Ph 21 Homework 5

Emily Springer

March 5, 2020

1 Question 1.

I used emcee for MCMC sampling. To get the log of the posterior function, I wrote the following function:

```
def getPosterior(a, nHeads):
    sigma = 0.3
    mu = 0.5
    prior = -(np.log(sigma) + np.log(np.sqrt(2 * np.pi)))\
    - (a - mu)**2 / (2 * sigma**2)
    if a <= 0 or a >=1:
        return -np.inf

    likelihood = np.log(math.factorial(n)) -(np.log((math.factorial(nHeads)) + \
                        np.log (math.factorial(n - nHeads)))) + \
                        nHeads * np.log(a)  + (n - nHeads) * np.log(1 - a)
    return prior + likelihood
```

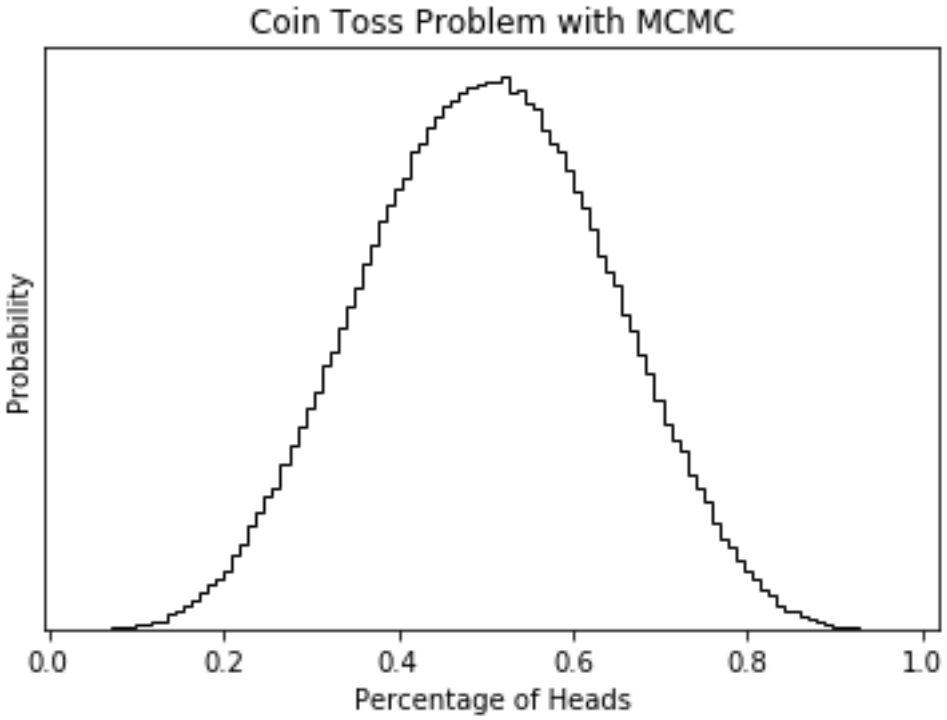
To fun the sampler, I wrote

```
p0 = np.random.rand(nwalkers, ndim)
sampler = emcee.EnsembleSampler(nwalkers, ndim, getPosterior, args = [nHeads])
state = sampler.run_mcmc(p0, 100)
sampler.reset()
sampler.run_mcmc(state, chainLength);
```

