Improving Parameter Estimation on Gravitational-Wave Signals

Alexander Mellus
Department of Physics and Astronomy, Ursinus College

Mentors: Ilya Mandel, Trevor Sidery, Alberto Vecchio
School of Physics and Astronomy, University of Birmingham

August 2012

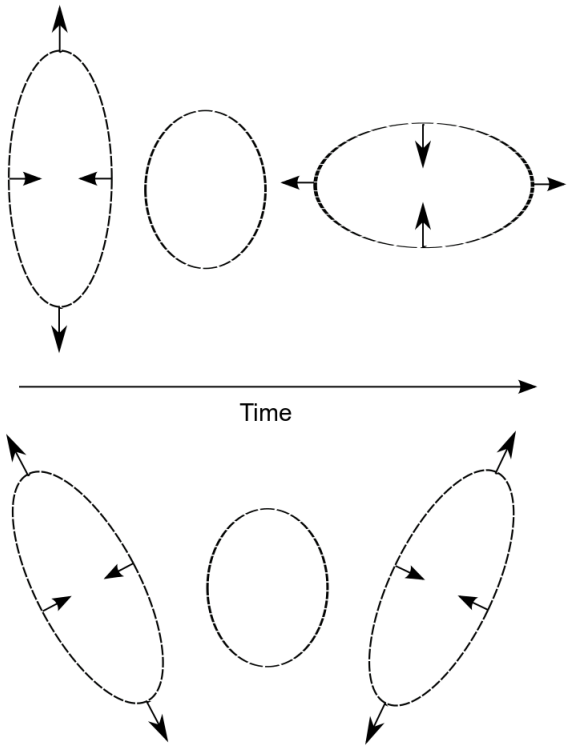


Figure 1: Effect of a gravitational wave with the passage of time (top) with plus polarization and (bottom) with cross polarization.

1 Introduction

According to Einstein's General Theory of Relativity, matter and energy cause spacetime to curve. The theory predicts that, as a result of certain astrophysical events, ripples in the curvature of spacetime are emitted. These ripples are known as gravitational waves. The waves can polarized in two ways, known as the "plus" and "cross" polarizations. As a gravitational wave passes by an object, it causes the object to stretch along one axis and compress along another, depending on the polarization of the wave, see Fig. 1. Energy, called gravitational energy, is carried away from the source by these waves.

Gravitational wave detection is done with large laser interferometers. There are several such detectors around the world and in development. Among these are LIGO, with sites in Washington State and Louisiana, VIRGO, based in Italy, GEO 600 in Germany, and TAMA 300 in Japan. A space-based detector called LISA and an underground detector called the Einstein Telescope are planned for launches in the future. These experiments should be able to detect gravitational waves from a variety of sources such as supernovae explosions from dying stars, supermassive black holes at the center of galaxies, and binary systems such as neutron star/neutron star(NS/NS) binaries, black hole/neutron star (BH/NS) binaries and black hole/black hole(BH/BH) binaries, among others.

2 Parameter Estimation

After a gravitational wave is detected, we will want to extract the astrophysical information contained in the signal. This process is known as parameter estimation. There are, in general, fifteen parameters associated with a gravitational wave signal from a compact binary system: the masses of the two objects in the system, two three-dimensional vectors that characterize each object's spin, the luminosity distance, the time and phase of coalescence, two parameters for sky position, and two angles that define the total angular momentum of the system [1]. There is a variety of information to be learned by accurately estimating the values of these parameters. For example, knowing the masses of the components of the binary would allow us to infer what type of objects the system is composed of. Locating the source precisely in the sky would allow us to follow up the gravitational wave detection with electromagnetic telescopes and learn more than we could have with just the gravitational wave signal.

2.1 Bayesian Analysis

The data collected by gravitational wave detectors can be expressed in the following way:

$$d = n + h(\vec{\theta}), \tag{1}$$

where n is the noise in the detector and h, which depends on parameters $\vec{\theta}$ and may or may not be present, is the gravitational wave signal. Bayesian analysis is used for parameter estimation because, due to the high dimensionality of the parameter vector, a grid based approach is impossible. In a grid based approach, the number of points necessary to calculate grows as

$$N = n^d \tag{2}$$

where n is the number of points in one dimension of the grid and d is the number of dimensions. If, for example, we wanted to calculate ten points in each dimension, which is already far too small for any practical problem, for fifteen dimensions, we would need to evaluate 10^{15} points. Therefore, we must use more efficient algorithms. Bayesian techniques rely on Bayes' theorem

$$p(\vec{\theta}|d) = \frac{p(d|\vec{\theta})p(\vec{\theta})}{p(d)}. \tag{3}$$

The quantity $p(\vec{\theta}|d)$ is called the posterior probability density function (pdf) and gives the probability of a set of parameters given the data. The likelihood of the data given the parameters is given by $p(d|\vec{\theta})$. The prior probability, $p(\vec{\theta})$, reflects all prior knowledge about the system before a measurement is taken. The quantity in the denominator is called the evidence and is a normalizing factor. It may or may not be important depending on whether the interest is in relative or absolute posterior probabilities.

