HW4_answers

September 4, 2019

1 ARE 106 Summer Session II

2 Homework 4

This homework will be due on **September 2nd, at 4:10pm**

2.1 Name:

2.2 SSID:

Please put your name and SSID in the corresponding cells above.

The homework is worth 13.5 points.

For each of the following questions, show as much of your steps as you can (without going overboard). If you end up getting the wrong answer, but we spot where you made a mistake in the algebra, partial credit will be more readily given. If you only put the final answer, you will be marked either right or wrong.

Answer questions in the correct cell. For problems where you have to input math, make sure that you know that it's a markdown cell (It won't have a `In: []` on the left) and make sure you run the cell by either pressing `Ctrl + Enter` or going to `Cell -> Run Cell`. Alternatively, write all your answers and then go to `Cell -> Run All Cells` after you're done.

Please ignore cells that read `\pagebreak`. These are so your document converts to PDF in a way that will make it possible to grade your homework. Ignore them and only write your answers where it is specified.

When you are finished export your homework to a PDF by going to `File -> Download as -> PDF`.

2.3 Question 1: Unbiasedness

For the following question, assume that we have a population model:

$$Y_i = \mu + \varepsilon_i$$

where $E(\varepsilon_i) = 0$.

We're able to get a sample of N observations of Y_i. For the following linear estimators of μ, find whether or not they are unbiased estimators of Y_i.

- a. $\frac{1}{N} \sum_i^N Y_i$
- b. $\sum_i^N Y_i$
- c. Y_3

Please write your answer here. If you need to use more than one line, you may do so.

- a. $E(\frac{1}{N}\sum_i^N Y_i) = \frac{1}{N}E(\sum_i^N Y_i) = \frac{1}{N}\sum_i^N E(Y_i) = \frac{1}{N}\sum_i^N \mu = \frac{1}{N} \cdot N \cdot \mu = \mu$. Unbiased
- b. $E(\sum_i^N Y_i) = \sum_i^N E(Y_i) = \sum_i^N \mu = N \cdot \mu$ Biased.
- c. $E(Y_3) = \mu$ Unbiased.

2.4 Question 2: More Unbiasedness

Now suppose that the population model is:

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$$

where X_i is some data and is treated as *given* (i.e. not a random variable). Once again, $E(\varepsilon_i) = 0$. Assume that all Gauss-Markov assumptions are satisfied.

We're able to get a sample of N observations of Y_i. For the following linear estimators of β_1, answer the following questions.

- a. What is the expected value of: $\sum_i^N k_i Y_i$, where k_i is some weight for every observation, which is not dependent on Y_i (not necessarily the OLS weight!!). Also do not bring in the expectation operator, just leave the operator outside the expression.
- b. What has to be true in order for the estimator in a. to be unbiased?
- c. What Gauss-Markov assumption do we use to make sure that $\sum_i^N k_i \varepsilon_i = 0$ in the OLS case?
- d. What is the expected value of: $\frac{1}{N} \sum_i^N Y_i$. Let $E(X_i) = \mu_x$
- e. What needs to be true, so that the estimator in d. is unbiased?

Please write your answer here. If you need to use more than one line, you may do so.

- a. $E(\sum_i^N k_i Y_i) = E(\sum_i^N k_i(\beta_0 + \beta_1 X_i + \varepsilon_i)) = E(\beta_0 \sum_i^N k_i + \beta_1 \sum_i^N k_i X_i + \sum_i^N k_i \varepsilon_i)$
- b. We need $\sum_i^N k_i = 0$, $\sum_i^N k_i X_i = 1$, and $\sum_i^N k_i \varepsilon_i = 0$
- c. CR 5: exogeneity of the X_i variables.
- d. $E(\frac{1}{N}\sum_i^N Y_i) = E(\frac{1}{N}\sum_i^N(\beta_0 + \beta_1 X_i + \varepsilon_i)) = \frac{1}{N}\sum_i^N E(\beta_0) + \beta_1 \frac{1}{N}\sum_i^N E(X_i) + \frac{1}{N}\sum_i^N E(\varepsilon_i) = \beta_0 + \beta_1 \mu_x$
- e. $\beta_0 = 0$ and $\mu_x = 1$.

2.5 Question 3: Variance of an Estimator

For the following question, assume that we have two possible population models:

$$Y_i = \mu + \varepsilon_i$$

where $E(\varepsilon_i) = 0$.

We're able to get a sample of N observations of Y_i. For the following linear estimators of μ, find the variance of the estimator.

Note: It need not be the case that $Cov(\varepsilon_i, \varepsilon_j) = 0$ or that $Var(\varepsilon_i) = \sigma^2$, unless explicitly stated.