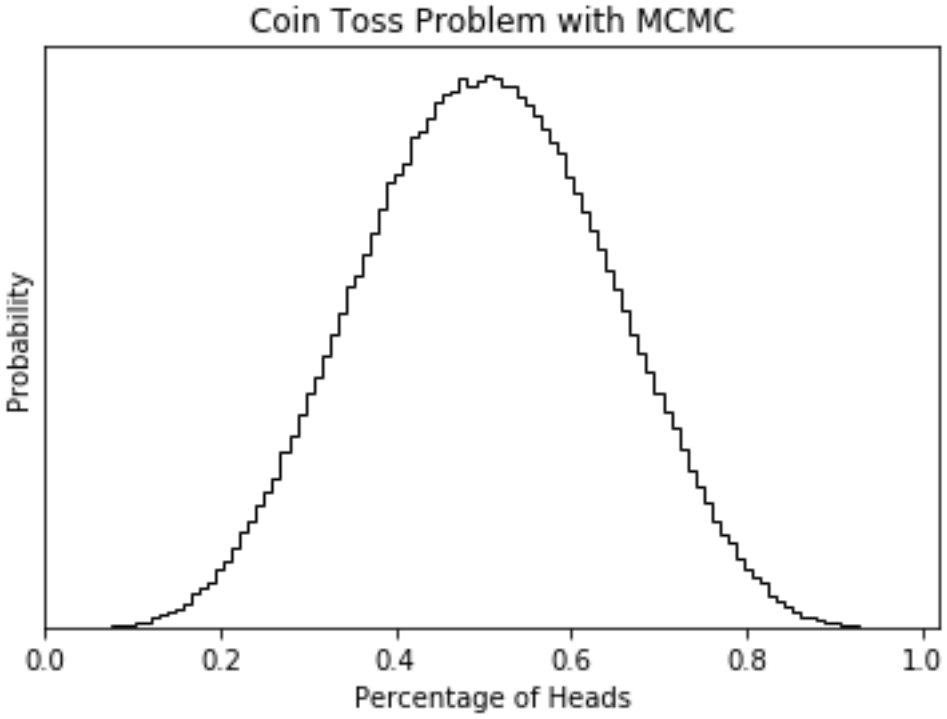
To draw the graphs, I wrote

```
samples = sampler.get_chain(flat=True)
plt.hist(samples[:, 0], 100, color="k", histtype="step")
plt.gca().set_yticks([]);
plt.xlabel(r"Percentage of Heads")
plt.ylabel(r"Probability")
plt.title('Coin Toss Problem with MCMC')
```

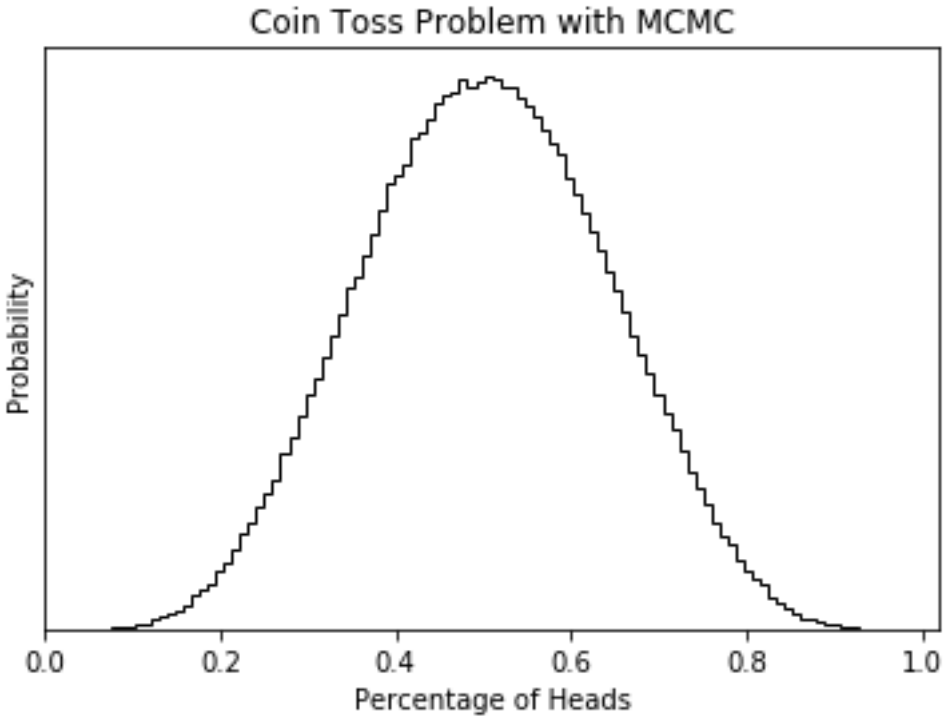
I started with a flat prior of 0.5.
I used n = 10. For 64 walkers and a chain length of 10,000, I got the following graph:



