Week 2 Assignments

Inferring the Astrophysical Population of Black Hole Binaries

Name: Osase Omoruyi
Mentor: Alan Weinstein
LIGO SURF 2017

Caltech, CA
LIGO Laboratory Astrophysics Group

12th July 2017

Abstract

During the second week, we focused on simulating the remaining parameters of Binary Black Holes (BBH) and becoming familiar with Bayesian Analysis.

Contents

1 Parameter Simulations

1.1 Individual Mass

In the previous week, we simulated the total mass of the BBH. This week, we investigated the mass parameter even further by simulating the mass of each black hole within the BBH. To solve for each black hole's mass, we created a system of equations using the total mass and the symmetric mass ratio parameter eta.

Eta describes how the individual black hole masses relate to one another.

$$Eta = \frac{(M_1 * M_2)}{(M_1 + M_2)^2}$$

We also know the total mass.

$$M_{total} = M_1 + M_2$$

Using these two equations, we could solve for M_1 and M_2 in terms of total mass and eta.

$$M_1 = \frac{M_{total} + \sqrt[2]{(M_{total}^2 * (1 - 4 * eta))}}{2}$$

$$M_2 = \frac{M_{total} - \sqrt[2]{(M_{total}^2 * (1 - 4 * eta))}}{2}$$

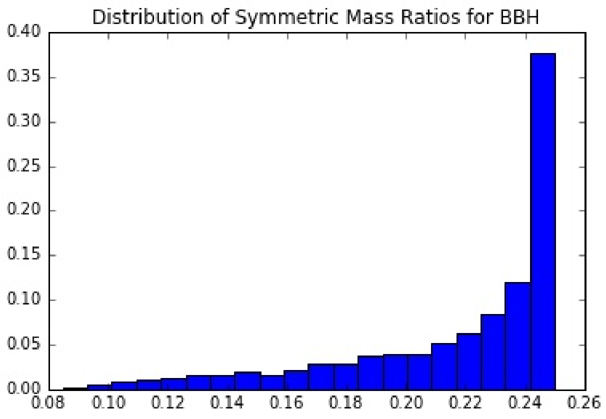

Figure 1: Distribution of Symmetric Mass Ratios (ETA) for BBH

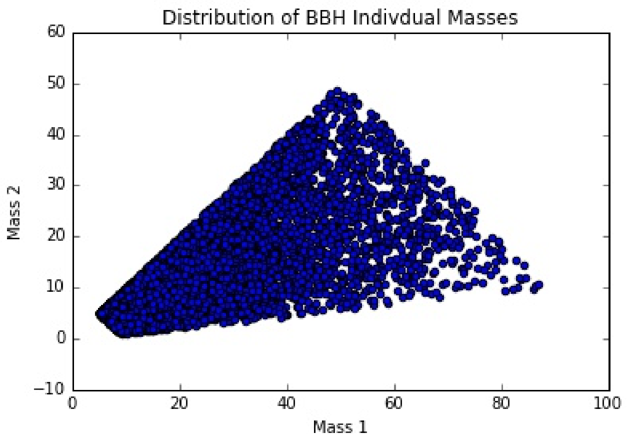

Figure 2: Distribution of BBH Individual Masses

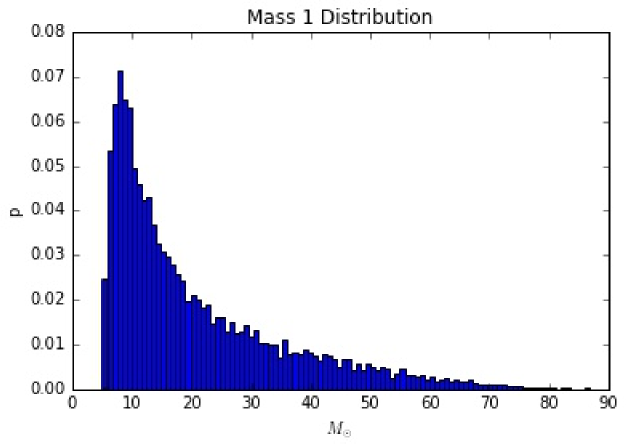

Figure 3: Distribution of BBH Individual Masses

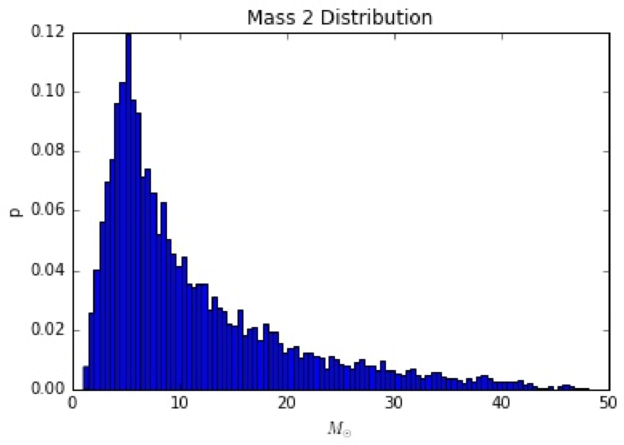

Figure 4: Distribution of BBH Individual Masses

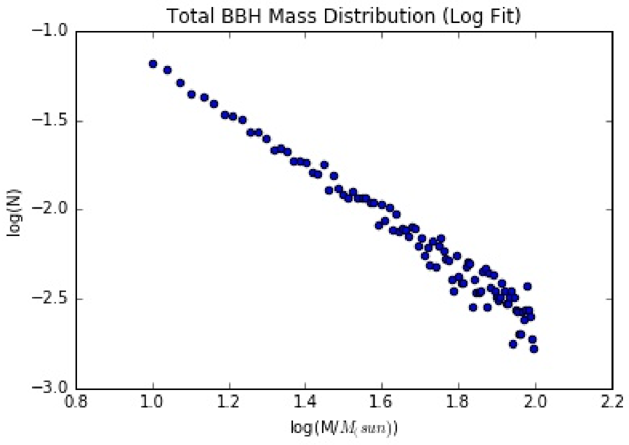

Figure 5: Total BBH Mass Distribution (Log Fit)

1.2 Spin

To simulate the distribution of BBH spin, we assumed the spins were distributed normally with a mean of 0.7 and std of 0.1.

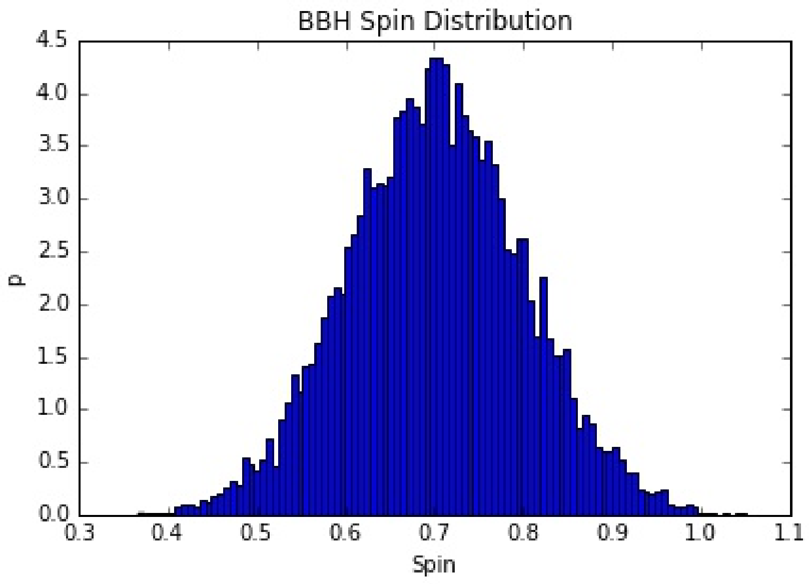

Figure 6: Radial Distribution of BBHs within 1 Gpc.

1.3 Detector Duty Cycle

Another factor we need to take into account to simulate BBH is when they may be detected. Because the detectors are only on an average of 60% of the time during an observing run, we will not be able to detect all events that occur during that period.

