Coursework2

March 8, 2022

1

(a) The covariance is defined as the expected value of the product of two variables' deviations from their expected values. And Pearson's correlation coefficient is the covariance of the two variables divided by the product of their standard deviations.

$$\begin{aligned}
Cov(X,Y) &= E[(X-E[X])(Y-E[Y]))] \\
&= E[XY - E[X]Y - E[Y]X + E[X]E[Y]] \\
&= E[XY] - E[X]E[Y] - E[X]E[Y] + E[X]E[Y] \\
&= E[XY] - E[X]E[Y]
\end{aligned}$$

$$p_{X,Y} = Cox(X,Y)/\sigma_X\sigma_Y \tag{1}$$

Essentially, we only need to consider the numerator of $p_{X,Y}$, when it comes to 'covariance and Pearson correlation vanish. Using this equation (1) (2) (and the fact that the expectation of the product of two independent random variables is equal to the product of the expectations), it is easy to see that if two random variables are independent their covariance is 0. Similarly, Pearson correlation is also 0.

(b) From given distribution, we can easily get the $E[X] = 1/3*(-1) + 1/3*0 + 1/3*1 = 0$ and $E[XY] = E[X^3] = 1/3*(-1)^3 + 1/3*0^3 + 1/3*1^3 = 0$.When we substitute the obtained value into the following equation, then we can easily get the covariance $Cov(x,y)$ (or Pearson correlation $p_{X,Y}$) is equal to zero. However, because $Y = X^2$, X and Y must be dependent. In conclusion, it is true that two random variables X and Y can be dependent but have zero Pearson correlation.

$$\begin{aligned}
Cov(X,Y) &= Cov(X,X^2) \\
&= E[X*X^2] - E[X]E[X^2] \\
&= E[X^3] - E[X]E[X^2] \\
&= 0 - 0*E[X^2] \\
&= 0
\end{aligned}$$

2

The code of this part is attached in Appendix.

(a) As seen in Figure 1, we can see that when ignoring the confounder X the regression result of β_T is far from the correct estimate in Figure 2. Ignoring the confounder, the error bar (within 2σ) of β_T is [2.285, 2.904], while the error bar is [5.985, 6.015] when taking confounder into consideration. According to the coefficient $\beta_T = 6$ we assumed in advance, we can prove that including the confounder yields the correct causal effect.

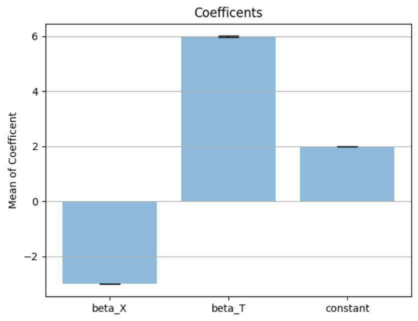

Figure 1: Simulation 1, without ignoring the confounder.

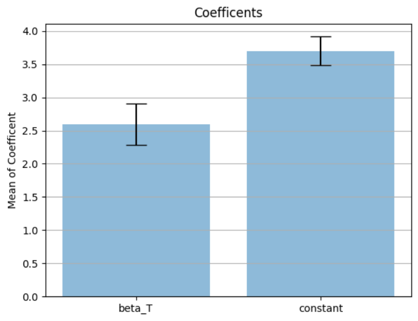

Figure 2: Simulation 1, ignoring the confounder.

(b) As seen in Figure 3, we can see that when ignoring the confounder X the regression result of β_T is approximately same as the correct estimate in Figure 4. Ignoring the confounder, the error bar (within 1σ) of β_T is [5.820, 6.185], while the error bar is [5.994, 6.006] when taking confounder into consideration. According to the coefficient $\beta_T = 6$ we assumed in advance, we can prove that including the confounder yields the correct causal effect.

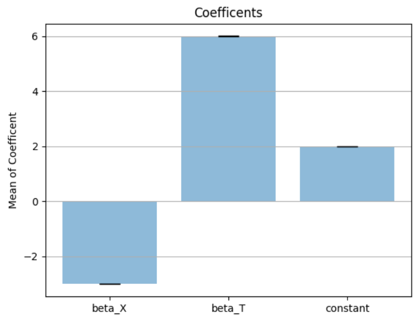

Figure 3: Simulation 1, without ignoring the confounder.

