LASER INTERFEROMETER GRAVITATIONAL WAVE OBSERVATORY
- LIGO -
CALIFORNIA INSTITUTE OF TECHNOLOGY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Technical Note **LIGO-T1700223—v1** 2017/07/12

Inferring the Astrophysical Population of Black Hole Binaries

LIGO SURF Progress Report 1

Osase Omoruyi, Yale University
Alan Weinstein, California Institue of Technology

California Institute of Technology
LIGO Project, MS 18-34
Pasadena, CA 91125
Phone (626) 395-2129
Fax (626) 304-9834
E-mail: info@ligo.caltech.edu

Massachusetts Institute of Technology
LIGO Project, Room NW22-295
Cambridge, MA 02139
Phone (617) 253-4824
Fax (617) 253-7014
E-mail: info@ligo.mit.edu

LIGO Hanford Observatory
Route 10, Mile Marker 2
Richland, WA 99352
Phone (509) 372-8106
Fax (509) 372-8137
E-mail: info@ligo.caltech.edu

LIGO Livingston Observatory
19100 LIGO Lane
Livingston, LA 70754
Phone (225) 686-3100
Fax (225) 686-7189
E-mail: info@ligo.caltech.edu

http://www.ligo.caltech.edu/

1 Motivation and Overview: Exploring Binary Black Hole Formation and Evolution From Simulated Data

With massive gravitational influence and head-scratching unknowns at their singularities, black holes are easily some of the universe's most interesting phenomena. Arguably, the only objects more interesting than black holes themselves are two black holes trapped in orbit around one another: Binary Black Holes (BBH). Before LIGO's detection of Gravitational Waves from BBH on September 14, 2015, there was no physical proof that systems like these could exist– only Einstein's equations. However, since then, these systems which were once thought to be imaginary are now being detected in mass through gravitational wave observations. Only two years after the first detection, LIGO has detected an additional 3 events. In the future of Gravitational Wave Astronomy, we expect this number of events to grow rapidly as more detectors are built and refined. With an increasing number of events, we will then be able to make concrete inferences about BBH systems, how they formed, and how they evolved over time.

However, before we jump into the future, we must focus on the present and prepare for the data to come. With 4 known events, we have begun to form a small population of BBH mergers. This number, however, is too small to make substantial inferences about the nature of Binary Black Hole systems. Because we will not have a large sample of events for decades to come, we must devise methods to make inferences about BBH from the data and knowledge we have available now. This is where our research becomes relevant.

The overarching goal of our project is to construct a simulation of Binary Black Holes (BBH) that extends to the most distant galaxies in the observable universe. From these simulations, we will make inferences about the formation and evolution of binary black hole systems. To simulate BBH, we need to create a model that describes how they are distributed throughout space. To create this model, we may identify the parameters that characterize black holes as well as the binaries they lie in. For the purposes of our project, we have employed 15 parameters to describe the binary: sky location (θ, ϕ), luminosity distance (d_L), mass (M, m_1, m_2), spin magnitude (a_1, a_2), spin azimuthal and polar angles (ϕ_{s1}, ϕ_{s2}, μ_{s1}, μ_{s2}), orbital inclination and polarization (i, ψ), time of coalescence (t_c) and phase of coalescence(ψ_c).

Utilizing our current astrophysical knowledge, we may construct simulations of the distribution of BBH in each of the 15 parameters. From these simulations, we may predict the gravitational wave form we will see when the black holes within the binary merge and determine if our detectors would detect the event. After this stage, the goals of each individual in the research group begin to diverge.

My line of the project entails using the simulations we create to discover more about the natural rate density of BBH in the universe. As a result, I am primarily concerned with the simulated mass parameters and the rate at which BBH of certain masses naturally appear in the universe. Knowing this rate will help us understand how most BBH are formed and how they evolve over time.

2 Method: Simulating Black Hole Binaries

We used Python code to create simulations of the distribution BBH in each parameter (See Appendix). To write the code, we needed to create models that would describe the distribution of BBH based on each parameter. To generate these models, we used our astrophysical knowledge about each parameter and built models that would match our expectations given our understanding of black holes and the universe.

2.1 Parameters Describing The Binary

Parameter	Symbol	BBH Distribution
Right Ascension	α	Uniform
Declination	δ	Unifrom in cosδ
Luminosity Distance	d_L	Radial
Orbital Inclination	ι	Uniform
Time of Coalescence	t_c	Uniform
Phase of Coalescence	φ_c	Uniform

Table 1: Summary of Parameters Describing The Binary

Sky Location: Right Ascension and Declination

According to the cosmological principal, the spatial distribution of matter in the universe is homogeneous. Therefore, the universe should look the same when viewed on a large enough scale, and there should be no observable pattern anywhere. Following the cosmological principal, we expect to see a random distribution of BBH across the sky because BBH are not concentrated in one single area of the universe.

We may see the distribution of BBH according to sky location best by plotting the distribution on a Mollweide Projection Map. Because the sky is not flat, we need to project the BBH's celestial coordinates (Right Ascension and Declination) to ecliptic coordinates. Taking advantage of spherical coordinates, we may project each BBH's celestial coordinates onto a Mollweide Map of the Earth by taking the cosine of the inclination angle (declination) and using the azimuthal angle (right ascension) as is.

Luminosity Distance

To simulate the distribution of binary black holes according to luminosity distance, we may assume population of BBH lie within a certain radial distance. Given the cosmological principal and the radial distribution of BBH, we expect the distribution of BBH to increase exponentially as we observe further from Earth. This is because as the distance increases, the volume of BBH increases exponentially due to the r^3 factor in the formula for volume.

$$Equation..1 : V = \frac{4}{3}\pi r^3$$

Orbital Inclination

Binary black holes do not always directly face our detectors– they may be inclined at a certain angle. The inclination angle of the system directly effects the magnitude of the gravitational wave strain we detect. Depending on the angle, the strain may exaggerate or diminish the properties of the BBH we extract from the gravitational wave, such as mass and luminosity distance. For example, if the BBH system directly faces us, the gravitational waves we detect from the system will be strong and we may expect the BBH system to be closer or larger than it actually is. By taking into account the inclination of the system, we can predict the true properties of the BBH system. We expect the inclination angle of the system to range anywhere between 0 and π and be uniform in $cos(\iota)$.

Time and Phase of Coalescence

Binary black holes may coalesce for different periods of time and merge at any time during our observation runs. The amount of time two black holes spend coalescing is known as the phase of coalescence. In the frequency domain, the φ_c oscillates in a sine wave between 0 and 2π. Therefore, we expect BBH systems to have random φ_c between 0 and 2π.

As for time, BBH may occur at any given time in the day during an observation run. However, our detectors are not on all day. The detector in Livingston is only on 61.5% of the time and the detector in Hanford is on about 56% of the time. Therefore, we expect to detect BBH at any point in time during which our detectors are both simultaneously on.

2.2 Parameters Describing The Black Holes Within the Binary

Parameter	Symbol	BBH Distribution
Total Mass	M	Exponential
Symmetric Mass Ratio	η (m_1, m_2)	Exponential
Spin Magnitude	a_1, a_2	Gaussian
Spin Azimuthal Angle	ϕ_{a1}, ϕ_{a2}	Uniform
Spin Polar Angle	μ_{a1}, μ_{a2}	Uniform

Table 2: Summary of Parameters Describing The Black Holes Within the Binary

Total Mass

To simulate the distribution of BBH masses, we used the Initial Mass Function for stars derived by Edwin Salpeter[1].

$$Equation..2 : \xi(M)\Delta M = \xi_0(M/M_\odot)^{-2.35}(\Delta/M_\odot)$$

[2] According to the Initial Mass Function, there are more low mass stars than high mass stars because low mass stars live longer and therefore contribute to most of the stellar mass in a system. Since black holes are formed from the collapse of stars, we may also apply the IMF to the distribution of BBH. The BBH our detectors are most capable of detecting range from masses of 10 to $100M_\odot$ because comparable mass BBH generate the strongest

gravitational waves. As a result, we simulated the mass distribution of BBH in the range from 10-100$M_\odot$ and expected the majority of the BBH to be between 10-30$M_\odot$.

