PS1_P3

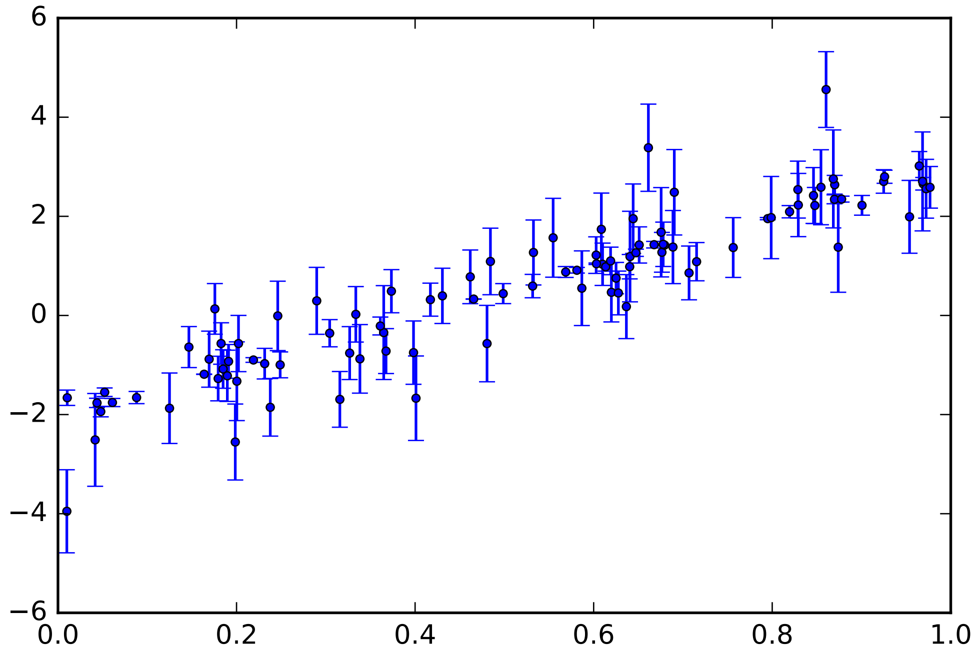
September 13, 2016

```
In [1]: %matplotlib inline
        import numpy as np
        import matplotlib.pyplot as plt
        from IPython.display import set_matplotlib_formats
        set_matplotlib_formats('png','pdf')
```

0.0.1 Problem 3: Linear Best-Fit Parameters

First, we generate some fake data with Gaussian noise, given linear parameters $\theta = (m, b) = (5.0, -2.0)$

```
In [465]: m,b = 5.0, -2.0

          def gen_data(num):
              xdat = 1.*np.random.rand(num)
              ydat = m*xdat+b
              sigma = np.random.rand(num)
              y_err = np.random.normal(0.0,1.*sigma,num)
              return np.stack((xdat,ydat+y_err,sigma),axis=-1)


          data = gen_data(100)
          plt.errorbar(data[:,0],data[:,1],yerr=data[:,2],fmt='o',markersize=3)

Out[465]: <Container object of 3 artists>
```



Next, we define the likelihood, proposition, and Markov chain functions as in Problem 2. The main differences are:

1.) We transform to log-space for the likelihood. The reason for this is, assuming Gaussian errors, the log probability becomes identical to chi-squared minimization and makes the math easier (see Hogg 2010).

2.) The proposition function becomes multivariate. We here opt to assign the same variances for m and b, which may not be correct.

In [412]:
```
#plt.errorbar(data[:,0],data[:,1],yerr=data[:,2],fmt='o',markersize=3)

def log_likelihood(theta,data):
    chi_sq = np.sum((data[:,1]-theta[0]*data[:,0]-theta[1])**2/(data[:,2]*data[:,2]))
    return -.5*chi_sq

def prop(theta_old,mc_params):
    mean = [0,0]
    cov = [[mc_params[0],0],[0,mc_params[0]]]
    theta_prop = theta_old + mc_params[1]*np.random.multivariate_normal(mean,cov,1)
    return theta_prop[0]

def new_state(t_old,t_prop,data):
    alpha = log_likelihood(t_prop,data)-log_likelihood(t_old,data)
    if alpha >= np.log(1.0):
        return t_prop
    else:
        if alpha >= np.log(np.random.rand(1)):
            return t_prop
    return t_old
```

Here we define the actual run function, where we pick an offset starting point $(3.0, -3.0)$ to test the code's capability to properly converge to the expected result.

This function returns a set containing the vector of parameters (dimensions $2xN$), the log likelihoods of the parameters, and the acceptance ratio.

```
In [419]: def run_mcmc(num_dat,num_theta,mc_params):
              dat = gen_data(num_dat)
              t_start = [3.0,-3.0]
              theta = np.array((t_start))
              ln_l = np.array((log_likelihood(t_start,dat)))

              theta_old = t_start
              theta_prop = t_start
              theta_new = t_start

              while np.size(ln_l) < num_theta:
                  theta_old = theta_new
                  theta_prop = prop(theta_old,mc_params)
                  theta_new = new_state(theta_old,theta_prop,dat)
                  theta = np.vstack((theta,theta_new))
                  ln_l = np.append(ln_l,log_likelihood(theta_new,dat))

              accept = np.size(np.unique(ln_l))/num_theta

              return [theta,ln_l,accept]
```

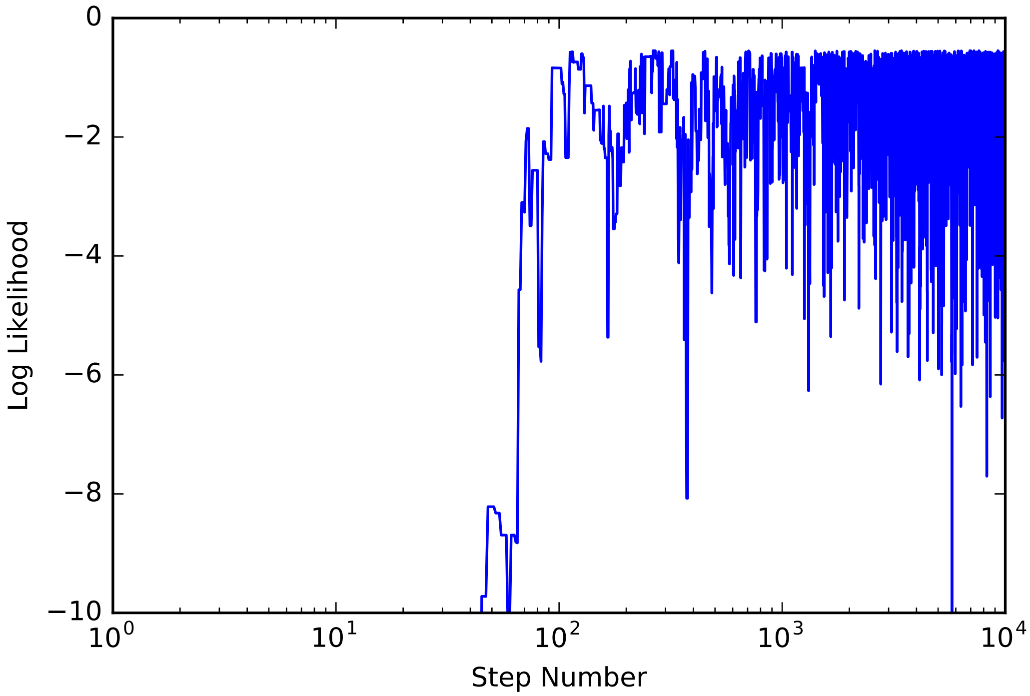
Run for 10 data points:

```
In [421]: run_10 = run_mcmc(10,10000,(2.0,0.1))
          print(run_10[2])
```