I proceeded to change the number of walkers to 128 walkers and got



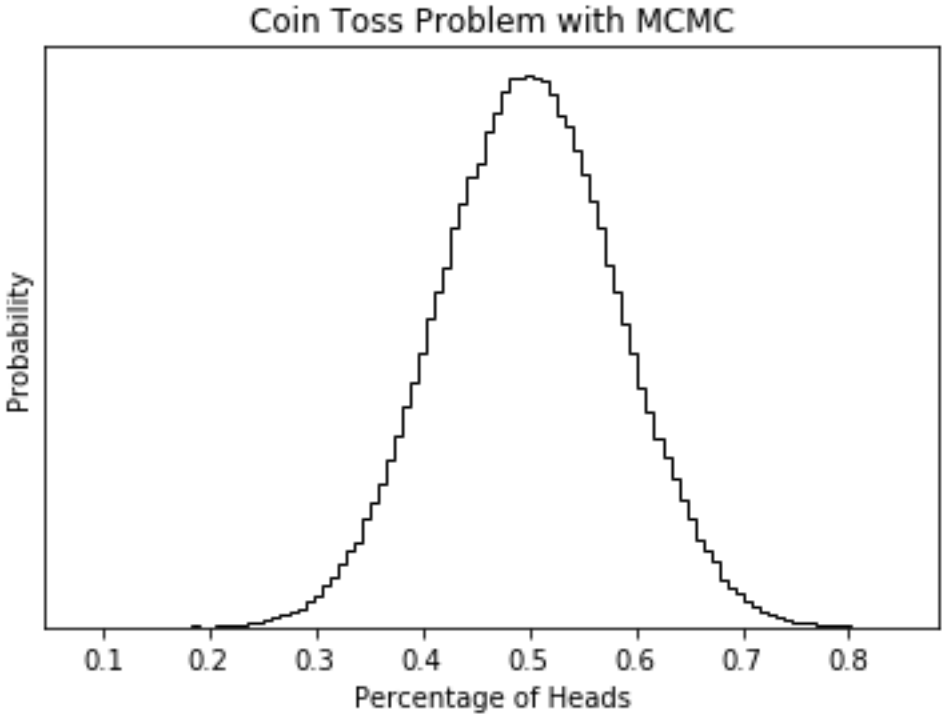
For 64 walkers, but 1,000 chain length, I got