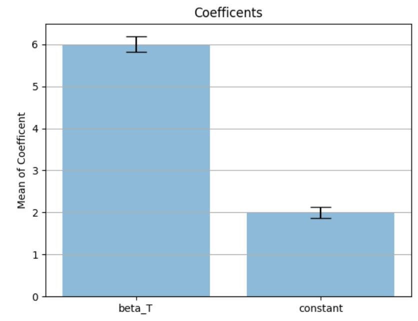

Figure 4: Simulation 2, ignoring the confounder.

3

(a) There are three variables set satisfy the demand, i.e., $\{T, X_1, X_2\}\{T, X_1, X_2, X_3\}\{T, X_2, X_3\}$. The sets of variables that d-separate X_0 and Y means that they block all the paths from X_0 to Y. First, we list all the undirected paths from X_0 to Y. We first analyze the sets of variables that d-separate each path between X_0 and Y separately.

$$
\begin{aligned}
&Path\ 1 : X_0 - X_1 - X_2 - Y\\
&Path\ 2 : X_0 - X_1 - X_3 - Y\\
&Path\ 3 : X_0 - X_1 - T - Y\\
&Path\ 4 : X_0 - T - Y\\
&Path\ 5 : X_0 - T - X_1 - X_2 - Y\\
&Path\ 6 : X_0 - T - X_1 - X_3 - Y
\end{aligned}
$$

For Path 1, let define s_n as the sets that could block the $Path\ n$, the triplet "$X_0 - X_1 - X_2$" is a collider and the triplet "$X_1 - X_2 - Y$" is a fork. Because the collider itself could block the path, there are two situations, one is not condition on X_1 or its descendants, $\emptyset \parallel X_2$ could block the path, thus $s_1 = \{$'not condition on X_1':$\emptyset \parallel X_2\}$. On the other hand, if conditioning on X_1 or at least one of its descendants, the collider unblocked path, and X_2 is needed to be conditioned on for blocking the path again. $s_1 = \{$'not condition on X_1 or its descendants':$\emptyset \parallel X_2$, 'condition on X_1 or its descendants': $X_1, X_2 \parallel X_2, X_3 \parallel X_2, T \parallel X_1, X_2, X_3 \parallel X_1, X_2, T \parallel X_2, X_3, T \parallel X_1, X_2, X_3, T\}$

For Path 2, the triplet "$X_0 - X_1 - X_3$" is a chain and the triplet "$X_1 - X_3 - Y$" is also a chain. Conditioning on one or more chain nodes could block the path. Thus, $s_2 := X_1 \parallel X_3 \parallel X_1, X_3$

For Path 3, the triplet "$X_0 - X_1 - T$" is a chain, and the triplet "$X_1 - T - Y$" is also a chain. Conditioning on one or more chain nodes could block the path. Thus, $s_3 := X_1 \parallel T \parallel X_1, T$

For Path 4, the triplet "$X_0 - T - Y$" is a chain. Conditioning on one or more chain nodes could block the path. Thus, $s_4 := T$

For Path 5, the triplet "$X_0 - T - X_1$" is a collider, the triplet "$T - X_1 - X_2$" is a chain and "$X_1 - X_2 - Y$" is a fork. Conditioning on one or more chain nodes could block the path. There are two situations, one is not condition on T, $\emptyset \parallel X_1 \parallel X_2 \parallel X_1, X_2$ could block the path. On the other side, when condition on T, $T, X_1 \parallel T, X_2 \parallel T, X_1, X_2$ could block the path. $s_5 = \{$'not condition on T':$\emptyset \parallel X_1 \parallel X_2 \parallel X_1, X_2$, 'condition on T': $T, X_1 \parallel T, X_2 \parallel T, X_1, X_2\}$