2.2 Markov Chain Monte Carlo

There are several algorithms used for Bayesian inference of gravitational wave parameters. One of the most common is called Markov Chain Monte Carlo (MCMC). The algorithm

works by choosing a starting point randomly from the prior distribution. The likelihood is calculated at that point using the equation

$$L(\vec{\theta}) = exp(-\frac{< d - h(\vec{\theta})|d - h(\vec{\theta}) >}{2}) \tag{4}$$

where d is the data, h is the value of the waveform evaluated at the point $\vec{\theta}$ and the inner product of two vectors a and b is given by

$$< a|b >= 4\Re \int_0^\infty \frac{a(f)b^*(f)}{S_n(|f|)} df. \tag{5}$$

S_n is the one-sided noise power spectral density, which depends on the details of the detector that took the data.

The value of the point is then added to a Markov chain, a new point is proposed based on a chosen jump proposal function, and the likelihood for the proposed point is calculated. If it is greater than the previous point's likelihood, the point is added to the Markov chain. If the likelihood is less than the previous point's likelihood, a number between 0 and 1 is picked randomly. If the ratio of the proposed likelihood to the previous likelihood is greater than the random number, the point is still added to the Markov chain. Otherwise, the proposed point is discarded, and the previous point is added to the chain.

This process is repeated until a termination condition is met. A histogram of the resulting Markov chain yields the desired posterior probability density function. Marginalizing to one dimension allows us to plot the pdf for each parameter individually, as shown in Fig. 2. This algorithm does not require that the evidence be calculated because the relative density of points in the chain yields the pdf, and no normalizing is needed.

2.3 Nested Sampling

Nested Sampling [2] is another parameter estimation technique used frequently for gravitational wave research. In contrast to Markov Chain Monte Carlo, Nested Sampling is primarily interested in calculating the evidence, though the posterior density function can be recovered from this algorithm. The evidence is given by the integral

$$Z = \int L(\theta)\pi(\theta)d\theta \tag{6}$$

where $L(\theta)$ is the likelihood and $\pi(\theta)$ is the prior. The algorithm works by generating N live points from the prior. The typical number of live points used in our runs was $N = 1000$. For each iteration i, the likelihood is calculated for each of the points, and the point with the lowest likelihood L_i is removed and replaced with a point of higher likelihood. The evidence value, Z, is then incremented by an amount $L_i w_i$, where $w_i = X_{i-1} - X_i$ and $X_i = exp(-i/N)$. To ensure the new live point is uncorrelated with the previous one, a short Markov Chain Monte Carlo is used to pick the new point. Through a process of reweighting, the live points can be used to produce the posterior density function.

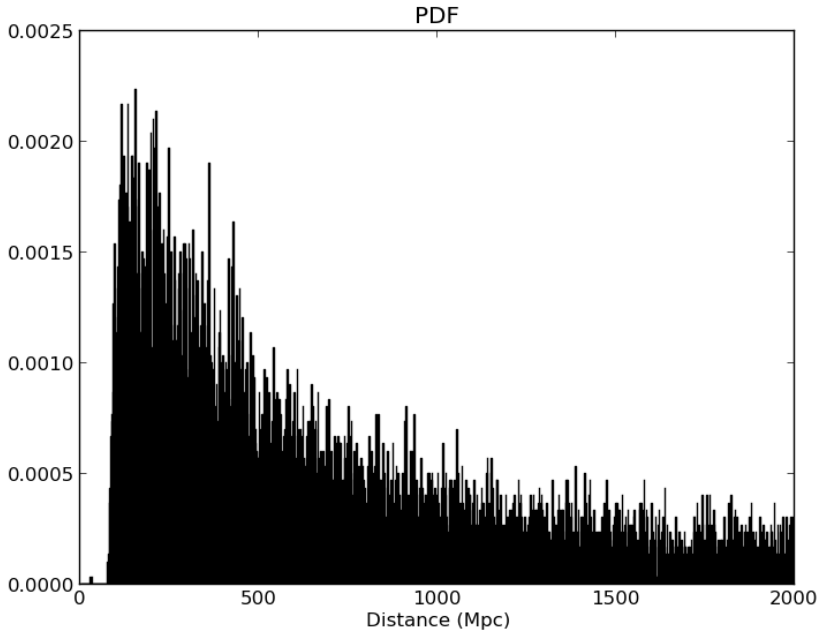


Figure 2: Example posterior density function from MCMC marginalized for luminosity distance. The injected value for the luminosity distance was 200Mpc.

