lab07_student_version

October 26, 2021

1 Lab 7: Estimating Causal Effects via Instrumental Variables

Welcome to the seventh DS102 lab!

The goals of this lab is to implement and get better understanding of Instrumental Variables discussed in Lecture. Along with Inverse Propensity Scaling which you will see in Lecture, Instrumental Variables are often used to determine causal effects.

The code you need to write is commented out with a message "TODO: fill in".

1.1 Collaboration Policy

Data science is a collaborative activity. While you may talk with others about the labs, we ask that you **write your solutions individually**. If you do discuss the assignments with others please **include their names** in the cell below.

1.2 Gradescope Submission

To submit this assignment, rerun the notebook from scratch (by selecting Kernel > Restart & Run all), and then print as a pdf (File > download as > pdf) and submit it to Gradescope.

This assignment should be completed and submitted before Wednesday, October 27, 2021 at 11:59 PM. PST

1.3 Collaborators

Write the names of your collaborators in this cell.

`<Collaborator Name> <Collaborator e-mail>`

[27]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import statsmodels.api as sm
import seaborn as sns
import itertools
from ipywidgets import interact, interactive

import hashlib

sns.set(style="dark")
```

```
plt.style.use("ggplot")
%matplotlib inline
```

2 Instrumental Variables Background

Suppose that we measure Z, the number of books a student read in the last year, and we are intrested in determing how Z affects an observed target outcome Y, the student's SAT score. The effect we are interested in is **causal** because we want to know how Y changes if all randomness other than Z remains fixed, and only Z changes. We will refer to Z as the "treatment". In general, Z might be multi-dimensional, however for the purpose of this exercise we take $Z \in \mathbb{R}$.

Suppose there's also a confounder X, which is the income of the student's family. We don't observe X, but it affects both the number of books the student reads (wealthier families may have more access to books) and the student's SAT score (wealthier students may have more access to SAT tutoring).

We assume that the outcome is generated as a linear function of the confounder X and treatment Z, with additive noise ϵ:

$$Y = \beta_1 Z + \beta_2 X + \epsilon$$

The goal is to estimate β_1, the true causal effect of the number of books a student reads on their SAT score.

2.0.1 Danger of bias

As we saw in the instrumental variable lecture note (link here), if the confounder X is highly correlated with Z, performing ordinary least squares (OLS) on the observed data Z, Y can lead to very biased results.

2.0.2 Instrumental variables (IVs) and two-stage least squares (2SLS)

One way to get around this issue is by using **instrumental variables (IVs)**. A valid instrument W is a variable which is independent of the confounder X_2, and affects Y only through X_1. For example, we can create such an instrument W by employing *encouragement design*, where we randomly assign students to "readathons" of different durations. See the figure below for a causal diagram:

Using the instrumental variable Z, we can estimate β_1 by first "guessing" X_1 from W using ordinary least squares (OLS) (denoted $\hat{X}_1$), and then regressing Y onto $\hat{X}_1$ (instead of X_1) using OLS as well. This procedure is known as **two-stage least squares (2SLS)**.

In this lab, we will observe the bias that can occur when naively performing OLS on the observed data X_1, Y, and also how employing 2SLS can achieve a better estimate of β_1.

3 Model setup

Suppose that we have historical data from $n = 10,000$ different students. Suppose we observe the following variables:

$Z^{(i)}$ = number of books the student read in the last year,

$W^{(i)} = $ duration of the "readathon" at the student's school. *Note: This is sligtly different from the setup in Discussion 7 where it was a binary variable*

$Y^{(i)} = $ the student's SAT score.

Suppose that the student's family income $X^{(i)}$ affects both $Z^{(i)}$ and $Y^{(i)}$, but is **not observed**.

3.1 Data Generation

The student's SAT score is linear in the number of books the student read and the student's family income:

$$Y^{(i)} = \beta_1 Z^{(i)} + \beta_2 X^{(i)} + \epsilon^{(i)}.$$

The number of books a student reads is linear in whether or not there was a readathon and the student's family income:

$$Z^{(i)} = \gamma_1 W^{(i)} + \gamma_2 X^{(i)} + \epsilon'^{(i)},$$

3.1.1 The true model was generated in the following manner:

- Sample $W^{(i)} \sim N(20, 5)$ $\longleftarrow$ *Duration of Readathon for student* i
- Sample $X^{(i)} \sim \text{Normal}(50, 10)$ $\longleftarrow$ *Income in tens of thousands (10,000) dollars for the family of student* i **(unobserved variable)**
- Generate $Z^{(i)}$ by setting $\gamma_1 = \gamma_2 = 1$ and sampling a noise $\epsilon'^{(i)} \sim N(0, 5)$ $\longleftarrow$ *Number of books read by student* i
- Generate $Y^{(i)}$ by setting $\beta_1 = 5$, $\beta_2 = 12$ and sampling a noise $\epsilon^{(i)} \sim N(0, 10)$ $\longleftarrow$ *SAT score for student* i.

Note: The data in this lab are not observed in real life. They are instead synthetic data generated according to the procedure described above.

3.2 Load the data

Run the cells below to load and plot the data.

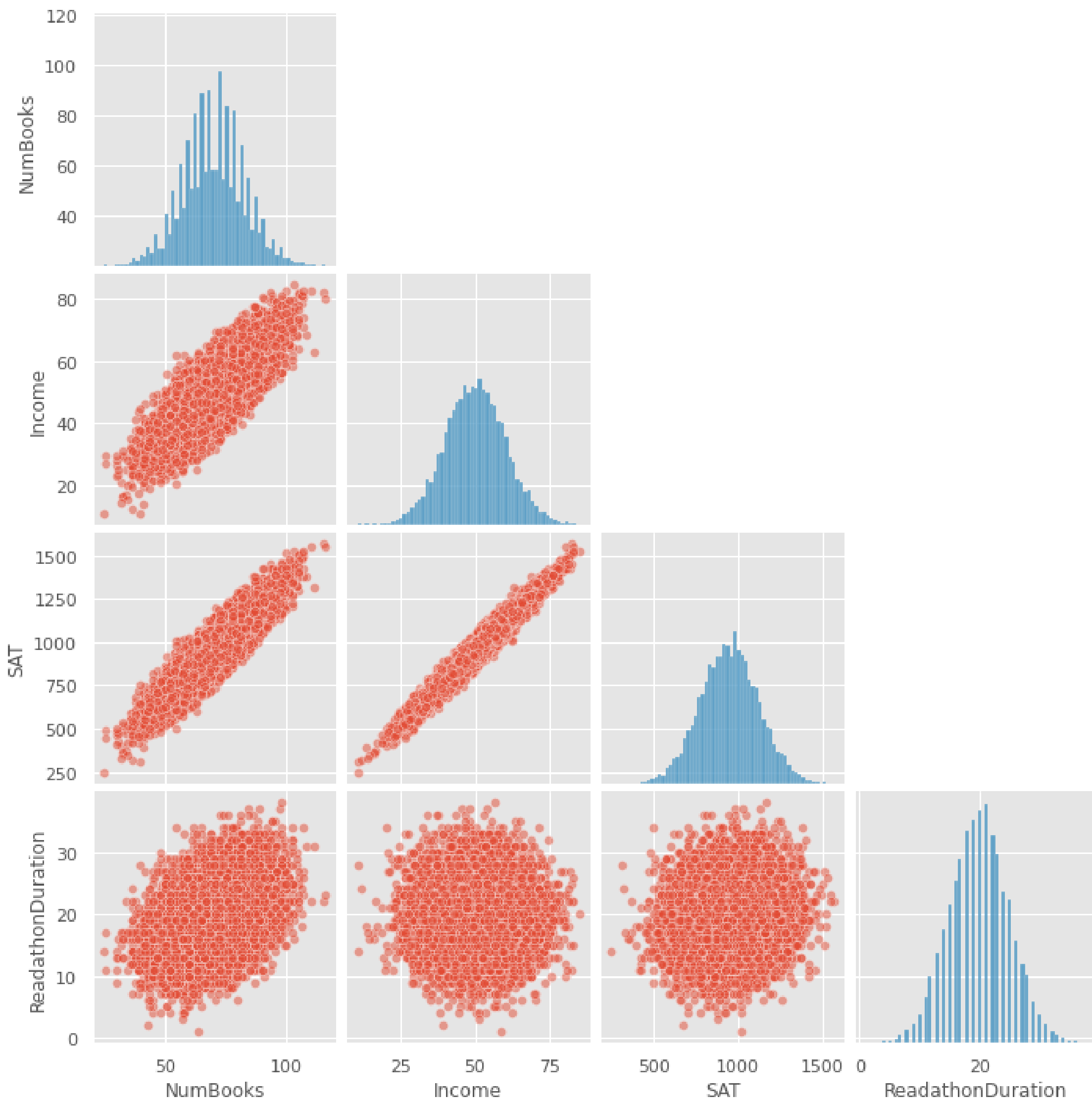
[28]:
```
# Do not modify: Just run this to load the data
student_data = pd.read_csv("SAT_data.csv")
student_data.head()
```

[28]:

	NumBooks	Income	SAT	ReadathonDuration
0	54.0	45.473122	799.0	16.0
1	71.0	53.064868	991.0	15.0
2	79.0	52.522466	1017.0	19.0
3	92.0	61.322441	1197.0	23.0
4	65.0	53.453239	982.0	12.0

[29]:
```
# make a pairplot illustrating the pairwise correlations between different␣
↪columns in the dataset
fig = sns.pairplot(student_data,  plot_kws=dict(marker="o", alpha = 0.5))
for i, j in zip(*np.triu_indices_from(fig.axes, 1)):
    fig.axes[i, j].set_visible(False)
```