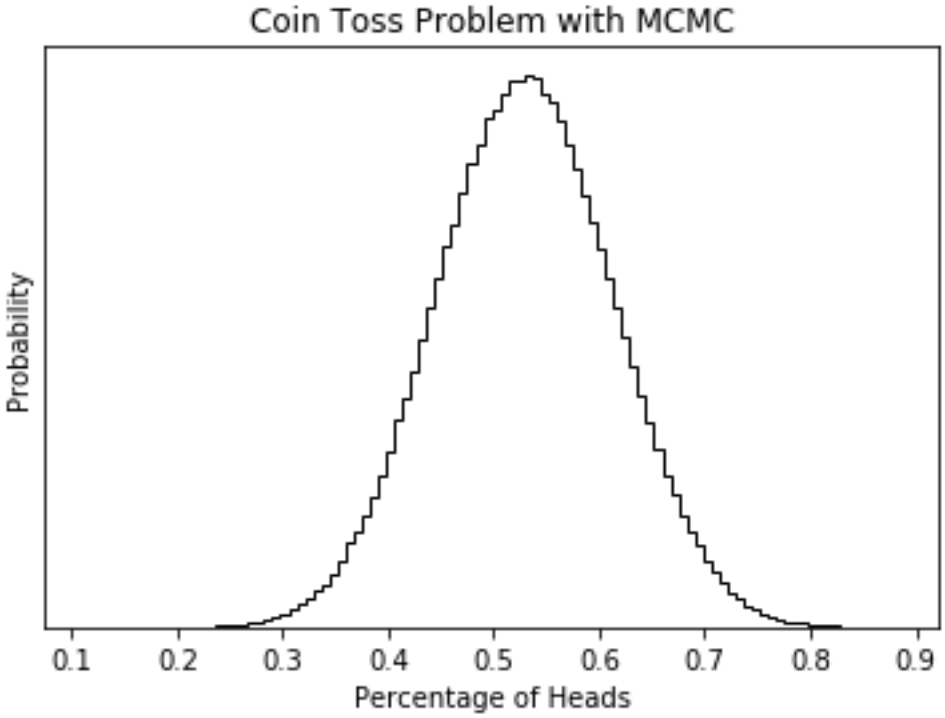
For a prior of a Gaussian distribution, I used the line

```
prior = -(np.log(sigma) + np.log(np.sqrt(2 * np.pi)))\
    - (a - mu)**2 / (2 * sigma**2)
```

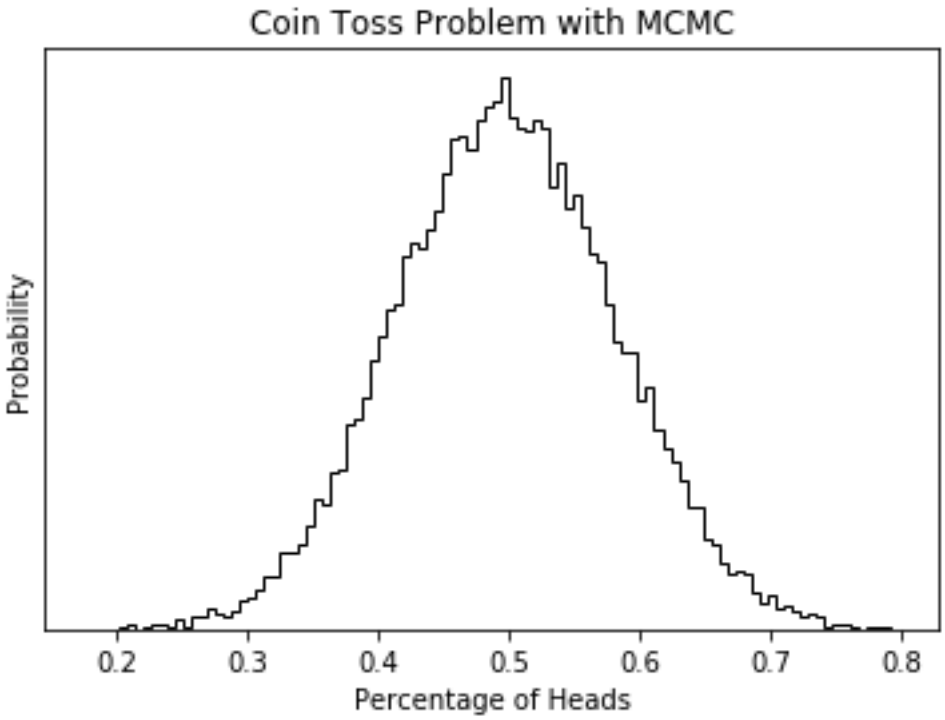
I centered the gaussain at 0.5 with a sigma of 0.1. My resulting graph for 64 walkers and 10,000 chain length was



