Worksheet 18

Name: Jeong Yong Yang, Junho Son UID: U95912941, U64222022

Topics

- Linear Regression

Optional Challenge

Every day my alarm goes off at seemingly random times… I've recorded the times at which it goes off for the past year of so (1 - 355 days). Today is day 356. Can you predict when my alarm will ring using `data.csv`?

Please fill out the piazza poll if you think you found the answer.

Linear Regression

Where does randomness come from?

```
1  pip install ipympl
```

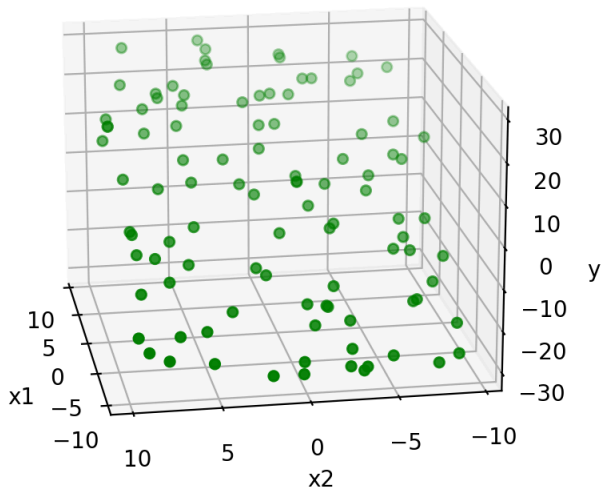
```
Collecting ipympl
  Downloading ipympl-0.9.3-py2.py3-none-any.whl (511 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 511.6/511.6 kB 6.0 MB/s eta 0:00:00
Requirement already satisfied: ipython<9 in /usr/local/lib/python3.10/dist-packages (from ipympl) (7.34.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from ipympl) (1.25.2)
Requirement already satisfied: ipython-genutils in /usr/local/lib/python3.10/dist-packages (from ipympl) (0.2.0)
Requirement already satisfied: pillow in /usr/local/lib/python3.10/dist-packages (from ipympl) (9.4.0)
Requirement already satisfied: traitlets<6 in /usr/local/lib/python3.10/dist-packages (from ipympl) (5.7.1)
Requirement already satisfied: ipywidgets<9,>=7.6.0 in /usr/local/lib/python3.10/dist-packages (from ipympl) (7.7
Requirement already satisfied: matplotlib<4,>=3.4.0 in /usr/local/lib/python3.10/dist-packages (from ipympl) (3.7
Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.10/dist-packages (from ipython<9->ipymp
Collecting jedi>=0.16 (from ipython<9->ipympl)
  Downloading jedi-0.19.1-py2.py3-none-any.whl (1.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 45.8 MB/s eta 0:00:00
Requirement already satisfied: decorator in /usr/local/lib/python3.10/dist-packages (from ipython<9->ipympl) (4.4
Requirement already satisfied: pickleshare in /usr/local/lib/python3.10/dist-packages (from ipython<9->ipympl) (0
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: pygments in /usr/local/lib/python3.10/dist-packages (from ipython<9->ipympl) (2.16
Requirement already satisfied: backcall in /usr/local/lib/python3.10/dist-packages (from ipython<9->ipympl) (0.2.
Requirement already satisfied: matplotlib-inline in /usr/local/lib/python3.10/dist-packages (from ipython<9->ipym
Requirement already satisfied: pexpect>4.3 in /usr/local/lib/python3.10/dist-packages (from ipython<9->ipympl) (4
Requirement already satisfied: ipykernel>=4.5.1 in /usr/local/lib/python3.10/dist-packages (from ipywidgets<9,>=7
Requirement already satisfied: widgetsnbextension~=3.6.0 in /usr/local/lib/python3.10/dist-packages (from ipywidg
Requirement already satisfied: jupyterlab-widgets>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from ipywidg
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib<4,>=3
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib<4,>=3.4.0
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib<4,>=
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib<4,>=
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib<4,>=3.
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib<4,>=3
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib<4
Requirement already satisfied: jupyter-client in /usr/local/lib/python3.10/dist-packages (from ipykernel>=4.5.1->
Requirement already satisfied: tornado>=4.2 in /usr/local/lib/python3.10/dist-packages (from ipykernel>=4.5.1->ip
Requirement already satisfied: parso<0.9.0,>=0.8.3 in /usr/local/lib/python3.10/dist-packages (from jedi>=0.16->i
Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.10/dist-packages (from pexpect>4.3->ipyt
Requirement already satisfied: wcwidth in /usr/local/lib/python3.10/dist-packages (from prompt-toolkit!=3.0.0,!=3
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->ma
Requirement already satisfied: notebook>=4.4.1 in /usr/local/lib/python3.10/dist-packages (from widgetsnbextensio
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from notebook>=4.4.1->widgetsnb
Requirement already satisfied: pyzmq<25,>=17 in /usr/local/lib/python3.10/dist-packages (from notebook>=4.4.1->wi
Requirement already satisfied: argon2-cffi in /usr/local/lib/python3.10/dist-packages (from notebook>=4.4.1->widg
Requirement already satisfied: jupyter-core>=4.6.1 in /usr/local/lib/python3.10/dist-packages (from notebook>=4.4
Requirement already satisfied: nbformat in /usr/local/lib/python3.10/dist-packages (from notebook>=4.4.1->widgets
Requirement already satisfied: nbconvert>=5 in /usr/local/lib/python3.10/dist-packages (from notebook>=4.4.1->wid
Requirement already satisfied: nest-asyncio>=1.5 in /usr/local/lib/python3.10/dist-packages (from notebook>=4.4.1
Requirement already satisfied: Send2Trash>=1.8.0 in /usr/local/lib/python3.10/dist-packages (from notebook>=4.4.1
Requirement already satisfied: terminado>=0.8.3 in /usr/local/lib/python3.10/dist-packages (from notebook>=4.4.1-
Requirement already satisfied: prometheus-client in /usr/local/lib/python3.10/dist-packages (from notebook>=4.4.1
```

```
Requirement already satisfied: nbclassic>=0.4.7 in /usr/local/lib/python3.10/dist-packages (from notebook>=4.4.1-
Requirement already satisfied: platformdirs>=2.5 in /usr/local/lib/python3.10/dist-packages (from jupyter-core>=4
Requirement already satisfied: jupyter-server>=1.8 in /usr/local/lib/python3.10/dist-packages (from nbclassic>=0.
Requirement already satisfied: notebook-shim>=0.2.3 in /usr/local/lib/python3.10/dist-packages (from nbclassic>=0
Requirement already satisfied: lxml in /usr/local/lib/python3.10/dist-packages (from nbconvert>=5->notebook>=4.4.
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-packages (from nbconvert>=5->note
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages (from nbconvert>=5->notebook>=4.
Requirement already satisfied: defusedxml in /usr/local/lib/python3.10/dist-packages (from nbconvert>=5->notebook
Requirement already satisfied: entrypoints>=0.2.2 in /usr/local/lib/python3.10/dist-packages (from nbconvert>=5->
Requirement already satisfied: jupyterlab-pygments in /usr/local/lib/python3.10/dist-packages (from nbconvert>=5-
```

```
from google.colab import output
output.enable_custom_widget_manager()
```

