omnibus):                  0.528   Jarque-Bera (JB):                0.759
Skew:                           0.114   Prob(JB):                        0.684
Kurtosis:                       3.361   Cond. No.                         1.20
==============================================================================

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

In [12]:
```
# Perform linear regression of x onto y with intercept
y_withIntercept = sm.add_constant(y)
model = sm.OLS(x, y_withIntercept)
result = model.fit()
print(result.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.741
Model:                            OLS   Adj. R-squared:                  0.738
Method:                 Least Squares   F-statistic:                     280.0
Date:                Tue, 26 Sep 2023   Prob (F-statistic):           1.74e-30
Time:                        16:01:40   Log-Likelihood:                -58.325
No. Observations:                 100   AIC:                             120.6
Df Residuals:                      98   BIC:                             125.9
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0095      0.044      0.216      0.829      -0.078       0.097
x1             0.3763      0.022     16.734      0.000       0.332       0.421
==============================================================================
Omnibus:                       13.123   Durbin-Watson:                   2.035
Prob(Omnibus):                  0.001   Jarque-Bera (JB):               22.501
Skew:                          -0.528   Prob(JB):                     1.30e-05
Kurtosis:                       5.070   Cond. No.                         1.98
==============================================================================

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

We see that the t-statistics for β_1 for both model is 16.734.

Problem 6.

In [27]:
```
# Importing abalone data
filepath = "/Users/raymondtsao/Desktop/STAT 154/HW 2/abalone.csv"
abalone = pd.read_csv(filepath)
```

In [31]:
```
abalone
```

Out[31]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings

5.6 3.7.11f **1 / 1**

✓ **- 0 pts** *Entirely correct*

- 0.5 pts Partial credit shown for Problem 5.6

- 1 pts No work shown for Problem 5.6

```
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0760      0.101     -0.756      0.451      -0.276       0.124
x1             1.9686      0.118     16.734      0.000       1.735       2.202
==============================================================================
Omnibus:                        1.277   Durbin-Watson:                   2.198
Prob(Omnibus):                  0.528   Jarque-Bera (JB):                0.759
Skew:                           0.114   Prob(JB):                        0.684
Kurtosis:                       3.361   Cond. No.                         1.20
==============================================================================

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

In [12]:
```
# Perform linear regression of x onto y with intercept
y_withIntercept = sm.add_constant(y)
model = sm.OLS(x, y_withIntercept)
result = model.fit()
print(result.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.741
Model:                            OLS   Adj. R-squared:                  0.738
Method:                 Least Squares   F-statistic:                     280.0
Date:                Tue, 26 Sep 2023   Prob (F-statistic):           1.74e-30
Time:                        16:01:40   Log-Likelihood:                -58.325
No. Observations:                 100   AIC:                             120.6
Df Residuals:                      98   BIC:                             125.9
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0095      0.044      0.216      0.829      -0.078       0.097
x1             0.3763      0.022     16.734      0.000       0.332       0.421
==============================================================================
Omnibus:                       13.123   Durbin-Watson:                   2.035
Prob(Omnibus):                  0.001   Jarque-Bera (JB):               22.501
Skew:                          -0.528   Prob(JB):                     1.30e-05
Kurtosis:                       5.070   Cond. No.                         1.98
==============================================================================

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

We see that the t-statistics for β_1 for both model is 16.734.

Problem 6.

In [27]:
```
# Importing abalone data
filepath = "/Users/raymondtsao/Desktop/STAT 154/HW 2/abalone.csv"
abalone = pd.read_csv(filepath)
```

In [31]:
```
abalone
```

Out[31]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings

0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.1500	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550	7
...	...	...	...	...	...	...	...	...	...
4172	F	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490	11
4173	M	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605	10
4174	M	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080	9
4175	F	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960	10
4176	M	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950	12

4177 rows × 9 columns

Part 1.

In [84]:
```
# Filter all rows whose "Sex" is not "M" or "F"
abalone = abalone[(abalone["Sex"] == "M") | (abalone["Sex"] == "F")]