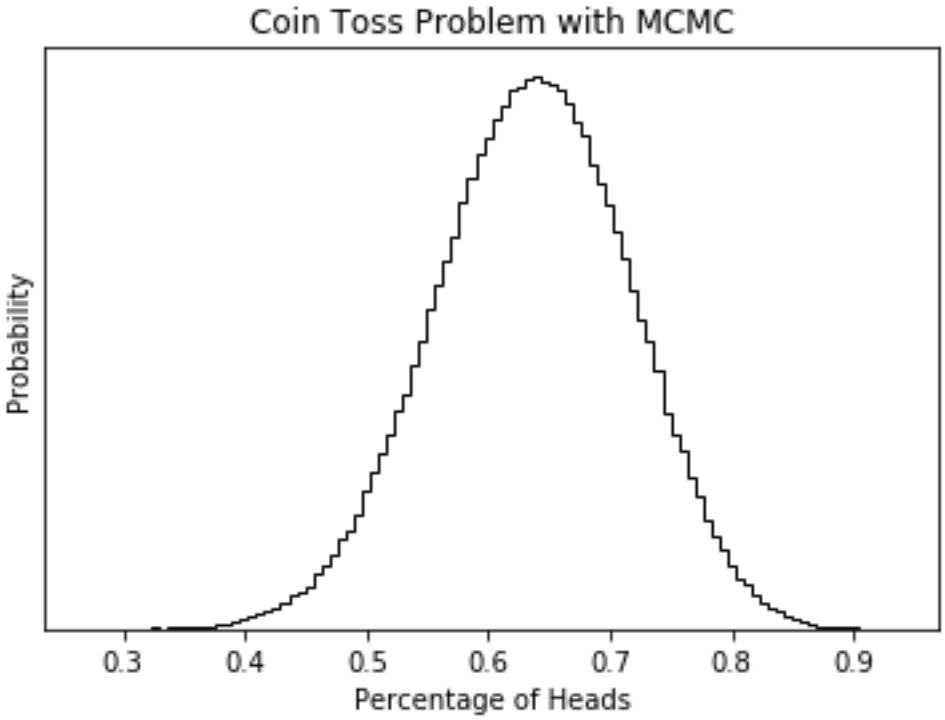
For 128 walkers and 10,000 chain length, it was



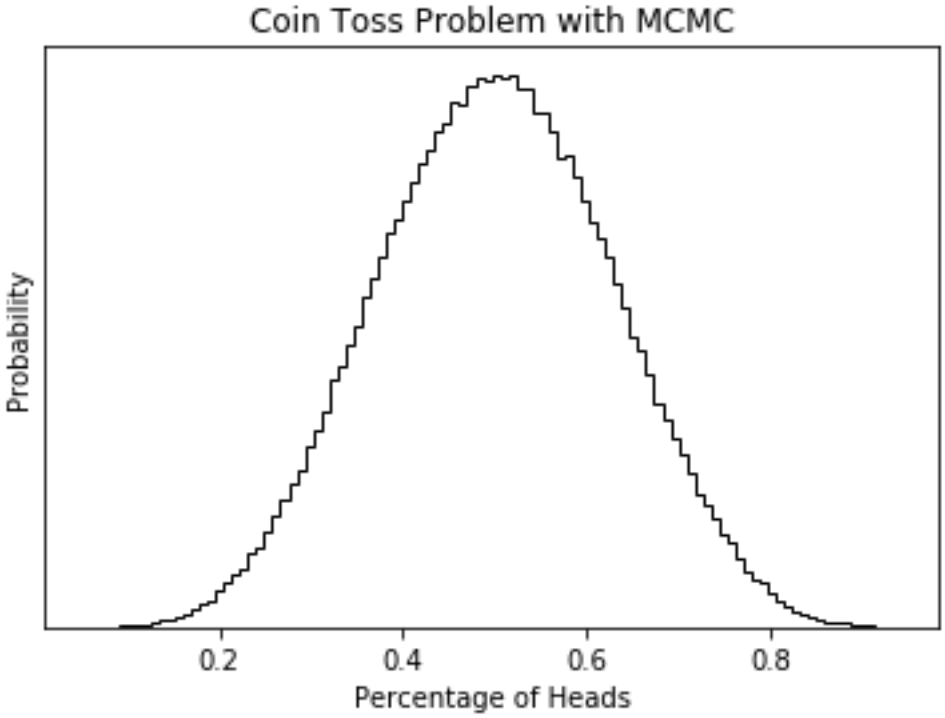
For 64 walkers and 1,000 chain length, it was