Symmetric Mass Ratio

Each black hole within the binary has its own individual mass. Since our detectors have a better chance of detecting gravitational waves from binary systems of comparable mass, we can expect the mass ratio, q, between the two black holes to range from 1 to 10.

With this ratio q, we can use the symmetric mass ratio parameter eta, η, to relate the mass of each black hole in the binary to each other.

$$Equation..3 : \eta = \frac{q}{(1+q^2)} = \frac{(M_1 * M_2)}{(M_1 + M_2)^2}$$

We know the total mass of the binary.

$$Equation..4 : M_{total} = M_1 + M_2$$

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

$$Equation..5a : M_1 = \frac{M_{total} + \sqrt[2]{(M_{total}^2 * (1 - 4 * eta))}}{2}$$

$$Equation..5b : M_2 = \frac{M_{total} - \sqrt[2]{(M_{total}^2 * (1 - 4 * eta))}}{2}$$

Spin Magnitude

The spin of a black hole is a dimensionless parameter that describes the ratio of black hole's observed angular momentum to the maximal angular momentum predicted by physics.

We assumed the spins of BBH are distributed in a Gaussian distribution of 0.7 mean and 0.1 std.

Spin Azimuthal and Polar Angles

If the spin of each black hole within the binary points in the same direction as the total angular momentum of the system, the magnitude of the spins will not be influenced by the inclination of the black holes. However, if the black holes do not spin in the direction of the total angular momentum of the system, the black hole will spin at certain angles away from the total angular momentum of the system. These angles are known as the polar and azimuthal angles of the spin. We expect the polar angle of the spin to range from 0 to π and the azimuthal angle of the spin to range from 0 to 2π.

2.3 Using Bayesian Inference to Estimate the Probability of Our Predicted Models

After creating models to describe the 15 parameters of BBH, we used Bayesian inference to corroborate and refine our models. Bayesian analysis chooses to estimate the probability of the parameters in models rather than the Frequentest approach of estimating the probability

of the data. We chose to employ Bayesian inference here because we wanted to see how the parameters of each model related to one another and modify our models if necessary. In the end, the Bayesian inference allowed us to estimate how well our predicted parameter distributions fit our simulated data.

3 Progress

3.1 Simulations Created

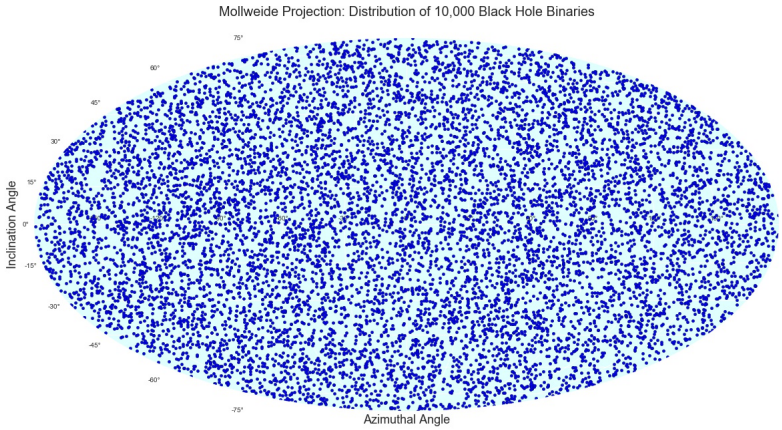

Figure 1: The distribution of Binary Black Holes in a Mollweide Projection. As expected, the BBH are randomly distributed throughout the sky.

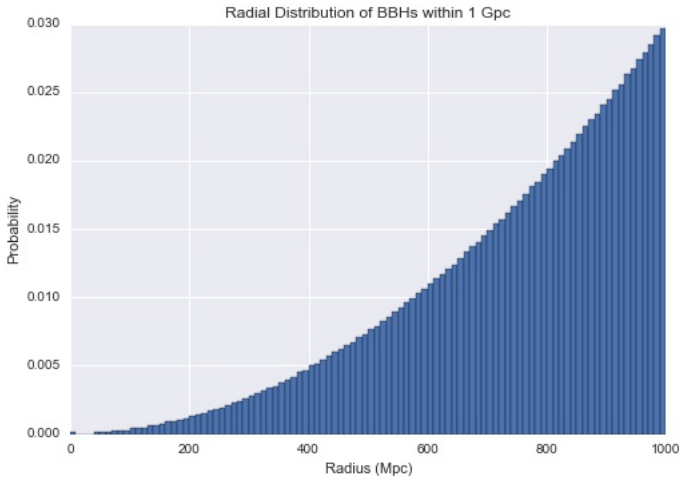

Figure 2: The radial distribution of Binary Black Holes within a luminosity distance of 1 Gpc. We expected the distribution of BBH to increase exponentially as we observe further from Earth.

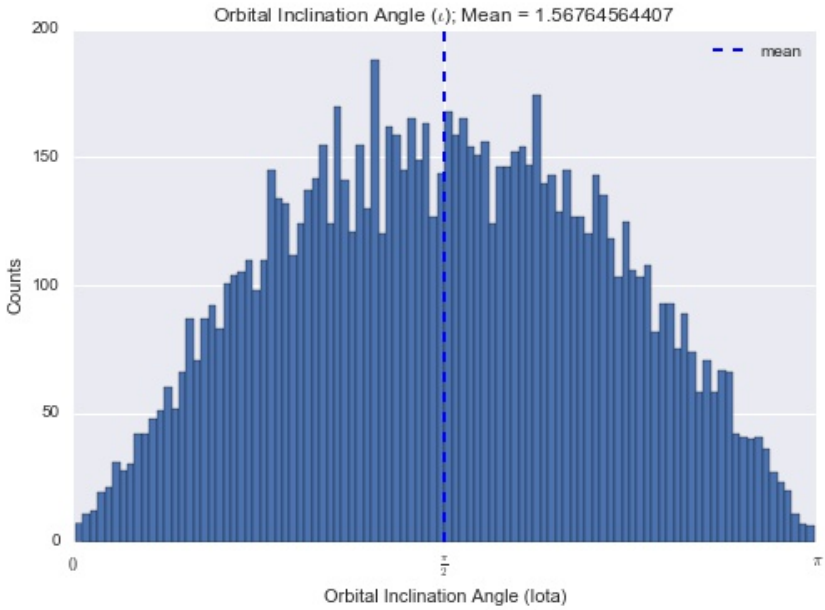

Figure 3: Orbital Inclination Angle (ι). We can see the inclination angle of the system ranges randomly between 0 and π.

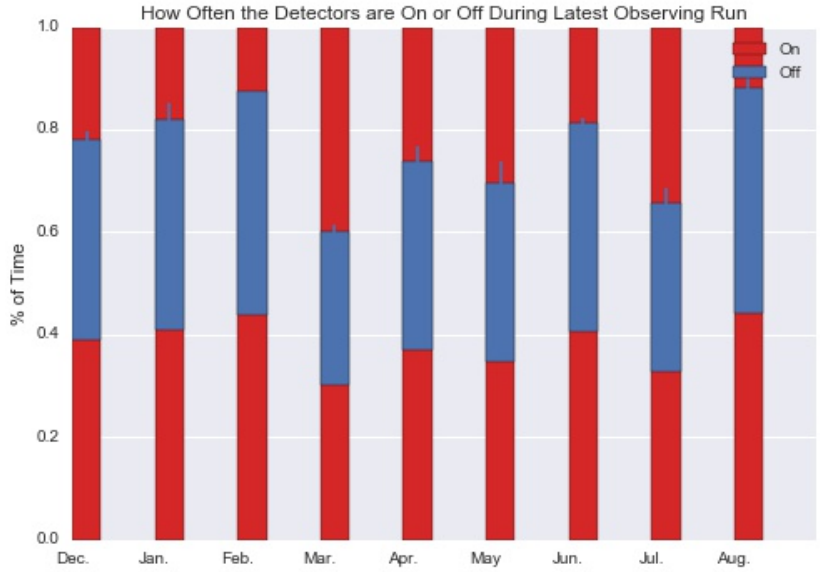

Figure 4: How Often the Detectors are On or Off During Latest Observing Run. Since LIGO Livingston is only on 61.5% of the time and LIGO Hanford is on 56% of the time, we generally assumed that the detectors were off about 40% of the time, as we can estimate in the figure.