- a. $\frac{1}{N}\sum_i^N Y_i$, with $Cov(\varepsilon_i, \varepsilon_j) = 0$ and $Var(\varepsilon_i) = \sigma^2$
- b. $\frac{1}{N}\sum_i^N Y_i$, with $Cov(\varepsilon_i, \varepsilon_j) = 0$ and $Var(\varepsilon_i) = \sigma_i^2$. **Note the difference!**
- c. $\frac{Y_1+Y_2}{2}$, with $Cov(\varepsilon_i, \varepsilon_j) \neq 0$ and $Var(\varepsilon_i) = \sigma^2$
- d. Would the estimator from c. be an unbiased estimator of σ^2?

Please write your answer here. If you need to use more than one line, you may do so.

- a. $Var(\frac{1}{N}\sum_i^N Y_i) = \frac{1}{N^2}\sum_i^N Var(Y_i) = \frac{1}{N^2}N\sigma^2 = \frac{1}{N}\sigma^2$
- b. $Var(\frac{1}{N}\sum_i^N Y_i) = \frac{1}{N^2}\sum_i^N Var(Y_i) = \frac{1}{N^2}\sum_i^N \sigma_i^2$
- c. $Var(\frac{Y_1+Y_2}{2}) = Var(\frac{1}{2}Y_1) + Var(\frac{1}{2}Y_2) + 2Cov(\frac{1}{2}Y_1, \frac{1}{2}Y_2) = \frac{1}{4}\sigma^2 + \frac{1}{4}\sigma^2 + \frac{1}{2}Cov(Y_1, Y_2) = \frac{1}{2}\sigma^2 + \frac{1}{2}Cov(Y_1, Y_2)$
- d. No, because there is a non-zero covariance term and σ^2 is being multiplied by $\frac{1}{2}$

2.6 Question 4: Showing Unbiasedness with Data

For this question, we're going to create some random data, and see how well the OLS estimator is able to estimate this data.

Suppose we have a population model:

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$$

First, we're going to create random data for Y_i and X_i. This will involve using the `random` library in Python.

This the code to generate the random data and construct our population model. **Note that we in constructing this population model, we are following the Gauss Markov assumptions**:

```
## imports
import numpy as np
import pandas as pd
import statsmodels.formula.api as sm

## This is for making graphs
import matplotlib.pyplot as plt


## define error
## makes 1000 normally distributed observations with mean 0 and standard deviaion 1
epsilon = np.random.normal(0,1, 1000)

## define X data
## makes 1000 normally distributed observations with mean 3 and standard deviaion 2
X = np.random.normal(3, 2, 1000)

## Define population parameters
beta_0 = 2
beta_1 = .5

## Define Y data
Y = beta_0 +  beta_1*X + epsilon

## Put it all in a pandas dataframe
df = pd.DataFrame({'Y': Y,
             'X': X})
```

a. Before running the code, look at the code. What Gauss-Markov assumption is responsible for the fact that ε_i comes from a *normal distribution*?

Please write your answer here. If you need to use more than one line, you may do so.

CR4- normally distributed errors

b. Before running the code, What Gauss-Markov assumption is responsible for the fact that ε_i comes from the fact that ε_i comes from a distribution whose variance (or in this case standard deviation) that doesn't change?

Please write your answer here. If you need to use more than one line, you may do so.

CR3- homoskedasticity

c. Run the regression of Y on X.

What is the estimate? What is the normalized z-score for this estimate? Put your anwer in a print statement f-string (Put this print statement before the `summary()` call so both show up).

Hint: you know what the population parameter actually is.

Hint: You can either write down the parameters from the regression or use `results.params['X']` for $\hat{b}_1$ and `results.bse['X']` for the standard error of $\hat{b}_1$.

Note: By normalized z-score, I mean the transformation that makes the estimate in terms of a Normal distribution of mean 0 and variance 1.

```
[19]: ## Put your answer in this cell.

      ## imports
      import numpy as np
      import pandas as pd
      import statsmodels.formula.api as sm

      ## This is for making graphs
      import matplotlib.pyplot as plt

      ## define error
      ## makes 1000 normally distributed observations with mean 0 and standard␣
       ↪deviaion 1
      epsilon = np.random.normal(0,1, 1000)

      ## define X data
      ## makes 1000 normally distributed observations with mean 3 and standard␣
       ↪deviaion 2
      X = np.random.normal(3, 2, 1000)

      ## Define population parameters
      beta_0 = 2
      beta_1 = .5

      ## Define Y data
      Y = beta_0 +  beta_1*X + epsilon

      ## Put it all in a pandas dataframe
      df = pd.DataFrame({'Y': Y,
                   'X': X})

      ## Running regression

      mod = sm.ols("Y ~ X", data=df)
      results = mod.fit()


      print(f"The normalized z-score is {(results.params['X']-0.5)/results.bse['X']}")

      results.summary()
```

```
     The normalized z-score is 65.10924550509961

[19]: <class 'statsmodels.iolib.summary.Summary'>
      """
                                  OLS Regression Results
      ==============================================================================
Dep. Variable:                      Y   R-squared:                       0.833
Model:                            OLS   Adj. R-squared:                  0.832
Method:                 Least Squares   F-statistic:                     4966.
Date:                Tue, 27 Aug 2019   Prob (F-statistic):               0.00
Time:                        21:15:59   Log-Likelihood:                -3204.3
No. Observations:                1000   AIC:                             6413.
Df Residuals:                     998   BIC:                             6422.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -3.1819      0.339     -9.390      0.000      -3.847      -2.517
X              6.5762      0.093     70.467      0.000       6.393       6.759
==============================================================================
Omnibus:                      844.815   Durbin-Watson:                   1.990
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            28824.463
Skew:                           3.688   Prob(JB):                         0.00
Kurtosis:                      28.247   Cond. No.                         6.87
==============================================================================

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

d. Now let's get a sample of this and data and run our regression on it. To do this, use the `sample` method on the `df` dataframe:

```
df.sample(n=<number>, replace=True)
```

We set `replace=True`, so that we can sample observations with replacement (this is more important if we keep taking random samples).

Sample 100 observations with replacement from `df` and call this `sample_df`.

Run the regression of Y on X with the sample data. Run it a few times and see how it changes (you do need to need rerun the previous cells if you rerun this one a few times. That's because the initial data has already been created.)

[145]:
```python
## Put your answer in this cell.

## Taking random sample
sample_df = df.sample(n=100, replace=True)


## Running regression

mod = sm.ols("Y ~ X", data=sample_df)
results = mod.fit()
results.summary()
```

[145]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      Y   R-squared:                       0.542
Model:                            OLS   Adj. R-squared:                  0.537
Method:                 Least Squares   F-statistic:                     115.8
Date:                Fri, 23 Aug 2019   Prob (F-statistic):           2.74e-18
Time:                        17:48:53   Log-Likelihood:                -136.39
No. Observations:                 100   AIC:                             276.8
Df Residuals:                      98   BIC:                             282.0
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      2.0240      0.160     12.663      0.000       1.707       2.341
X              0.5187      0.048     10.760      0.000       0.423       0.614
==============================================================================
Omnibus:                        1.283   Durbin-Watson:                   2.068
Prob(Omnibus):                  0.527   Jarque-Bera (JB):                0.839
Skew:                           0.200   Prob(JB):                        0.657
Kurtosis:                       3.202   Cond. No.                         5.88
==============================================================================

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

e. Which regression has a higher standard error for $\hat{b}_1$ Why?

Please write your answer here. If you need to use more than one line, you may do so.

The regression with the lower sample size (in this case 1,000) will have the lower standard error.

f. Now let's see what happens when we keep sampling. This will allow us to construct a distribution for $\hat{b}_1$. This will involve making a loop much like in HW2.

Write a loop that loops over the many regressions and stores the results in a list. Run the loop 1000 times. Call this list `b_dist`.

Instead of appending a number multiplied by another as in HW2, append `results.params['X']` (I'm assuming that you're using `results= mod.fit()`).

Remember to put in a sample command in the loop, so that new data can be sampled every iteration.

So your loop should look like this:

```
b_dist = []

for reg in <<the range from 0 to 1000>>:

    <<sampling command>>
    <<regression commands>>
    <<parameter appending to b_dist>>
```

After the loop, make this list into a `pandas` dataframe and call it `b_dist_df`: `b_dist_df = pd.DataFrame({'b_dist':b_dist})`

Note: It might take a while.

[146]:
```
## Put your answer in this cell.

b_dist = []

for reg in range(1000):

    sample_df = df.sample(n=100, replace=True)


    mod = sm.ols("Y ~ X", data=sample_df)
    results = mod.fit()
    b_dist.append(results.params['X'])