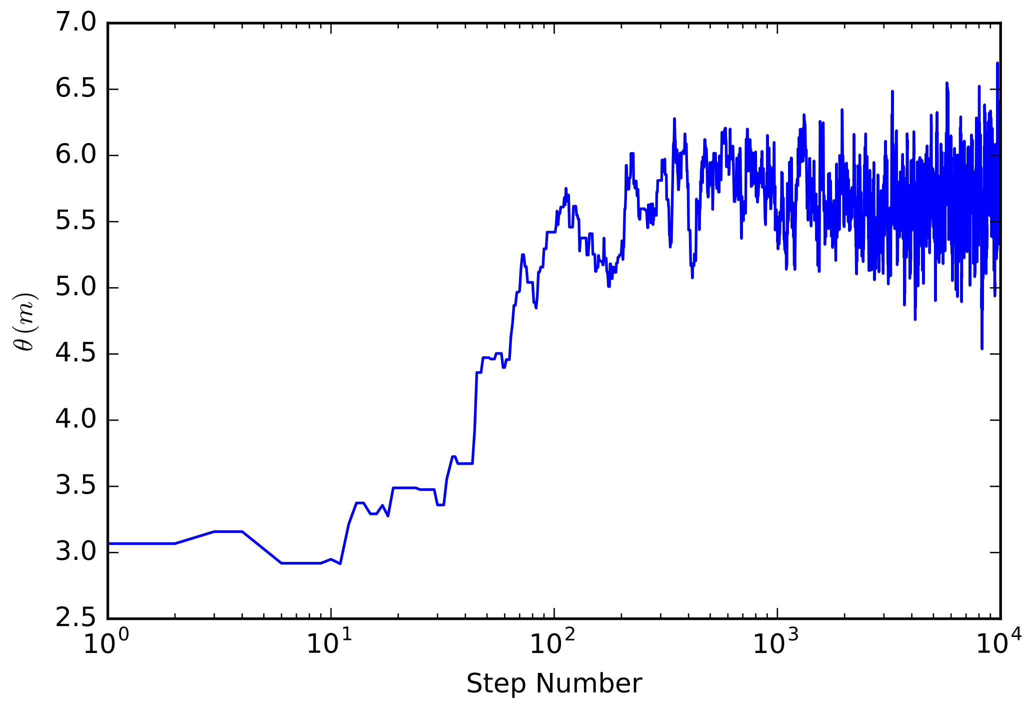
```
0.3681
```

```
In [466]: plt.plot(range(10000),2.5+run_10[1])
          plt.ylim(-10,0)
          plt.xscale('log')
          plt.xlabel("Step Number")
          plt.ylabel("Log Likelihood")
```

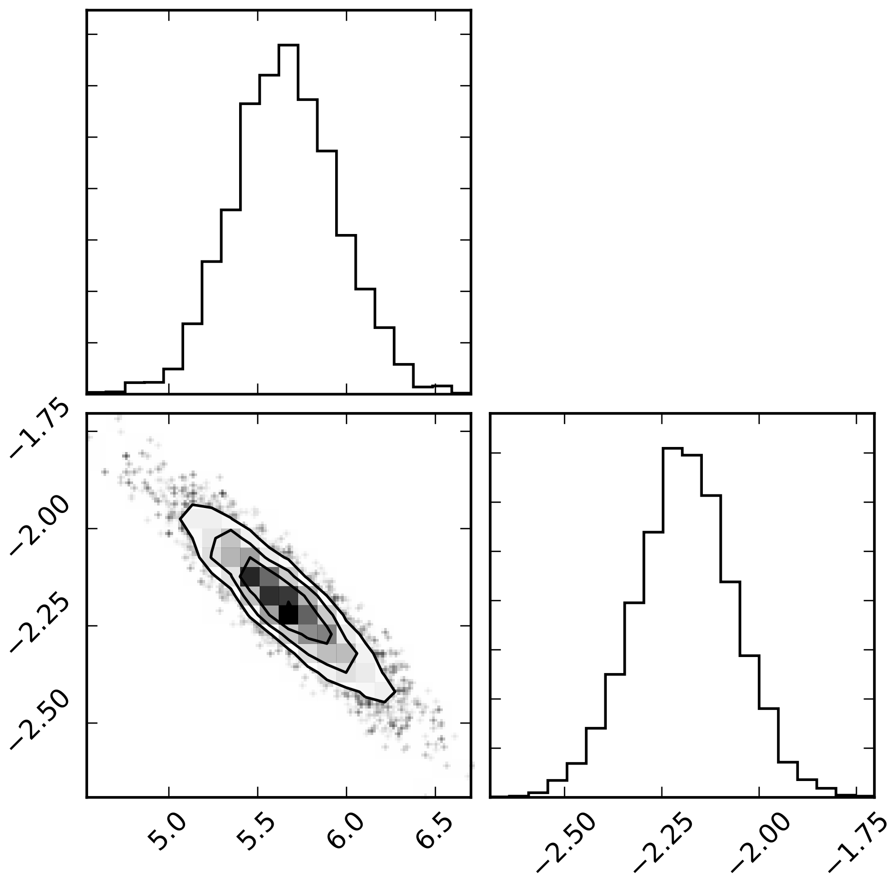
```
Out[466]: <matplotlib.text.Text at 0x115442208>
```