3 Convergence Tests

When using Markov Chain Monte Carlo, Nested Sampling, or any other stochastic technique, it is important to carefully consider whether a given run has converged and should be trusted. In general, if the algorithm were allowed to run for an infinite number of iterations, it would reach the target distribution. For obvious reasons, this is impossible, so we must construct methods of cutting runs short, but still being confident the correct distribution has been reached. One way of doing this, utilized by MCMC, is to execute several runs with different starting points in parallel. If the separate runs have reached the same distribution, they have likely converged. In Nested Sampling, the algorithm is run until the value $L_i w_i$ produces a negligible change in Z. There are other possible conditions that could be used, but none ensure that the target distribution has been found.

The goal of my project was to make a variety of information about the convergence of runs available in one place, so that a user could quickly and easily decide whether to trust the runs. I worked specifically with Nested Sampling runs. The Nested Sampling code, available in the LIGO Scientific Collaboration Algorithm Library (LAL), creates several independent runs, which are eventually merged by post processing code. The live points are then reweighted, and the resulting posterior probability density functions are output to a web page along with other information about the runs.

Merging the separate Nested Sampling runs is not valid if one of the runs has not reached the target distribution, and there is no guarantee that this has happened for any run. To

account for this, my project focused on comparing the independent runs before they have been merged. If they do not appear to be drawn from the same distribution, at least one run has not converged.

My code works in conjunction with the post processing code. It performs a number of comparisons and statistical checks on the multiple runs and outputs the results to a web page dedicated to convergence information. In addition, it writes out a warning file if certain tests were not passed, as defined by the user. This is particularly useful for users who have a large number of events to analyze. It is impractical to look at every page individually, so the existence of the warning file associated with a particular event would indicate that the event may be suspicious and should be analyzed further.

I implemented several tests in the interest of providing a more comprehensive overview of the convergence of the Nested Sampling runs. The first was a simple comparison of certain values of the runs. At the top of the web page, I output a table containing the maximum log likelihood, the value of the log evidence, and the maximum log posterior for each run. For well converged runs, these values should closely agree. I also performed two statistical tests to compare the distributions found by the individual runs, the Kolmogorov-Smirnov test and the Gelman Rubin test.

3.1 Kolmogorov-Smirnov Test

The Kolmogorov-Smirnov test (KS test) evaluates whether two data sets are drawn from the same distribution. It works by making a cumulative fraction plot, shown in Fig. 3, for each set. The maximum difference, the D statistic, is then calculated, Fig. 4, between the two plots. Assuming the hypothesis that the two data sets are drawn from the same distribution, the probability of obtaining a D statistic value as extreme as the calculated value, called the p-value, is then determined. If the p-value is greater than a chosen value, say 0.05, then the null hypothesis, the hypothesis that assumes the sets are drawn from the same distribution, cannot be rejected. This does not allow us to claim they were drawn from the same distribution. Passing the KS test is necessary but not sufficient to make this conclusion.

In my code, I implemented the KS test individually for each parameter for all pairings of separate Nested Sampling runs. I created two different visual representations of this on the web page. The first is a histogram plot of the p-values of each parameter. The second is a table for each parameter that shows the p-value calculated from the KS test for each pairing of runs. The idea behind including the two visualizations is that a quick glance at the plots would give the user a general feeling for how the runs performed in the KS test, and the tables would give more detailed information about specific pairings of runs. Examples of the plots and tables can be found in the Appendix. As can be seen in the tables in the KS Test section of the page, the p-values along the diagonal are always exactly equal to 1.0 because performing a KS test on a data set with itself will always yield a perfect fit. The tables