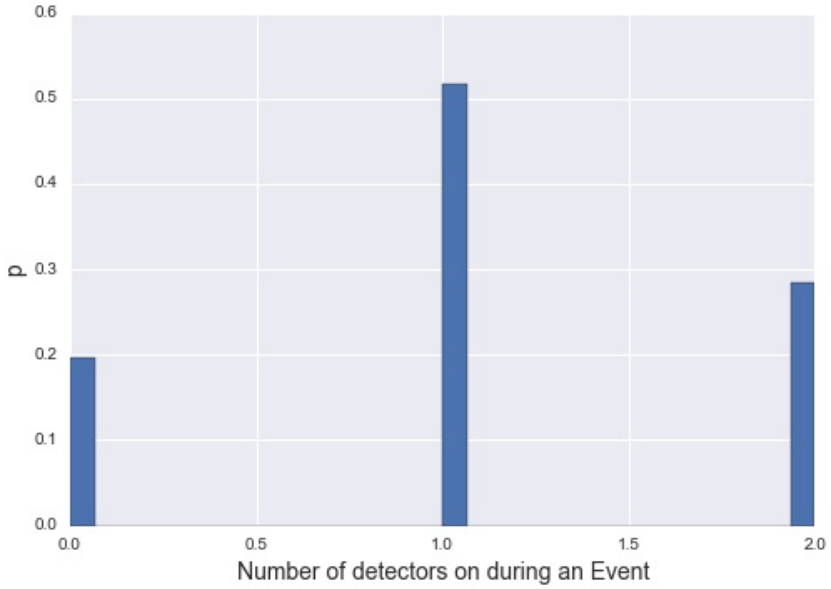

Figure 5: Number of detectors on during an Event. We expect both detectors to be on at the same time for about 30% of the observing run.

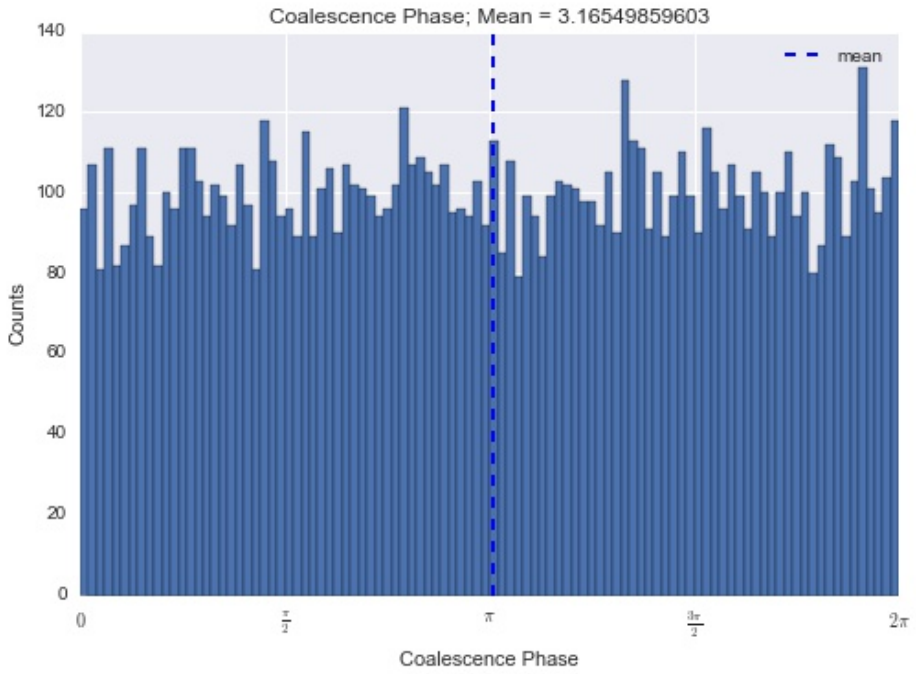

Figure 6: Phase of Coalescence. As see, we expect the distribution of phases for BBH to range from 0 to 2π.

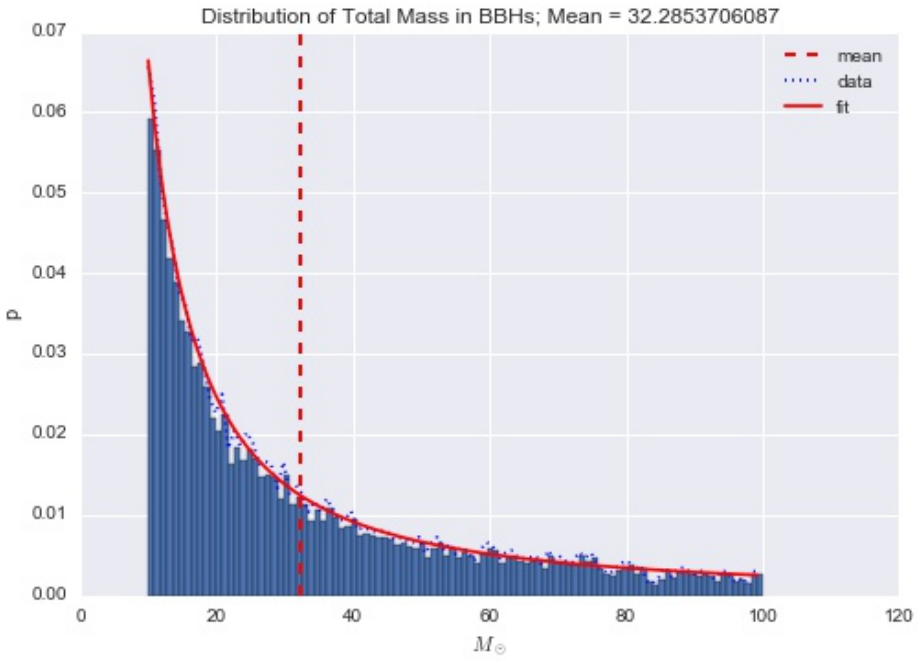

Figure 7: Distribution of Total Mass in BBHs. As seen, in the distribution of BBH in the range from 10-100$M_\odot$,the majority of the BBH are between 10-30$M_\odot$.

