Tuto_2.5_Parameter_estimation_for_compact_object_mergers

November 21, 2020

1 Gravitational Wave Open Data Workshop #3

Tutorial 2.5: Parameter estimation for compact object mergers – Using and interpreting posterior samples This is a simple demonstration to loading and viewing data released in associaton with the publication titled **GWTC-1: A Gravitational-Wave Transient Catalog of Compact Binary Mergers Observed by LIGO and Virgo during the First and Second Observing Runs** avaliable through DCC and arXiv. This should lead to discussion and interpretation.

The data used in these tutorials will be downloaded from the public DCC page LIGO-P1800370.

Click this link to view this tutorial in Google Colaboratory

1.1 Installation (execute only if running on a cloud platform!)Âű

[1]:
```
# -- Use the following line for google colab
#! pip install -q 'corner==2.0.1'
```

Important: With Google Colab, you may need to restart the runtime after running the cell above.

1.2 Initialization

[2]:
```
from __future__ import division, print_function
import numpy as np
import matplotlib.pyplot as plt
import h5py
import pandas as pd
import corner
```

1.3 Get the data

Selecting the event, let's pick GW150914.

[3]:
```
label = 'GW150914'
```

```
# if you do not have wget installed, simply download manually
# https://dcc.ligo.org/LIGO-P1800370/public/GW150914_GWTC-1.hdf5
# from your browser
! wget https://dcc.ligo.org/LIGO-P1800370/public/{label}_GWTC-1.hdf5
```

```
--2020-11-21 19:28:28--
https://dcc.ligo.org/LIGO-P1800370/public/GW150914_GWTC-1.hdf5
Resolving dcc.ligo.org (dcc.ligo.org)... 131.215.125.144
Connecting to dcc.ligo.org (dcc.ligo.org)|131.215.125.144|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://dcc.ligo.org/public/0157/P1800370/005/GW150914_GWTC-1.hdf5
[following]
--2020-11-21 19:28:30--
https://dcc.ligo.org/public/0157/P1800370/005/GW150914_GWTC-1.hdf5
Reusing existing connection to dcc.ligo.org:443.
HTTP request sent, awaiting response... 200 OK
Length: 7026464 (6.7M)
Saving to: âĂŸGW150914_GWTC-1.hdf5âĂŹ

GW150914_GWTC-1.hdf 100%[===================>]   6.70M  2.60MB/s    in 2.6s

2020-11-21 19:28:32 (2.60 MB/s) - âĂŸGW150914_GWTC-1.hdf5âĂŹ saved [7026464/
↪7026464]
```

[4]:
```
posterior_file = './'+label+'_GWTC-1.hdf5'
posterior = h5py.File(posterior_file, 'r')
```

1.3.1 Looking into the file structure

[5]:
```
print('This file contains four datasets: ',posterior.keys())
```

```
This file contains four datasets:  <KeysViewHDF5 ['IMRPhenomPv2_posterior',
'Overall_posterior', 'SEOBNRv3_posterior', 'prior']>
```

This data file contains several datasets, two using separate models for the gravitaional waveform (`IMRPhenomPv2` and `SEOBNRv3` respectively, see the paper for more details).

It also contiains a joint dataset, combining equal numbers of samples from each individual model, these datasets are what is shown in the paper.

Finally, there is a dataset containing samples drawn from the prior used for the analyses.

[6]:
```
print(posterior['Overall_posterior'].dtype.names)
```

```
('costheta_jn', 'luminosity_distance_Mpc', 'right_ascension', 'declination',
'm1_detector_frame_Msun', 'm2_detector_frame_Msun', 'spin1', 'spin2',
'costilt1', 'costilt2')
```

Here are some brief descriptions of these parameters and their uses:

- `luminosity_distance_Mpc`: luminosity distance [Mpc]
- `m1_detector_frame_Msun`: primary (larger) black hole mass (detector frame) [solar mass]
- `m2_detector_frame_Msun`: secondary (smaller) black hole mass (detector frame) [solar mass]
- `right_ascension`, `declination`: right ascension and declination of the source [rad].
- `costheta_jn`: cosine of the angle between line of sight and total angular momentum vector of system.
- `spin1`, `costilt1`: primary (larger) black hole spin magnitude (dimensionless) and cosine of the zenith angle between the spin and the orbital angular momentum vector of system.
- `spin2`, `costilt2`: secondary (smaller) black hole spin magnitude (dimensionless) and cosine of the zenith angle between the spin and the orbital angular momentum vector of system.

