CS146 Assignment 3

Isaac Schaal

```python
import numpy as np
import matplotlib.pyplot as plt
import pystan
from scipy import stats
```

1. Exponential Likelihood with Gamma Prior

In this exercise we are modeling the number of phone calls we expect into the call center during the 11th hour of the day.

We have a data set of the intervals between phone calls arriving during one day. Each value is a time in minutes indicating the amount of time that passed between recieving two consecutive phone calls. The data for the whole day was provided, but only the data from the 11th hour was used.

For this model, we are using an exponential liklihood function. The exponential distribution describes the time between events in a process in which events occur continuosuly and indipendently at a constant average rate. This describes our telephone scenario, and thus the likelihood function is appropriate.

The exponential liklihood has one parameter, the rate, λ. We are using a gamma distribution with alpha = 1 and beta = 0.25 as our prior. These fixed prior hyperparameters were provided in the task.

Because the gamma is a conjugate prior distribution to the exponential distribution, the posterior distribution is also a gamma distribution. This distrtribution was found analytically, and produced the following confidence interval.

Posterior 95% confidence interval over λ: [13.9, 15.9]

Results

The model was coded into stan. It produced the following confidence interval.

Posterior 95% confidence interval over λ: [13.93, 15.82]

This matches the result from the analytical computation.

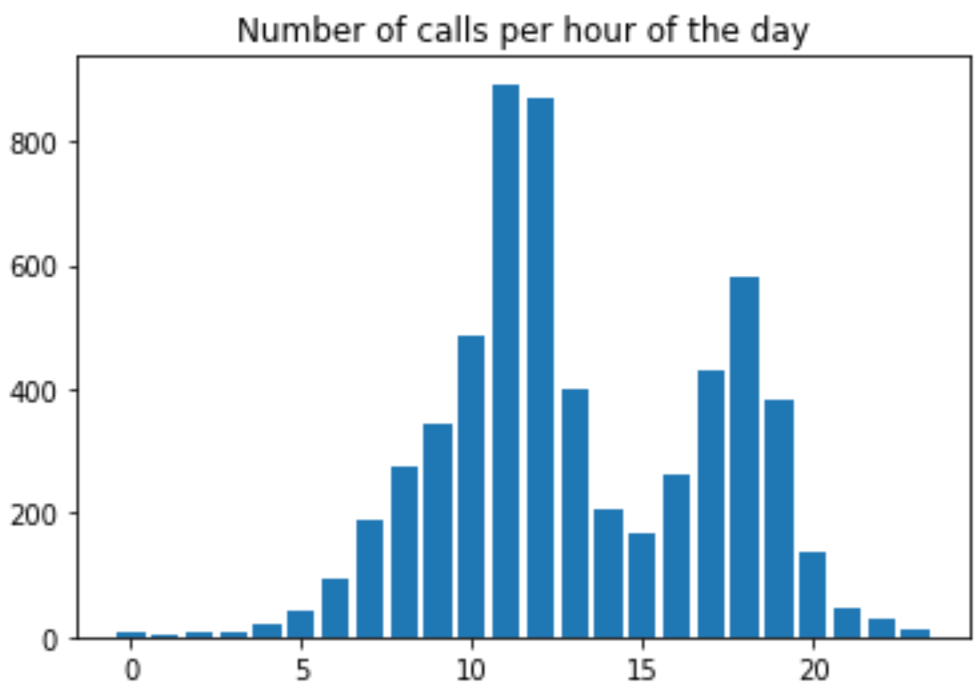
The `n_eff` value for lambda was well over 1000, and the `rhat` value was 1.0, which both indicate that sampling was succesful.

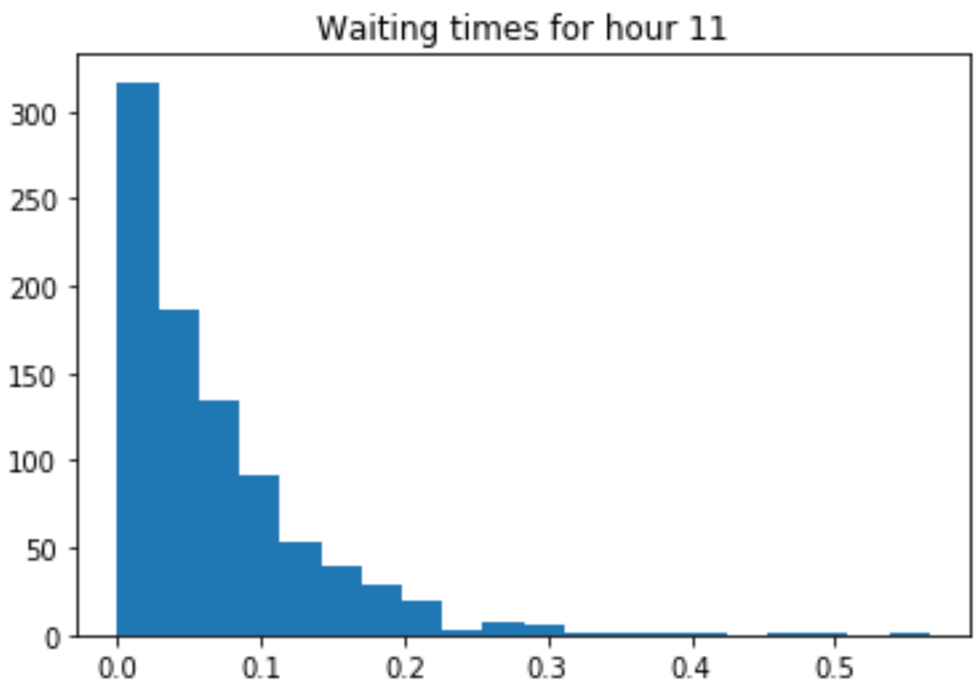
The histogram of the posterior distribution can be seen below.