# Remove all columns other than "Sex", "Lenght", and "Diameter"
abalone = abalone[["Sex", "Length", "Diameter"]]
```

In [85]:
```
abalone
```

Out[85]:

	Sex	Length	Diameter
0	M	0.455	0.365
1	M	0.350	0.265
2	F	0.530	0.420
3	M	0.440	0.365
6	F	0.530	0.415
...	...	...	...
4172	F	0.565	0.450
4173	M	0.590	0.440
4174	M	0.600	0.475
4175	F	0.625	0.485
4176	M	0.710	0.555

2835 rows × 3 columns

Part 2.

In [86]:
```
from sklearn.model_selection import train_test_split
```

In [87]:
```
# Split into training and testing set
abalone_train, abalone_test = train_test_split(abalone, test_size=0.5, random_state=0)
```

6.1 6.1 **1 / 1**

✓ **- 0 pts** *Entirely correct*

- 0.5 pts Partial credit for Problem 6.1

- 1 pts No work shown for Problem 6.1

0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.1500	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550	7
...	...	...	...	...	...	...	...	...	...
4172	F	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490	11
4173	M	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605	10
4174	M	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080	9
4175	F	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960	10
4176	M	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950	12

4177 rows × 9 columns

Part 1.

In [84]:
```
# Filter all rows whose "Sex" is not "M" or "F"
abalone = abalone[(abalone["Sex"] == "M") | (abalone["Sex"] == "F")]

# Remove all columns other than "Sex", "Lenght", and "Diameter"
abalone = abalone[["Sex", "Length", "Diameter"]]
```

In [85]:
```
abalone
```

Out[85]:

	Sex	Length	Diameter
0	M	0.455	0.365
1	M	0.350	0.265
2	F	0.530	0.420
3	M	0.440	0.365
6	F	0.530	0.415
...	...	...	...
4172	F	0.565	0.450
4173	M	0.590	0.440
4174	M	0.600	0.475
4175	F	0.625	0.485
4176	M	0.710	0.555

2835 rows × 3 columns

Part 2.

In [86]:
```
from sklearn.model_selection import train_test_split
```

In [87]:
```
# Split into training and testing set
abalone_train, abalone_test = train_test_split(abalone, test_size=0.5, random_state=0)
```

In [88]:
```
abalone_train
```

Out[88]:

	Sex	Length	Diameter
1705	M	0.640	0.525
852	M	0.565	0.450
754	M	0.650	0.515
2265	F	0.720	0.575
2122	F	0.435	0.350
...	...	...	...
1013	F	0.625	0.475
1145	M	0.580	0.455
2403	M	0.290	0.225
3833	M	0.535	0.410
4015	M	0.635	0.480

1417 rows × 3 columns

In [89]:
```
abalone_test
```

Out[89]:

	Sex	Length	Diameter
1405	M	0.655	0.535
877	F	0.635	0.500
231	M	0.565	0.440
1718	M	0.650	0.505
2270	F	0.600	0.475
...	...	...	...
3693	M	0.650	0.495
2989	M	0.560	0.425
3592	M	0.655	0.525
428	F	0.560	0.455
4175	F	0.625	0.485

1418 rows × 3 columns

Part 3.

In [90]:
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis, QuadraticDiscrimin
```

In [91]:
```
# Extracting features and response variables
X_train, y_train = abalone_train[["Length", "Diameter"]], abalone_train["Sex"].values
X_test, y_test = abalone_test[["Length", "Diameter"]], abalone_test["Sex"].values
```

In [92]:
```
# Fitting LDA model
lda = LinearDiscriminantAnalysis()
lda.fit(X_train, y_train)
```

6.2 6.2 **1 / 1**

✓ **- 0 pts** *Entirely correct*

- 0.5 pts Partial credit for Problem 6.2

- 1 pts No work shown for Problem 6.2

In [88]:
```
abalone_train
```

Out[88]:

	Sex	Length	Diameter
1705	M	0.640	0.525
852	M	0.565	0.450
754	M	0.650	0.515
2265	F	0.720	0.575
2122	F	0.435	0.350
...	...	...	...
1013	F	0.625	0.475
1145	M	0.580	0.455
2403	M	0.290	0.225
3833	M	0.535	0.410
4015	M	0.635	0.480

1417 rows × 3 columns

In [89]:
```
abalone_test
```

Out[89]:

	Sex	Length	Diameter
1405	M	0.655	0.535
877	F	0.635	0.500
231	M	0.565	0.440
1718	M	0.650	0.505
2270	F	0.600	0.475
...	...	...	...
3693	M	0.650	0.495
2989	M	0.560	0.425
3592	M	0.655	0.525
428	F	0.560	0.455
4175	F	0.625	0.485

1418 rows × 3 columns

Part 3.

