Bayesean Data Analysis - Exercise 7

November 4, 2018

1 Linear model: drowning data with Stan

This is how the modified Stan code looks like:

```
data {
    int <lower=0> N; //number of data points
    vector[N] x; //observation year
    vector[N] y; //observation number of drowned
    real xpred; //prediction year
    real pmubeta;
    real psbeta;
}

parameters {
    real alpha;
    real beta;
    real <lower=0> sigma;
}

transformed parameters {
    vector[N] mu;
    mu = alpha + beta*x;
}

model {
    beta ~ normal(pmubeta, psbeta);
    y ~ normal(mu, sigma);
}

generated quantities {
    real ypred;
    ypred = normal_rng(alpha + beta*xpred, sigma);
}
```

The code that was provided for us has two mistakes in it.
The first one is in the declaration of **sigma**:

```
real sigma;
```

which is not correct because the standard deviation can't have negative values and needs to have a lower bound of 0.
Here is how we fix that:

```
real <lower=0> sigma;
```

The second mistake is in the calculation of the **ypred**, where it is given as:

```
ypred = normal_rng(mu, sigma);
```

and it should be modified to look like this:

```
ypred = normal_rng(alpha + beta*xpred, sigma);
```

because it is a linear model and the formula should be $y = a + bx$.

It is also stated that we should use weakly informative prior $N(0, \tau^2)$. The τ value that I am using is **26.788**. The τ value is calculated with the following Python code:

```
alpha_val = 0
tau = 26.788
beta_val = tau
prior = norm(alpha_val, beta_val)
a = np.arange(-1000, 1000, 0.1)
b = prior.pdf(a)
print(prior.interval(0.99))
```

The prior is implemented in the following way:
First we declare mean and std for beta in the **data** block as follows:

```
real pmubeta;
real psbeta;
```

then in the **model** block we calculate beta:

```
beta ~ normal(pmubeta, psbeta);
```

We also need to add it as a parameter in the **parameter** block:

```
real beta;
```

From the prediction we can see that the predicted value from drownings in 2019 is **120**.
We can also that the number of drownings is **decreasing** every year.

```
import pystan
import numpy as np
import pandas as pd
from scipy.stats import norm, beta
import matplotlib.pyplot as plt
import matplotlib as mpl

df = pd.read_csv('drowning.csv', sep=' ', header=None)
year = df[0].values
number_deaths = df[1].values
number_datapoints = len(df)
prediction_year = 2019

alpha_val = 0
tau = 26.788
beta_val = tau
prior = norm(alpha_val, beta_val)
a = np.arange(-1000, 1000, 0.1)
b = prior.pdf(a)
print(prior.interval(0.99))


drowning_code = '''
data {
    int <lower=0> N; //number of data points
    vector[N] x; //observation year
    vector[N] y; //observation number of drowned
    real xpred; //prediction year
    real pmubeta;
    real psbeta;
}

parameters {
    real alpha;
    real beta;
    real <lower=0> sigma;
}

transformed parameters {
    vector[N] mu;
    mu = alpha + beta*x;
}

model {
    beta ~ normal(pmubeta, psbeta);
    y ~ normal(mu, sigma);
}

generated quantities {
    real ypred;
    ypred = normal_rng(alpha + beta*xpred, sigma);
}
'''

drowning_data = {'N': number_datapoints,
```

```
                'x': year,
                'y': number_deaths,
                 'xpred': prediction_year,
                 'pmubeta': 0,
                 'psbeta': tau

               }

sm = pystan.StanModel(model_code = drowning_code)
fit = sm.sampling(data = drowning_data)
```