Importing and Cleaning

```
# Load the data set containing durations between calls arriving at the call ce
# All values are in minutes.
waiting_times_day = np.loadtxt('call-center.csv')
print('Size of data set:', len(waiting_times_day))
print('First 3 values in data set:', waiting_times_day[:3])
print('Sum of data set:', sum(waiting_times_day))

# Split the data into 24 separate series, one for each hour of the day
current_time = 0
waiting_times_per_hour = [[] for _ in range(24)]  # Make 24 empty lists, one p
for t in waiting_times_day:
    current_hour = int(current_time // 60)
    current_time += t
    waiting_times_per_hour[current_hour].append(t)

# Plot the number of calls per hour
plt.bar(range(24), [len(w) for w in waiting_times_per_hour])
plt.title('Number of calls per hour of the day')
plt.show()

# Plot histogram of waiting times for hour
hour_index = 11
waiting_times_hour = waiting_times_per_hour[hour_index]
plt.hist(waiting_times_hour, bins=20)
plt.title('Waiting times for hour %i' % hour_index)
plt.show()
```

```
Size of data set: 5891
First 3 values in data set: [5.36 2.48 8.08]
Sum of data set: 1442.145437310004
```





Modeling

```
call_center_data = {
        'alpha': 1,  # fixed prior hyperparameters for the
        'beta': 0.25,   # gamma distribution
        'calls': waiting_times_hour,  # number of patients per trial
        'num_calls' : len(waiting_times_hour)}


stan_code = """

data {
    real<lower=0> alpha; // fixed prior hyperparameter
    real<lower=0> beta; // fixed prior hyperparameter
    int<lower=0> num_calls ;// Number of calls
    real<lower=0> calls[num_calls]; // wait time for each call
    }

parameters {

    real<lower=0> lambda; // Rate parameter of the Exponential Liklihood

    }

model {

    lambda ~ gamma(alpha, beta); // Prior over Lambda
    for (i in 1:num_calls) {
        calls[i] ~ exponential(lambda); // Likelihood function
        }
    }

"""
```

```
stan_model = pystan.StanModel(model_code=stan_code)
```