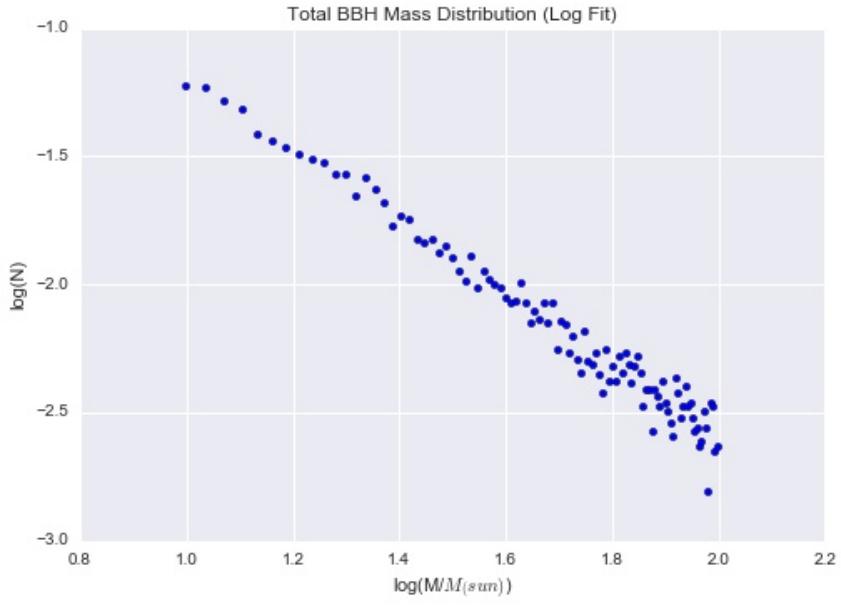

Figure 8: Log Fit of Mass Distribution. The distribution is linear, as expected from a log fit of an exponential function.

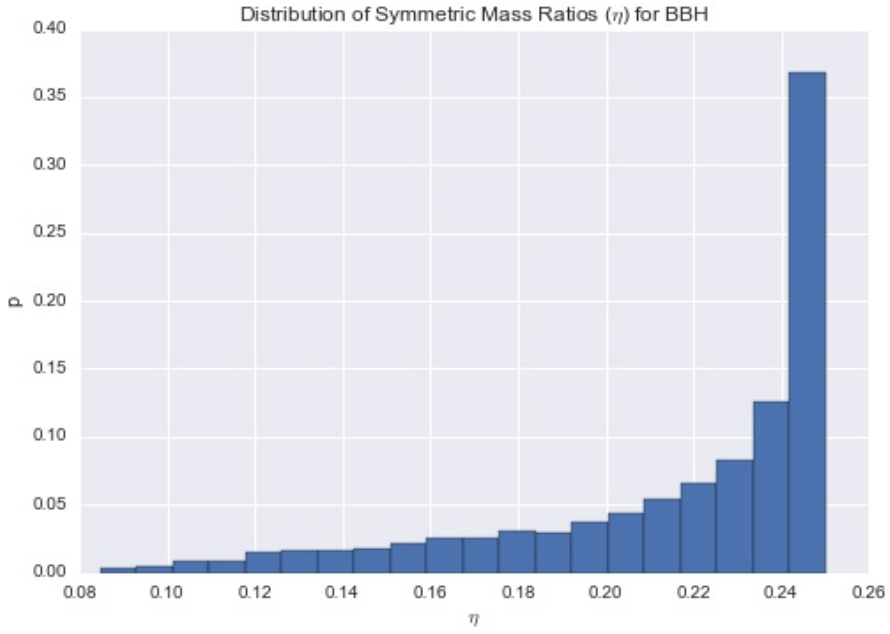

Figure 9: Distribution of Symmetric Mass Ratios (η) for BBH. When η is 0.25, q is 1. Seeing how the mass is distributed in the IMF, we should expect η to be near 0.25.

Figure 10: Distribution of BBH Individual Masses. The average value for q appears to be less than 2, which is expected due to the distribution of total mass according to the IMF.

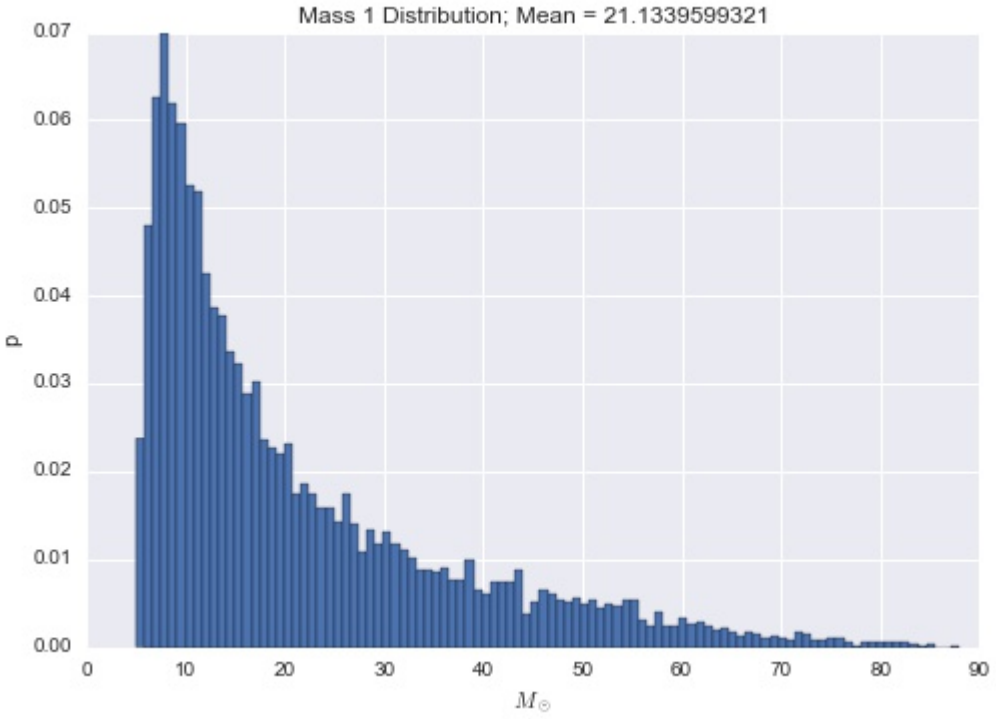

Figure 11: Mass 1 Distribution. The average mass is 21 $m_\odot$.

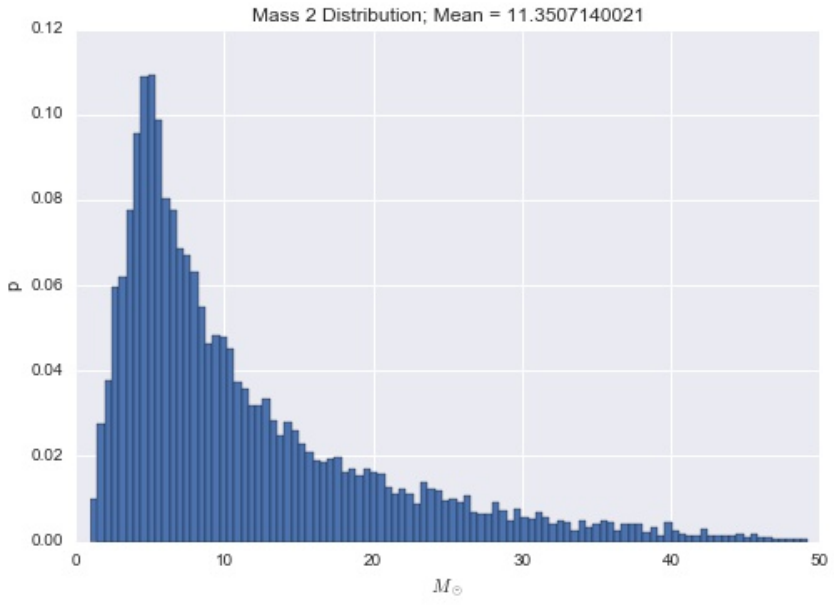

Figure 12: Mass 2 Distribution. The average mass is 11 $m_\odot$.