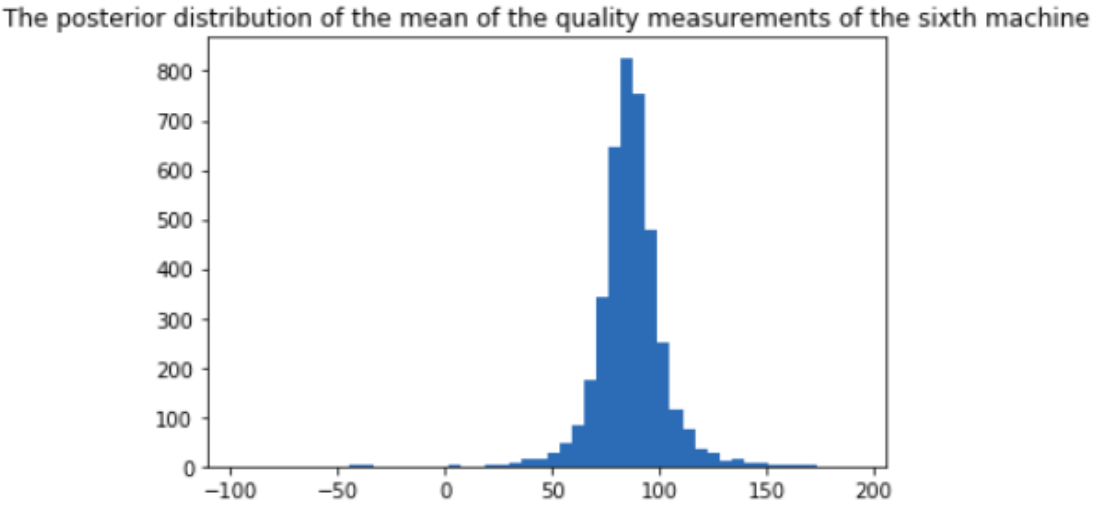
2 Hierarchical model: factory data with Stan

2.1 Separate model

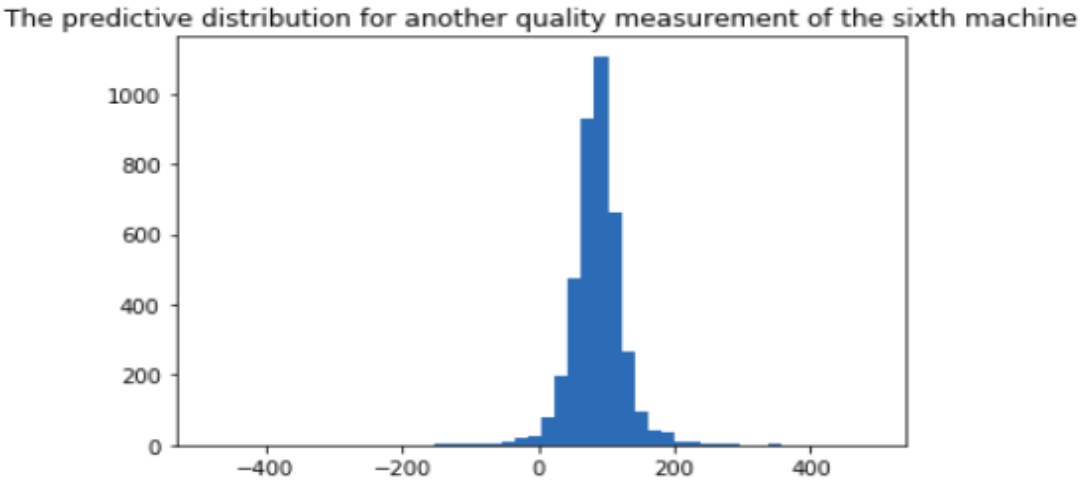
Here is that Stan code for the separate model:

```
data {
    int<lower=0> N; // number of data points
    int<lower=0> K; // number of groups
    int<lower=1,upper=K> x[N]; // group indicator
    vector[N] y; //
}

parameters {
    vector[K] mu;       // group means
    //real<lower=0> sigma;
    vector<lower=0>[K] sigma;
}

model {
  y ~ normal(mu[x], sigma[x]);
}

generated quantities {
    real ypred;
    ypred = normal_rng(mu[6], sigma[6]);
}
```