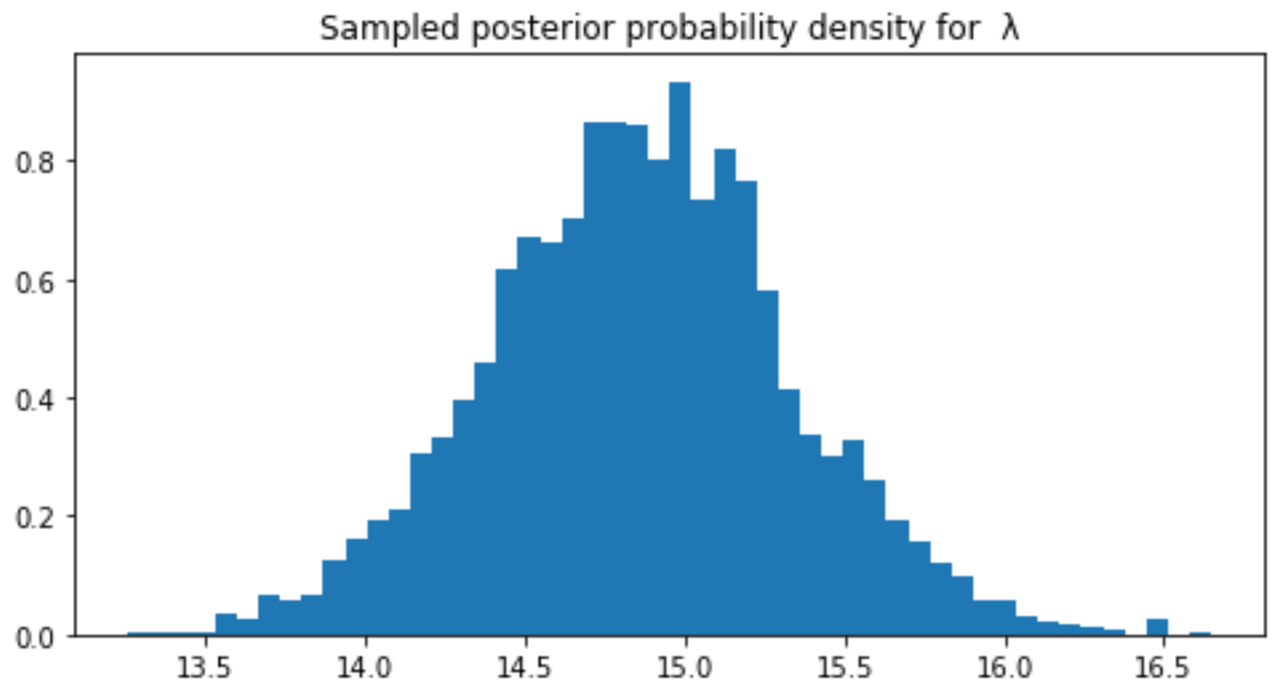
```
INFO:pystan:COMPILING THE C++ CODE FOR MODEL anon_model_95de4b763fa0b52ab6b55c
/usr/local/lib/python3.7/site-packages/Cython/Compiler/Main.py:367: FutureWarn
```

```
  tree = Parsing.p_module(s, pxd, full_module_name)
```

```
# Generate posterior samples
results = stan_model.sampling(data=call_center_data)
print(results.stansummary(pars = ['lambda']))
```

```
Inference for Stan model: anon_model_95de4b763fa0b52ab6b55c7fdee05d3b.
4 chains, each with iter=2000; warmup=1000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=4000.

         mean se_mean     sd   2.5%    25%    50%    75%  97.5%  n_eff   Rhat
lambda  14.86    0.01   0.47  13.93  14.54  14.86  15.17  15.82   1671    1.0

Samples were drawn using NUTS at Fri Oct 19 22:14:40 2018.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

```
# Plot posterior histograms

samples = results.extract()

plt.figure(figsize=(8,4))
plt.hist(samples['lambda'], bins=50, density=True)
plt.title('Sampled posterior probability density for  λ')
plt.show()

print('Posterior 95% confidence interval over λ :',
      np.percentile(samples['lambda'], [2.5, 97.5]))
```



```
Posterior 95% confidence interval over λ : [13.93119791 15.82184776]
```

Normal Likelihood with Normal-Inverse-Gamma Prior

In this task, we compute the posterior distribution over the mean x and variance σ^2 of normally distributed data.

For this model, we are using a normal liklihood function. We can see from its histogram that the data is approximately normally distributed. The normal liklihood has two parameters, the mean, x, and the variance, σ^2. We are using a normal-inverse-gamma distribution as our prior. It has the following hyperperameters:

- mu = 0
- nu = 0.054
- alpha = 1.12
- beta = 0.4

Mu is the prior mean, and the smaller nu is, the more uncertain we are about the prior mean. Alpha and beta control the marginal prior over the variance. These fixed prior hyperparameters were provided in the task.

Because the normal-inverse-gamma is a conjugate prior distribution to the normal distribution, the posterior distribution is also a normal-inverse-gamma distribution. This distribution was found analytically, and produced the following confidence intervals.

Posterior 95% confidence interval over x: [2.800, 3.328]

Posterior 95% confidence interval over σ^2: [2.975, 4.396]

Results

The model was coded into stan. It produced the following confidence intervals.

Posterior 95% confidence interval over x : [2.800 3.333]

Posterior 95% confidence interval over sigma2 : [3.004 4.358]

This matches the result from the analytical computation.

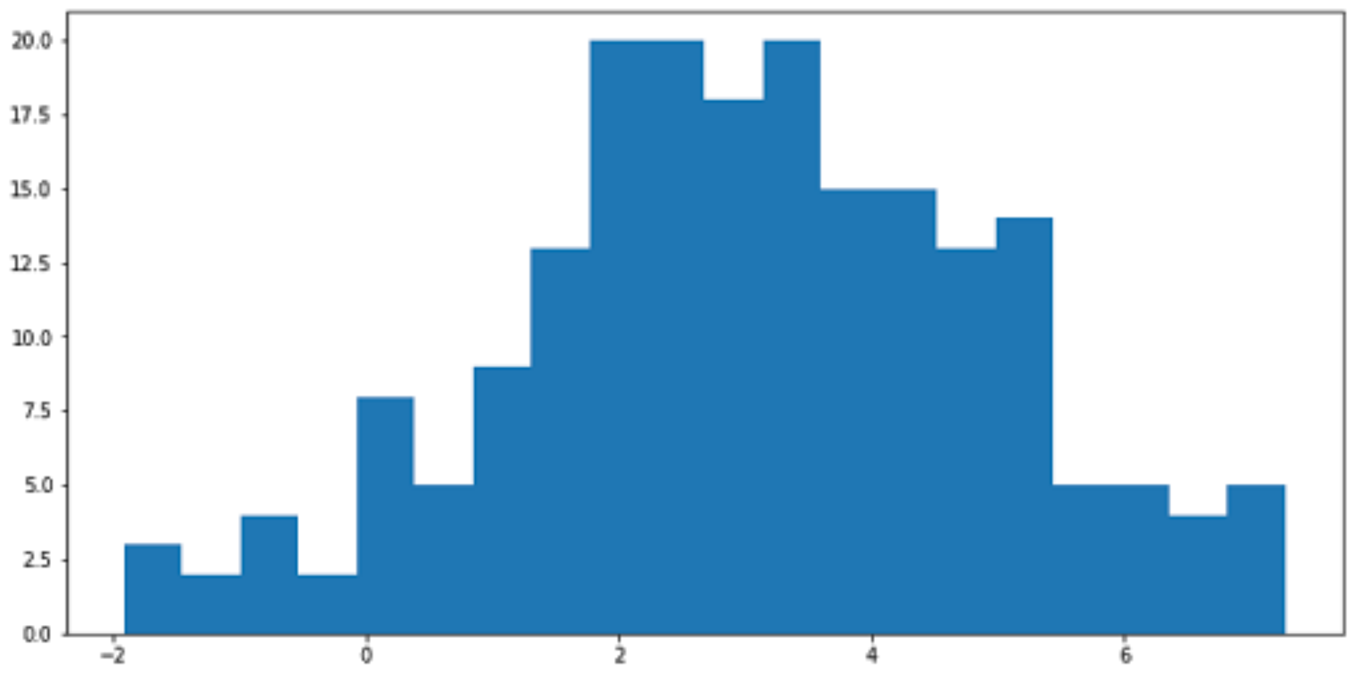
The `n_eff` values for both x and σ^2 were over 3000, and the `rhat` values were 1.0, which both indicate that sampling was succesful.

10 samples from the normal-inverse-gamma posterior distribution an be seen below.

Importing