For Path 6, the triplet "$X_0 - T - X_1$" is a collider, the triplet "$T - X_1 - X_3$" is a fork and "$X_1 - X_3 - Y$" is a chain. Conditioning on one or more chain nodes could block the path. There are two situations, one is not condition on T, $\emptyset \parallel X_1 \parallel X_3 \parallel X_1, X_3$ could block the path. On the other side, when condition on T, $T, X_1 \parallel T, X_3 \parallel T, X_1, X_3$ could block the path. $s_6 = \{$'not condition on T':$\emptyset \parallel X_1 \parallel X_3 \parallel X_1, X_3$, 'condition on T': $T, X_1 \parallel T, X_3 \parallel T, X_1, X_3\}$

First, we have to condition on T, because T is the only elements in s_4 that could block the Path 4. Once we condition on T, the condition of s_1's 'not condition on X_1', s_5's 'not condition on T' and s_6's 'not condition on T' could be eliminated, since T is the collider node or its descendant. Next, by listing the combinations that meet all s_n, we can get $\{T, X_1, X_2\}\{T, X_1, X_2, X_3\}\{T, X_2, X_3\}$ as our final results.

(b) No. For the path $X_0 - X_1 - X_2$, the triplet $X_0 - X_1 - X_2$ is a collider. X_3 is a descendant of collider node X_1, and conditioning on X_3 unblocks the path $X_0 - X_1 - X_2$ from X_0 to X_2. Since one of its paths between X_0 and X_2 is unblocked, X_0 and X_2 are not independent conditional on X_3.

(c) No arrow can be reversed in this DAG. There are 8 arrows in this DAG. Let us analyze the consequence of reversing the direction of each arrow.

Arrow '$X_0 -> T$'. If reverse this arrow, there will a cycle '$X_0 - X_1 - X_2$' created.

Arrow '$X_0 -> X_1$'. If reverse this arrow, the collider '$X_0 -> X_1 <- X_2$' will be destoryed and the independence $X_0 \perp\!\!\!\perp X_2$ will be destoryed.

Arrow '$X_1 -> T$'. If reverse this arrow, the collider '$X_0 -> X_1 <- X_2$' will be destoryed, the independence $T \perp\!\!\!\perp X_2 | \{X_1, X_0\}$ will be destoryed, and a new collider '$T -> X_1 <- X_2$' will be created.

Arrow '$X_1 - > X_3$'. If reverse this arrow, the independence $X_0 \perp\!\!\!\perp Y|\{X_1, X_2, T\}$ will be destoryed, and new colliders '$X_0 - > X_1 < -X_3$' and '$X_2 - > X_1 < -X_3$' will be created.

Arrow '$X_2 - > X_1$'. If reverse this arrow, the collider '$X_0 - > X_1 < -X_2$' will be destoryed and the independence $X_0 \perp\!\!\!\perp X_2$ will be destoryed.

Arrow '$X_2 - > Y$'. If reverse this arrow, there will a cycle '$X_3 - X_1 - X_2 - Y$' created.

Arrow '$X_3 - > Y$'. If reverse this arrow, the colliders '$X_3 - > Y < -T$' and '$X_3 - > Y < -X_2$' will be destoryed and the independence $X_1 \perp\!\!\!\perp Y|\{X_2, X_3, T\}$ will be destoryed.

Arrow '$T - > T$'. If reverse this arrow, the colliders '$X_3 - > Y < -T$' and '$T - > Y < -X_2$' will be destoryed and the independence $X_1 \perp\!\!\!\perp Y|\{X_2, X_3, T\}$ will be destoryed.

In conclusion, the reversing of any arrows will affect the statistical test of DAG independence. Also, the CPDAG of this DAG is itself, which means that no arrow is reversible.

(d) The adjustment sets for the causal effect of T on Y are listed as follows, i.e., $\{X_0, X_1\}$, $\{X_0, X_1, X_3\}$, $\{X_0, X_1, X_2\}$, $\{X_0, X_2, X_3\}$, $\{X_0, X_1, X_2, X_3\}$, $\{X_1, X_2\}$, $\{X_1, X_2, X_3\}$, $\{X_2, X_3\}$.