I then proceeded to shift the center of the gaussian to 0.7 while keeping the standard 64 walkers and 10,000 chain length.



Then, I changed the mean back to 0.5, but increased the sigma to 0.3.



Shifting the mean slightly shifted the center of the posterior towards 0.7. Increasing the sigma widened the posterior.

2 Question 2.

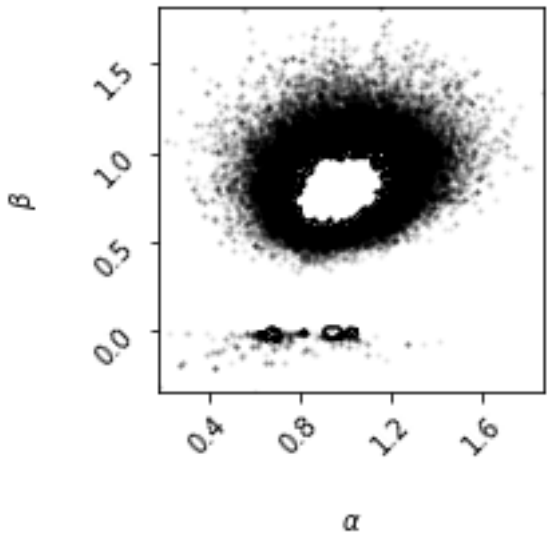
The getPosterior function for this was very similar to last assignment. It was

```
def getPosterior(params, data):
    logB = 0
    if (params[1] > 0):
        logB = math.log(params[1])

    likelihood = np.ones(len(data)) * logB - np.ones(len(data)) * \
    math.log(math.pi) - np.log(params[1]**2 + (data - params[0]) ** 2)

    return np.sum(likelihood)
```

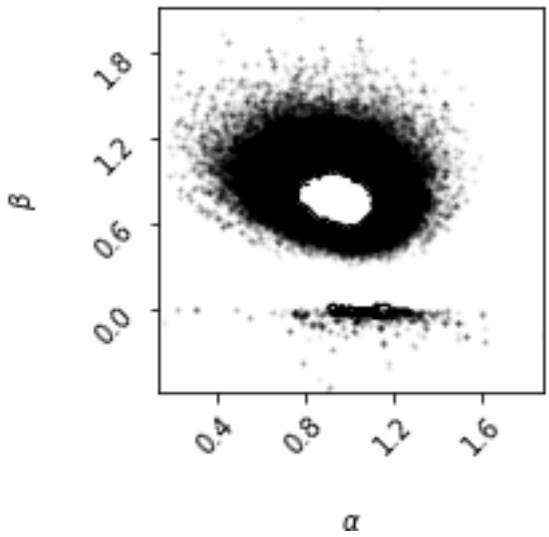
I got data in the exact same way as the previous assignment. For this part, I set the number of dimensions to 2 since there are 2 unknown variables. I plotted the 2D contour graph using the corner package with the line

```
fig = corner.corner(sampler.flatchain, labels=[r"$\alpha$", r"$\beta$"], bins=100)
```