The posterior mean for the sixth machine is **86.341**



The mean for another quality measurement of the sixth machine is **86.322**



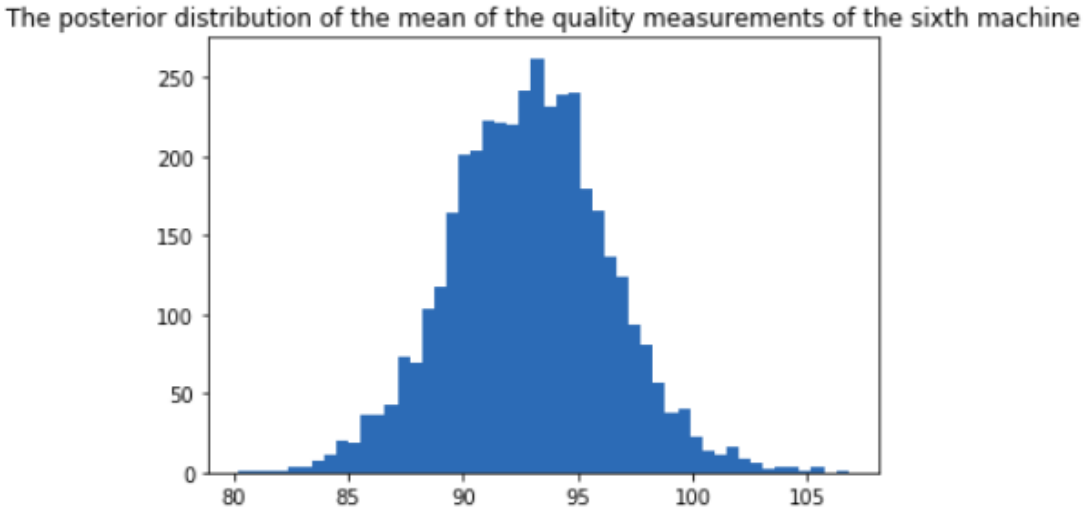
The posterior distribution of the mean of the quality measurements of the seventh machine can't be calculated because we are calculating for every machine separately and there is not data for the seventh machine.

2.2 Pooled model

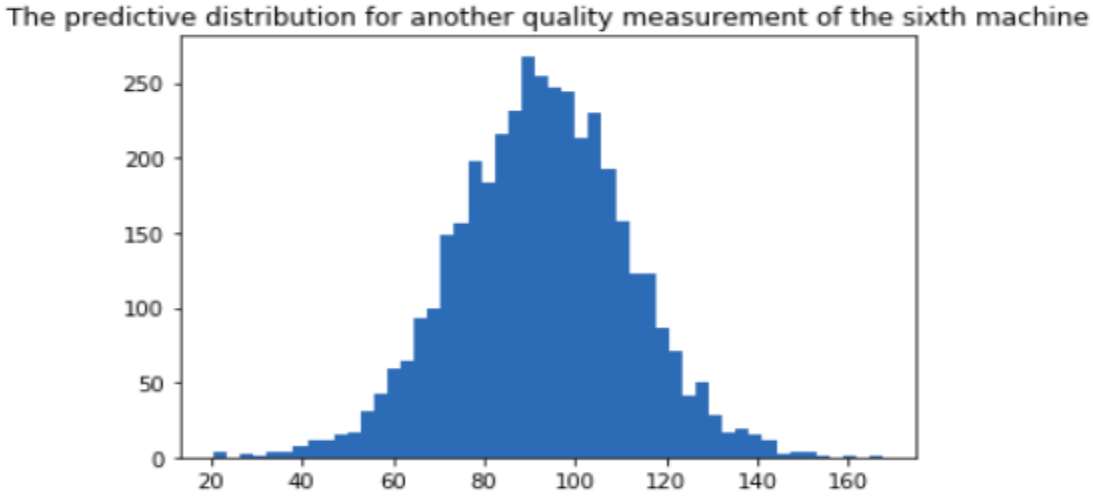
Here is that Stan code for the pooled model:

```
data {
    int<lower=0> N; // number of data points
    vector[N] y;
}

parameters {
    real mu;           // prior mean
    real<lower=0> sigma;
}

model {
  y ~ normal(mu, sigma);
}

generated quantities {
    real ypred;
    ypred = normal_rng(mu, sigma);
}
```

The posterior mean for the sixth machine is **92.903**



The mean for another quality measurement of the sixth machine is **92.637**



The posterior distribution of the mean of the quality measurements of the seventh machine is the same and for the sixth machine because all the measurements are combined together.