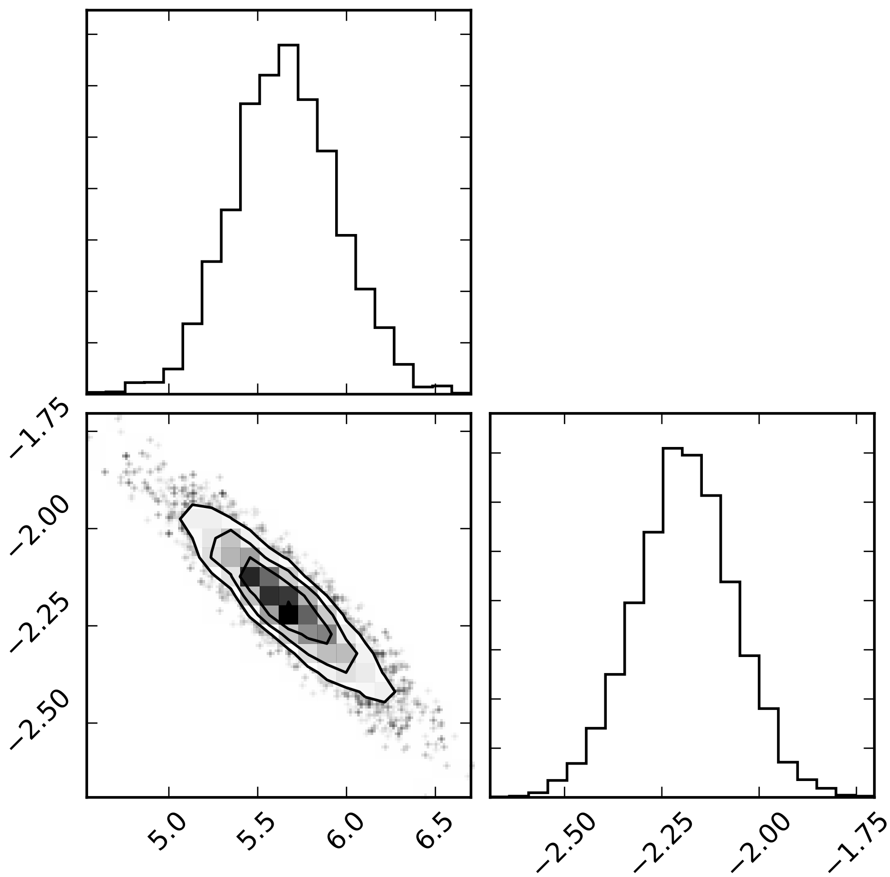

```
In [469]: plt.plot(range(10000),run_10[0][:,0])
          plt.xscale('log')
          plt.xlabel("Step Number")
          plt.ylabel(r'$\theta\,(m)$')

Out[469]: <matplotlib.text.Text at 0x1121d5ef0>
```




```
In [481]: import corner
          corner.corner(run_10[0][1000:9999,])

Out[481]:
```



−1.75
−2.00
−2.25
−2.50
5.0
5.5
6.0
6.5
−2.50
−2.25
−2.00
−1.75



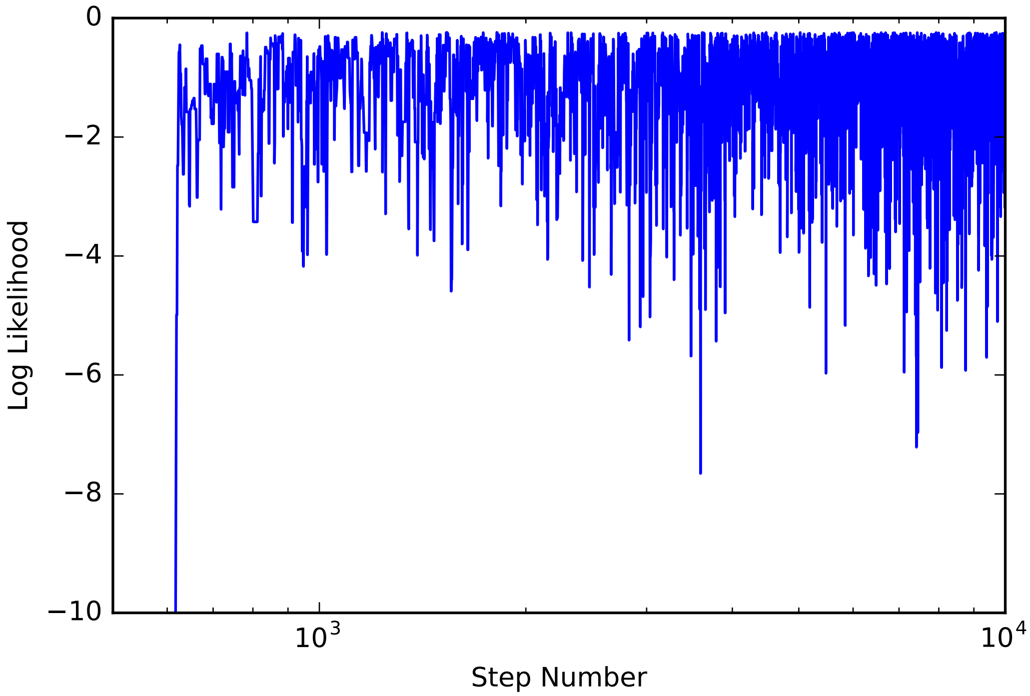
Run for 100 data points:

```
In [423]: run_100 = run_mcmc(100,10000,(1.0,0.0125))
          print(run_100[2])
```

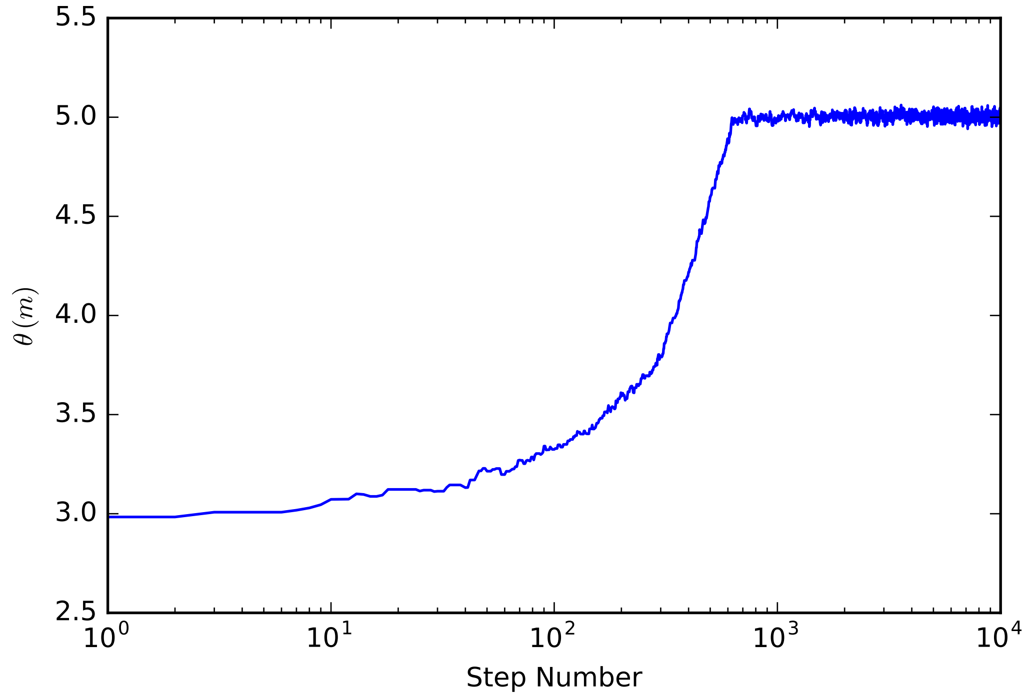
```
0.4185
```

```
In [470]: plt.plot(range(10000),60+run_100[1])
          plt.ylim(-10,0)
          plt.xscale('log')
          plt.xlim(5e2,1e4)
          plt.xlabel("Step Number")
          plt.ylabel("Log Likelihood")
```

```
Out[470]: <matplotlib.text.Text at 0x11343ca20>
```