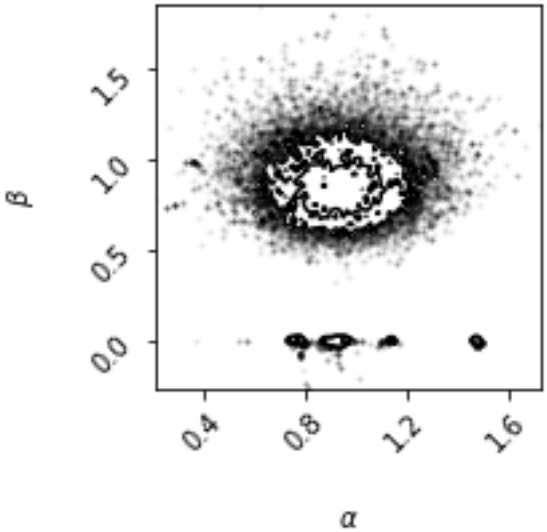
My resulting graph for n = 50, 64 walkers, and 10,000 chain length was



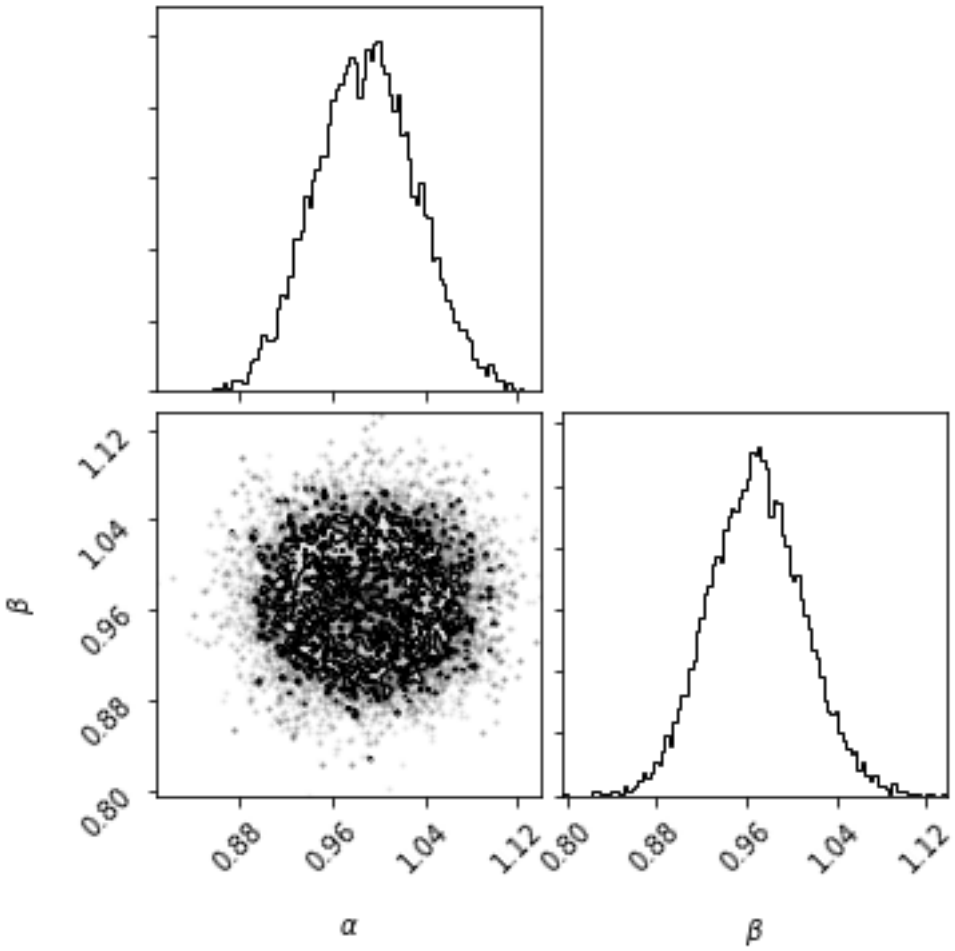
It is centered at about (1,1) which is expected. For 128 walkers and 10,000 chain length, I got