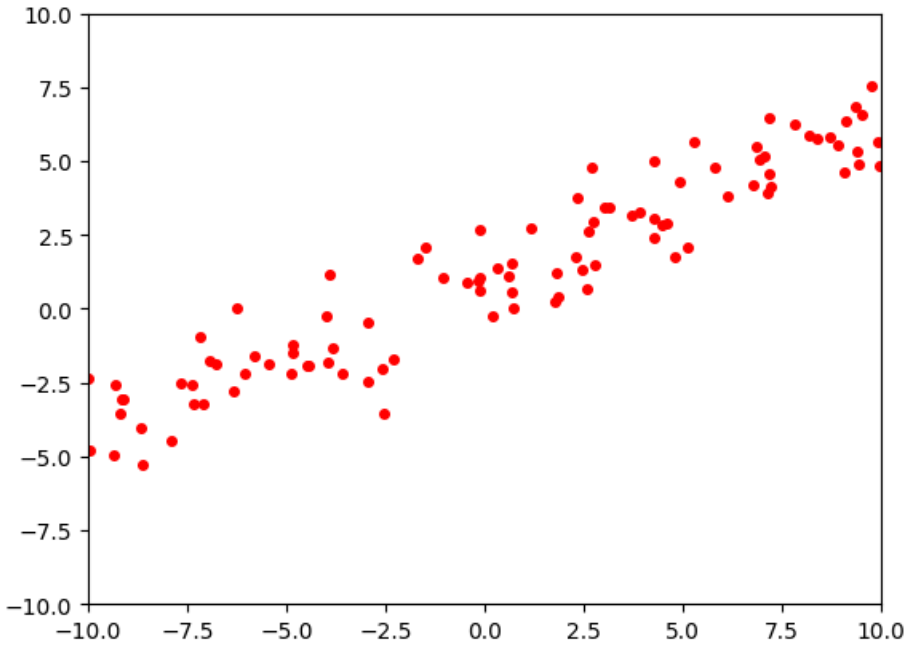
```
%matplotlib widget
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

SAMPLE_SIZE = 100

# it's possible for y = 3 * x_1 + (1/4) * x_2
# but we don't know to look for x_2 and we only have x_1 as a feature
x1 = -10.0 + 20.0 * np.random.random(SAMPLE_SIZE)
x2 = -10.0 + 20.0 * np.random.random(SAMPLE_SIZE)
y = 3 * x1 + (1/4) * x2

# Create the figure
fig = plt.figure()

# Add an axes
ax = fig.add_subplot(111,projection='3d')

# and plot the point
ax.scatter(x1 , x2 , y,  color='green')
ax.set_xlabel("x1")
ax.set_ylabel("x2")
ax.set_zlabel("y")
plt.show()
```

