Assignment 2

- Aditi Balaji; ab231

Code:

```
import math
import numpy as np
from scipy.stats import invgamma

# load the data and put it in a dictionary
allData = {}
with open('data.txt', 'r') as data:
  for line in data:
    vals = [float(x) for x in line.split()]
    allData[int(vals[0])] = (vals[1], vals[2])

# parameters on the prior for m
mu_zero_m = 5.0
sigma_zero_m = 10.0

# parameters on the prior for c
mu_zero_c = 50.0
sigma_zero_c = 100.0

# parameters on the prior for sigma^2
alpha = 10.0
beta = 1.0

# initial estimates for the three model parameters
m = 20.0
c = 50.0
sigma = 200.0


# write this for 1a)
def SampleSigma():
    data = allData
    n = len(data)
    alpha_p = alpha + float(n)/2
    sum_var = 0
    for i, (h, w) in data.items():
```

```
        x_i = float(w - (h * m + c))
        sum_var = sum_var + (x_i**2)
    beta_p = beta + sum_var/2
    pos = invgamma.rvs(a=alpha_p, scale=beta_p)
    return math.sqrt(pos)

print("sigma")
for x in range(10):
        print(SampleSigma())
print("\n")

# write this for 1b)
def SampleC ():
    data = allData
    n = float(len(data))
    mu_prior = mu_zero_c
    sigma_prior = sigma_zero_c
    sum_wd = 0
    for i, (h, w) in data.items():
        x_i = float(w - (h * m))
        sum_wd = sum_wd + x_i
    mu_pos = (1 / (1/sigma_prior**2 + n/sigma**2)) * (mu_prior/ sigma_prior**2 + sum_wd/ sigma**2)
    sigma_pos = math.sqrt((1/sigma_prior**2 + n/sigma**2)**(-1))
    return np.random.normal(mu_pos, sigma_pos)

print("c")
for x in range(10):
        print(SampleC ())
print("\n")

# write this for 1c)
def SampleM ():
    data = allData
    mu_prior = mu_zero_m
    sigma_prior = sigma_zero_m
    sum_wd = 0
    d_i = 0
    n_i = 0
    for i, (h, w) in data.items():
        x_i = float((w - c)/h)
        sum_wd = sum_wd + x_i
        d_i = d_i + h**2/ sigma**2
        n_i = n_i + x_i * h**2
    mu_pos = (1 / (1/sigma_prior**2 + d_i)) * (mu_prior/ sigma_prior**2 + n_i/ sigma**2)
```

```
    sigma_pos = math.sqrt((1/sigma_prior**2 + d_i)**(-1))
    return np.random.normal(mu_pos, sigma_pos)

print("M")
for x in range(10):
        print(SampleM ())
print("\n")

# this computes the error of the current model
def getError ():
        error = 0.0
        count = 0
        for x in allData:
                y = allData[x]
                error += (c + y[0] * m - y[1]) * (c + y[0] * m - y[1])
                count += 1
        return error / count

# for part 2, you run 1000 iteratins of a Gibbs sampler
print("The initial values:")
print("m =", m)
print("c =", c)
print("sigma =",sigma)
print("\n")
error_vals = []
for xz in range(1000):
        error_vals.append(getError ());
        sigma = SampleSigma ();
        m = SampleM ();
        c = SampleC ();

print("The first 5 error values are", error_vals[:5])
print("\n")
print("The last 5 error values are", error_vals[-5:])
print("\n")
print("The final values:")
print("m =", m)
print("c =", c)
print("sigma =",sigma)
```

Outputs:

sigma
1305.2571431115825
1275.2822249152557
1256.6720668511336
1252.0223214886475
1287.1215300791969
1300.9297180672716
1306.556900956295
1287.2339768330896
1327.0898382081884
1307.6082692597627

c
-1222.6189729670325
-1228.1210896619293
-1227.1190623182524
-1239.7758700200343
-1225.3603437050288
-1224.145047400733
-1222.3124641803415
-1235.1337476216988
-1229.3233512924817
-1230.9558614706489

M
1.2403624778943365
0.9354790416575297
1.160425453908761
1.0821807872698312
1.1680468310242347
1.0237702634120154
1.171493329232447
1.0943860330374995
1.0936580185193432
1.0887425035787082

The initial values:
m = 20.0
c = 50.0

sigma = 200.0

The first 5 error values are [1648445.9364176341, 1163.5568166723922, 113.95325227222177, 113.02279041836007, 113.27174403147208]

The last 5 error values are [106.38402445519797, 106.35666451931206, 106.34156188295448, 106.35703882289374, 106.84727411070364]

The final values:
m = 2.350257231894672
c = -32.34996345719724
sigma = 10.215975458617152