For 64 walkers and 1,000 chain length, I got

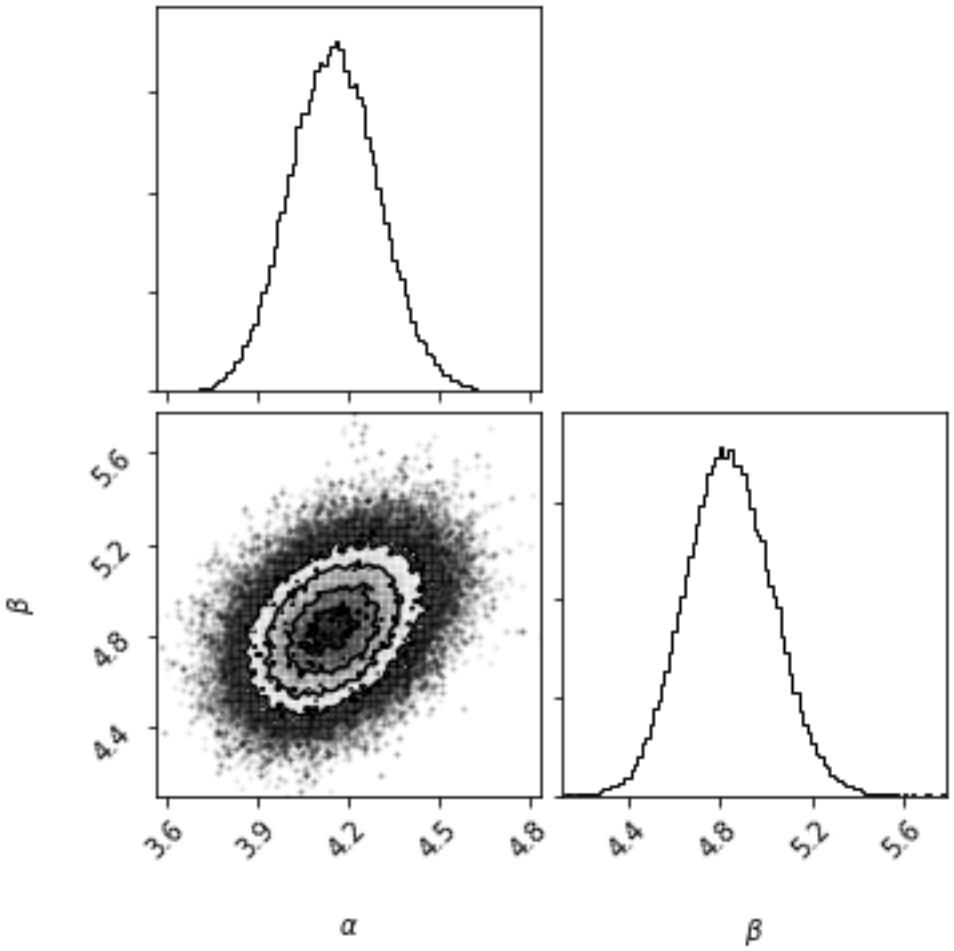


The most probable position of the lighthouse can be found by manually changing the nWalkers while computing the median for each parameter. For 1000 data points, 32 walkers, and a chain length of 1000, I got the following graph



where the medians for alpha and beta were 0.9885793949999999 and 0.9676138599999999 respectively.

When putting a second source, the system cannot detect the interloper because MCMC calculates the likelihood of each alpha/beta combination to get the data as if it was from one source. It averages the values and goes farther away in the beta direction to lessen the angle. Here is the resulting graph for 1000 data points, 2000 walkers, and a chain length of 1000 where one source was at (3, 1) and the other was at (15, 10) for (alpha, beta).


β
5.6
5.2
4.8
4.4
3.6
3.9
4.2
4.5
4.8
α
4.4
4.8
5.2
5.6
β