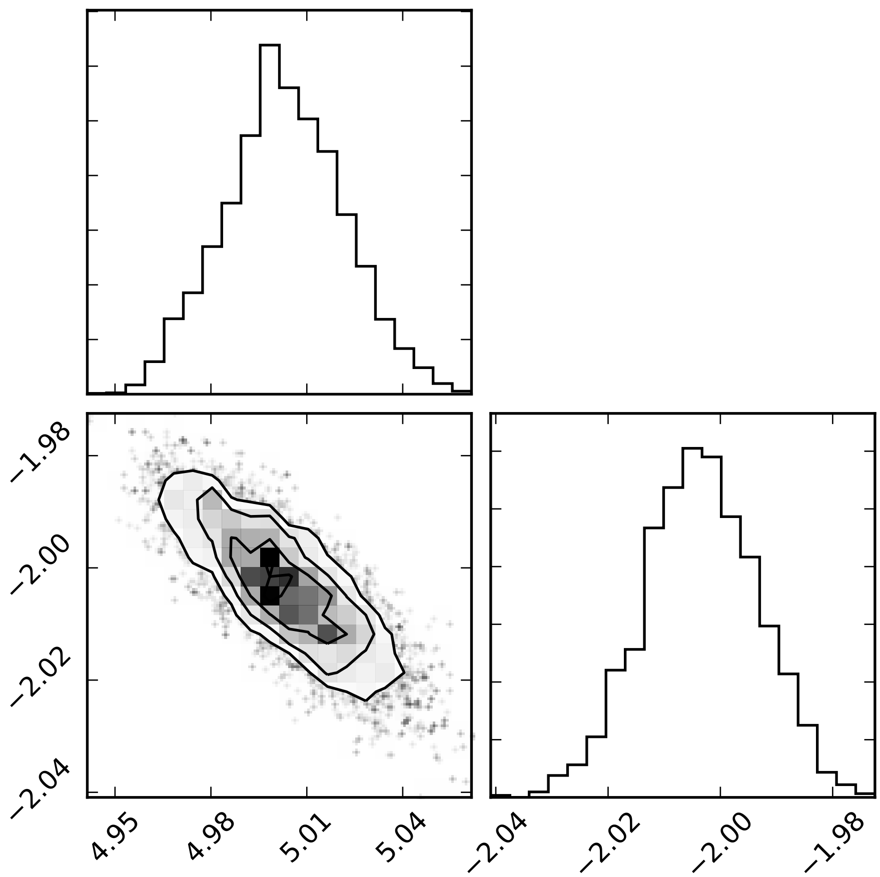


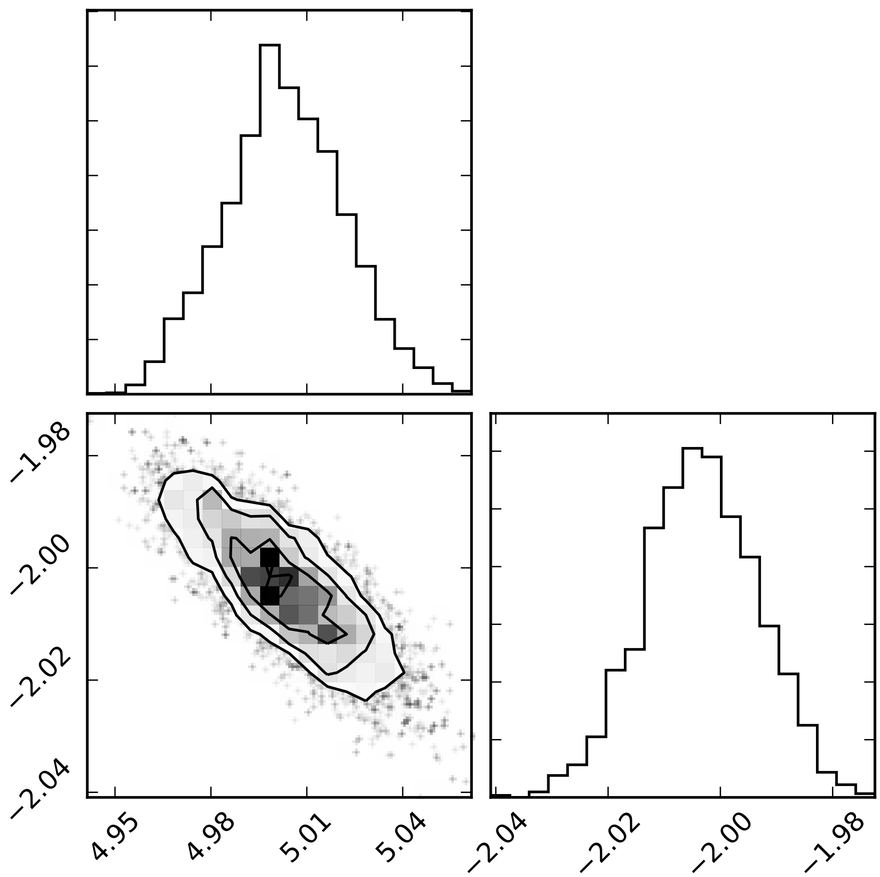

```
In [471]: plt.plot(range(10000),run_100[0][:,0])
          plt.xscale('log')
          plt.xlabel("Step Number")
          plt.ylabel(r'$\theta\,(m)$')

Out[471]: <matplotlib.text.Text at 0x1135e1dd8>
```




```
In [480]: corner.corner(run_100[0][1000:9999,])
```