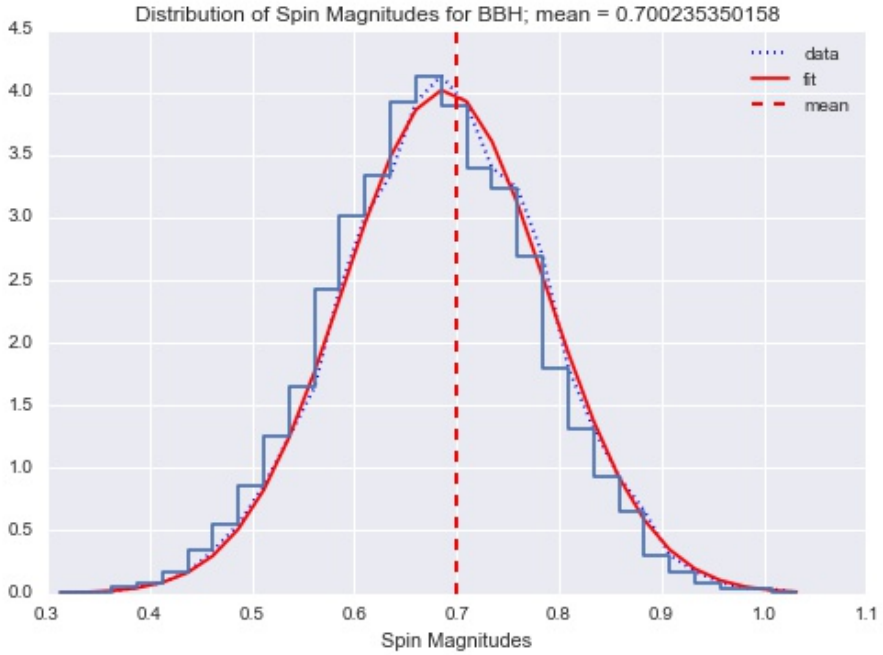

Figure 13: Distribution of Spin Magnitudes for BBH. The spins of BBH are distributed in a Gaussian distribution of 0.7 mean and 0.1 std.

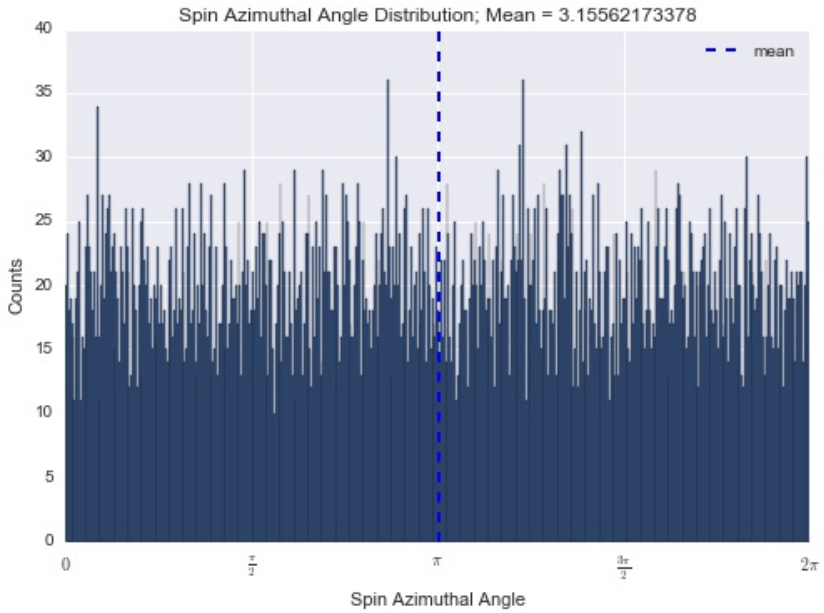

Figure 14: Spin Azimuthal Angle Distribution. We can see the azimuthal angle of the spin ranges from 0 to 2π.

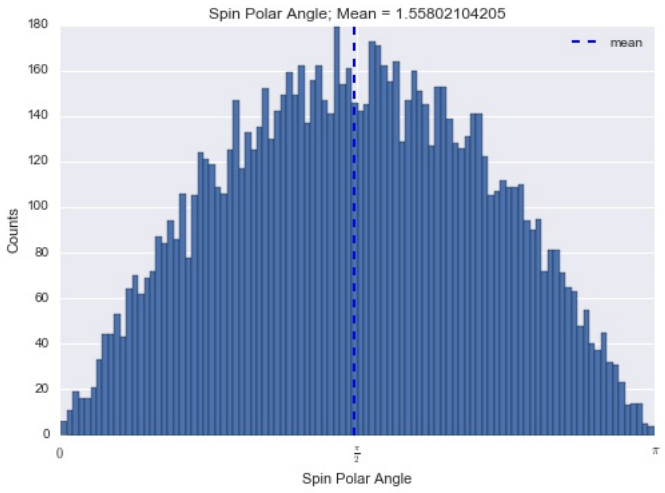

Figure 15: Spin Polar Angle Distribution. We can see the polar angle of the spin ranges from 0 to π.

3.2 Bayesian Parameter Estimation Plots Created With PyMC

We estimated the probability of the parameters of the linear function that describes the log fit of the mass distribution.

$$LinearModel : y = ax + b$$

"a" is the slope and "b" is the y-intercept. In terms of the log fit of the mass distribution, the slope should be around α-1, which is approximately -1.35, and the y-intercept should be about 0.5.

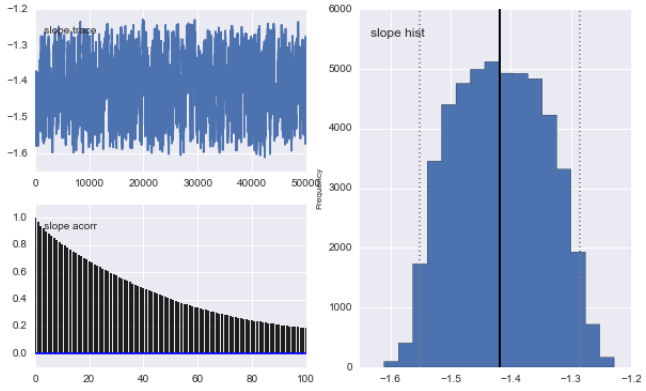

Figure 16: Slope Estimation. The average slope is -1.418 with a standard deviation of 0.075. This estimation aligns with the slope we expect of -1.35.

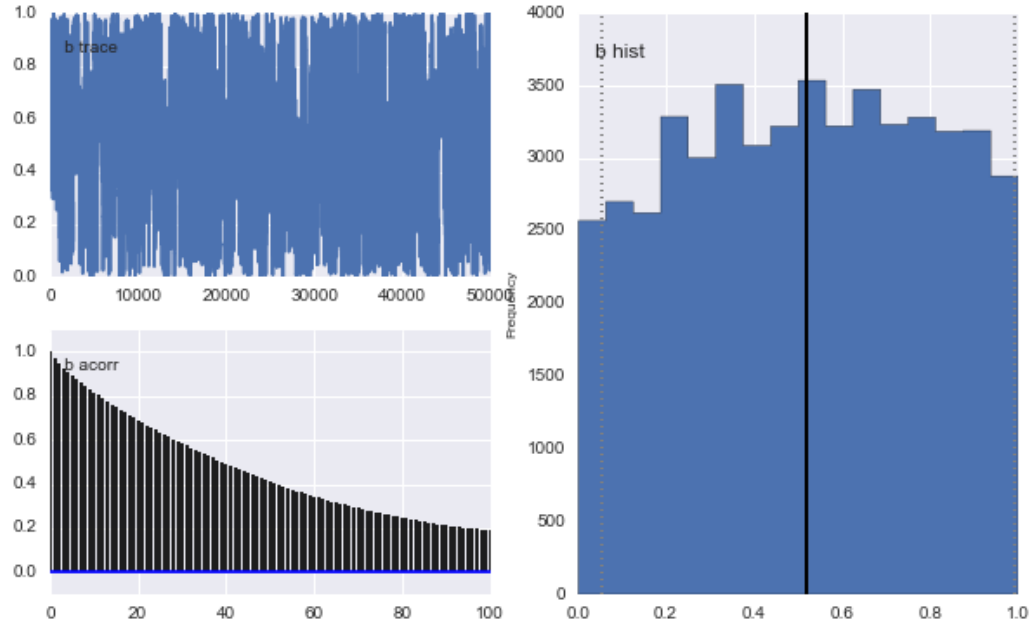

Figure 17: Y-Intercept Estimation. The average value of the y-intercept is 0.513 with a standard deviation of 0.28. This estimation aligns with the y-intercept we expect of 0.5.