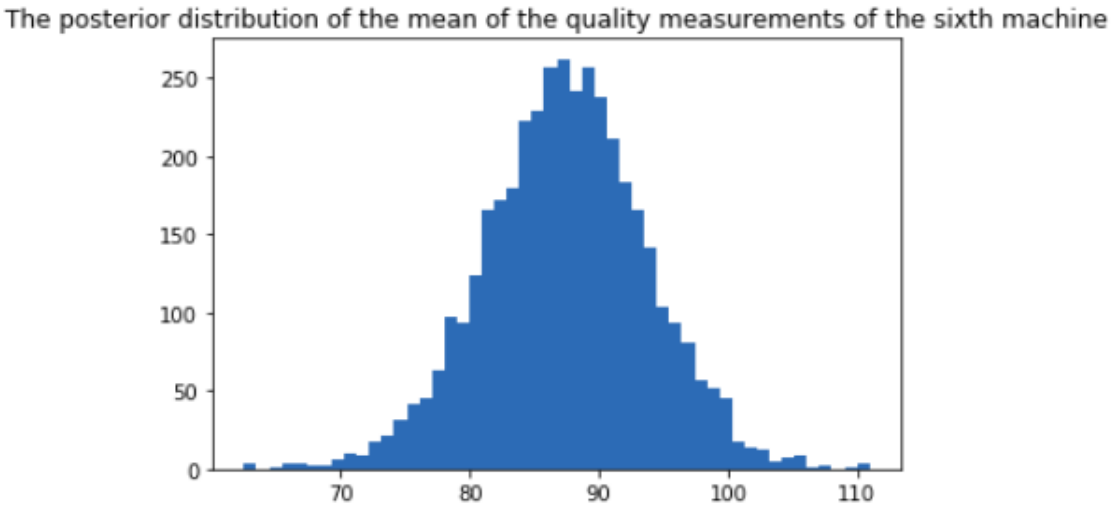
2.3 Hierarchical model

Here is that Stan code for the hierarchical model:

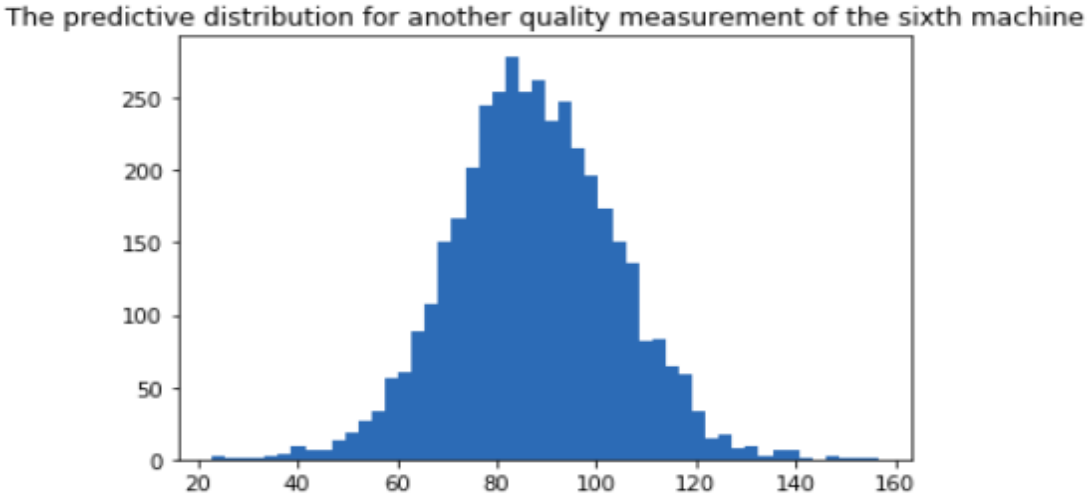
```
data {
    int<lower=0> N; // number of data points
    int<lower=0> K; // number of groups
    int<lower=1,upper=K> x[N]; // group indicator
    vector[N] y; //
}

parameters {
    real mu0;           // prior mean
    real<lower=0> sigma0; // prior std
```

```
    vector[K] mu;       // group means
    real<lower=0> sigma;
}

model {
    mu ~ normal(mu0, sigma0);
    y ~ normal(mu[x], sigma);
}

generated quantities {
    real mu7;
    real ypred6;

    mu7 = normal_rng(mu0, sigma0);
    ypred6 = normal_rng(mu[6], sigma);
}
```