```
plt.show()
```



In the plot above on the main diagonal we have the histograms of each variable, and on the off-diagonals we have scatter plots illustrating the corelations between pair of variables.

4 Question 1: Understanding the Model

4.1 1.a Correlations between variables

i) Just by inspecting the pairplot above rank in order from most correlated to least correlated the following pairwise relationships: Z & X, Z & Y, Z & W, X & Y, X & W, Y & W

TODO:

W = Duration of Readathon

X = Income

Y = SAT Score

Z = Number of books read

X & Y, Z & Y, Z & X, Z & W, Y & W, X & W

ii) Which of the above pairs appear to be independent?

TODO:

X & W, Y & W

4.2 1.b Understanding the marginal impacts

4.2.1 Inspect the Data Generation section above, and answer the following questions.

i) What is the true causal effect of an extra book read on the SAT score (i.e. if you hold everything else constant and you read one more book by how much will the SAT score change)?

ii) What is the true causal effect of increasing income by 10000 on the SAT score?

iii) What is the true causal effect of an extra readathon day on the number of books read?

iv) What is the true causal effect of increaing income by 10000 on the number of books read?

i) **TODO**: _1=5

ii) **TODO**: _2=12

iii) **TODO**: _1=1

iv) **TODO**: _2=1

4.3 Ordinary Least Squares

If we had access to income data X, then we could estimate directely β_1, β_2, γ_1, γ_2 from the data by setting up a linear regression problem and finding Ordinary Least Squares estimator.

$$\hat{\beta}_1, \hat{\beta}_2 = \arg\min_{\beta_1, \beta_2} \|Y - \beta_1 Z - \beta_2 X\|_2^2$$

$$\hat{\gamma}_1, \hat{\gamma}_2 = \arg\min_{\gamma_1, \gamma_2} \|Z - \gamma_1 W - \gamma_2 X\|_2^2$$

To find OLS estimators we will use `sm.OLS` from `statsmodels.api`.

[30]:
```
# No TODOs here: Just examine the code
def fit_OLS_model(df, target_variable, explanatory_variables, intercept =␣
↪False):
    """
    Fits an OLS model from data.
```