Figure 1



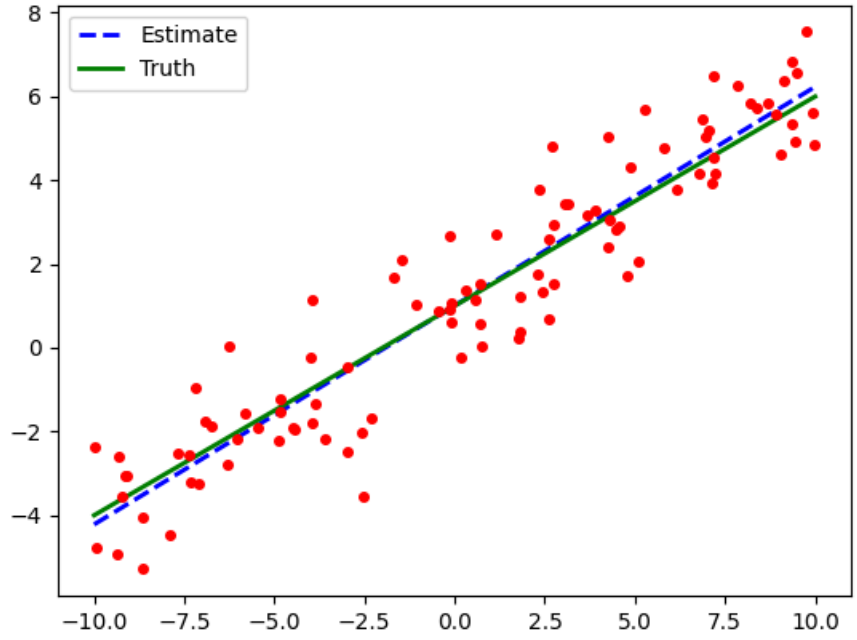
a) Create and plot a dataset of 100 (Y, X) points where `Y = 1 + 0.5 * X + eps`.

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

SAMPLE_SIZE = 100

fig = plt.figure()
ax = fig.subplots()

true_beta = np.array([ 1 , 0.5 ])
x = -10.0 + 20.0 * np.random.random(SAMPLE_SIZE)
y = true_beta[0] + true_beta[1] * x + np.random.randn(SAMPLE_SIZE)
"""
x_plot = np.linspace(-10,10,50)
plt.plot(x_plot, true_beta[0] + true_beta[1] * x_plot, lw=2, c='g', label='Truth')
"""

plt.plot(x,y,'ro',markersize=4)
ax.set_xlim(-10, 10)
ax.set_ylim(-10,10)
plt.show()
```



b) Plot the least squares estimate line through the scatter plot.