```
normal_data = np.array([3.54551763569501, 4.23799861761927, 4.72138425951628,
print(len(normal_data), "data")

# Plot a histogram of the data. The data looks normally distributed
plt.figure(figsize=(12, 6))
plt.hist(normal_data, bins=20)
plt.show()

# Calculate the sample mean and variance of the data
print('Data sample mean:', np.mean(normal_data))
print('Data sample variance:', np.var(normal_data))
```

```
200 data
```



```
Data sample mean: 3.065080189542003
Data sample variance: 3.615214178769922З
```

Modeling

```
stan_data = {
        'mu' : 0,       # fixed prior hyperparameters for the
        'nu' : 0.054,   # normal-inverse-gamma distribution
        'alpha' : 1.12,
        'beta' : 0.4,
        'normal_data' : normal_data,
        'num_data' : len(normal_data)}

stan_code = """

data {
    real<lower=0> mu; // fixed prior hyperparameters
    real<lower=0> nu;
    real<lower=0> alpha;
    real<lower=0> beta;
    int<lower=0> num_data ;// Number of data
    real normal_data[num_data]; // Data from normal distribution
    }

parameters {

    real<lower=0> sigma2; // The variance of the likelihood function
    real  x; // The mean of the likelihood function
    }

model {

    sigma2 ~ inv_gamma(alpha, beta); // Normal-Inverse-Gamma Prior
    x ~ normal(mu,sqrt(sigma2 / nu)); // over sigma2 and x
    for (i in 1:num_data) {
        normal_data[i] ~ normal(x, sqrt(sigma2)); // Likelihood function
        }
    }

"""
```

```
stan_model = pystan.StanModel(model_code=stan_code)
```

```
INFO:pystan:COMPILING THE C++ CODE FOR MODEL anon_model_623d005bbca361f51c11d4
/usr/local/lib/python3.7/site-packages/Cython/Compiler/Main.py:367: FutureWarn
  tree = Parsing.p_module(s, pxd, full_module_name)
```