In [90]:
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis, QuadraticDiscrimin
```

In [91]:
```
# Extracting features and response variables
X_train, y_train = abalone_train[["Length", "Diameter"]], abalone_train["Sex"].values
X_test, y_test = abalone_test[["Length", "Diameter"]], abalone_test["Sex"].values
```

In [92]:
```
# Fitting LDA model
lda = LinearDiscriminantAnalysis()
lda.fit(X_train, y_train)
```

Out[92]:
```
▾ LinearDiscriminantAnalysis
LinearDiscriminantAnalysis()
```

In [93]:
```
# Fitting QDA model
qda = QuadraticDiscriminantAnalysis()
qda.fit(X_train, y_train)
```

Out[93]:
```
▾ QuadraticDiscriminantAnalysis
QuadraticDiscriminantAnalysis()
```

Part 4.

We compare the LDA/QDA model prediction with the ground truth using the accuracy metric, which is given by </br>

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of samples}}$$

In [94]:
```
def getAccuracy(model, X_test, y_test):
    predictions = model.predict(X_test)
    numCorrect = np.count_nonzero(predictions == y_test)
    return numCorrect / len(y_test) * 100
```

In [135…
```
# Prediction using LDA and QDA model
lda_predictions = lda.predict(X_test)
qda_predictions = qda.predict(X_test)
```

In [130…
```
# Assess LDA model accuracy
lda_accuracy = getAccuracy(lda, X_test, y_test)
print(f"The accuracy of LDA model is {lda_accuracy}%")
```

```
The accuracy of LDA model is 52.256699576868826%
```

In [131…
```
# Assess QDA model accuracy
qda_accuracy = getAccuracy(qda, X_test, y_test)
print(f"The accuracy of QDA model is {qda_accuracy}%")
```

```
The accuracy of QDA model is 49.647390691114246%
```

Based on the results, we see that LDA model performs slightly better than QDA model.</br>

Part 5.

Since plotting training data and testing data together makes the graph hard to visualize, we plot training and testing data on two separate graphs. In the plot for training data, pink dots represents female and blue dot represents male.</br> This holds true for the testing data plot, except points are colored red if the prediction is wrong

In [132…
```
def plot(predictions, modelName):
    fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(10, 3))

    # Plotting training data
    ax[0].set_title(f"Training data")
    for label in ['F', 'M']:
```

6.3 6.3 **2 / 2**

✓ **- 0 pts** *Entirely correct*

- 1 pts Partial credit for Problem 6.3

- 2 pts No work shown for Problem 6.3

Out[92]:
```
▾ LinearDiscriminantAnalysis
LinearDiscriminantAnalysis()
```

In [93]:
```
# Fitting QDA model
qda = QuadraticDiscriminantAnalysis()
qda.fit(X_train, y_train)
```

Out[93]:
```
▾ QuadraticDiscriminantAnalysis
QuadraticDiscriminantAnalysis()
```

Part 4.

We compare the LDA/QDA model prediction with the ground truth using the accuracy metric, which is given by </br>

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of samples}}$$

In [94]:
```
def getAccuracy(model, X_test, y_test):
    predictions = model.predict(X_test)
    numCorrect = np.count_nonzero(predictions == y_test)
    return numCorrect / len(y_test) * 100
```

In [135…
```
# Prediction using LDA and QDA model
lda_predictions = lda.predict(X_test)
qda_predictions = qda.predict(X_test)
```

In [130…
```
# Assess LDA model accuracy
lda_accuracy = getAccuracy(lda, X_test, y_test)
print(f"The accuracy of LDA model is {lda_accuracy}%")
```

```
The accuracy of LDA model is 52.256699576868826%
```

In [131…
```
# Assess QDA model accuracy
qda_accuracy = getAccuracy(qda, X_test, y_test)
print(f"The accuracy of QDA model is {qda_accuracy}%")
```