```
intercept = np.ones(np.shape(x)[0])
X = np.array([intercept, x]).T
beta_hat = np.linalg.inv(X.T @ X) @ X.T @ y

x_plot = np.linspace(-10,10,50)
y_est = beta_hat[0] + beta_hat[1] * x_plot
plt.plot(x_plot, y_est,'b--', lw=2, label='Estimate')
plt.plot(x_plot, true_beta[0] + true_beta[1] * x_plot, lw=2, c='g', label='Truth')
plt.plot(x, y,'ro',markersize=4)
plt.legend()

plt.show()
```



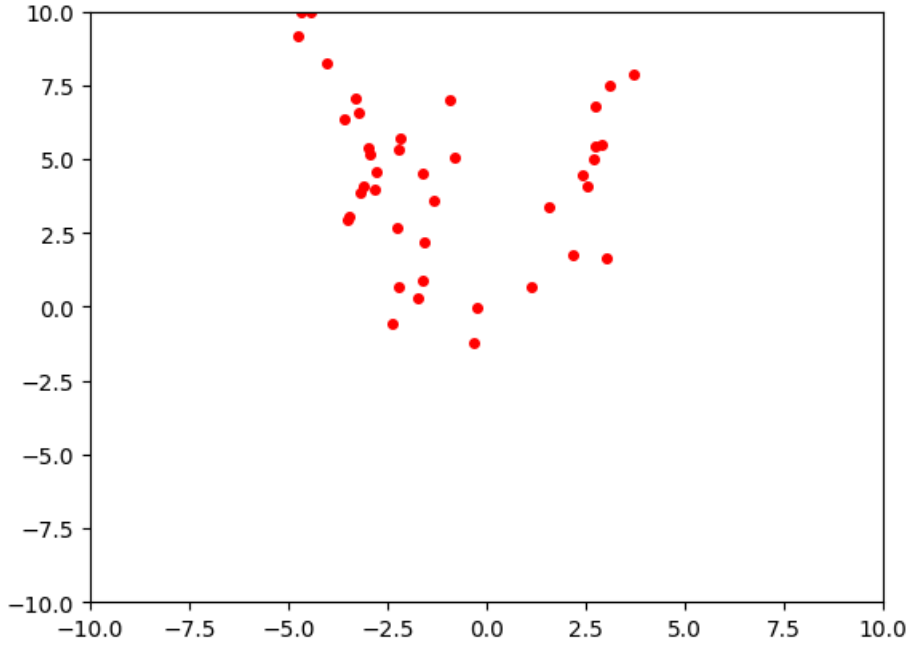
c) How does the estimate of `beta` compare to the parameters we used to generate the data?

```
print(f"The true beta is {true_beta}")
print(f"The estimated beta is {beta_hat}")
```

```
The true beta is [1.  0.5]
The estimated beta is [1.00651849 0.52254326]
```

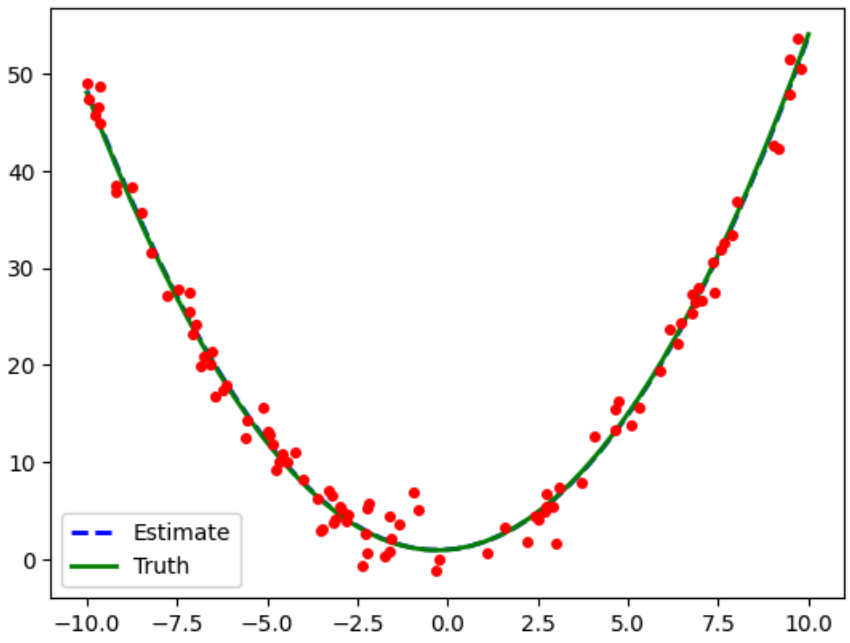
d) Create and plot a dataset of 100 (Y, X) points where `Y = 1 + 3 * X + .5 * X^2 + eps`.

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

SAMPLE_SIZE = 100

fig = plt.figure()
ax = fig.subplots()

true_beta = np.array([ 1 , 0.3, 0.5 ])
x = -10.0 + 20.0 * np.random.random(SAMPLE_SIZE)
y = true_beta[0] + true_beta[1] * x + true_beta[2] * (x**2) + np.random.randn(SAMPLE_SIZE) * 2
"""
x_plot = np.linspace(-10,10,50)
plt.plot(x_plot, true_beta[0] + true_beta[1] * x_plot, lw=2, c='g', label='Truth')
"""

plt.plot(x,y,'ro',markersize=4)
ax.set_xlim(-10, 10)
ax.set_ylim(-10,10)
plt.show()
```