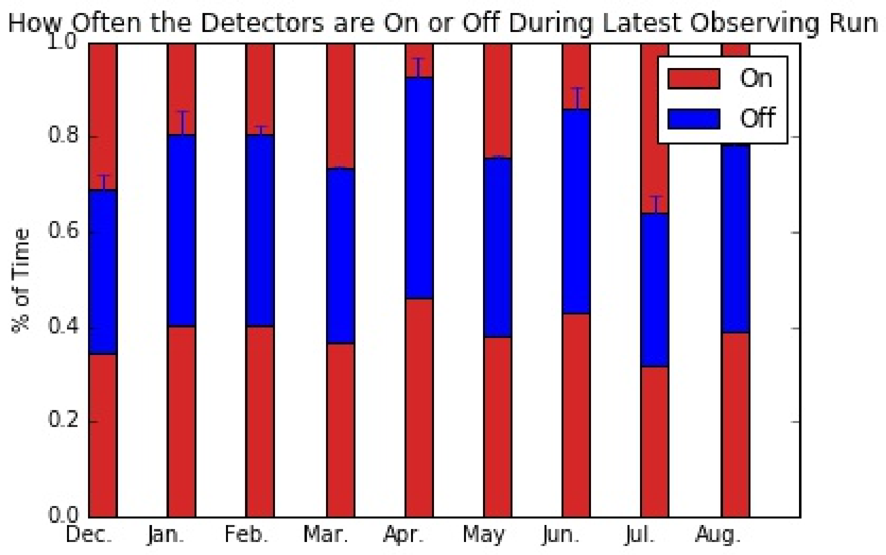

Figure 7: How Often the Detectors are On or Off During Latest Observing Run.

When detecting events, we need to detect the event in both detectors to ensure the event actually occurred. As a result, we need to know the probability of detecting an event in both detectors given that the Livingston detector is on 61.5% of the time and the Hanford detector is on 56% of the time. To do this, we assumed 10000 BBH mergers occurred over the course of a 9 month observing run. The duty cycle (time detector is on) for each detector is a gaussian distribution with std 0.05. We threw random number between 0.6 and 0.8 to determine the actual duty cycle of the detector that week since the duty cycle varies per week. We then generated random times for BBH occurrences between 0 and 1. If the number was greater than duty cycle, the detector was on. If less, the detector was off.

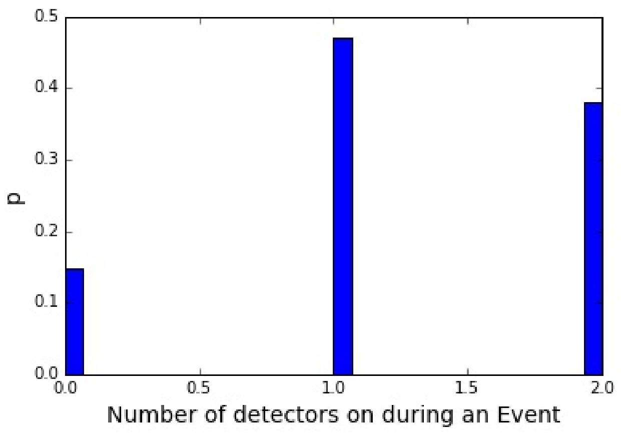

Figure 8: Number of detectors on during an Event

Within the next 10 years, LIGO expects to have 5 detectors running. So we plotted the probability of more than two detectors being on during an event.

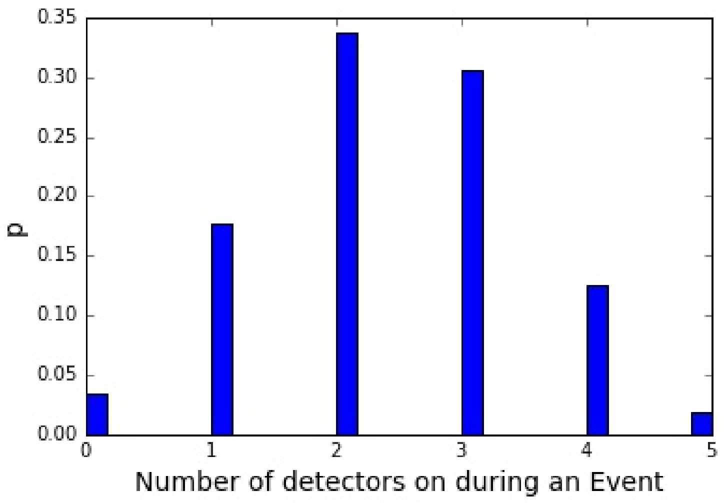

Figure 9: Number of detectors on during an Event

1.4 Using Bayesian Analysis To Plot the Mass and Spin Distribution of BBH

This week, we focused largely on becoming familiar with Bayesian Analysis and how to apply it to our simulations of BBH.

We used PyMC to simulate the mass and spin distributions.