```
The accuracy of QDA model is 49.647390691114246%
```

Based on the results, we see that LDA model performs slightly better than QDA model.</br>

Part 5.

Since plotting training data and testing data together makes the graph hard to visualize, we plot training and testing data on two separate graphs. In the plot for training data, pink dots represents female and blue dot represents male.</br> This holds true for the testing data plot, except points are colored red if the prediction is wrong

In [132…
```
def plot(predictions, modelName):
    fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(10, 3))

    # Plotting training data
    ax[0].set_title(f"Training data")
    for label in ['F', 'M']:
```

6.4 6.4 **2 / 2**

✓ **- 0 pts** *Entirely correct*

- 1 pts Partial credit for Problem 6.4

- 2 pts No work shown for Problem 6.4

Out[92]:
```
▾ LinearDiscriminantAnalysis
LinearDiscriminantAnalysis()
```

In [93]:
```
# Fitting QDA model
qda = QuadraticDiscriminantAnalysis()
qda.fit(X_train, y_train)
```

Out[93]:
```
▾ QuadraticDiscriminantAnalysis
QuadraticDiscriminantAnalysis()
```

Part 4.

We compare the LDA/QDA model prediction with the ground truth using the accuracy metric, which is given by </br>

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of samples}}$$

In [94]:
```
def getAccuracy(model, X_test, y_test):
    predictions = model.predict(X_test)
    numCorrect = np.count_nonzero(predictions == y_test)
    return numCorrect / len(y_test) * 100
```

In [135…
```
# Prediction using LDA and QDA model
lda_predictions = lda.predict(X_test)
qda_predictions = qda.predict(X_test)
```

In [130…
```
# Assess LDA model accuracy
lda_accuracy = getAccuracy(lda, X_test, y_test)
print(f"The accuracy of LDA model is {lda_accuracy}%")
```

```
The accuracy of LDA model is 52.256699576868826%
```

In [131…
```
# Assess QDA model accuracy
qda_accuracy = getAccuracy(qda, X_test, y_test)
print(f"The accuracy of QDA model is {qda_accuracy}%")
```