e) Plot the least squares estimate line through the scatter plot.

```
intercept = np.ones(np.shape(x)[0])
X = np.array([intercept, x, x**2]).T
beta_hat = np.linalg.inv(X.T @ X) @ X.T @ y

x_plot = np.linspace(-10,10,50)
y_est = beta_hat[0] + beta_hat[1] * x_plot + beta_hat[2] * (x_plot ** 2)
plt.plot(x_plot, y_est,'b--', lw=2, label='Estimate')
plt.plot(x_plot, true_beta[0] + true_beta[1] * x_plot + true_beta[2] * (x_plot ** 2), lw=2, c='g', label='Truth')
plt.plot(x, y,'ro',markersize=4)
plt.legend()

plt.show()
```



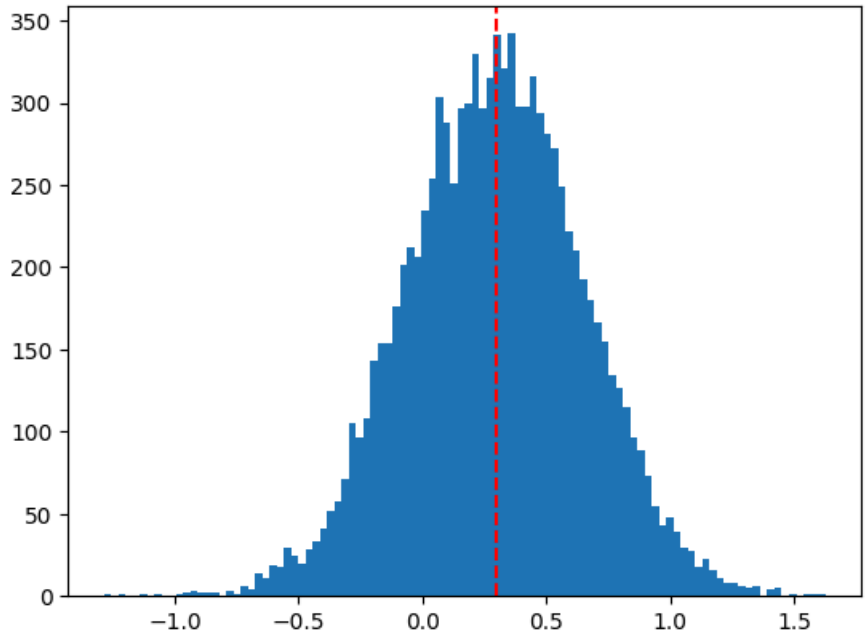
f) How does the estimate of `beta` compare to the parameters we used to generate the data?

```
print(f"The true beta is {true_beta}")
print(f"The estimated beta is {beta_hat}")
```

```
The true beta is [1.  0.3 0.5]
The estimated beta is [1.01106703 0.27819109 0.49990981]
```

g) Let's repeat d) and f) a large number of times to see how close our estimates are on average and what that distribution looks like.

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

betas = []
true_beta = np.array([ 1, 0.3, 0.5 ])
for _ in range(10000):
    x = -1 + 2 * np.random.random(SAMPLE_SIZE)
    y = true_beta[0] + true_beta[1] * x + true_beta[2] * (x**2) + np.random.randn(SAMPLE_SIZE) * 2
    intercept = np.ones(np.shape(x)[0])
    X = np.array([intercept, x, x ** 2]).T
    betas.append(np.linalg.inv(X.T @ X) @ X.T @ y)

plt.hist(np.array(betas)[:, 1], bins=100)
plt.axvline(x=true_beta[1], c='r', linestyle='dashed')
plt.show()
```




```
코딩을 시작하거나 AI로 코드를 생성하세요.
```