```
    Inputs:
        df: pandas DataFrame
        target_variable: string, name of the target variable
        explanatory_variables: list of strings, names of the explanatory␣
↪variables
        intercept: bool, if True add intercept term
    Outputs:
        fitted_model: model containing OLS regression results
    """

    target = df[target_variable]
    inputs = df[explanatory_variables]
    if intercept:
        inputs = sm.add_constant(inputs)

    fitted_model = sm.OLS(target, inputs).fit()
    return(fitted_model)
```

[31]:
```
# Computing the OLS estimators for gamma_1 and beta_2
gammas_model = fit_OLS_model(student_data, 'NumBooks', ['ReadathonDuration',␣
↪'Income'])
print(gammas_model.summary())
```

```
                                 OLS Regression Results
================================================================================
=======
Dep. Variable:               NumBooks   R-squared (uncentered):
0.995
Model:                            OLS   Adj. R-squared (uncentered):
0.995
Method:                 Least Squares   F-statistic:
9.849e+05
Date:                Tue, 26 Oct 2021   Prob (F-statistic):
0.00
Time:                        10:01:33   Log-Likelihood:
-30390.
No. Observations:               10000   AIC:
6.078e+04
Df Residuals:                    9998   BIC:
6.080e+04
Df Model:                           2
Covariance Type:            nonrobust
================================================================================
=====
                        coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
```

```
-----
ReadathonDuration      1.0027      0.008     125.670      0.000       0.987
1.018
Income                 1.0004      0.003     311.124      0.000       0.994
1.007
==============================================================================
Omnibus:                        1.085   Durbin-Watson:                   1.998
Prob(Omnibus):                  0.581   Jarque-Bera (JB):                1.104
Skew:                          -0.025   Prob(JB):                        0.576
Kurtosis:                       2.985   Cond. No.                         9.25
==============================================================================

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

[32]:
```python
# Print the fitted_estimators
gammas = gammas_model.params
print("The estimated causal effect on number of books read of an additional␣
↪Readathon day is {:.2f}".format(gammas[0]))
print("The estimated causal effect on number of books read of an additional␣
↪$10000 is {:.2f}".format(gammas[1]))
# The numbers you get should be very close to you answer in 1.b
```

```
The estimated causal effect on number of books read of an additional Readathon
day is 1.00
The estimated causal effect on number of books read of an additional $10000 is
1.00
```

4.4 1.c Estimate causal effect of NumBooks and Income on the SAT Score

Fill in the code below (similar as above) to estimate the causal effect of NumBooks and Income on the SAT Scores.

[33]:
```python
# Compute OLS estimators for beta_1 and beta_2
betas_model = fit_OLS_model(student_data, 'SAT', ['NumBooks', 'Income']) # TODO:
↪ fill in
print(betas_model.summary())
```

```
                                 OLS Regression Results
================================================================================
=======
Dep. Variable:                    SAT   R-squared (uncentered):
1.000
Model:                            OLS   Adj. R-squared (uncentered):
1.000
```

```
Method:                 Least Squares   F-statistic:
4.546e+07
Date:                Tue, 26 Oct 2021   Prob (F-statistic):
0.00
Time:                        10:01:34   Log-Likelihood:
-37348.
No. Observations:               10000   AIC:
7.470e+04
Df Residuals:                    9998   BIC:
7.471e+04
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
NumBooks       4.9814      0.012    398.950      0.000       4.957       5.006
Income        12.0265      0.017    690.963      0.000      11.992      12.061
==============================================================================
Omnibus:                        0.753   Durbin-Watson:                   2.020
Prob(Omnibus):                  0.686   Jarque-Bera (JB):                0.785
Skew:                           0.010   Prob(JB):                        0.675
Kurtosis:                       2.961   Cond. No.                         18.4
==============================================================================

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

[34]:
```python
# Print the fitted_estimators
betas = betas_model.params
print("The estimated causal effect on SAT score of an additional book read is {:
→.2f}".format(betas[0]))
print("The estimated causal effect on SAT score of an additional $10000 is {:.
→2f}".format(betas[1]))
# The numbers you get should be very close to you answer in 1.b
```