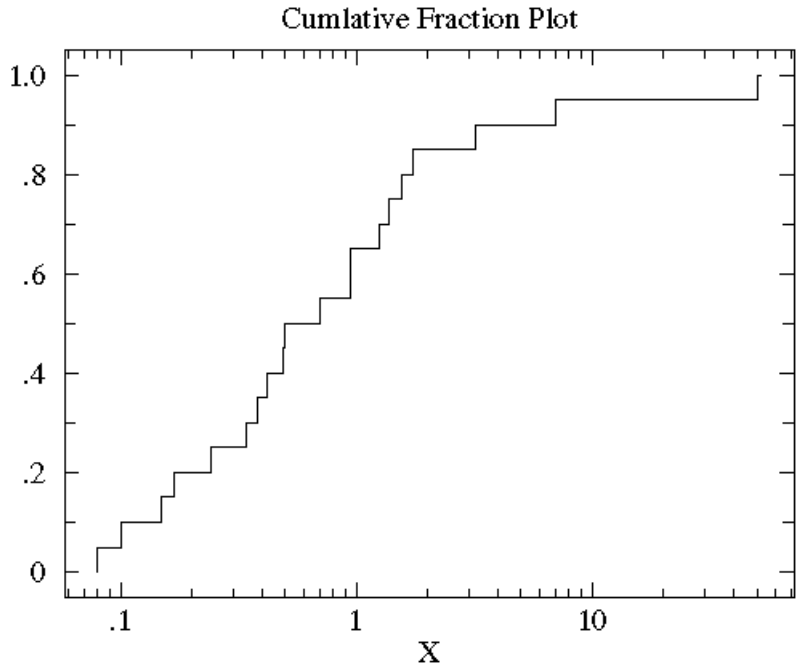


Figure 3: An example of a cumulative fraction plot. The value of y at a given point is the percentage of the data set less than the x-value at that point. (source: http://www.physics.csbsju.edu/stats/KS-test.html)

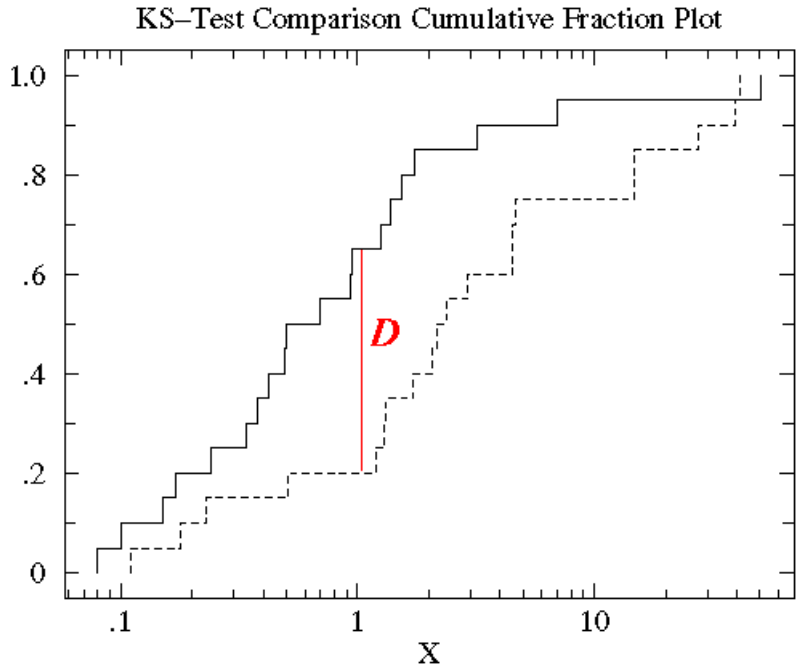
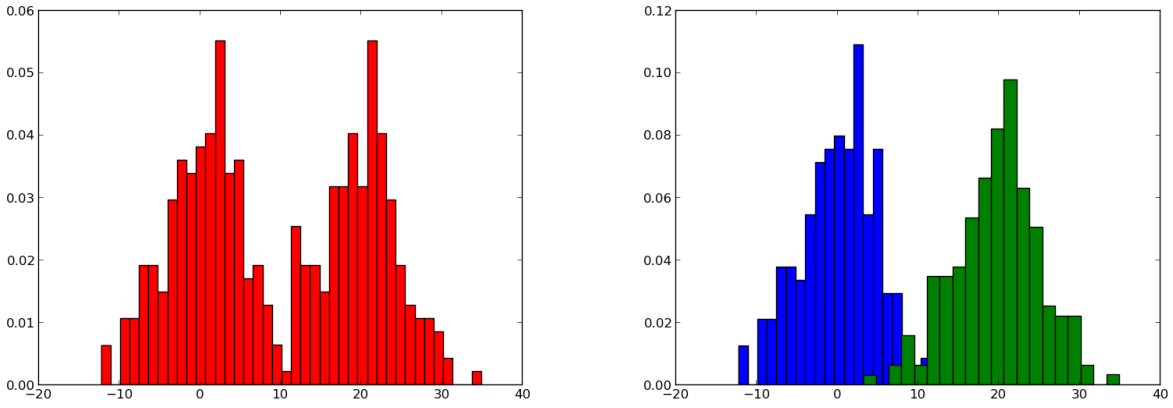


Figure 4: Cumulative fraction plots for two data sets with the maximum difference D shown in red. (source: http://www.physics.csbsju.edu/stats/KS-test.html)



(a) Actual distribution

(b) Distributions found by two different runs

Figure 5: A bimodal distribution and the result after two runs each find only one mode.

are also symmetric because the result of a KS test between run i and run j is the same as between run j and run i.

3.2 Gelman Rubin Test

The Gelman Rubin R statistic was first described by Andrew Gelman and Donald Rubin in 1992 [3]. The basic idea is to compare the variance of each individual data set with the variance of the merged data sets. The R statistic, also called the potential scale reduction, is given by

$$\sqrt{\hat{R}} = \sqrt{(\frac{n-1}{n} + \frac{m+1}{mn}\frac{B}{W})\frac{df}{df-2}}. \tag{7}$$

The quantities m and n are the number of data sets and the number of elements in each data set respectively. B is the variance between the means of the data sets, and W is the variance within a given set, averaged over all m sets. The number of degrees of freedom is given by df.