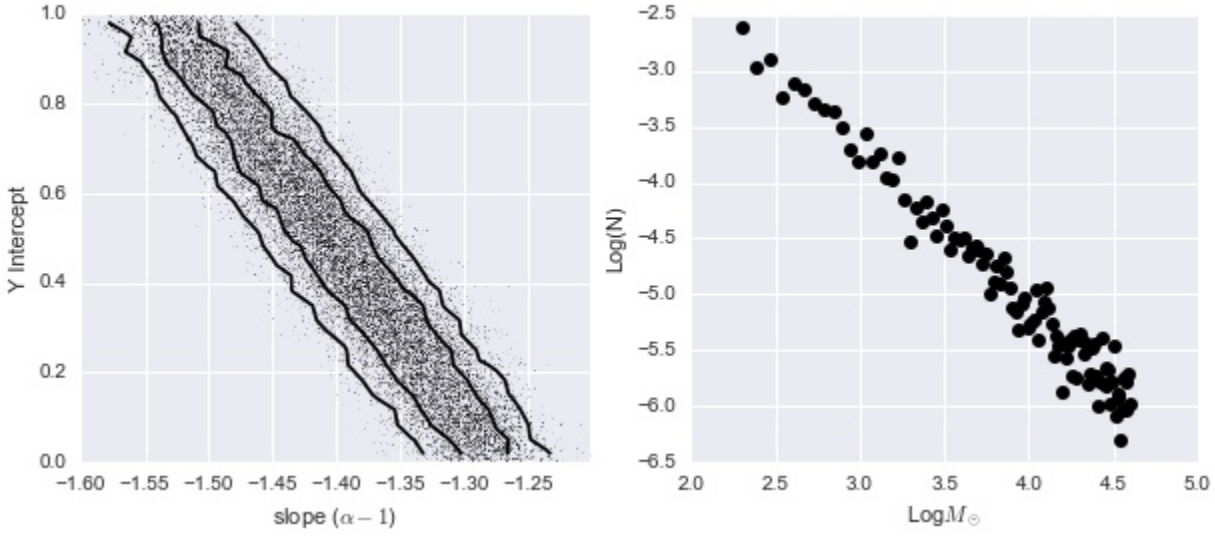

Figure 18: Another Visualization of the Linear Model Fit to the mass distribution.

We estimated the probability of the parameters of the gaussian function that describes the spin distribution.

$$GaussianModel : P(x) = a * e^{-(x-x_0)^2/2\sigma^2}$$

"a" is the amplitude, "x_0" is the mean and "σ" is the standard deviation. We expected the amplitude to be around 4, the mean to be around 0.7, and sigma to be around 0.1.

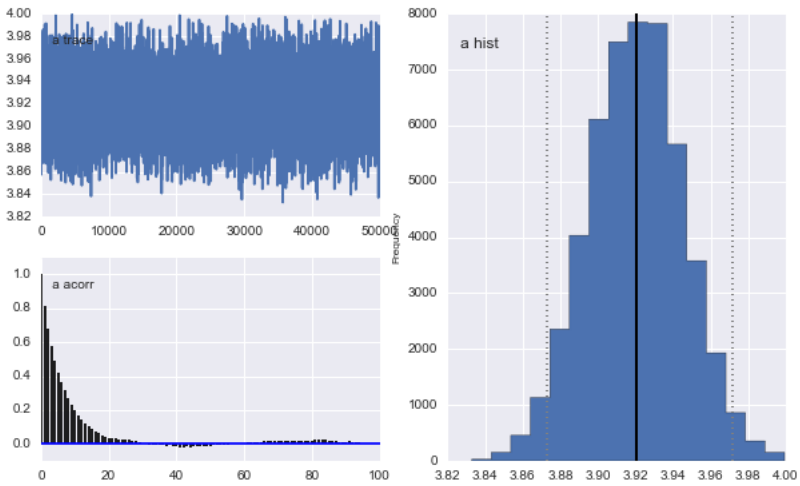

Figure 19: Amplitude Estimation. The average value of the amplitude is 3.92 with a standard deviation of 0.025. This estimation aligns with the amplitude we expect of 4.

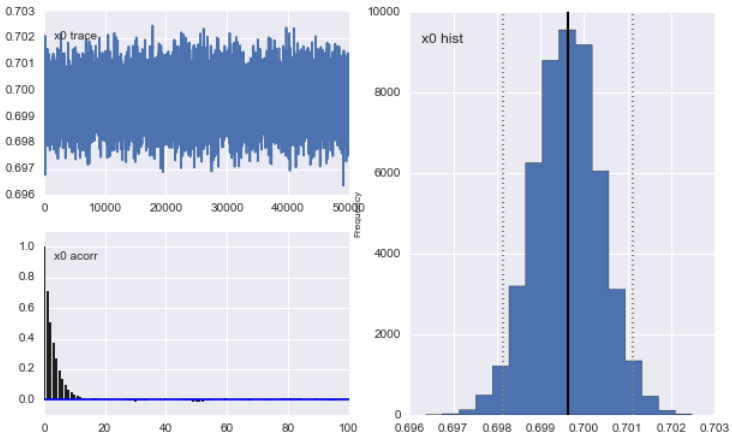

Figure 20: Mean Estimation. The average value of the mean is 0.7 with a standard deviation of 0.001. This estimation aligns with the mean we expect of 0.7.

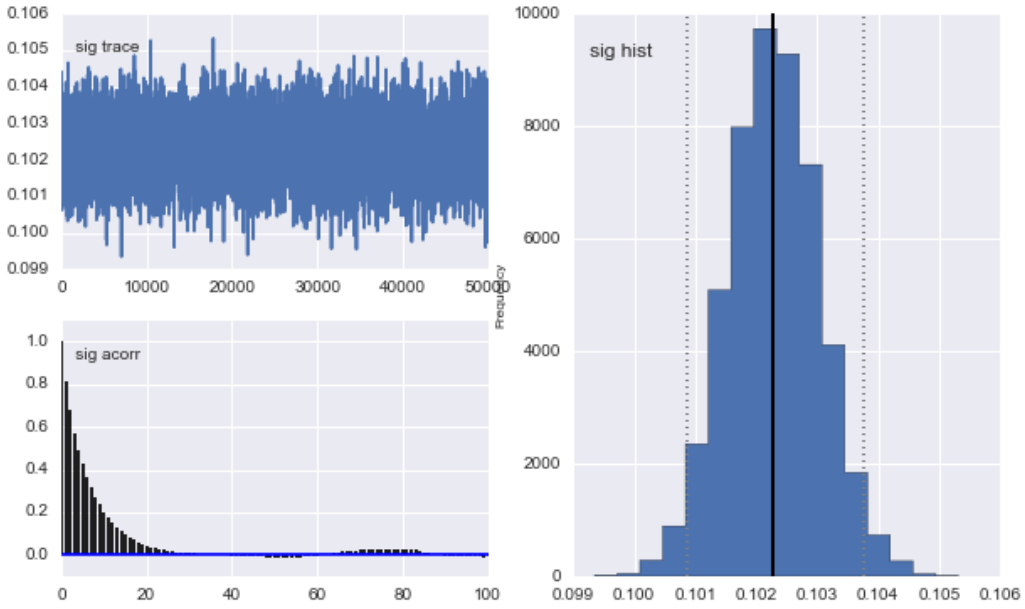

Figure 21: Standard Deviation Estimation. The average value of the standard deviation is 0.102 with a standard deviation of 0.001. This estimation aligns with the standard deviation we expect of 0.1.