```
The estimated causal effect on SAT score of an additional book read is 4.98
The estimated causal effect on SAT score of an additional $10000 is 12.03
```

[35]:
```python
# Validation tests: Do not modify
assert np.abs(betas[0]-5)< 0.1
assert np.abs(betas[1]-12)< 0.1
print("Test passed!")
```

```
Test passed!
```

In Question 1 we saw how we can estimate all causal relationships if we have access to the income variable. However in our actual data we **do not observe Income**.

Goal: estimate β_1, the true causal effect of the number of books a student reads on their SAT score without access to the Income variable.

5 2. Naive OLS: OLS on the observed variables Z, Y.

The confounding variable X (family income) is unfortunately unobserved. We will start by somewhat "naively" attempting to estimate the causal effect β_1 by using plain linear regression (OLS) on the observed variables Z and Y. This time we will include an intercept term:

$$\hat{\beta}_1, \hat{c} = \arg\min_{\beta_1, c} \|Y - \beta_1 Z - c\|_2^2$$

5.1 2.a. Fit Naive OLS

[36]:
```
# TODO: Fit OLS parameters to predict Y from X_1.
beta_naive_model = fit_OLS_model(student_data, 'SAT', 'NumBooks',␣
↪intercept=True) # TODO:fill in
print(beta_naive_model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    SAT   R-squared:                       0.837
Model:                            OLS   Adj. R-squared:                  0.837
Method:                 Least Squares   F-statistic:                 5.117e+04
Date:                Tue, 26 Oct 2021   Prob (F-statistic):               0.00
Time:                        10:01:39   Log-Likelihood:                -56726.
No. Observations:               10000   AIC:                         1.135e+05
Df Residuals:                    9998   BIC:                         1.135e+05
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         42.8488      4.074     10.518      0.000      34.863      50.834
NumBooks      12.9591      0.057    226.209      0.000      12.847      13.071
==============================================================================
Omnibus:                        1.482   Durbin-Watson:                   2.011
Prob(Omnibus):                  0.477   Jarque-Bera (JB):                1.479
Skew:                           0.004   Prob(JB):                        0.477
Kurtosis:                       2.941   Cond. No.                         412.
==============================================================================

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

```
/opt/conda/lib/python3.9/site-packages/statsmodels/tsa/tsatools.py:142:
FutureWarning: In a future version of pandas all arguments of concat except for
the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
```

[37]:
```python
# Validation tests: Do not modify
params = beta_naive_model.params
assert len(params)==2
assert np.abs(params[0] - 42.85)<0.5
assert np.abs(params[1] - 12.96)<0.2
print('Test Passed!')
```

```
Test Passed!
```

[38]:
```python
print("The Naive OLS estimate of beta_1 is {:.2f}, while the true beta_1 is {}".
↪format(beta_naive_model.params[1], 5))
```

```
The Naive OLS estimate of beta_1 is 12.96, while the true beta_1 is 5
```

5.2 2.b Does the Naive approach overestimate or under estimate the value of reading books?

Answer this question by comparing the naive estimate and the true value of β_1

TODO:

It overestimates the value of reading books. This is because it predicts that the value of reading a book is 12.96 points, while it is only 5 points.

6 3. Instrumental variables and 2SLS

To eliminate the bias, we turn to instrumental variables. In the first stage, we “predict” the number of books a student read from whether or not they had a readathon, W, producing an estimate $\hat{Z}$. Then, in the second stage, we regress the SAT score Y onto the predicted number of books read $\hat{Z}$.