The usefulness of this test is easily seen when considering a simple example. Say two runs are done on a distribution with two modes, Fig.5a. If each run finds only one of the modes, shown in Fig. 5b, clearly neither can be said to have converged. Analyzing the results of a Gelman Rubin test in this situation will make it abundantly clear that the runs should not be trusted. As with the KS test, passing the Gelman Rubin test is necessary but not sufficient for claiming that runs have converged. For instance, if the two runs in the bimodal example each found the same mode, it would pass the Gelman Rubin test despite the fact that an entire mode has been missed.

In my code, I implement the Gelman Rubin test for each parameter and output the R value to the convergence web page. An R value of 1 would indicate that additional iterations

would not improve the agreement between runs. An optional argument in the code allows the user to set a threshold for the R value. If any parameter returned an R value greater than the given threshold value, a warning file would be output containing the name of the parameter and R value. The default value for the threshold, if no value is given, is 1.01. In addition, for all parameters, I output a scatter plot of the value of the parameter versus the sample number. Each run is plotted in a different color on the same plot. The reason for creating these plots is that if a particular run was not well converged, it may be easily seen on the plot that it had not completely sampled the prior volume.

4 Conclusion

Being confident that runs have converged is very important in parameter estimation of gravitational waves. While some checks have been in place for some time, implementing furthur tests can only be beneficial. Additionally, having a variety of information about convergence all in one place makes it much easier to analyze large quantities of data. In the future, it would be helpful to investigate other statistical tests that may applicable to checking the convergence of gravitational wave parameter estimation code. It may also be worthwhile to more carefully consider the termination conditions of the simulations themselves to determine whether more robust solutions than the ones currently implemented exist.

References

[1] M. van der Sluys, I. Mandel, V. Raymond, V. Kalogera, C. Röver, and N. Christensen *Class. Quantum Grav.*, 26(204010), 2009.

[2] J. Skilling. *Bayesian Analysis*, 1(4), 2006.

[3] A. Gelman, and D. B. Rubin. *Statistical Science*, Vol. 7, No. 457-511, 1992.

5 Appendix

Shown below is an example of a webpage output from my code, cbcBayesConvergence.py. The script will be available in the pylal section of LAL after it has been merged successfully with the existing version.

Convergence Information

Summary

Max difference in loglikelihood: 0.910419

Run	maxloglikelihood
0	-202914.115798
1	-202914.200738
2	-202914.001669
3	-202914.201863
4	-202913.291444

KS Test

phi

	Run0	Run1	Run2	Run3	Run4
Run0	1.000000	0.292218	0.434546	0.523679	0.967845
Run1	0.292218	1.000000	0.067141	0.192643	0.207739
Run2	0.434546	0.067141	1.000000	0.515962	0.416565
Run3	0.523679	0.192643	0.515962	1.000000	0.446401
Run4	0.967845	0.207739	0.416565	0.446401	1.000000

psi

	Run0	Run1	Run2	Run3	Run4
Run0	1.000000	0.628057	0.618003	0.481243	0.997455
Run1	0.628057	1.000000	0.554275	0.686312	0.502825
Run2	0.618003	0.554275	1.000000	0.936718	0.491920
Run3	0.481243	0.686312	0.936718	1.000000	0.332749
Run4	0.997455	0.502825	0.491920	0.332749	1.000000

dist

	Run0	Run1	Run2	Run3	Run4
Run0	1.000000	0.523131	0.003801	0.645922	0.105299
Run1	0.523131	1.000000	0.042828	0.957236	0.332944
Run2	0.003801	0.042828	1.000000	0.069782	0.511985
Run3	0.645922	0.957236	0.069782	1.000000	0.407524
Run4	0.105299	0.332944	0.511985	0.407524	1.000000

m2

	Run0	Run1	Run2	Run3	Run4
Run0	1.000000	0.002573	0.004832	0.400622	0.101712

Run1	0.002573	1.000000	0.467916	0.091386	0.053107
Run2	0.004832	0.467916	1.000000	0.104992	0.019980
Run3	0.400622	0.091386	0.104992	1.000000	0.532769
Run4	0.101712	0.053107	0.019980	0.532769	1.000000

m1

	Run0	Run1	Run2	Run3	Run4
Run0	1.000000	0.001748	0.005616	0.470322	0.062841
Run1	0.001748	1.000000	0.620769	0.078391	0.030975
Run2	0.005616	0.620769	1.000000	0.079926	0.006978
Run3	0.470322	0.078391	0.079926	1.000000	0.491475
Run4	0.062841	0.030975	0.006978	0.491475	1.000000