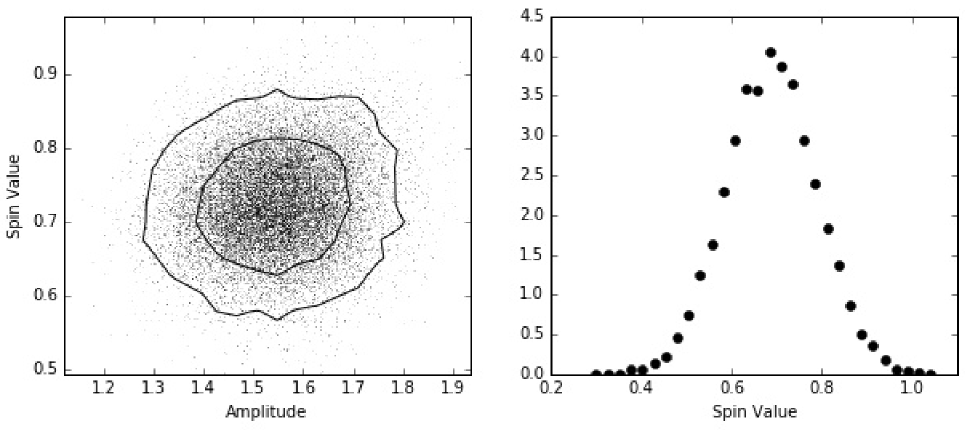

Figure 10: Spin Fit

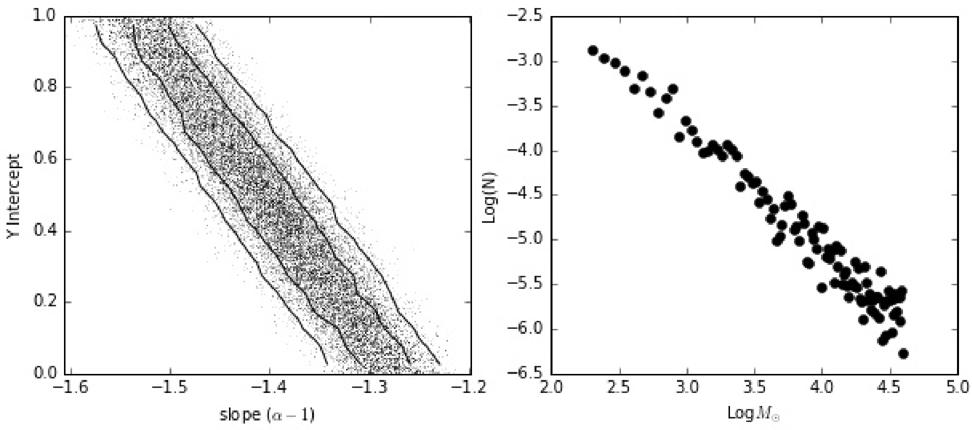

Figure 11: Mass Fit

Appendix A Code

Code used to obtain simulated BBH distributions according to sky location, luminosity distance and mass.

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import LogNorm
import scipy.stats as stats
from scipy.stats import rv_continuous
import pymc as mc
from pylab import *
import scipy
from scipy.interpolate import UnivariateSpline
from scipy.stats import norm
from numpy.random import normal
from scipy.optimize import curve_fit
from scipy import asarray as ar,exp
from __future__ import division
import math
from scipy.special import gamma
import thinkbayes2
from thinkbayes2 import Pmf, Suite
import thinkplot

Assignment 4: Simulate the spin distribution of BBH

Approach: Spins are distributed in a gaussian distribution of 0.7 mean and 0.1 std

#create a gaussian function

def gaus(x,a,x0,sigma):
     #http://www.upscale.utoronto.ca/PVB/Harrison/ErrorAnalysis/Images/GaussFormula.gif
    return a*exp(-(x-x0)**2/(2*sigma**2))

gaussian_numbers = normal(0.7,0.1,size=10000)
plt.hist(gaussian_numbers, bins=100, normed=True)
plt.title("BBH Spin Distribution")
plt.xlabel("Spin")
plt.ylabel("p")
plt.savefig("Spinhi.jpg")
plt.show()

Assignment 5: Plot a function showing how the individual masses of BBH are related.

Approach: We know that the symmetric mass ratio parameter eta is (mass1*mass2)/(mass1
     + mass2)^2. Eta is largest (0.25) when mass1=mass2. Plot this from 0 to 0.25.