6.1 Stage 1: Predict treatment variable $\hat{Z}$ from instrumental variable W

[39]:
```python
# No TODOs here, just run this call and understand what this function is doing.
def compute_OLS_predictions(input_array, input_params):
    """Calculates OLS predictions from fitted OLS parameters, input_params.

    Args:
      input_array: numpy array with n entries, where each entry corresponds␣
↪with a feature value for a given student.
      input_params: numpy array with 2 entries, where the entries are␣
↪[intercept, beta_hat].
        The intercept is a constant term, so the final OLS predictions should␣
↪be
        predictions = intercept + beta_hat*input_array.
```

```
    Returns:
      numpy array with n entries containing predictions from input_array.
    """
    predictions = input_params[0] + input_params[1] * input_array
    return predictions
```

6.2 3.a Predict $\hat{Z}$

6.2.1 3.a.i Complete the code below to fit an OLS model that predicts $\hat{Z}$ (estimated number of books read) using W (whether they had a readathon).

$$\hat{\gamma}_1, \hat{c} = \arg\min_{\gamma_1, c} \|Z - \gamma_1 W - c\|_2^2$$

[40]:
```
# TODO: Fit OLS parameters to predict X_1 from Z
gamma1_model = fit_OLS_model(student_data, 'NumBooks', 'ReadathonDuration',␣
 ↪intercept=True) # TODO: fill in
print(gamma1_model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:               NumBooks   R-squared:                       0.162
Model:                            OLS   Adj. R-squared:                  0.162
Method:                 Least Squares   F-statistic:                     1927.
Date:                Tue, 26 Oct 2021   Prob (F-statistic):               0.00
Time:                        10:01:44   Log-Likelihood:                -38391.
No. Observations:               10000   AIC:                         7.679e+04
Df Residuals:                    9998   BIC:                         7.680e+04
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
=====
                        coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-----
const                50.1628      0.467    107.518      0.000      49.248
51.077
ReadathonDuration     0.9963      0.023     43.903      0.000       0.952
1.041
==============================================================================
Omnibus:                        0.272   Durbin-Watson:                   2.014
Prob(Omnibus):                  0.873   Jarque-Bera (JB):                0.240
Skew:                          -0.001   Prob(JB):                        0.887
Kurtosis:                       3.024   Cond. No.                         85.5
==============================================================================
```

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

/opt/conda/lib/python3.9/site-packages/statsmodels/tsa/tsatools.py:142:
FutureWarning: In a future version of pandas all arguments of concat except for
the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
```

[41]:
```python
# Validation tests: Do not modify
params = gamma1_model.params
assert len(params)==2
assert np.abs(params[0] - 50.16)<0.5
assert np.abs(params[1] - 1)<0.1
print('Test Passed!')
```

```
Test Passed!
```

[42]:
```python
print("The OLS estimate of gamma_1 is {:.3f}, while the true gamma_1 is {}".
 ↪format(gamma1_model.params[1], 1))
```

```
The OLS estimate of gamma_1 is 0.996, while the true gamma_1 is 1
```

6.2.2 3.a.ii We observe that the estimate of γ_1 above is very close to the true value, even though we don't make use of the `Income` variable. How can you explain this?

Hint: Think about independence

TODO: Here, we are using W (whether they had a readathon) to predict Z (estimated number of books read). The reason this is very close to the true value despite not using X (income) is because W is independent of X. Due to this, we can just use the data we have for W to predict Z and get an accurate prediction.

6.2.3 Now we can use the OLS model above to create $\hat{Z}$ predictions

[43]:
```python
# Compute predictions for number of books read
intercept_OLS = gamma1_model.params[0]
gamma1_OLS = gamma1_model.params[1]
X_1_hat = intercept_OLS + gamma1_OLS*student_data['ReadathonDuration']

