Bayesian Statistical Methods: Homework 2

Due on Feb 14, 2013

Andrew Kurzawski

Problem 1

Reading Assignment.

Problem 2

I began by doing a test run of 1000 iterations with no burn in or thinning. From the traces of μ and σ, I saw that both had converged within a few iterations. So, I burned 100 iterations because running the model is not computationally expensive. Then, I ran the model again and observed significant autocorrelation of μ within about 15 itterations, so I selected to thin the results by 15. Fig. 1 and Fig. 2 are the resulting posterior summaries after 10,000 iterations.

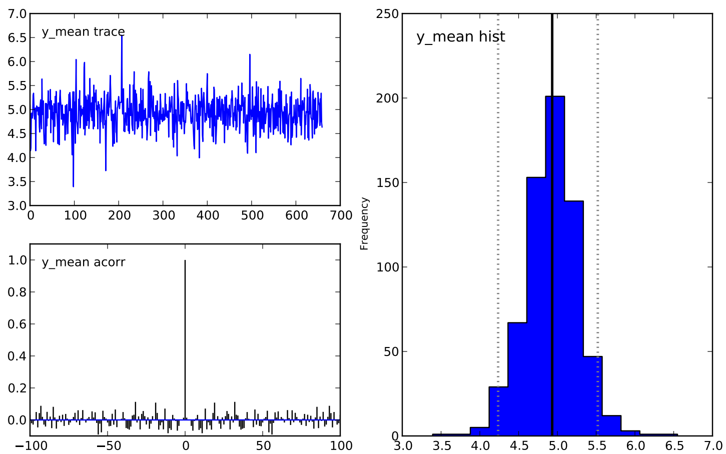

Figure 1: Posterior summary for μ using PyMC

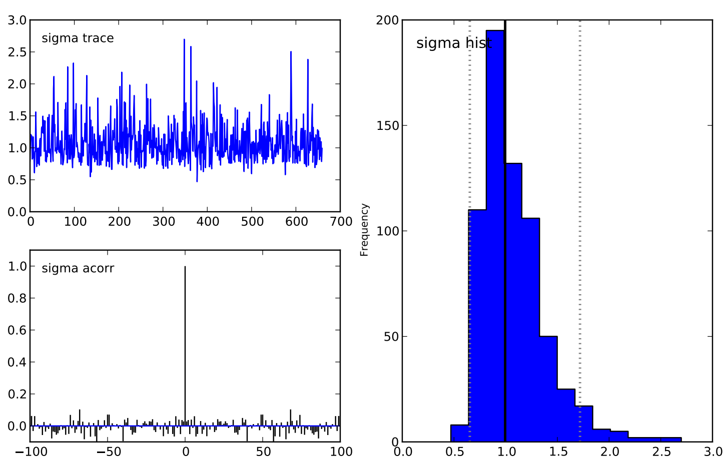

Figure 2: Posterior summary for σ using PyMC

Problem 3

Write out the steps for your slice sampler, use h1 tails to boost U?

Problem 4

```
from __future__ import division
import numpy as np
import scipy.stats
import matplotlib as plt
from pylab import *

def mh_sample(n, data, mu, walk_sig):
    # n is the number of iterations
    # data is what we are conditioning on
    # mu is an initial value for the chain of samples
    # walk_sig is the variance of the random walk

    # Initialize arrays
    x_list = np.zeros(n)
    n_list = np.zeros(n)
    x_list[0] = mu
    n_list[0] = 1

    # Variance is 1 for now
    sigma = 1.0

    # Pointer for last accepted value
    acc = 0

    for i in range(n):
        # Random Walk to generate next value of x
        z = x_list[acc] + np.random.normal(0.0,walk_sig**2,1)

        # Compute Denominator of MH alogrithm
        den = 1
        for j in range(0,data.shape[0]):
            den = den * np.exp((-(data[j]-x_list[acc])**2)/(2.0*sigma**2))

        # Compute Numerator of MH alogrithm
        num = 1
        for j in range(0,data.shape[0]):
            num = num * np.exp((-(data[j]-z)**2)/2.0)

        # Compute the probability of a move, alpha
        alpha = min(1, num/den)

        # Accept / Reject
        if alpha == 1:
```

Problem 4 continued on next page. . .

```
            x_list[acc+1] = z
            n_list[acc+1] = i+1
            acc = acc + 1
        elif alpha < 1:
            p_a = np.random.binomial(1, alpha, 1)
            if p_a == 1:
                x_list[acc+1] = z
                n_list[acc+1] = i+1
                acc = acc + 1

    return x_list, n_list, acc
```