```
# Generate posterior samples
results = stan_model.sampling(data=stan_data)
print(results.stansummary(pars = ['sigma2', 'x']))
```

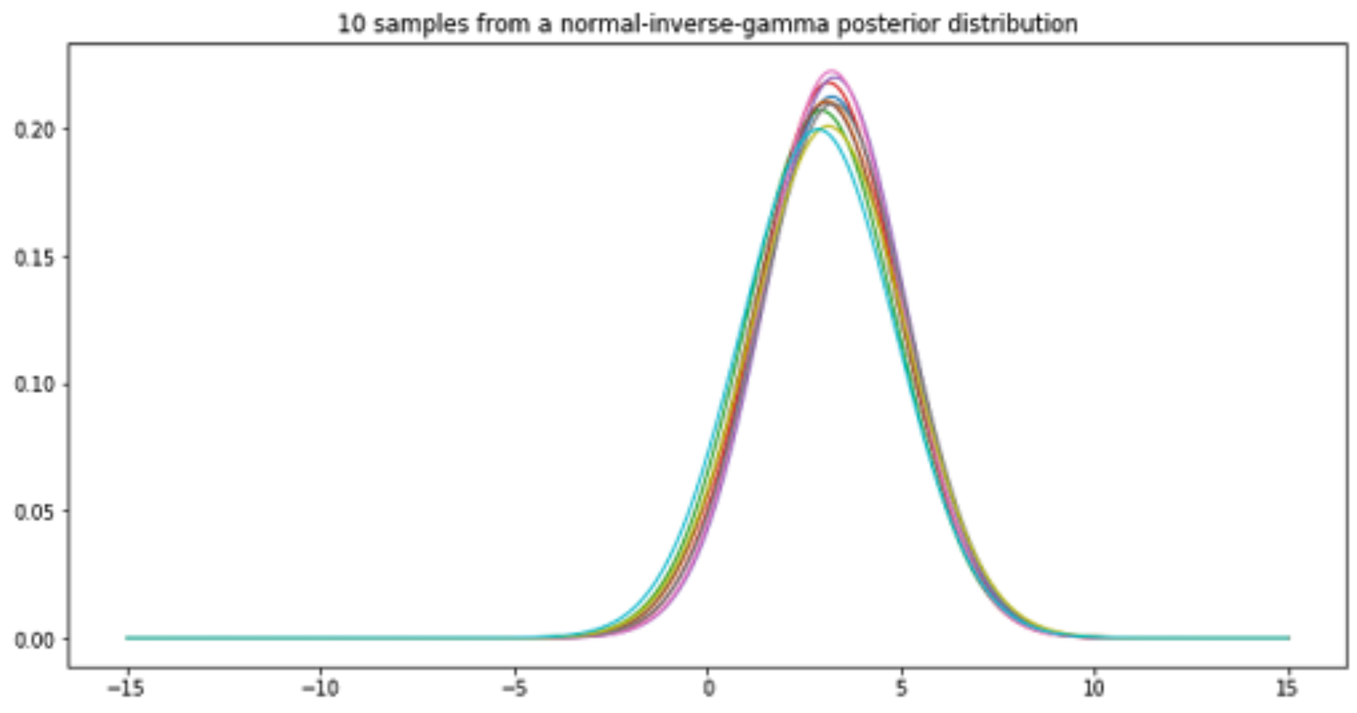
```
Inference for Stan model: anon_model_623d005bbca361f51c11d426964cb408.
4 chains, each with iter=2000; warmup=1000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=4000.

         mean se_mean     sd   2.5%    25%    50%    75%  97.5%  n_eff   Rhat
sigma2   3.62  6.0e-3   0.35    3.0   3.37   3.59   3.84   4.36   3506    1.0
x        3.06  2.5e-3   0.13    2.8   2.97   3.06   3.15   3.33   2870    1.0
```

```
Samples were drawn using NUTS at Fri Oct 19 22:16:23 2018.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

```
# Show posterior confidence intervals

samples = results.extract()

print('Posterior 95% confidence interval over sigma2 :',
      np.percentile(samples['sigma2'], [2.5, 97.5]))
print('Posterior 95% confidence interval over x :',
      np.percentile(samples['x'], [2.5, 97.5]))
```

```
Posterior 95% confidence interval over sigma2 : [3.00366153 4.35839605]
Posterior 95% confidence interval over x : [2.80000198 3.33325104]
```

```
# Take the first ten samples
sigma2s = samples['sigma2'][:10]
xs = samples['x'][:10]

# Plot the normal distributions corresponding to the samples
plt.figure(figsize=(12, 6))
plot_x = np.linspace(-15, 15, 500)
for i in range(10):
    plot_y = stats.norm.pdf(plot_x, loc= xs[i], scale=np.sqrt(sigma2s[i]))
    plt.plot(plot_x, plot_y)
plt.title('10 samples from a normal-inverse-gamma posterior distribution')
plt.show()
```



Log-Normal with Normal-Inverse-Gamma Prior

In this task, we compute the posterior distribution over the mean x and variance σ^2 of log-normally distributed data.

We first see that the data is not normally distributed. It is skewed to the right and all values are positive. In order to use a normal liklihood function, we take the log of all of the data. Thus, we are making a log-normal model. Once we log the data, we can use a normal liklihood function.

For this model, we are using a normal liklihood function. The normal liklihood has two parameters, the mean, x, and the variance, σ^2. We are using a normal-inverse-gamma distribution as our prior. It has the following hyperperameters:

- mu = 2.3
- nu = 0.1
- alpha = 2
- beta = 5

Mu is the prior mean, and the smaller nu is, the more uncertain we are about the prior mean.