ra

	Run0	Run1	Run2	Run3	Run4
Run0	1.000000	0.201508	0.743754	0.764783	0.230777
Run1	0.201508	1.000000	0.536610	0.558272	0.560259
Run2	0.743754	0.536610	1.000000	0.990747	0.631620
Run3	0.764783	0.558272	0.990747	1.000000	0.346973
Run4	0.230777	0.560259	0.631620	0.346973	1.000000

time

	Run0	Run1	Run2	Run3	Run4
Run0	1.000000	0.046553	0.614730	0.156160	0.999998
Run1	0.046553	1.000000	0.578169	0.051761	0.122214
Run2	0.614730	0.578169	1.000000	0.432635	0.888380
Run3	0.156160	0.051761	0.432635	1.000000	0.216273
Run4	0.999998	0.122214	0.888380	0.216273	1.000000

logl

	Run0	Run1	Run2	Run3	Run4
Run0	1.000000	0.820067	0.005427	0.000126	0.000000
Run1	0.820067	1.000000	0.035500	0.000218	0.000001
Run2	0.005427	0.035500	1.000000	0.000692	0.000004
Run3	0.000126	0.000218	0.000692	1.000000	0.327041
Run4	0.000000	0.000001	0.000004	0.327041	1.000000

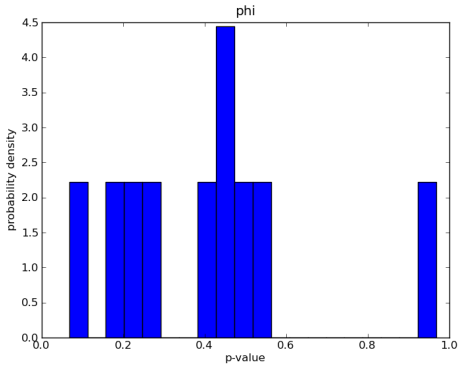
dec

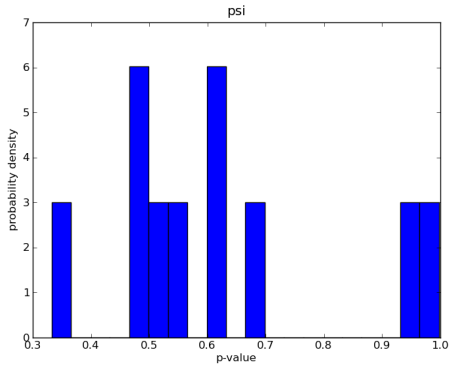
	Run0	Run1	Run2	Run3	Run4
Run0	1.000000	0.487430	0.061363	0.224131	0.903475

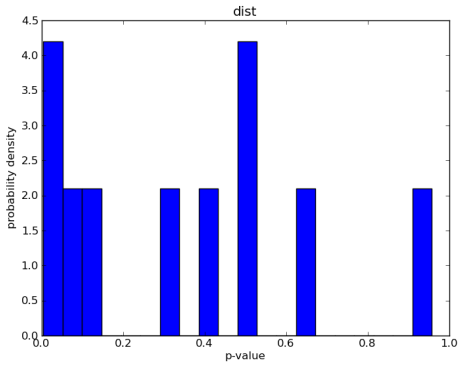
Run1	0.487430	1.000000	0.017553	0.161292	0.668317
Run2	0.061363	0.017553	1.000000	0.897464	0.061615
Run3	0.224131	0.161292	0.897464	1.000000	0.206087
Run4	0.903475	0.668317	0.061615	0.206087	1.000000

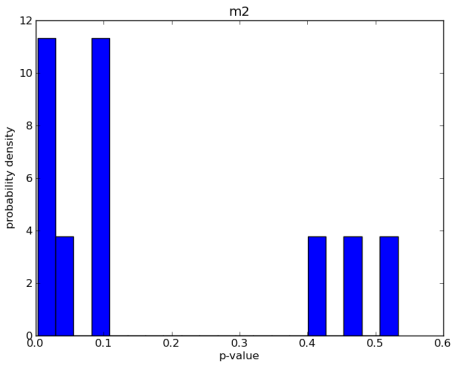
iota

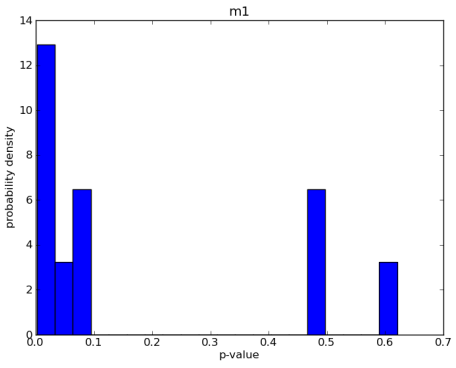
	Run0	Run1	Run2	Run3	Run4
Run0	1.000000	0.486184	0.009256	0.342233	0.055983
Run1	0.486184	1.000000	0.212610	0.991111	0.559128
Run2	0.009256	0.212610	1.000000	0.225952	0.733580
Run3	0.342233	0.991111	0.225952	1.000000	0.661550
Run4	0.055983	0.559128	0.733580	0.661550	1.000000