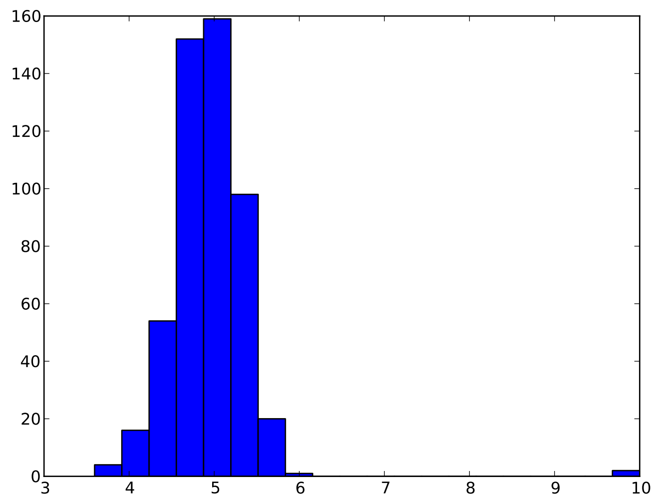

Figure 3: Posterior summary for μ using Metropoils-Hastings

Problem 5

```
from __future__ import division
import numpy as np
import scipy.stats
import matplotlib as plt
from pylab import *

def slice_sample(n, data, mu):
    # n is the number of iterations
    # data is what we are conditioning on
    # mu is an initial value for the chain of samples

    # Initialize arrays
```

```
    x_list = np.zeros(n+1)
    x_list[0] = mu
    sigma = 1.0

    width = 1.0

    for i in range(n):
        # Calculate h(x) at data
        h_x = 1
        for j in range(0,data.shape[0]):
            h_x = h_x * np.exp((-(data[j]-x_list[i])**2)/(2.0*sigma**2))

        # Sample from 0 to h(x) to get U
        U = np.random.uniform(0, h_x, 1)

        # Find bounds x_L and x_R
        x_L = x_list[i] - width
        x_R = x_list[i] + width
        left_bound = False
        right_bound = False
        while left_bound == False:
            h_x_L = 1.0
            for j in range(0,data.shape[0]):
                h_x_L = h_x_L * np.exp((-(data[j]-x_L)**2)/(2.0*sigma**2))
            if h_x_L < U:
                left_bound = True
            elif h_x_L > U:
                x_L = x_L - width

        while right_bound == False:
            h_x_R = 1.0
            for j in range(0,data.shape[0]):
                h_x_R = h_x_R * np.exp((-(data[j]-x_R)**2)/(2.0*sigma**2))
            if h_x_R < U:
                right_bound = True
            elif h_x_R > U:
                x_R = x_R + width

        # Sample from uniform on interval x_L to x_R
        x_list[i+1] = np.random.uniform(x_L, x_R, 1)

    return x_list
```

Problem 6

For each Prior: Differences in posterior, what was the thin and burn?

Compute probability that lambda is greater than 0.5. Need to get full sample data somehow and run a script on it. P(lambda¿0.5) = Number of samples over 0.5 / Total number of samples. do this for each prior

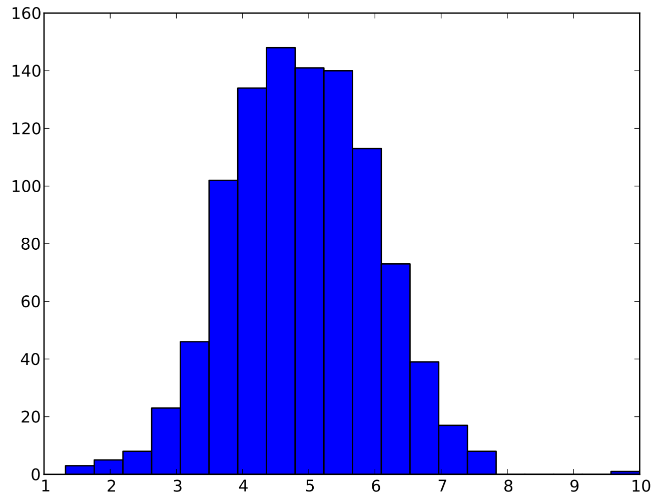

Figure 4: Posterior summary for μ using Slice Sampling

also.

Problem 7

For each Prior: Differences in posterior, what was the thin and burn?

Compute the probability that the difference of the means is greater than zero. Compute the posterior probability that the ratio of the standard deviations is greater than 1.

Is this a good diagnostic?