Alpha and beta control the marginal prior over the variance. These fixed prior hyperparameters were provided in the task.

Because the normal-inverse-gamma is a conjugate prior distribution to the normal distribution, the posterior distribution is also a normal-inverse-gamma distribution. This distribution was found analytically, and produced the following confidence intervals.

Posterior 95% confidence interval over x: [1.83, 1.95]

Posterior 95% confidence interval over σ^2: [0.44, 0.56]

Results

The model was coded into stan. It produced the following confidence intervals.

Posterior 95% confidence interval over x : [1.83, 1.96]

Posterior 95% confidence interval over sigma2 : [0.44, 0.56]

This matches the result from the analytical computation.

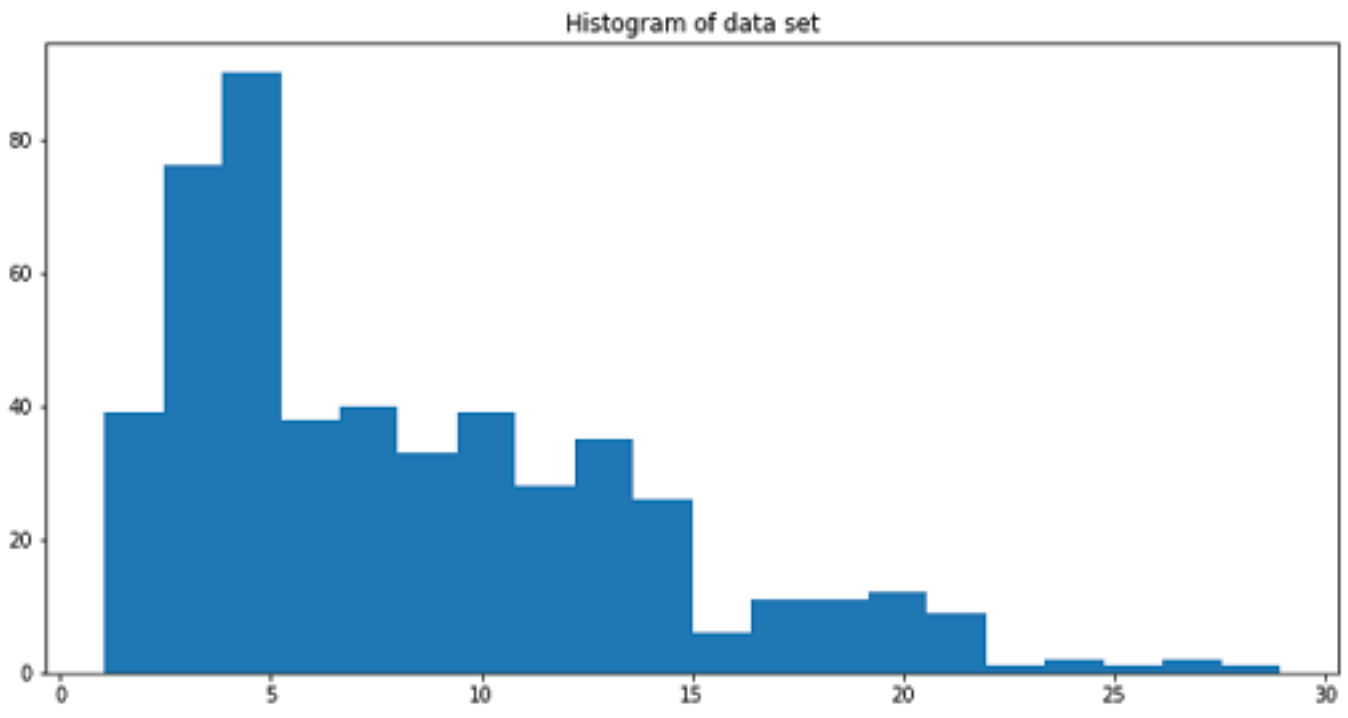
The `n_eff` values for both x and σ^2 were over 3000, and the `rhat` values were 1.0, which both indicate that sampling was succesful.