Since no variable z intercepts all directed paths from T to Y, there are only back-door adjustment sets. The sets need to satisfy two criterion, i.e. (1) no nodes in our sets that is the descendant of T. Obviously, there are no descendants of T in our graph. (2) The sets block every path from T to Y that contains an arrow into T. The paths that meet this criterion include:

$$\begin{aligned} &Path\ 1 : Y - X_3 - X_1 - T \\ &Path\ 2 : Y - X_2 - X_1 - T \\ &Path\ 3 : Y - X_3 - X_1 - X_0 - T \\ &Path\ 4 : Y - X_2 - X_1 - X_0 - T \end{aligned}$$

For Path 1, the triplet "$Y - X_3 - X_1$" is a chain and the triplet "$X_3 - X_1 - T$" is a fork. At least conditioning on either chain node or fork node, the path could be blocked. $s_1 = \{X_1 \parallel X_3 \parallel X_1, X_3\}$

For Path 2, the triplet "$Y - X_2 - X_1$" is a fork and the triplet "$X_2 - X_1 - T$" is a chain. At least conditioning on either chain node or fork node, the path could be blocked. $s_2 = \{X_1 \parallel X_2 \parallel X_1, X_2\}$

For Path 3, the triplet "$Y - X_3 - X_1$" is a chain, the triplet "$X_3 - X_1 - X_0$" is a chain and the triplet "$X_1 - X_0 - T$" is a fork. At least conditioning on either one or both chain nodes or fork node, the path could be blocked. $s_3 = \{X_0 \parallel X_1 \parallel X_3 \parallel X_0, X_1 \parallel X_0, X_3 \parallel X_1, X_3 \parallel X_0, X_1, X_3\}$

For Path 4, the triplet "$Y - X_2 - X_1$" is a fork, the triplet "$X_2 - X_1 - X_0$" is a collider and the triplet "$X_1 - X_0 - T$" is a fork. Since there is a collider, the path itself is blocked and if condition on collider node X_1, the path is unblocked. Thus, $s_4 = \{$'no condition on X_1 or its descendants': ,$\emptyset \parallel X_0 \parallel X_2 \parallel X_0, X_2$, 'condition on X_1 or it descendants':$X_0, X_1 \parallel X_1, X_2 \parallel X_0, X_1, X_2 \parallel X_0, X_3 \parallel X_2, X_3 \parallel X_0, X_2, X_3 \parallel X_0, X_1, X_3 \parallel X_1, X_2, X_3 \parallel X_0, X_1, X_2, X_3\}$

Then, by listing the combinations that meet all s_n, we can get $\{X_0, X_1\}$, $\{X_0, X_1, X_3\}$, $\{X_0, X_1, X_2\}$, $\{X_0, X_2, X_3\}$, $\{X_0, X_1, X_2, X_3\}$, $\{X_1, X_2\}$, $\{X_1, X_2, X_3\}$, $\{X_2, X_3\}$ as our final results.

(e) $\{X_2, X_3\}$

From the results we get from (d), we can easily get $\{X_2, X_3\}$ and $\{X_0, X_2, X_3\}$ as our adjustment sets if X_1 is unobserved. And $\{X_2, X_3\}$ is obviously the smallest one.