A convenient (and pretty) way to load up this array of samples is to use pandas:

[7]:
```
samples=pd.DataFrame.from_records(np.array(posterior['Overall_posterior']))
```

[8]:
```
samples
```

[8]:

	costheta_jn	luminosity_distance_Mpc	right_ascension	declination	\
0	-0.976633	517.176717	1.456176	-1.257815	
1	-0.700404	401.626864	2.658802	-0.874661	
2	-0.840752	369.579071	1.106548	-1.136396	
3	-0.583657	386.935268	2.077180	-1.246351	
4	-0.928271	345.104345	0.993604	-1.069243	
5	-0.861517	312.266527	0.884342	-0.875526	
6	-0.993057	482.702041	1.254024	-1.220087	
7	-0.951662	473.665172	1.419747	-1.271510	
8	-0.896320	469.099680	2.236555	-1.234226	
9	-0.963932	556.990637	1.700747	-1.262282	
10	-0.820570	500.368921	2.276248	-1.202984	
11	-0.674318	393.492394	2.341810	-1.190604	
12	-0.650114	420.066518	2.509839	-1.105781	
13	-0.933496	468.648499	1.637059	-1.261151	
14	-0.906077	507.140697	1.434600	-1.239394	
15	-0.884868	377.765909	1.158292	-1.171804	
16	-0.767135	394.088489	1.830805	-1.287126	
17	-0.963054	545.922299	1.163536	-1.148894	
18	-0.956892	420.316660	1.192301	-1.186619	
19	-0.903831	550.460213	2.082583	-1.262594	
20	-0.911840	501.145664	2.086447	-1.261614	
21	-0.946788	588.954034	2.416206	-1.192111	
22	-0.977904	513.966913	1.831148	-1.262297	
23	-0.935906	433.154879	1.479955	-1.253897	
24	-0.905283	341.298103	0.998063	-1.072230	

25	-0.988916	331.863993	0.896731	-0.893949
26	-0.734077	329.644247	2.422131	-1.157909
27	-0.968690	407.794381	1.430451	-1.247261
28	-0.901933	466.915429	1.635602	-1.277849
29	-0.911420	520.304212	1.898219	-1.255650
...	...	...	...	...
8320	-0.963754	479.627679	1.421226	-1.237403
8321	-0.797818	291.640379	0.921329	-1.037707
8322	-0.903869	302.571288	0.894647	-0.926900
8323	-0.995566	535.871423	2.426527	-1.161662
8324	-0.128536	251.468407	1.351458	-1.186373
8325	-0.916900	481.455085	1.402443	-1.226163
8326	-0.750079	398.735004	1.256436	-1.252159
8327	-0.952350	362.100941	1.094760	-1.108855
8328	-0.863589	409.387362	1.266844	-1.216554
8329	-0.867119	584.992707	2.544252	-1.071247
8330	-0.916898	396.680219	1.293819	-1.225568
8331	-0.964043	556.990459	1.609491	-1.258841
8332	-0.936385	550.553176	2.260506	-1.191693
8333	-0.991132	589.459904	2.267003	-1.215404
8334	-0.980707	515.160072	1.755466	-1.274265
8335	-0.951436	406.233345	1.080483	-1.137830
8336	-0.543811	382.763475	2.350371	-1.165626
8337	-0.314194	273.768244	1.907840	-1.238437
8338	-0.792538	484.979160	2.563289	-1.019216
8339	-0.868087	339.484613	0.955686	-1.021462
8340	-0.924296	495.492685	1.701333	-1.279951
8341	-0.781792	440.172496	1.538598	-1.270290
8342	-0.890317	569.832475	2.387177	-1.112518
8343	-0.906677	497.608021	1.878699	-1.267034
8344	-0.975888	513.885473	2.012550	-1.287542
8345	-0.691637	306.985025	1.485646	-1.269228
8346	-0.834615	462.649414	2.065362	-1.265618
8347	-0.911463	448.930876	1.536913	-1.257956
8348	-0.856914	561.020036	2.367289	-1.211824
8349	-0.919556	519.641782	1.916675	-1.250801

	m1_detector_frame_Msun	m2_detector_frame_Msun	spin1	spin2	\
0	39.037380	37.044563	0.417147	0.867740	
1	34.620096	34.184416	0.125709	0.260679	
2	37.894343	33.970520	0.581047	0.926893	
3	36.412973	35.684463	0.235808	0.094391	
4	39.477251	31.645008	0.511521	0.868009	
5	40.053864	31.991598	0.270972	0.248762	
6	41.302817	32.510163	0.296694	0.073529	
7	38.622335	36.758122	0.568455	0.548404	
8	41.520437	28.397390	0.538265	0.712628	