Out[480]:


−1.98
−2.00
−2.02
−2.04
4.95
4.98
5.01
5.04
−2.04
−2.02
−2.00
−1.98



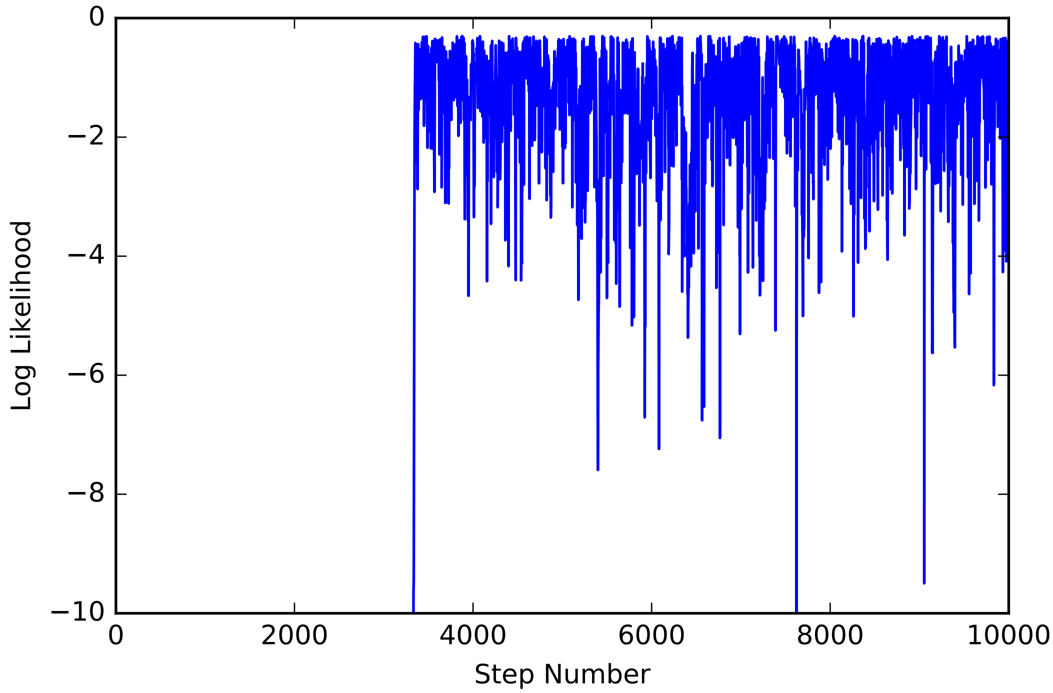
Run for 1000 data points:

```
In [438]: run_1000 = run_mcmc(1000,10000,(5.0,.001))
          print(run_1000[2])
```

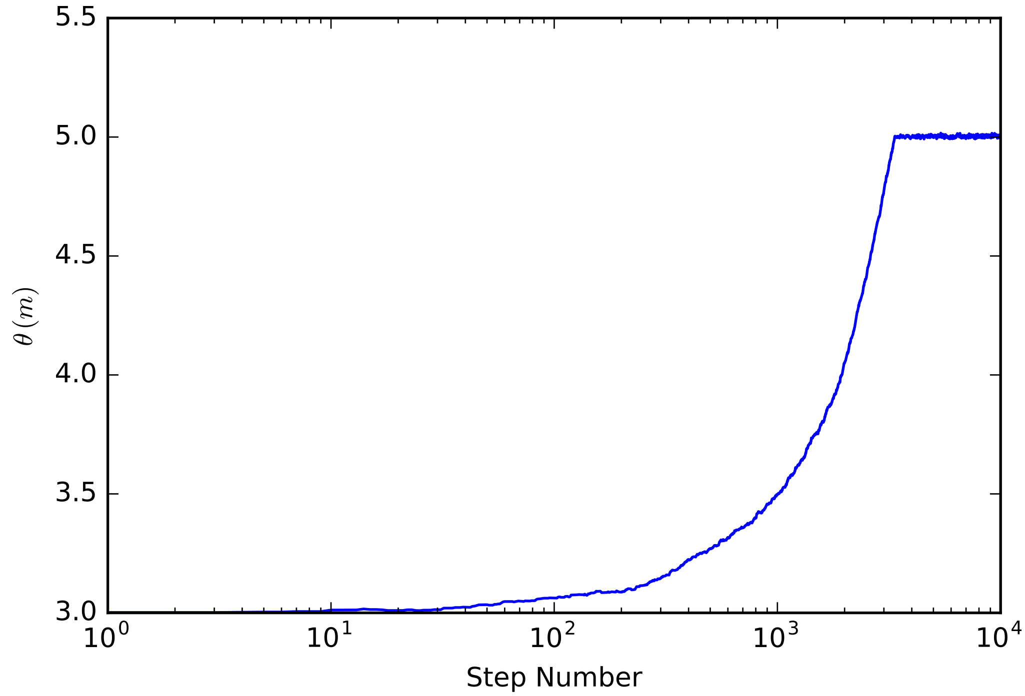
```
0.445
```

```
In [472]: plt.plot(range(10000),475+run_1000[1])
          plt.ylim(-10,0)
          plt.xlabel("Step Number")
          plt.ylabel("Log Likelihood")
```

```
Out[472]: <matplotlib.text.Text at 0x112b24f28>
```

