```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
```

```
# Matplotlib and seaborn for visualization
import matplotlib.pyplot as plt
%matplotlib inline

import seaborn as sns

# Linear Regression to verify implementation
from sklearn.linear_model import LinearRegression

# Scipy for statistics
import scipy

# PyMC3 for Bayesian Inference
import pymc3 as pm
```

```
exercise = pd.read_csv('/content/exercise.csv')
calories = pd.read_csv('/content/calories.csv')
df = pd.merge(exercise, calories, on = 'User_ID')
df = df[df['Calories'] < 300]
df = df.reset_index()
df['Intercept'] = 1
df.head()
```

| | index | User_ID | Gender | Age | Height | Weight | Duration | Heart_Rate | Body_Temp | Calories | Intercept |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 14733363 | male | 68 | 190.0 | 94.0 | 29.0 | 105.0 | 40.8 | 231 | 1 |
| **1** | 1 | 14861698 | female | 20 | 166.0 | 60.0 | 14.0 | 94.0 | 40.3 | 66 | 1 |
| **2** | 2 | 11179863 | male | 69 | 179.0 | 79.0 | 5.0 | 88.0 | 38.7 | 26 | 1 |
| **3** | 3 | 16180408 | female | 34 | 179.0 | 71.0 | 13.0 | 100.0 | 40.5 | 71 | 1 |
| **4** | 4 | 17771927 | female | 27 | 154.0 | 58.0 | 10.0 | 81.0 | 39.8 | 35 | 1 |

```
# Create the features and response
X = df.loc[:, ['Intercept', 'Duration']]
y = df.loc[:, 'Calories']
```

```
from sklearn.linear_model import LinearRegression
reg = LinearRegression().fit(X, y)
print('Coefficient of determination or R2:\n',round(reg.score(X, y),3))
```

```
Coefficient of determination or R2:
 0.913
```

```
print('Coefficients: \n', reg.coef_)
```

```
Coefficients: 
 [0.         7.16978335]
```

```
print('Intercept:\n', reg.intercept_)
```

```
Intercept:
 -21.8281025260508
```

Build the model with 500 observations and then all observations.

```
with pm.Model() as linear_model_500:
    # Intercept
    intercept = pm.Normal('Intercept', mu = 0, sd = 10)

    # Slope
    slope = pm.Normal('slope', mu = 0, sd = 10)

    # Standard deviation
    sigma = pm.HalfNormal('sigma', sd = 10)

    # Estimate of mean
    mean = intercept + slope * X.loc[0:499, 'Duration']

    # Observed values
    # Observed values
    Y_obs = pm.Normal('Y_obs', mu = mean, sd = sigma, observed = y.values[0:500])

    # Sampler
    step = pm.NUTS()

    # Posterior distribution
    linear_trace_500 = pm.sample(1000, step)
```

```
INFO:pymc3:Sequential sampling (2 chains in 1 job)
INFO:pymc3:NUTS: [sigma, slope, Intercept]
100%|██████████| 1500/1500 [00:02<00:00, 642.71it/s]
100%|██████████| 1500/1500 [00:02<00:00, 611.80it/s]
WARNING:pymc3:The acceptance probability does not match the target. It is 0.8803147835931578, but should be close to
WARNING:pymc3:The acceptance probability does not match the target. It is 0.9061663130361706, but should be close to
```

```
with pm.Model() as linear_model:
    # Intercept
    intercept = pm.Normal('Intercept', mu = 0, sd = 10)

    # Slope
    slope = pm.Normal('slope', mu = 0, sd = 10)

    # Standard deviation
    sigma = pm.HalfNormal('sigma', sd = 10)

    # Estimate of mean
    mean = intercept + slope * X.loc[:, 'Duration']

    # Observed values
    Y_obs = pm.Normal('Y_obs', mu = mean, sd = sigma, observed = y.values)

    # Sampler
    step = pm.NUTS()

    # Posterior distribution
    linear_trace = pm.sample(1000, step)
```

```
INFO:pymc3:Sequential sampling (2 chains in 1 job)
INFO:pymc3:NUTS: [sigma, slope, Intercept]
100%|██████████| 1500/1500 [00:12<00:00, 117.01it/s]
100%|██████████| 1500/1500 [00:09<00:00, 159.42it/s]
WARNING:pymc3:The acceptance probability does not match the target. It is 0.9895795767051766, but should be close to
WARNING:pymc3:The acceptance probability does not match the target. It is 0.9790896525672083, but should be close to
```

Compute posterior prediction for 15.5 minutes

```
bayes_prediction = linear_trace['Intercept'] + linear_trace['slope'] * 15.5
```

```
print('Prediction:\n:', bayes_prediction)
```

```
Prediction:
: [89.28474968 89.30393246 89.33247202 ... 89.35222443 89.23276362
 89.21184103]
```