#function for individual mass relation
def SMR(mass1, mass2):        #COM coordinate equation
    return (mass1*mass2)/(mass1 + mass2)**2
```

```
m_1 = np.random.uniform(10,100, size = 10000)
m_2 = np.random.uniform(10,100, size = 10000)
eta = SMR(m_1,m_2)

#Plot histogram
results2, edges2 = np.histogram(eta, bins=20, normed=True)
binWidth2 = edges2[1] - edges2[0]
plt.bar(edges2[:-1], results2*binWidth2, binWidth2)
plt.title("Distribution of Symmetric Mass Ratios (ETA) for BBH")
plt.savefig("SMR.jpg")
plt.show()

Approach: To find the individual mass, we can create a system of equations. We know
     that eta is (mass1*mass2)/(mass1 + mass2)^2 and that the total mass (M) is mass1
     + mass2. We can create a system of equations and solve.

mass1 = M - mass2
eta = ((M - mass2) * mass2)/(((M - mass2) + mass2)^2

After solving, we get:
mass1 = 0.5* (M + sqrt(M^2*(1-4*eta))
mass2 = 0.5* (M - sqrt(M^2*(1-4*eta))

#function for m1 and m2

M = p2
eta = eta

mass1 = 0.5* (M + np.sqrt(M**2*(1-4*eta)))
mass2 = 0.5* (M - np.sqrt(M**2*(1-4*eta)))

plt.scatter(mass1, mass2)
plt.xlabel("Mass 1")
plt.ylabel("Mass 2")
plt.title("Distribution of BBH Indivdual Masses")
plt.savefig("IM.jpg")
plt.show()

plt.hist(mass1, bins = 100, normed=True)
plt.title("Mass 1 Distribution")
plt.xlabel("$M_\odot$")
plt.ylabel("p")
plt.savefig("m1.jpg")

plt.hist(mass2, bins = 100, normed=True)
plt.title("Mass 2 Distribution")
plt.xlabel("$M_\odot$")
plt.ylabel("p")
plt.savefig("m2.jpg")

Assignment 6: Plot the probablity of detecting BBH vs time the detectors are up

Approach: The detectors are only on 70\% of the time. Assume 10000 BBH mergers occur
     over the course of two years. How many were detected when the detectors were on?

The duty cycle (time detector is on) for each detector is a gaussian distribution with
     mean 0.7 and std 0.05. Throw random number between 0.6 and 0.8 to determine the
     actual duty cycle of the detector that week.
Generate random time for BBH between 0 and 1. If the number is greater than 0.7, the
     detector was on. If less, the detector was off.
```

```
def detector_cycle(ave,std,events):
    return normal(ave,std,size=events)
def duty_cycle(cycle):
    return np.random.choice(cycle)
def on_and_off(events, duty_cycle):
    list_events = np.random.uniform(0,1, size =events)
    list_events[list_events <= duty_cycle] = 1    #on
    list_events[list_events < 1] = 0    #off
    return list_events

#was detector 1 on or off during this event?
d1time_on= detector_cycle(0.615,.05,10000)
d1_duty_cycle = duty_cycle(d1time_on)
eventsd1= on_and_off(10000,d1_duty_cycle)

#was detector 2 on or off during this event?
d2time_on= detector_cycle(0.56,.05,10000)
d2_duty_cycle = duty_cycle(d2time_on)
eventsd2= on_and_off(10000,d2_duty_cycle)

#was detector 3 on or off during this event?
d3time_on= detector_cycle(0.3,.05,10000)
d3_duty_cycle = duty_cycle(d3time_on)
eventsd3= on_and_off(10000,d3_duty_cycle)

#was detector 4 on or off during this event?
d4time_on= detector_cycle(0.6,.05,10000)
d4_duty_cycle = duty_cycle(d4time_on)
eventsd4= on_and_off(10000,d4_duty_cycle)

#was detector 5 on or off during this event?
d5time_on= detector_cycle(0.5,.05,10000)
d5_duty_cycle = duty_cycle(d5time_on)
eventsd5= on_and_off(10000,d5_duty_cycle)

#was detector 6 on or off during this event?

d6time_on= detector_cycle(0.4,.1,10000)
d6_duty_cycle = duty_cycle(d6time_on)
eventsd6= on_and_off(10000,d6_duty_cycle)