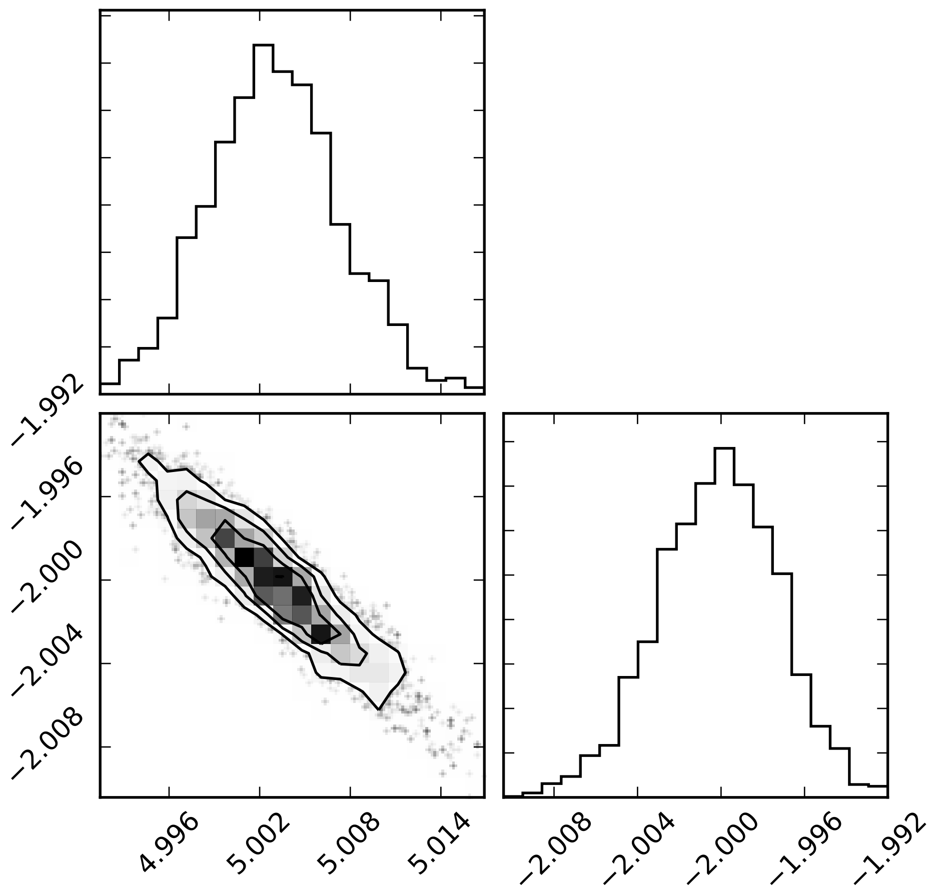



```
In [473]: plt.plot(range(10000),run_1000[0][:,0])
          plt.xscale('log')
          plt.xlabel("Step Number")
          plt.ylabel(r'$\theta\,(m)$')

Out[473]: <matplotlib.text.Text at 0x11537b748>
```

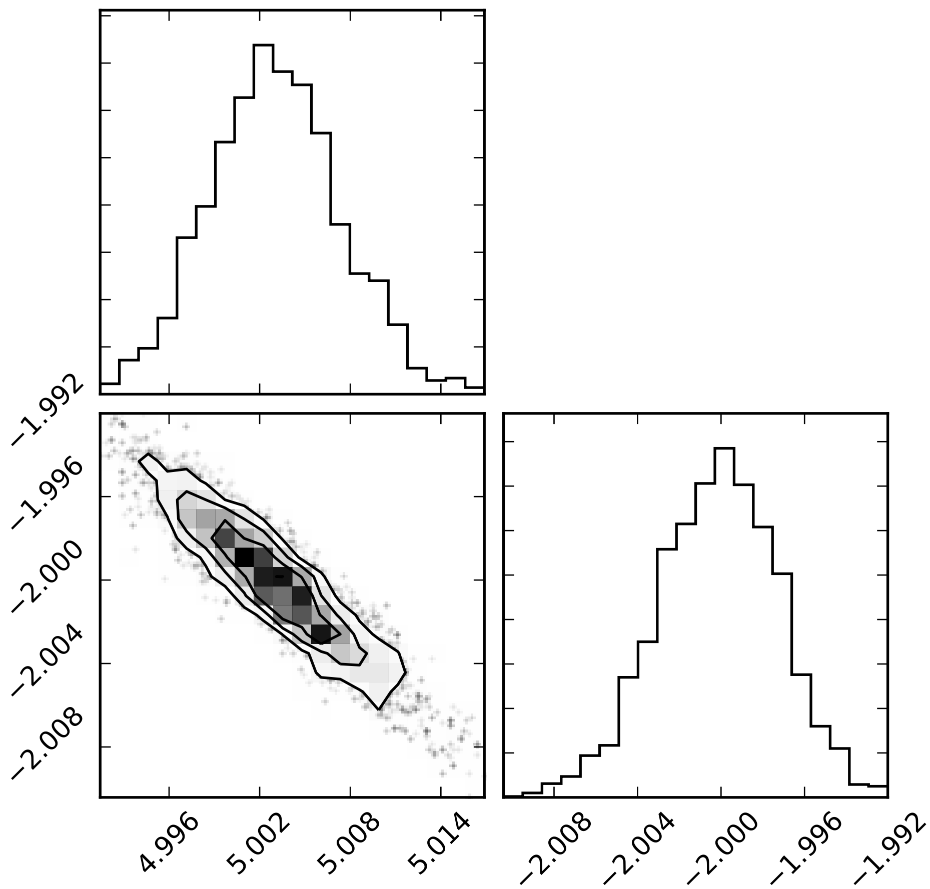



```
In [484]: corner.corner(run_1000[0][4000:9999,])
```

```
Out[484]:
```



−1.992
−1.996
−2.000
−2.004
−2.008
4.996
5.002
5.008
5.014
−2.008
−2.004
−2.000
−1.996
−1.992




```
In [485]: import emcee

In [498]: ndim, nwalkers = 2, 6
          p0 = [np.random.rand(ndim) for i in range(nwalkers)]

In [499]: data = gen_data(100)

          def lnprob(theta):
              return log_likelihood(theta,data)

In [502]: sampler = emcee.EnsembleSampler(nwalkers,ndim,lnprob)
          pos, prob, state = sampler.run_mcmc(p0,1000)
          sampler.reset()
          sampler.run_mcmc(p0,8000)

Out[502]: (array([[ 5.03319632, -2.0158658 ],
                  [ 5.03664352, -2.01525742],
                  [ 5.00926287, -2.00181753],
                  [ 5.07036828, -2.0214412 ],
                  [ 4.97610477, -2.00032446],
                  [ 5.01626723, -1.99940283]]),
           array([-39.40844419, -38.90450013, -38.87164836, -39.09543152,
```