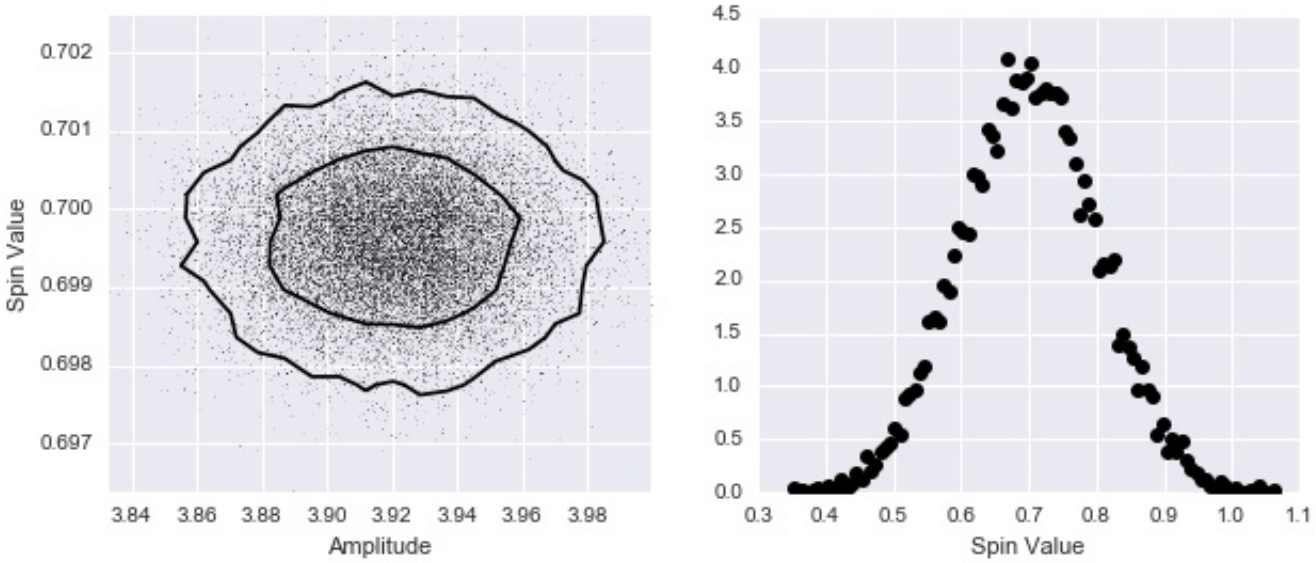

Figure 22: Another Visualization of the Gaussian Model Fit to the Spin distribution.

We estimated the probability of the parameters of the distance function we created that describes the luminosity distance's radial distribution.

$$DistanceModel : d_L = \sqrt[3]{\frac{3x}{4\pi n_G pc)}}$$

"x" is the number of events. This parameter did not need to be estimated. We assumed 10000 events. "n_{Gpc}" is the number of BBH per cubic Gpc, which we expected to be 2387 assuming 10000 BBH.

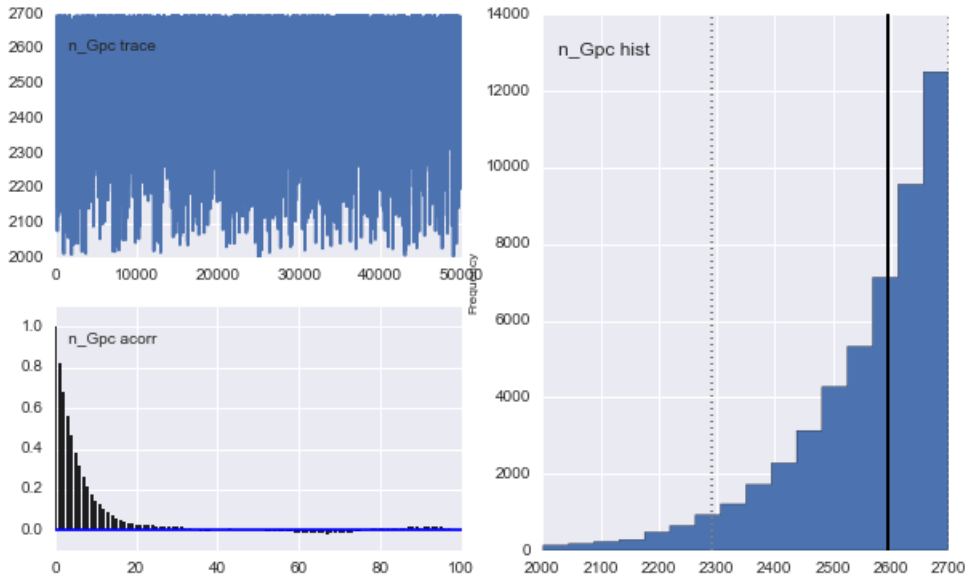

Figure 23: Number of BBH per cubic Gpc estimation. The average number density is 2557.672 with a standard deviation of 129.073. This is close to our predicted value of 2387.

4 Challenges Encountered and Moving Forward

I experienced mainly technical challenges while creating my simulations of BBH. Often times I knew what I wanted to do, but not how to do it in Python. As a result, I spent much of my days scouring Stack Overflow [3] and consulting my mentor, co-mentor and fellow SURF students to help me figure out what I wanted to do. While tedious at first, this has helped my Python coding skills greatly improve and I have become even more comfortable with the language than I was before.

Moving forward, my project will begin to focus on analyzing and modifying the mass parameter so that we may make inferences about the natural rate density of BBH in the universe. Once again, I expect the bulk of my future challenges to be of technical nature. Because I

have learned more about Python and none of my problems with code have gone unsolved so far, I actually expect there to be far less significant coding challenges in the future than there were before.

References

[1] E. E. Salpeter. The Luminosity Function and Stellar Evolution. *apj*, 121:161, January 1955.

[2] Wikipedia. Initial mass function — wikipedia, the free encyclopedia, 2017.

[3] Stack Overflow. https://stackoverflow.com/, 2017.

A Code

```
%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


Assignment 1: Plot the distribution of Black Hole binaries in a Mollweide Projection

Approach: BBH are uniform in the cosine of the inclination angle and azimuthal angle

#cosine inclination
random_inclination = []
cos_inc = np.random.uniform(-1, 1, size=1000)
inc = np.arccos(cos_inc)
inc *= np.array([57.2958])
inc -= np.array([90])
```

```
random_inclination.append(inc)

#azimuthal
random_phi = []
phi = np.random.uniform(0, 360, size= inc.size)
phi -= np.array([180.])
random_phi.append(phi)

#plot mollweide-- x: azimuthal, y: altitude
x = random_phi
y = random_inclination
fig = plt.figure(figsize=(20,10))
ax = fig.add_subplot(111, projection="mollweide", axisbg ='LightCyan')
ax.grid(True)
ax.scatter(np.radians(x),np.radians(y))
plt.title('Mollweide Projection: Distribution of 1000 Black Hole Binaries', fontsize = 2
plt.xlabel('Azimuthal Angle', fontsize = 15)
plt.ylabel('Inclination Angle', fontsize = 15)
plt.savefig("Mollweide.jpg")
plt.show()


Assignment 2: Plot distribution of black holes in binaries within 1 Gpc

Approach: Assume volume density for bbh and plot that then make the histogram to find th

#Assume BBH in volume of space of radius 1 Gpc
r = 1    #Gpc
volume = (4/3)*np.pi*(r)**3

#Assume 1000 BBH in the volume. How many BBH per meter?
nper_cubic_Gpc = 238.732                    #Wolfram Alpha to solve for n_per_cubic_Gpc
n = (volume)*(nper_cubic_Gpc)/(r**3)        #With n = 1000

#Generate random volumes
distances1 = np.linspace(0.001,1,1000)
volumes = (4/3)*np.pi*(distances1)**3

#Generate numbers of BBH by volume
n_BBH = volumes * np.array([nper_cubic_Gpc])
n_BBH = n_BBH /np.array([r**3])

#Make scatter plot

plt.plot(distances1, n_BBH, color="Red")
plt.title("Radial Distribution of BBHs within 1 Gpc")
```

```
plt.xlabel("Radius (Gpc)")
plt.ylabel("Number of BBH")

def distances(x,n):
    return np.cbrt((3*x)/(4*np.pi*n))
events = np.arange(0, 1000)
dist_distribution = distances(events,nper_cubic_Gpc)

#Plot histogram
plt.title("Radial Distribution of BBHs within 1 Gpc")
plt.xlabel("Radius (Gpc)")
plt.ylabel("Probability")
results1, edges1 = np.histogram(dist_distribution, bins=100, normed=True)
binWidth1 = edges1[1] - edges1[0]
plt.bar(edges1[:-1], results1*binWidth1, binWidth1)
plt.savefig("Radial.jpg")
```

Assignment 3: What is the mass distribution of black hole binaries?