# Add the predictions to the student_data dataframe
student_data['PredictedNumBooks'] = X_1_hat
student_data.head()
```

[43]:

	NumBooks	Income	SAT	ReadathonDuration	PredictedNumBooks
0	54.0	45.473122	799.0	16.0	66.104173
1	71.0	53.064868	991.0	15.0	65.107836
2	79.0	52.522466	1017.0	19.0	69.093186

```
3       92.0   61.322441   1197.0                23.0            73.078536
4       65.0   53.453239    982.0                12.0            62.118823
```

6.3 Stage 2: Estimate target Y from predicted treatment variable $\hat{X}_1$

6.3.1 3.b Fit OLS parameters to predict Y from the predicted $\hat{Z}$

$$\hat{\beta}_1, \hat{c} = \arg\min_{\beta_1, c} \|Y - \beta_1 \hat{Z} - c\|_2^2$$

[44]:
```
# TODO: Fit OLS parameters to predict Y from the predicted X_1_hat.
beta1_model = fit_OLS_model(student_data, 'SAT', 'PredictedNumBooks',␣
↪intercept=True) # TODO: fill in
print(beta1_model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    SAT   R-squared:                       0.019
Model:                            OLS   Adj. R-squared:                  0.019
Method:                 Least Squares   F-statistic:                     191.8
Date:                Tue, 26 Oct 2021   Prob (F-statistic):           3.24e-43
Time:                        10:01:54   Log-Likelihood:                -65687.
No. Observations:               10000   AIC:                         1.314e+05
Df Residuals:                    9998   BIC:                         1.314e+05
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
=====
                        coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-----
const               611.8673     24.515     24.959      0.000     563.814
659.921
PredictedNumBooks     4.8351      0.349     13.849      0.000       4.151
5.519
==============================================================================
Omnibus:                        0.460   Durbin-Watson:                   2.010
Prob(Omnibus):                  0.794   Jarque-Bera (JB):                0.424
Skew:                           0.003   Prob(JB):                        0.809
Kurtosis:                       3.031   Cond. No.                         999.
==============================================================================

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

/opt/conda/lib/python3.9/site-packages/statsmodels/tsa/tsatools.py:142:
FutureWarning: In a future version of pandas all arguments of concat except for
```

```
the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
```

[45]:
```
# Validation tests: Do not modify
params = beta1_model.params
assert len(params)==2
assert np.abs(params[0] - 612)<5
assert np.abs(params[1] - 4.84)<0.3
print('Test Passed!')
```

```
Test Passed!
```

[46]:
```
print("The 2SLS estimate of beta_1 is {:.3f}, while the true value is {}".
 ↪format(beta1_model.params[1], 5))
```

```
The 2SLS estimate of beta_1 is 4.835, while the true value is 5
```

6.4 3.c. Answer the following conceptual questions:

6.4.1 i) Which technique produced a better estimate of β_1, naive OLS or 2SLS?

6.4.2 ii) Give a plausible scenario where the organizing a Readathon would not serve as an appropriate Intrumental Variable (IV).

Hint Recall what properties should an Instrumental Variable satisfy.

i) `TODO`: The Naive OLS estimate of beta_1 is 12.96, while the true beta_1 is 5. The 2SLS estimate of beta_1 is 4.835, while the true value is 5. Hence, the 2SLS technique produced a better estimate.

ii) `TODO`: Organizing a readathon would not serve as an appropriate IV if it was not independent of income i.e. if some dependence between readathons and income existed (one would expect that higher incomes -> longer readathons).

[47]:
```
import matplotlib.image as mpimg
img = mpimg.imread('cute_gecko.jpg')
imgplot = plt.imshow(img)
imgplot.axes.get_xaxis().set_visible(False)
imgplot.axes.get_yaxis().set_visible(False)
print("Yay, you've made it to the end of Lab 7!")
plt.show()
```

```
Yay, you've made it to the end of Lab 7!
```

```
[ ]:
```