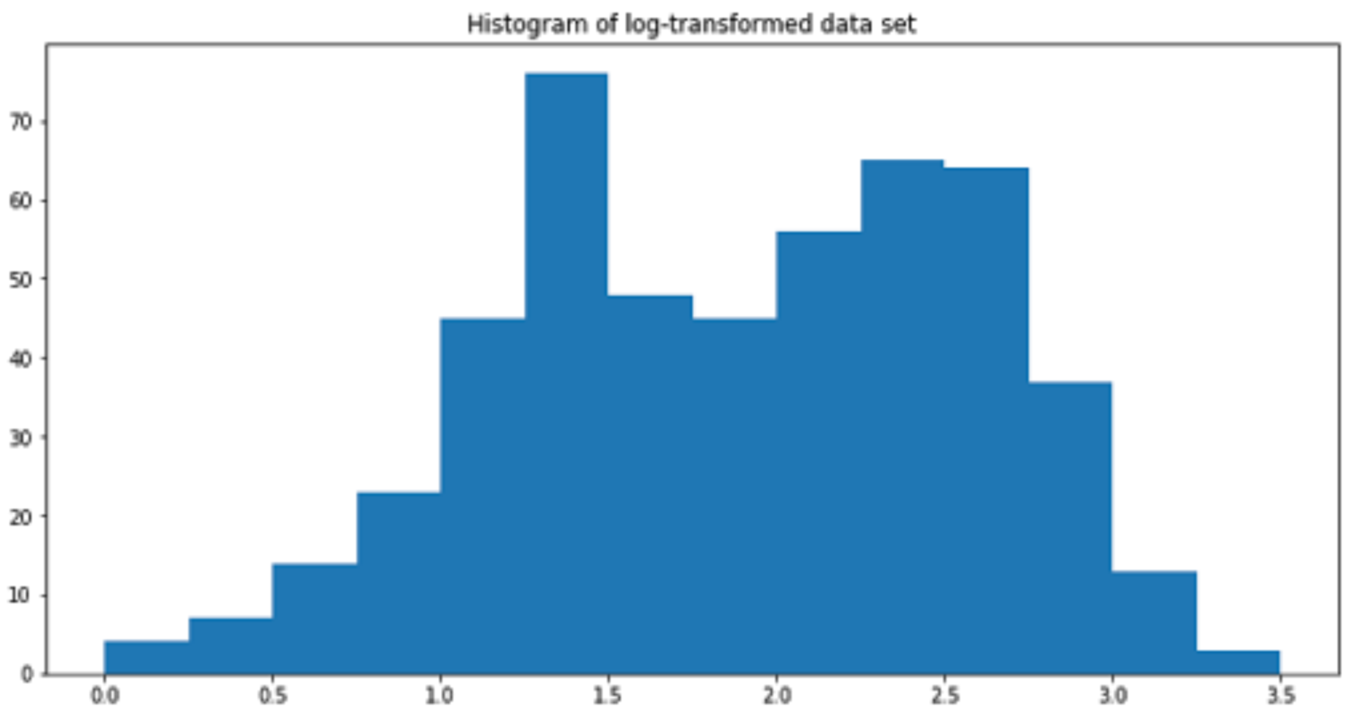
10 samples from the normal-inverse-gamma posterior distribution on top of the histogram of the original data can be seen below.

```
# Load data: read the particle sizes (in nanometers) from a CSV file.
hrtem_data = np.loadtxt('hrtem.csv')
print('%i data, min: %f, max: %f' % (len(hrtem_data), min(hrtem_data), max(hrt

# Data are very skewed and all values are positive, so probably non-normal.
plt.figure(figsize=(12,6))
plt.hist(hrtem_data, bins=20)
plt.title('Histogram of data set')
plt.show()

# Transform the data to make it normal
```

```
log_hrtem_data = np.log(hrtem_data)

plt.figure(figsize=(12,6))
plt.hist(log_hrtem_data, bins=np.linspace(0, 3.5, 15))
plt.title('Histogram of log-transformed data set')
plt.show()
```

```
500 data, min: 1.051827, max: 28.942578
```





```
stan_data = {
        'mu' : 2.3,    # fixed prior hyperparameters for the
        'nu' : 0.1,    # normal-inverse-gamma distribution
        'alpha' : 2,
        'beta' : 5,
        'log_hrtem_data' : log_hrtem_data,
        'num_data' : len(hrtem_data)}

stan_code = """

data {
    real<lower=0> mu; // fixed prior hyperparameters
    real<lower=0> nu;
    real<lower=0> alpha;
    real<lower=0> beta;

    int<lower=0> num_data ;// Number of data
    real <lower=0> log_hrtem_data[num_data]; // Log transformed hrtem data
    }

parameters {

    real<lower=0> sigma2; // The variance of the likelihood function
    real  x; // The mean of the likelihood function
    }

model {

    sigma2 ~ inv_gamma(alpha, beta); // Normal-Inverse-Gamma Prior
    x ~ normal(mu,sqrt(sigma2 / nu)); // over sigma2 and x
    for (i in 1:num_data) {
        log_hrtem_data[i] ~ normal(x, sqrt(sigma2)); // Likelihood function
        }
    }

"""
```

```
stan_model = pystan.StanModel(model_code=stan_code)
```

```
INFO:pystan:COMPILING THE C++ CODE FOR MODEL anon_model_b994a298c06ec4a3efde1f
/usr/local/lib/python3.7/site-packages/Cython/Compiler/Main.py:367: FutureWarn
  tree = Parsing.p_module(s, pxd, full_module_name)
```