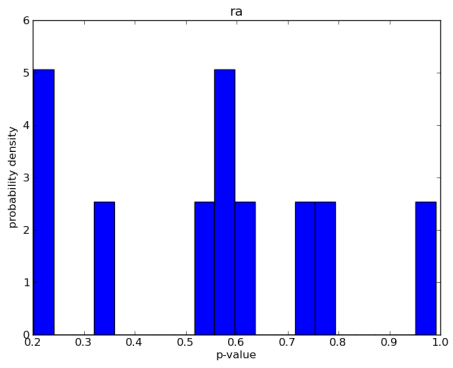


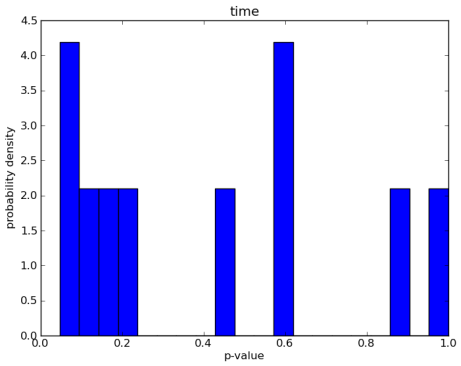


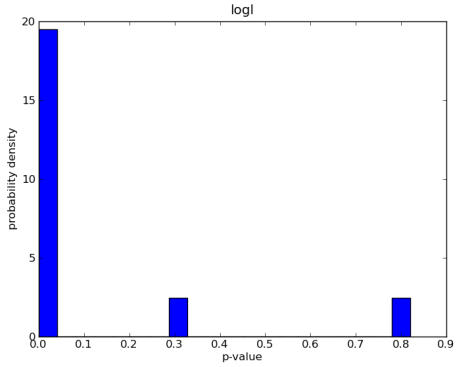


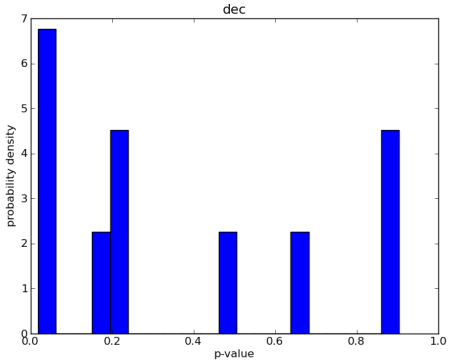


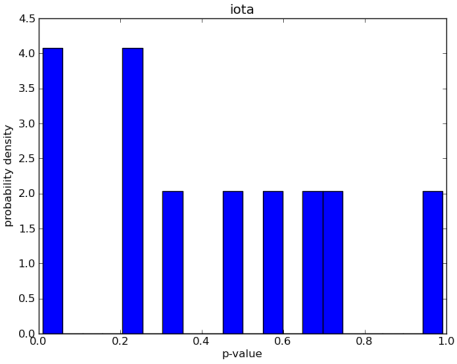




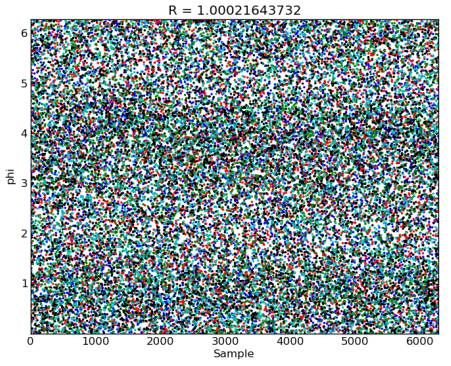


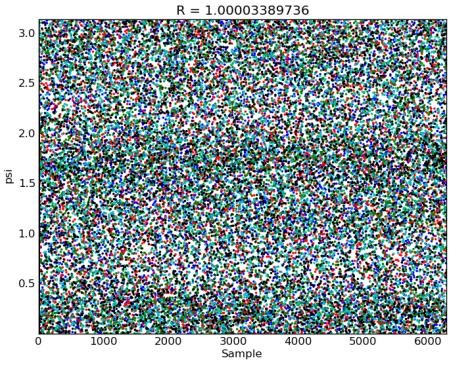


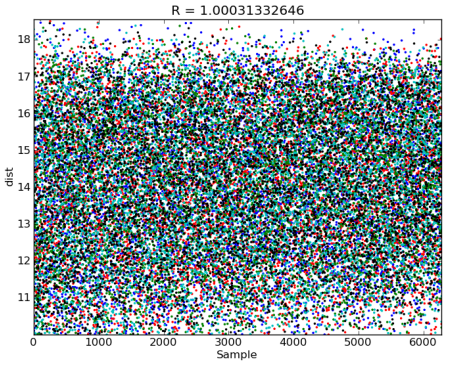


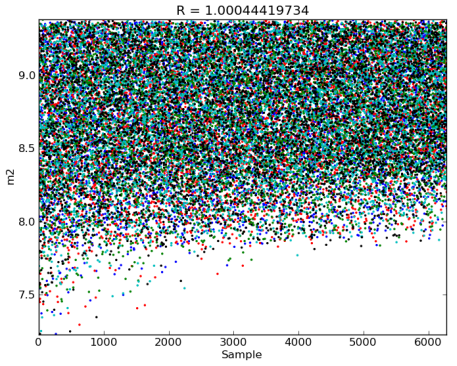


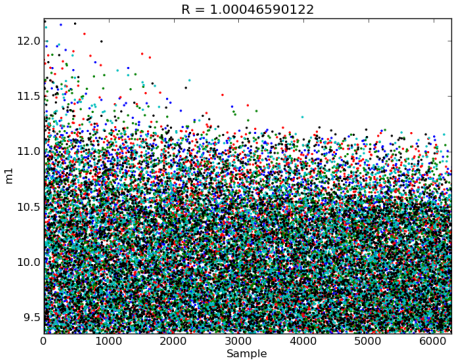