```
The accuracy of QDA model is 49.647390691114246%
```

Based on the results, we see that LDA model performs slightly better than QDA model.</br>

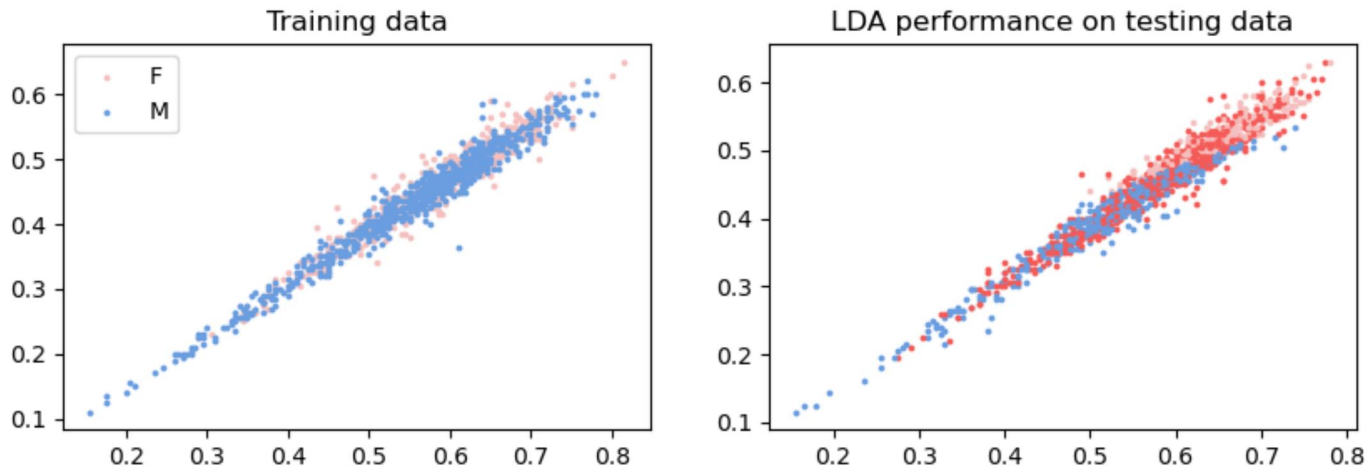
Part 5.

Since plotting training data and testing data together makes the graph hard to visualize, we plot training and testing data on two separate graphs. In the plot for training data, pink dots represents female and blue dot represents male.</br> This holds true for the testing data plot, except points are colored red if the prediction is wrong

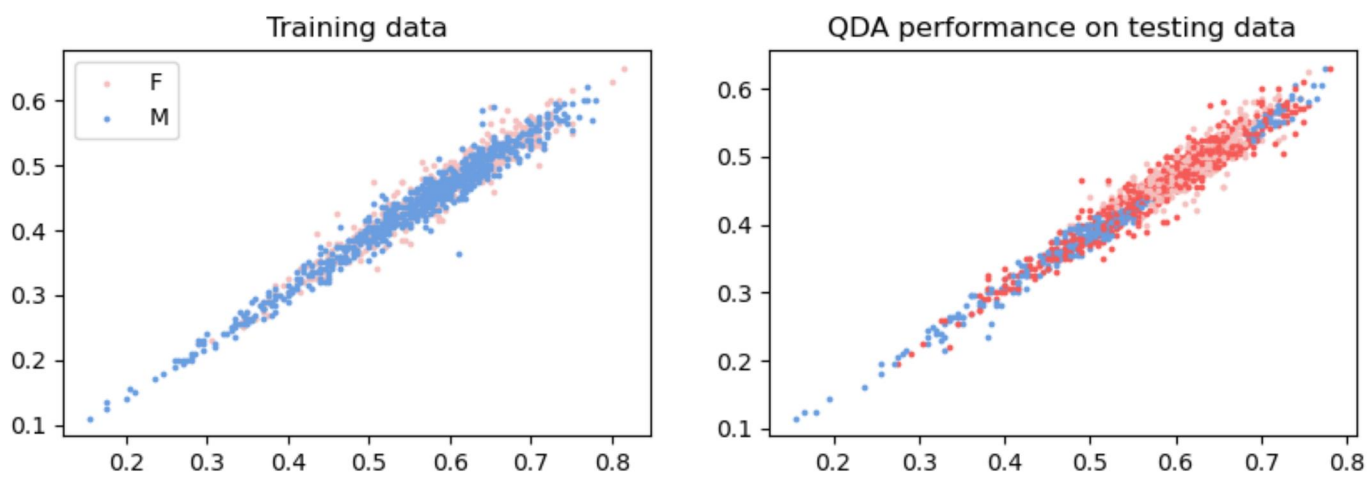
In [132…
```
def plot(predictions, modelName):
    fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(10, 3))

    # Plotting training data
    ax[0].set_title(f"Training data")
    for label in ['F', 'M']:
```

```
        ax[0].scatter(
            X_train[y_train == label]['Length'],
            X_train[y_train == label]['Diameter'],
            label=label,
            c="#f5c0bf" if label == 'F' else '#6A9EE1',
            marker='o',
            s=3
        )
    ax[0].legend()

    # Plotting testing data
    ax[1].set_title(f"{modelName} performance on testing data")
    for i in range(len(X_test)):
        if predictions[i] != y_test[i]:
            ax[1].scatter(
                X_test.iloc[i]['Length'],
                X_test.iloc[i]['Diameter'],
                c='#f55b58',
                marker='o',
                s=3
            )
        else:
            ax[1].scatter(
                X_test.iloc[i]['Length'],
                X_test.iloc[i]['Diameter'],
                c="#f5c0bf" if y_test[i] == 'F' else '#6A9EE1',
                marker='o',
                s=3
            )
```

In [136…
```
plot(lda_predictions, "LDA")
```



In [137…
```
plot(qda_predictions, "QDA")
```



From the above plot, we see that there's a significant amount of data being misclassified.

In [ ]:

6.5 6.5 **2 / 2**

✓ **- 0 pts** *Entirely correct*

- 1 pts Partial credit for Problem 6.5

- 2 pts No work shown for Problem 6.5