b_dist_df = pd.DataFrame({'b_dist': b_dist})
```

g.Now create a histogram of `b_dist` with:

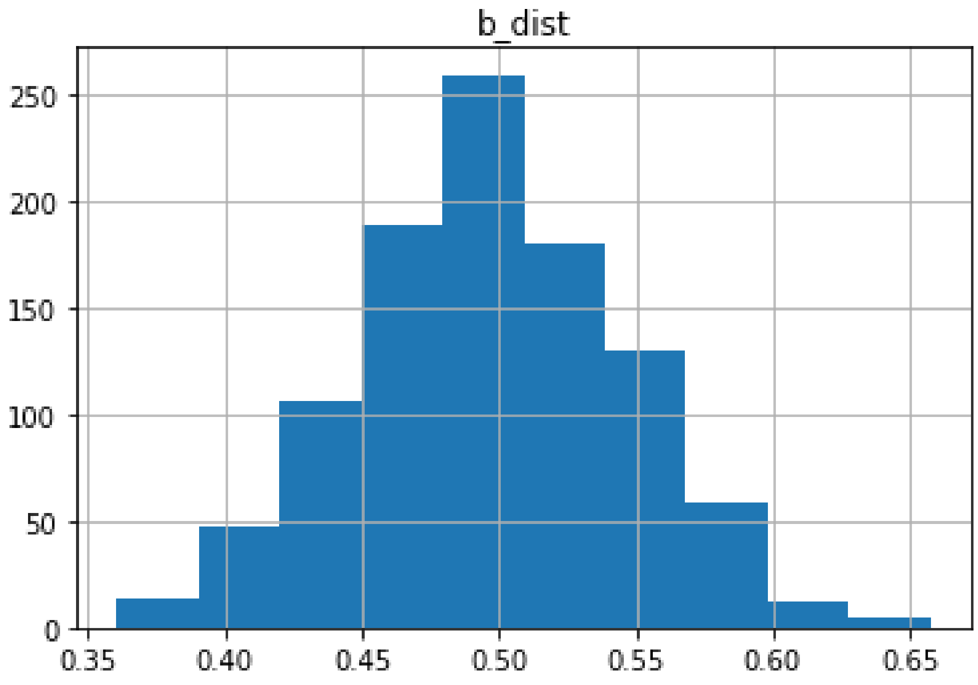
```
b_dist_df.hist()
```

```
[142]: %matplotlib inline
       ## Do not modify the above

       ## Put your answer in this cell.

       b_dist_df.hist()
```

```
[142]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f4322ff53d0>]],
             dtype=object)
```



h. What is the mean of `b_dist`?

```
[143]: ## Put your answer in this cell.\pagebreak


       b_dist_df['b_dist'].mean()
```

```
[143]: 0.4964843177758095
```

h. Plot a vertical line on the histogram, which denotes the population parameter, β_1. To do this, you can do this:

```
b_dist_df.hist()
plt.axvline(<population parameter here>, color='red')
```

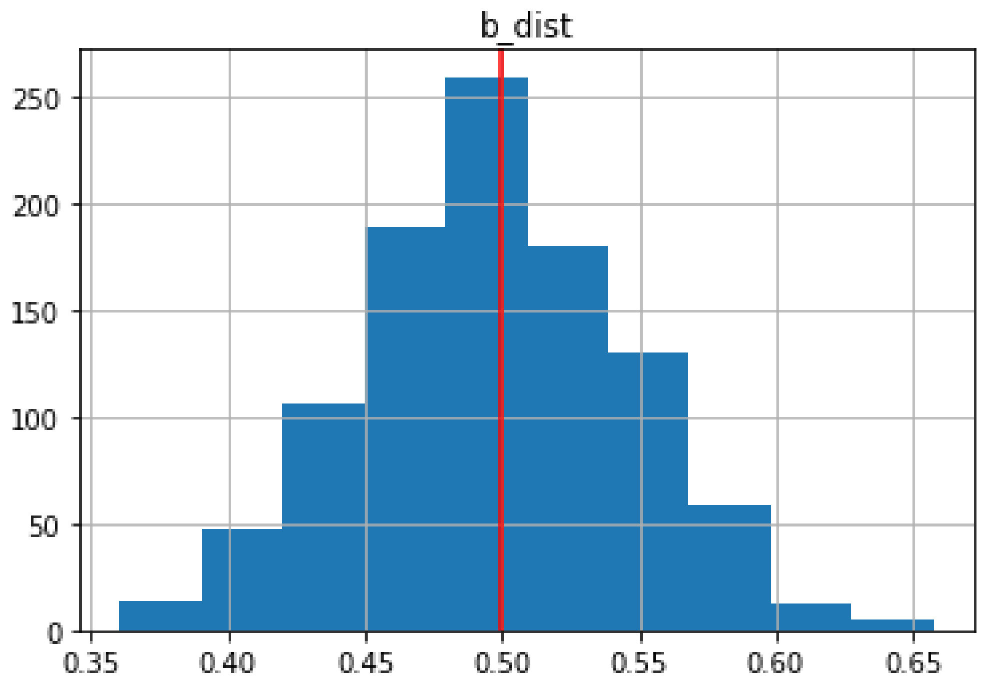
Does the OLS estimator seem unbiased? What would have made each estimate even closer to the population parameter? Write your answer in a print statement.

```
[144]: ## Put your answer in this cell.

       b_dist_df.hist()
       plt.axvline(0.5, color='red')

       print("The distribution is centered, more or less, around the population␣
        ↪parameter.\
       To make it better, we can try increasing the sample size of each random sample.
        ↪")
```

```
The distribution is centered, more or less, around the population parameter.To
make it better, we can try increasing the sample size of each random sample.
```


b_dist