#were all, some, one, or none on?
on_or_off = eventsd1 + eventsd2 #+eventsd3+eventsd4+eventsd5+eventsd6

#plot histograms

results3, edges3 = np.histogram(on_or_off, bins=30, normed=True)
binWidth3 = edges3[1] - edges3[0]
plt.bar(edges3[:-1], results3*binWidth3, binWidth3)
plt.xlabel("Number of detectors on during an Event", size=14)
plt.ylabel("p", size = 14)
plt.savefig("detect.jpg")
plt.show()
#plt.savefig("Five Detectors.jpg")

How often are the detectors on or off?

N = 9
timeon= np.full((9, 1), 1)
timeoff = (normal(0.4,0.05,size=9))
onStd = (np.random.uniform(0,0.05, size = 9))
offStd = np.random.uniform(0,0.05, size = 9)
```

```
ind = np.arange(N)    # the x locations for the groups
width = 0.35       # the width of the bars: can also be len(x) sequence

p1 = plt.bar(ind, timeon, width, color='#d62728')
p2 = plt.bar(ind, timeoff, width,
             bottom=timeoff, yerr=offStd)

#plt.ylabel('% On')
plt.title('How Often the Detectors are On or Off During Latest Observing Run')
plt.xticks(ind, ('Dec.', 'Jan.', 'Feb.', 'Mar.', 'Apr.','May', 'Jun.', 'Jul.', 'Aug.'))
plt.ylabel("% of Time")
plt.legend((p1[0], p2[0]), ('On', 'Off'))
plt.savefig("timeonoff.jpg")

plt.show()

Task: Using Pymc to plot mass distribution

Approach: Taking the log of the fit produces a linear function, with the slope equal
     to alpha. Therefore, y = mx+b. y is the log of the number of BBH. m is alpha-1. x
     is the log of the mass. b is the value when x

#Log of mass distribution fit is a linear function.
plt.scatter(np.log(edges[:-1]),np.log(results))
plt.xlabel("log(M/$M_(sun)$)")
plt.ylabel("log(N)")
plt.title("Mass Distribution (Log Fit)")

# A linear fit
# create data
x1 = np.log(edges[:-1])
f = np.log(results)
noise = np.random.normal(size=100) * .1     # create some Gaussian noise
f = f + noise                                # add noise to the data

#priors
sig = mc.Uniform("sig", 0.0, 100.0, value=1.)

a = mc.Uniform("a", -3, 1, value= -1.35)
b = mc.Uniform("b", 0, 1, value= 0.5)


#model
@mc.deterministic(plot=False)
def mod_linear(x=x1, a=a, b=b):
      return a*x + b

#likelihood
y = mc.Normal("y", mu=mod_linear, tau=1.0/sig**2, value=f, observed=True)
#--

S = mc.MCMC([y, a, b])
S.sample(iter=100000, burn=50000)

mc.Matplot.plot(S)

Task: Using Pymc to plot spin distribution

Approach: The spin as distributed in a gaussian function

#Spin distribution is a gaussian function.
gaussian_numbers = normal(0.7,0.1,size=10000)
```

```
histg, binsg = np.histogram(gaussian_numbers, normed=True, bins=30)

plt.plot(binsg[:-1],histg)
plt.xlabel("Spin")
plt.ylabel("Number")
plt.title("Spin Distribution")

# create data
x2 = binsg[:-1]
f2 = histg

#priors
sigma = mc.Uniform("sig", 0.0, 1.0, value=0.5)

a = mc.Uniform("a", 0, 2, value= 1)
x0 = mc.Uniform("x0", 0, 1.5, value= 0.7)

#model
@mc.deterministic(plot=False)
def mod_gaus(x=x2,a=a,x0=x0,sigma=sigma):
    return a*exp(-(x-x0)**2/(2*sigma**2))

#likelihood
y1 = mc.Normal("y1", mu=mod_gaus, tau=1.0/sigma**2, value=f2, observed=True)
#--

G = mc.MCMC([y1, a, x0])
G.sample(iter=100000, burn=50000)

mc.Matplot.plot(G)
```

References

[1] P. Erdős, *A selection of problems and results in combinatorics*, Recent trends in combinatorics (Matrahaza, 1995), Cambridge Univ. Press, Cambridge, 2001, pp. 1–6.