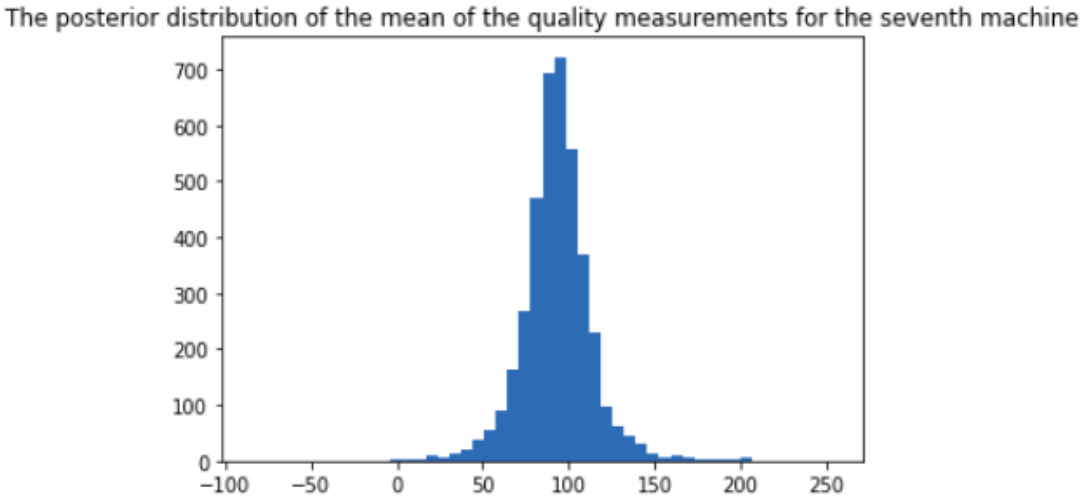
The posterior mean for the sixth machine is **87.456**