9	38.005237	36.227526	0.422409	0.185092
10	37.122250	36.083515	0.257545	0.332633
11	38.459260	31.799995	0.022493	0.364121
12	44.076230	29.201757	0.070059	0.241866
13	38.645475	32.398512	0.005890	0.119546
14	37.874577	36.509048	0.494393	0.636374
15	36.357493	33.687949	0.488504	0.254615
16	35.965565	34.205400	0.078071	0.234350
17	38.542505	35.798820	0.288123	0.559945
18	42.288312	33.486999	0.814305	0.373396
19	36.773857	36.210384	0.539052	0.563486
20	36.842878	35.179096	0.759660	0.827230
21	39.068531	36.506772	0.206433	0.108762
22	40.953668	33.062135	0.377102	0.178347
23	39.059075	34.346471	0.038759	0.469905
24	36.178895	33.058884	0.269331	0.046139
25	40.408669	30.338095	0.274004	0.457297
26	37.791702	35.097534	0.099933	0.120829
27	44.392623	27.101020	0.248231	0.152856
28	40.804260	34.357069	0.481804	0.262667
29	38.468793	38.112555	0.780451	0.456891
...	...	...	...	...
8320	37.418872	33.040471	0.213178	0.106581
8321	37.963982	36.136089	0.529649	0.409030
8322	39.094267	33.338767	0.559142	0.418611
8323	42.714378	30.441516	0.922040	0.989441
8324	36.806506	36.472964	0.928667	0.316016
8325	37.287401	36.838470	0.511637	0.173961
8326	41.601842	33.599982	0.018344	0.171433
8327	41.690405	30.406117	0.424441	0.358483
8328	37.583397	33.914142	0.229000	0.254653
8329	39.350488	33.336474	0.014724	0.014333
8330	36.265841	33.210403	0.491009	0.051018
8331	40.152133	33.299200	0.276635	0.282308
8332	40.094982	32.925201	0.361812	0.664824
8333	36.951265	36.662160	0.980342	0.814305
8334	38.136783	35.955693	0.184186	0.310562
8335	36.627502	36.258330	0.234235	0.082137
8336	38.205089	33.141560	0.173946	0.427770
8337	38.056001	33.102631	0.393329	0.455752
8338	36.365292	34.805373	0.234390	0.112930
8339	38.351915	33.147393	0.263137	0.024049
8340	43.020937	28.566935	0.023839	0.301614
8341	38.802564	34.771638	0.187242	0.128243
8342	39.873922	35.808303	0.556795	0.564091
8343	38.359023	35.279562	0.449313	0.311660
8344	41.548878	28.998181	0.280597	0.954617

8345	37.561962	33.355792	0.484003	0.627191
8346	37.824298	36.674075	0.589654	0.650758
8347	38.063291	35.757913	0.708407	0.714805
8348	44.884396	31.592433	0.389284	0.521304
8349	37.275183	35.445032	0.391824	0.516908

	costilt1	costilt2
0	-0.280624	0.403853
1	-0.757349	-0.312285
2	0.649781	-0.510843
3	0.116578	-0.720505
4	-0.438237	0.269333
5	-0.349028	0.545518
6	0.137425	-0.083697
7	-0.594185	0.877344
8	-0.045699	-0.426311
9	0.681972	-0.847753
10	-0.951930	0.814314
11	-0.049694	-0.468250
12	0.990392	-0.460822
13	-0.022517	-0.642337
14	0.590248	-0.352235
15	-0.534439	0.618918
16	-0.617526	-0.395582
17	0.267301	0.041915
18	0.438368	-0.855349
19	0.174670	-0.012382
20	-0.579448	0.395875
21	0.534922	0.544726
22	0.128910	0.550940
23	-0.219874	0.053649
24	-0.606615	0.512709
25	-0.881741	0.277930
26	0.336268	0.149600
27	-0.254594	0.346958
28	0.542702	-0.567035
29	-0.255877	0.806603
...	...	...
8320	-0.206470	-0.841896
8321	0.749740	-0.718774
8322	0.258864	-0.649596
8323	0.569637	-0.896173
8324	0.129004	-0.504672
8325	0.531790	-0.986684
8326	0.176509	0.850630
8327	-0.089335	0.007443
8328	0.470166	-0.730604