Approach: We can plot this using the Initial Mass Function (IMF)

```
#Plot Histogram of mass distribution
def my_mass(x,c,e):        #IMF
    return c*x**(-e)
c= 1.35
e = 2.35
number = np.random.uniform(0.160, 0.427, size = 1000)  #Wolfram Alpha for number range
p2 = my_mass(number,c,e)
results, edges = np.histogram(p2, bins=100, normed=True)
binWidth = edges[1] - edges[0]
plt.bar(edges[:-1], results*binWidth, binWidth)
plt.xlabel("Solar masses")
plt.ylabel("Probability")
plt.xlim(10,100)
plt.savefig("Mass.jpg")
```

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/Imag
    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 +

```
#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

```
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 ove

The duty cycle (time detector is on) for each detector is a gaussian distribution with m
Generate random time for BBH between 0 and 1. If the number is greater than 0.7, the det

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

```
#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)

G_trace = [G.trace('a')[:],
              G.trace('x0')[:]]

# Create some convenience routines for plotting

def compute_sigma_level(trace1, trace2, nbins=20):
```

```
    """From a set of traces, bin by number of standard deviations"""
    L, xbins, ybins = np.histogram2d(trace1, trace2, nbins)
    L[L == 0] = 1E-16
    logL = np.log(L)

    shape = L.shape
    L = L.ravel()

    # obtain the indices to sort and unsort the flattened array
    i_sort = np.argsort(L)[::-1]
    i_unsort = np.argsort(i_sort)

    L_cumsum = L[i_sort].cumsum()
    L_cumsum /= L_cumsum[-1]

    xbins = 0.5 * (xbins[1:] + xbins[:-1])
    ybins = 0.5 * (ybins[1:] + ybins[:-1])

    return xbins, ybins, L_cumsum[i_unsort].reshape(shape)


def plot_GMCMC_trace(ax, xdata, ydata, trace, scatter=False, **kwargs):
    """Plot traces and contours"""
    xbins, ybins, sigma = compute_sigma_level(trace[0], trace[1])
    ax.contour(xbins, ybins, sigma.T, levels=[0.683, 0.955], **kwargs)
    if scatter:
        ax.plot(trace[0], trace[1], ',k', alpha=0.1)
    ax.set_xlabel('Amplitude')
    ax.set_ylabel('Spin Value')


def plot_MCMC_model(ax, xdata, ydata, trace):
    """Plot the linear model and 2sigma contours"""
    ax.plot(xdata, ydata, 'ok')

    alpha, beta = trace[:2]
    xfit = np.linspace(0, 1.5, 10)
    yfit = alpha[:, None] + beta[:, None] * xfit
    mu = yfit.mean(0)
    sig = 2 * yfit.std(0)

    #ax.plot(xfit, mu, '-k')
    #ax.fill_between(xfit, mu - sig, mu + sig, color='lightgray')

    ax.set_xlabel('Spin Value')
    #ax.set_ylabel('Amplitude')
```

```
def plot_GMCMC_results(xdata, ydata, trace, colors='k'):
    """Plot both the trace and the model together"""
    fig, ax = plt.subplots(1, 2, figsize=(10, 4))
    plot_GMCMC_trace(ax[0], xdata, ydata, trace, True, colors=colors)
    plot_MCMC_model(ax[1], xdata, ydata, trace)

plot_GMCMC_results(x2, f2, G_trace)
plt.savefig("G.jpg")
plt.show()

# Create some convenience routines for plotting

def compute_sigma_level(trace1, trace2, nbins=30):
    """From a set of traces, bin by number of standard deviations"""
    L, xbins, ybins = np.histogram2d(trace1, trace2, nbins)
    L[L == 0] = 1E-16
    logL = np.log(L)

    shape = L.shape
    L = L.ravel()

    # obtain the indices to sort and unsort the flattened array
    i_sort = np.argsort(L)[::-1]
    i_unsort = np.argsort(i_sort)

    L_cumsum = L[i_sort].cumsum()
    L_cumsum /= L_cumsum[-1]

    xbins = 0.5 * (xbins[1:] + xbins[:-1])
    ybins = 0.5 * (ybins[1:] + ybins[:-1])

    return xbins, ybins, L_cumsum[i_unsort].reshape(shape)


def plot_SMCMC_trace(ax, xdata, ydata, trace, scatter=False, **kwargs):
    """Plot traces and contours"""
    xbins, ybins, sigma = compute_sigma_level(trace[0], trace[1])
    ax.contour(xbins, ybins, sigma.T, levels=[0.683, 0.955], **kwargs)
    if scatter:
        ax.plot(trace[0], trace[1], ',k', alpha=0.1)
    #ax.set_ylim([0, 5])
    ax.set_xlabel(r'slope ($\alpha -1$)')
    ax.set_ylabel('Y Intercept')
```

```
def plot_MCMC_model(ax, xdata, ydata, trace):
    """Plot the linear model and 2sigma contours"""
    ax.plot(xdata, ydata, 'ok')

    alpha, beta = trace[:2]
    xfit = np.linspace(0, 1.5, 10)
    yfit = alpha[:, None] + beta[:, None] * xfit
    mu = yfit.mean(0)
    sig = 2 * yfit.std(0)

    #ax.plot(xfit, mu, '-k')
    #ax.fill_between(xfit, mu - sig, mu + sig, color='lightgray')

    ax.set_xlabel('Log$M_\odot$')
    ax.set_ylabel('Log(N)')


def plot_SMCMC_results(xdata, ydata, trace, colors='k'):
    """Plot both the trace and the model together"""
    fig, ax = plt.subplots(1, 2, figsize=(10, 4))
    plot_SMCMC_trace(ax[0], xdata, ydata, trace, True, colors=colors)
    plot_MCMC_model(ax[1], xdata, ydata, trace)

plot_SMCMC_results(x1, f, S_trace)
plt.savefig("S.jpg")
plt.show()

Pymc Distance Distribution

#Distance distribution is an exponential function.
x3 = edges1[:-1]
f3= results1 # create data
plt.plot(x3,f3)
plt.xlabel("Distance (1Gpc)")
plt.ylabel("p")
plt.title("Distance Distribution")


#priors

n_Gpc= mc.Uniform("n_Gpc", 2000, 3000, value=2387)
sigma1= mc.Uniform("sigma1", 0, 1, value=0.5)

#model
@mc.deterministic(plot=False)
```

```
def mod_dist(x=x3,n_Gpc=n_Gpc):
    return np.cbrt((3*x)/(4*np.pi*n_Gpc))

#likelihood
y2 = mc.Normal("y2", mu=mod_dist, tau=1.0/sigma1**2, value=f3, observed=True)
#

D = mc.MCMC([y2, n_Gpc, x3])
D.sample(iter=100000, burn=50000)

mc.Matplot.plot(D)

D.summary()
```