The mean for another quality measurement of the sixth machine is **87.528**



The posterior distribution of the mean of the quality measurements of the seventh machine is **93.037**.



```
import pystan
import numpy as np
import pandas as pd
from scipy.stats import norm, beta
import matplotlib.pyplot as plt
import matplotlib as mpl


# POOLED MODEL
df = pd.read_csv('factory.csv', sep=' ', header=None)
number_datapoints = df.shape[0] * df.shape[1]
# num_groups = df.shape[1]
y = df.values
y = y.ravel()
x = np.arange(1, df.shape[1]+1, 1)

pooled_factory_code = '''
data {
    int<lower=0> N; // number of data points
    vector[N] y;
}

parameters {
    real mu;           // prior mean
    real<lower=0> sigma;
}

model {
  y ~ normal(mu, sigma);
}

generated quantities {
    real ypred;
    ypred = normal_rng(mu, sigma);
}
'''


pooled_factory_data = {'N': number_datapoints,
               'y': y
              }

sm = pystan.StanModel(model_code = pooled_factory_code)
pooled_fit = sm.sampling(data = pooled_factory_data)


summary = pooled_fit.summary()
summary = pd.DataFrame(summary['summary'], columns=summary['summary_colnames'], index=summary['summary_rownames'])

mu6 = pooled_fit.extract(permuted=True)['mu']
plt.hist(mu6, bins=50)
plt.title('The posterior distribution of the mean of the quality measurements of the sixth machine')
plt.show()

ypred = pooled_fit.extract(permuted=True)['ypred']
```

```
plt.hist(ypred, bins=50)
plt.title('The predictive distribution for another quality measurement of the sixth machine')
plt.show()
summary


# HIERARCHICAL MODEL
df = pd.read_csv('factory.csv', sep=' ', header=None)
number_datapoints = df.shape[0] * df.shape[1]
num_groups = df.shape[1]
y = df.values
y = np.reshape(y, 30, order='F')
y = y.ravel()
x = np.arange(1,7)
x = np.repeat(x, 5)

hierarchical_factory_code = '''
data {
    int<lower=0> N; // number of data points
    int<lower=0> K; // number of groups
    int<lower=1,upper=K> x[N]; // group indicator
    vector[N] y; //
}

parameters {
    real mu0;           // prior mean
    real<lower=0> sigma0; // prior std

    vector[K] mu;       // group means
    real<lower=0> sigma;
}

model {
    mu ~ normal(mu0, sigma0);
    y ~ normal(mu[x], sigma);
}

generated quantities {
    real mu7;
    real ypred6;

    mu7 = normal_rng(mu0, sigma0);
    ypred6 = normal_rng(mu[6], sigma);
}
'''

hierarchical_factory_data = {'N': number_datapoints,
               'K': num_groups,
               'x': x,
               'y': y
              }

sm = pystan.StanModel(model_code = hierarchical_factory_code)
hierarchical_fit = sm.sampling(data = hierarchical_factory_data)


summary = hierarchical_fit.summary()
summary = pd.DataFrame(summary['summary'], columns=summary['summary_colnames'], index=summary['summary_rownames'])
```

```
mu6 = hierarchical_fit.extract(permuted=True)['mu'][:,5]
plt.hist(mu6, bins=50)
plt.title('The posterior distribution of the mean of the quality measurements of the sixth machine')
plt.show()


ypred6 = hierarchical_fit.extract(permuted=True)['ypred6']
plt.hist(ypred6, bins=50)
plt.title('The predictive distribution for another quality measurement of the sixth machine')
plt.show()


mu7 = hierarchical_fit.extract(permuted=True)['mu7']
plt.hist(mu7, bins=50)
plt.title('The posterior distribution of the mean of the quality measurements for the seventh machine')
plt.show()

summary


# SEPARATE MODEL
df = pd.read_csv('factory.csv', sep=' ', header=None)
number_datapoints = df.shape[0] * df.shape[1]
num_groups = df.shape[1]
y = df.values
y = np.reshape(y, 30, order='F')
y = y.ravel()
x = np.arange(1,7)
x = np.repeat(x, 5)


separate_factory_code = '''
data {
    int<lower=0> N; // number of data points
    int<lower=0> K; // number of groups
    int<lower=1,upper=K> x[N]; // group indicator
    vector[N] y; //
}

parameters {
    vector[K] mu;       // group means
    //real<lower=0> sigma;
    vector<lower=0>[K] sigma;
}

model {
  y ~ normal(mu[x], sigma[x]);
}

generated quantities {
    real ypred;
    ypred = normal_rng(mu[6], sigma[6]);
}
'''

separate_factory_data = {'N': number_datapoints,
                'K': num_groups,
                'x': x,
                'y': y
```

```
            }

sm = pystan.StanModel(model_code = separate_factory_code)
separate_fit = sm.sampling(data = separate_factory_data)


summary = separate_fit.summary()
summary = pd.DataFrame(summary['summary'], columns=summary['summary_colnames'], index=summary['summary_rownames'])

mu6 = separate_fit.extract(permuted=True)['mu'][:, 5]
plt.hist(mu6, bins=50)
plt.title('The posterior distribution of the mean of the quality measurements of the sixth machine')
plt.show()

ypred = separate_fit.extract(permuted=True)['ypred']
plt.hist(ypred, bins=50)
plt.title('The predictive distribution for another quality measurement of the sixth machine')
plt.show()
summary
```