Appendices

A Code for Q2(a)

```
import numpy as np
import matplotlib.pyplot as plt

# set global variables
np.random.seed(42)
sample_n = 1000
n = 1000
# here we can choose regression with or without confounder X
without_confounder = False


T_coefficient_list = []
X_coefficient_list = []
constant_list = []

# sample for sample_n times
for _ in range(sample_n):
    # set up some noise for X,T and Y
    noise_X = 0.1 * np.random.randn(n)
    noise_T = 0.1 * np.random.randn(n)
    noise_Y = 0.1 * np.random.randn(n)
    # X is a normal distribution
    X = np.random.randn(n) + noise_X

    # turn T into a binomial distribution
    T = X + np.random.randn(n) + noise_T
    T[T>0] = 1
    T[T<=0] = 0

    Y = 2 + 6 * T - 3 * X + noise_Y

    # linear regression
    if without_confounder:
        ww = np.linalg.lstsq(np.array([T, np.ones(n)]).T, Y, rcond=None)[0]
        T_coefficient_list.append(ww[0])
        constant_list.append(ww[1])
    else:
        ww = np.linalg.lstsq(np.array([X, T, np.ones(n)]).T, Y, rcond=None)[0]
        X_coefficient_list.append(ww[0])
        T_coefficient_list.append(ww[1])
        constant_list.append(ww[2])

# calculate the mean of coefficients
T_mean = np.mean(T_coefficient_list)
constant_mean = np.mean(constant_list)

# calculate the std of coefficients
T_std = np.std(T_coefficient_list)
constant_std = np.std(constant_list)

if without_confounder:
    x_label = ['beta_T', 'constant']
    x_pos = np.arange(len(x_label))
    x_mean = [T_mean, constant_mean]
    errorbar = [2*T_std, 2*constant_std]

else:
    X_mean = np.mean(X_coefficient_list)
    X_std = np.std(X_coefficient_list)

    x_label = ['beta_X', 'beta_T', 'constant']
    x_pos = np.arange(len(x_label))
    x_mean = [X_mean, T_mean, constant_mean]
    errorbar = [2*X_std, 2*T_std, 2*constant_std]

# Build the plot
fig, ax = plt.subplots()
ax.bar(x_pos, x_mean,
       yerr=errorbar,
       align='center',
       alpha=0.5,
       ecolor='black',
       capsize=10)
ax.set_ylabel('Mean of Coefficent')
ax.set_xticks(x_pos)
ax.set_xticklabels(x_label)
ax.set_title('Coefficents')
ax.yaxis.grid(True)

plt.show()
print([T_mean - 2*T_std, T_mean + 2*T_std])
```

B Code for Q2(b)

```
import numpy as np
import matplotlib.pyplot as plt

# set global variables
np.random.seed(42)
sample_n = 1000
n = 1000
# here we can choose regression with or without confounder X
without_confounder = False

T_coefficient_list = []
X_coefficient_list = []
constant_list = []

# sample for sample_n times
for _ in range(sample_n):
    # set up some noise for X,T and Y
    noise_X = 0.1 * np.random.randn(n)
    noise_T = 0.1 * np.random.randn(n)
    noise_Y = 0.1 * np.random.randn(n)
    # X is a normal distribution
    X = np.random.randn(n) + noise_X

    # turn T into a binomial distribution
    T = np.random.randn(n) + noise_T
    T[T>0] = 1
    T[T<=0] = 0

    Y = 2 + 6 * T - 3 * X + noise_Y

    # linear regression
    if without_confounder:
        ww = np.linalg.lstsq(np.array([T, np.ones(n)]).T, Y, rcond=None)[0]
        T_coefficient_list.append(ww[0])
        constant_list.append(ww[1])
    else:
        ww = np.linalg.lstsq(np.array([X, T, np.ones(n)]).T, Y, rcond=None)[0]
        X_coefficient_list.append(ww[0])
        T_coefficient_list.append(ww[1])
        constant_list.append(ww[2])

# calculate the mean of coefficients
T_mean = np.mean(T_coefficient_list)
constant_mean = np.mean(constant_list)

# calculate the std of coefficients
T_std = np.std(T_coefficient_list)
constant_std = np.std(constant_list)

if without_confounder:
    x_label = ['beta_T', 'constant']
    x_pos = np.arange(len(x_label))
    x_mean = [T_mean, constant_mean]
    errorbar = [T_std, constant_std]
```

```
else:
    X_mean = np.mean(X_coefficient_list)
    X_std = np.std(X_coefficient_list)

    x_label = ['beta_X', 'beta_T', 'constant']
    x_pos = np.arange(len(x_label))
    x_mean = [X_mean, T_mean, constant_mean]
    errorbar = [X_std, T_std, constant_std]

# Build the plot
fig, ax = plt.subplots()
ax.bar(x_pos, x_mean,
       yerr=errorbar,
       align='center',
       alpha=0.5,
       ecolor='black',
       capsize=10)
ax.set_ylabel('Mean of Coefficent')
ax.set_xticks(x_pos)
ax.set_xticklabels(x_label)
ax.set_title('Coefficents')
ax.yaxis.grid(True)

plt.show()
print([T_mean - T_std, T_mean + T_std])
```