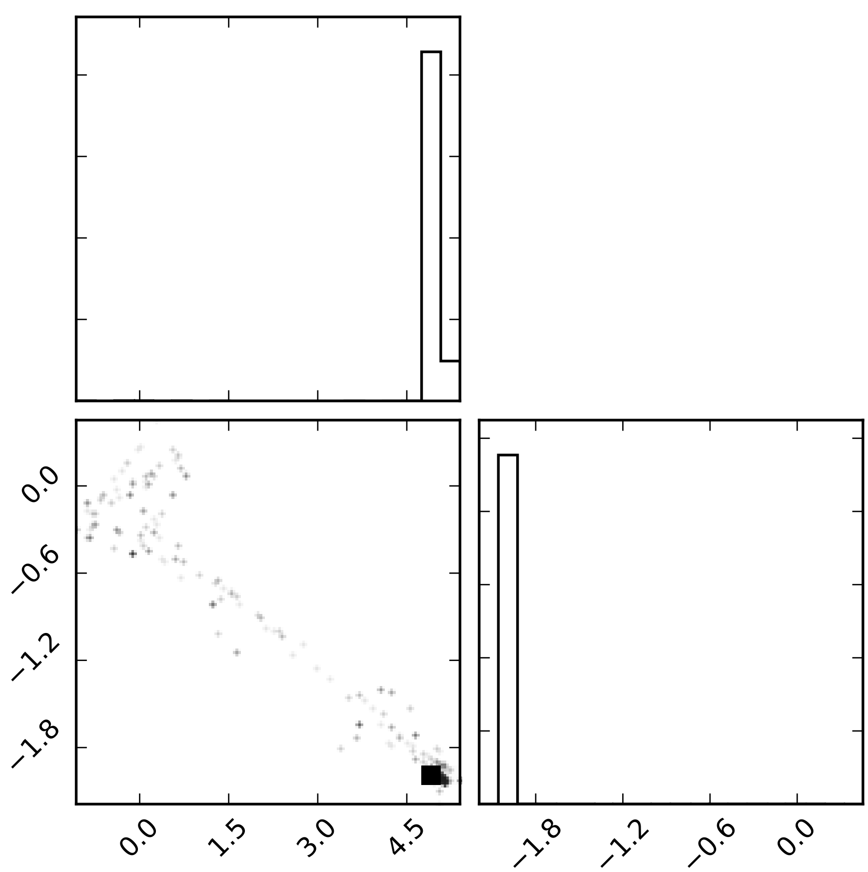
```
      -41.23293594, -39.9099519 ]),
('MT19937', array([2661258919, 3697918871, 2459200446,   98004938, 3149703370,
       3020146788, 1259653702, 3608480883, 3687483423,  573802388,
       1824062751,  931354260, 4219190740,  517753815,  400993887,
       1764447601, 1778658552, 4166509907, 1101277720, 3580021564,
       3793646193,  811147692,  368653999, 2178209415, 3766310206,
       3453690948, 2177375650,  149323420, 2946383535,  340037163,
       3094113982, 1411484420,  857918091, 2207518168, 4008908236,
       3285054760, 1288827374,  548385571, 2156970209, 2654972850,
       1687108606,  537678661,  794139077, 2638449707,   88648490,
        488920891, 3192449172, 3891755206, 2848612494, 4047259359,
       1614829591, 3290497305, 3476305391, 3453578956, 3844229439,
       3272121280, 3739119030, 2133876481,   54566626,  804522574,
       1735863256, 2752240509,  973283007,   82162923, 2956086716,
        905044800, 2570364579, 2897363129, 4116813455, 2597453366,
       1488389000,  235381623, 2386402437, 2995181847,  125318206,
        867792855,  269960185, 2481219129, 2386957789,  464490624,
       1444356856, 2351657574, 4260182640, 1964448331, 2744794301,
       1360095806,   54356087, 1402343900, 3194316935, 1519985666,
       2099224908, 2359673709, 2012676819,  165176117, 1599099697,
       2693482153,  781248910, 2193330103, 3489894264,  127674584,
       2996726085, 1544036181, 1121456503, 4105575946, 4253705949,
        516083347, 1130464085, 4101799286, 3190518265, 1742547151,
       1604013853, 1265640158, 1595724823, 1931526843, 1766534485,
       1286217755, 3036461970, 2949307324, 2729960633, 2392313960,
       3509354553, 2334337787, 2080416891, 3102348288,  315388779,
        498969020, 1025705918, 1537535690,   26066208, 2594036548,
       3099184017, 1002411653, 2119872196,  456621186, 3078731867,
       1236927211, 3366057120, 2253853815, 3940208292, 1616533234,
       3099681241, 3410080625, 2739568420,  768264650, 1759292221,
       1462776041, 3193524518,  585561949,  695554682, 1713248944,
       1621568322, 2285127863,  847990944, 4259530623,  615852485,
        728880301, 2837478419, 1543046627, 2583515339, 1686836869,
       2035779879,  451041903,  365560709,  593360919, 1333197843,
          8836679, 3398400767, 1922269836,  346456323,  533224391,
       1496220056, 2365153457, 4206207922, 3610373278, 3867910555,
       2611836961, 3686086476, 3112049904, 2918861451, 2890726876,
       1398970010, 3665429311, 1331875207, 1196551749, 1948239208,
       3779236023, 3763482875, 2061607416, 1626958185, 3486804747,
       4272479290, 3104881293, 3858287393, 3323536768, 1393132991,
        748261516,  637842887, 1019358114, 2903620533, 1371647287,
       1480349734,   59894214, 3345417630, 3735450306, 1442490872,
       3582752626, 1559366137, 2134518543, 1443003668, 2873775237,
         41006386, 1348662156, 3564563085,    1679324, 1738839637,
       1705414125, 2581797339, 3296032043, 3752977017, 1807705673,
       1635286298, 2325102538, 1533441978,  548165422, 3986224652,
        409398453,   68904900,  894543189, 2048653318,  398600067,
        641380916,  653600779, 3065135281,  695990355,  446625966,
       4146051018,  911449678, 2083888056, 2268385521, 4247951138,
       1577143570, 3535032115,  516492673, 1470190506,  128592933,
       3107304002, 3682091664, 1823524537, 1922628297,  114788602,
       3746393437,  901300160,   88841430, 4109783643, 2526926186,
        743602374, 3260343699, 1841260562, 3242645932, 3456552060,
       3477288871, 4287615955, 1786073627,  734872599,  444954215,
```