Gelman Rubin

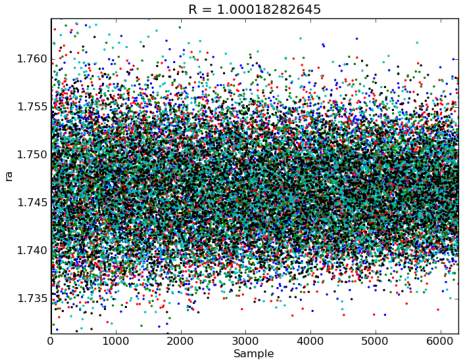


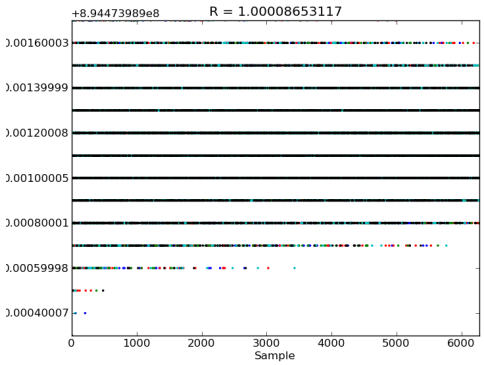


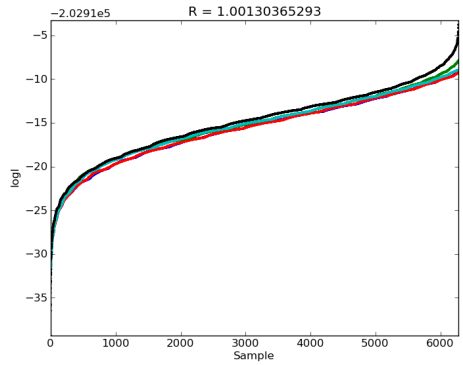


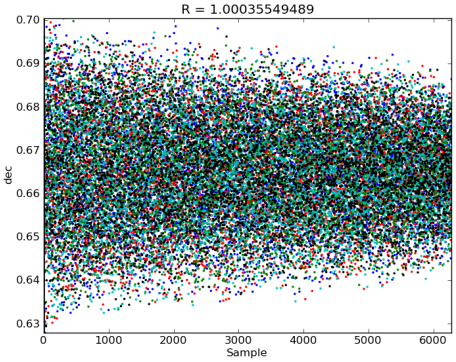


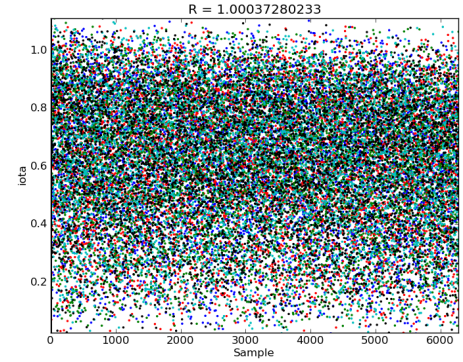

R = 1.00046590122
m1
Sample


R = 1.00018282645
ra
Sample


+8.94473989e8
R = 1.00008653117
Sample


−2.0291e5
R = 1.00130365293
logl
Sample


R = 1.00035549489
dec
Sample


R = 1.00037280233
iota
Sample