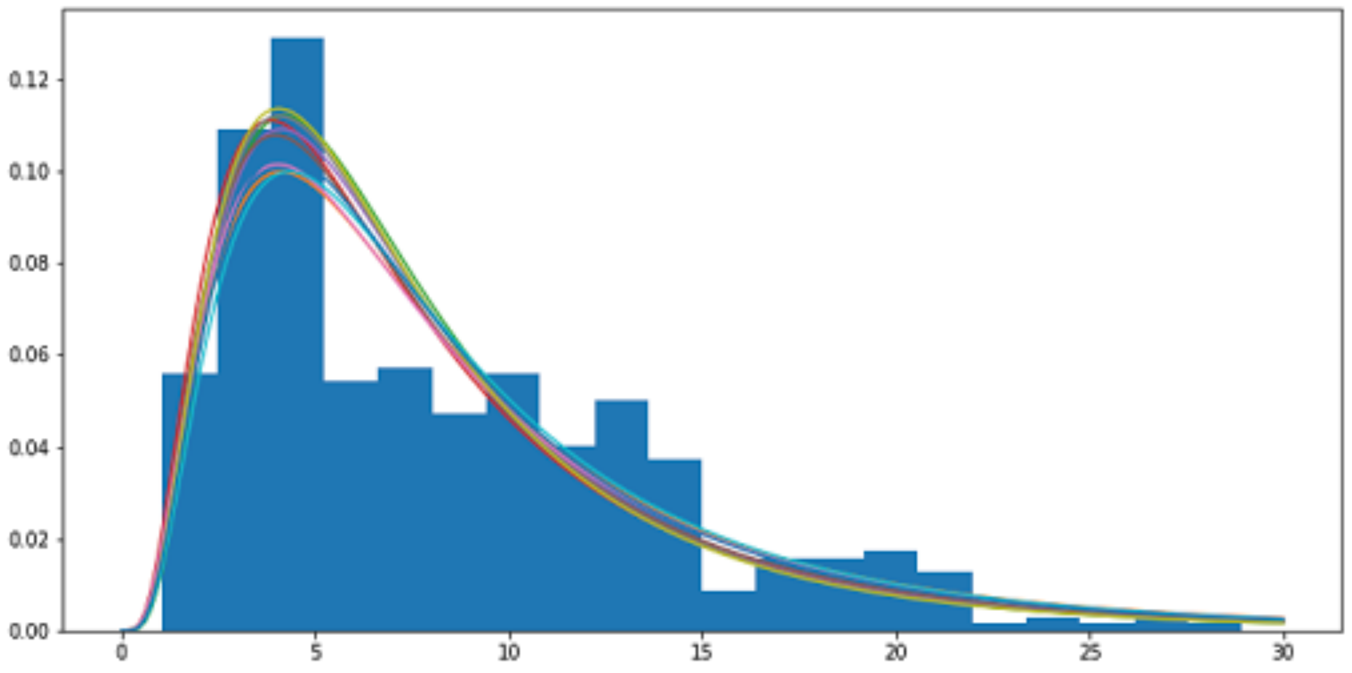
```python
# Generate posterior samples
results = stan_model.sampling(data=stan_data)
print(results.stansummary(pars = ['sigma2', 'x']))
```

```
Inference for Stan model: anon_model_b994a298c06ec4a3efde1f415a0722c9.
4 chains, each with iter=2000; warmup=1000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=4000.

         mean se_mean     sd   2.5%    25%    50%    75%  97.5%  n_eff   Rhat
sigma2    0.5  5.6e-4   0.03   0.44   0.47   0.49   0.52   0.56   3176    1.0
x        1.89  5.4e-4   0.03   1.83   1.87   1.89   1.91   1.96   3582    1.0

Samples were drawn using NUTS at Fri Oct 19 22:20:51 2018.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

```python
# Show posterior confidence intervals

samples = results.extract()

print('Posterior 95% confidence interval over sigma2 :',
      np.percentile(samples['sigma2'], [2.5, 97.5]))
print('Posterior 95% confidence interval over x :',
      np.percentile(samples['x'], [2.5, 97.5]))
```

```
Posterior 95% confidence interval over sigma2 : [0.43924298 0.56082155]
Posterior 95% confidence interval over x : [1.82804557 1.95715216]
```

```python
# Take the first ten samples
sigma2s = samples['sigma2'][:10]
xs = samples['x'][:10]

# Plot 10 pdfs over the histogram of the original data.
plt.figure(figsize=(12,6))
plt.hist(hrtem_data, bins=20, density=True)
plot_x = np.linspace(0, 30, 200)
for i in range(10):
    plot_y = stats.lognorm.pdf(plot_x, np.sqrt(sigma2s[i]),
                                scale=np.exp(xs[i]))
    plt.plot(plot_x, plot_y)
plt.show()
```


0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
5
10
15
20
25
30