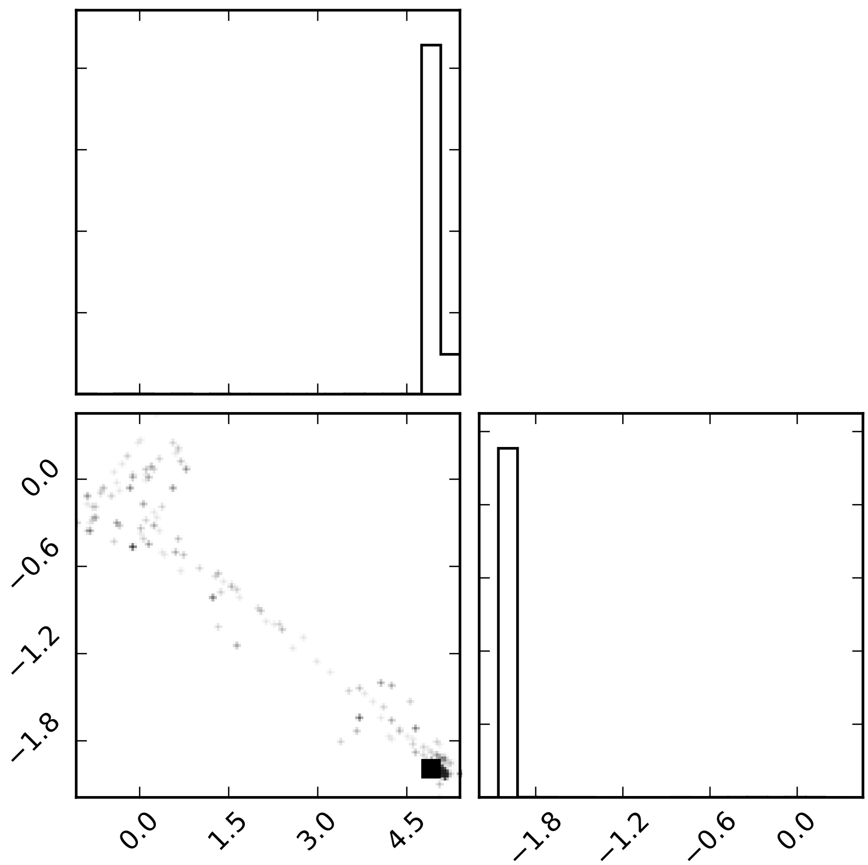
1460502496, 1342057216, 1807463123, 1580053592, 3433724266,
2124972501, 324034032, 4023857831, 2835506131, 2409734015,
1792050377, 2521746448, 391852997, 2022821660, 2467207687,
2338272014, 2714249801, 4258627247, 1113810078, 1741772531,
1363309285, 3040564733, 2643245200, 1070380311, 1443217662,
3097892768, 2207743916, 3186047176, 1849697144, 3886790969,
1332599185, 162041490, 272077598, 2260144688, 3981586664,
2222613972, 2095294064, 1820660063, 3788707914, 659943958,
1902353781, 2761641118, 2704846335, 1213677369, 2130842188,
1824521886, 828731607, 3140559056, 3039629921, 3308338590,
4157324304, 2449842014, 823254510, 663407208, 172808495,
2011845097, 633725854, 4153629459, 3425592303, 1427451345,
2144362500, 2523464056, 962693663, 3993933082, 1506675971,
3590722806, 4139809484, 3025505356, 2112152332, 1832883007,
3310408854, 3762790239, 1390796048, 2119124423, 997047573,
199478285, 256552336, 1466588891, 3537034272, 2397257837,
1584630123, 2662144646, 3822747357, 3220345615, 3267149828,
3325538862, 1407058986, 2197701384, 1011065300, 1969677030,
3565110838, 1114752863, 2269357580, 254978132, 25170603,
319440066, 3389441740, 904740390, 561474838, 2955133197,
1729178942, 52980212, 3065794667, 2073351123, 650515864,
4271666510, 883015947, 2021554044, 1743261567, 3549436402,
1520087715, 3057871412, 4241045549, 2214021630, 3490774802,
972787705, 1627137833, 3525936882, 2806302518, 2280156960,
669243583, 3215759603, 3237547003, 995840876, 3806275431,
4118885291, 2857119604, 2260065636, 2945054623, 822429400,
1918690324, 2093208820, 1511033418, 1471732166, 3452693700,
3670751957, 92022264, 877684979, 1239665924, 1014236471,
430120773, 3535625208, 3527053895, 746872316, 2363089066,
2118803410, 3167958327, 3919324817, 1564828839, 2671862992,
1141959534, 3389114404, 897601917, 2729600346, 4129177970,
1886162760, 882049105, 949779991, 3729864389, 3882116038,
890574073, 376474094, 2440044893, 1334994636, 3702879876,
3927805230, 2587890945, 4056035564, 2394656518, 2429712432,
1341391393, 4058627056, 114148185, 1798531748, 1753481157,
3420002761, 763036588, 2782589149, 4138304379, 3053906941,
3655245346, 2454253977, 3265921336, 3662029347, 971200755,
3265350447, 3467789651, 4123935480, 3991301160, 871543729,
2112426361, 4238280005, 1800601642, 2037711213, 684820792,
2159677874, 2468356770, 3712014923, 598230298, 315767406,
3515609187, 2289081709, 1405855321, 1948431448, 1151242331,
3441225580, 3123133600, 2911240738, 534205570, 2749738707,
3478581929, 3737585067, 3548904572, 1806271268, 2389191680,
3698890435, 1864731823, 2275207984, 1517077492, 176155452,
1199366503, 2299711064, 1697559433, 3511661529, 1607592603,
1582042318, 3463685511, 3910658227, 1666912121, 1115567693,
2927642676, 116523227, 1265943465, 2092106323, 3143719139,
2119196216, 2790737339, 1134798399, 503233803, 296216205,
3586523721, 4249926280, 3437480118, 3422360373, 1234310812,
3127068221, 2395436988, 1012501101, 313244527, 3257988763,
1197265201, 2946247269, 3609186909, 1579853997, 3343261206,
1288933358, 3717147210, 3313802648, 3338505523, 85643722,
2926310430, 517244091, 3142897220, 2499585011, 2983841704,
3849608147, 3957350941, 73359967, 2154295527, 1713879994,

```
2828967241, 3229727738, 3199458754, 3390060232, 2901418851,
4036537853, 3616854946, 1528096062, 3539667034,  313506324,
1984505936,  613914342, 1431506709, 2130093177, 2459775924,
2712245068,  157334357, 4013818056, 3464231809,  868190171,
2702326094,  415257256, 1576603394,  200046363, 1057181586,
3617593020, 1310801796,  993198146, 2741097379, 4169301057,
 934490946, 2536720273, 3085733843, 3063424773, 3447224831,
4077197740, 3307561011, 1040501468, 2316108127, 4271117109,
 621144321, 2196782258, 1256979843, 2754153678, 2040852444,
3332816500, 2988145151, 1478290377, 1213907149,  458544498,
2703467144,  797144241,  115087922, 1873580884, 3244383755,
3671358924, 3455751741, 1375927330, 2762712320,  940331925,
3161258867, 3866603049, 3381175571, 1697739856, 3229683311,
 621715535,  123631315, 2718598150, 3300284185, 2183767955,
 511740500, 3119625065, 3927867591, 2681326551,  600003293,
3129491921, 2497930207, 1541131539, 4188470939,  758193285,
3139691206, 3686017513, 2551215875, 1172109125,  757151641,
2320261930, 1583615904, 1331016175, 3934564216], dtype=uint32), 487, 0, 0.0))
```

In [503]:
```
import corner
corner.corner(sampler.flatchain)
```

Out[503]:






```
In [ ]:
```