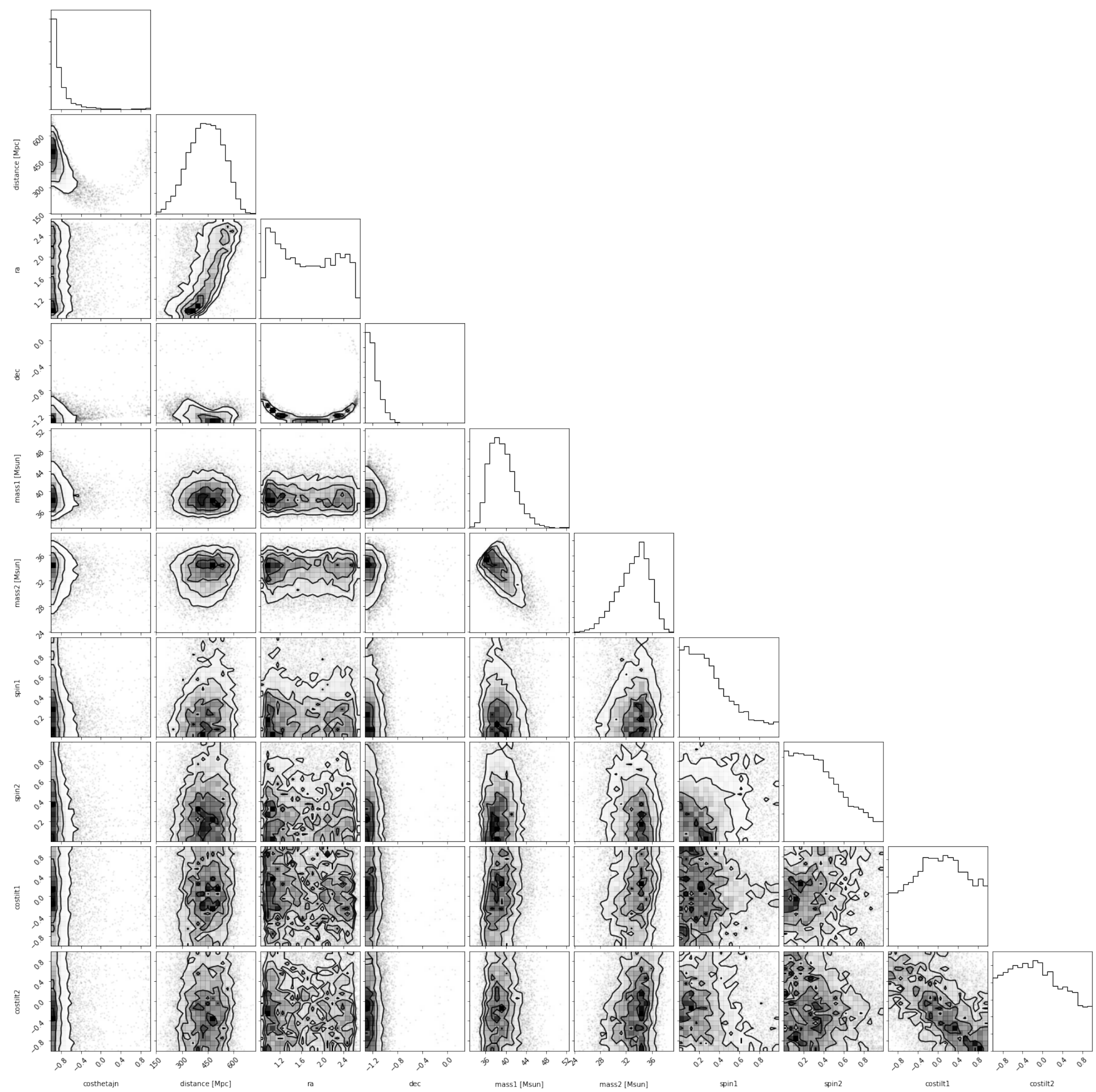
```
8329 -0.203959 -0.831735
8330 -0.411079  0.952778
8331  0.397194 -0.102864
8332 -0.296723  0.156690
8333  0.018503 -0.030134
8334  0.187724  0.264574
8335 -0.081511  0.262844
8336  0.969616 -0.554059
8337  0.492497 -0.758070
8338 -0.007013 -0.919706
8339 -0.410763  0.038883
8340  0.235901 -0.382719
8341 -0.124488  0.932231
8342  0.821992 -0.662277
8343 -0.029597  0.195098
8344  0.613993 -0.547786
8345  0.194507 -0.408345
8346 -0.737792  0.875384
8347  0.852085 -0.797475
8348 -0.251461  0.830526
8349 -0.705305  0.600727

[8350 rows x 10 columns]
```

Those are all the samples stored in the `Overall` dataset.

1.3.2 Plotting

We can plot all of them with, for instance, the corner package:

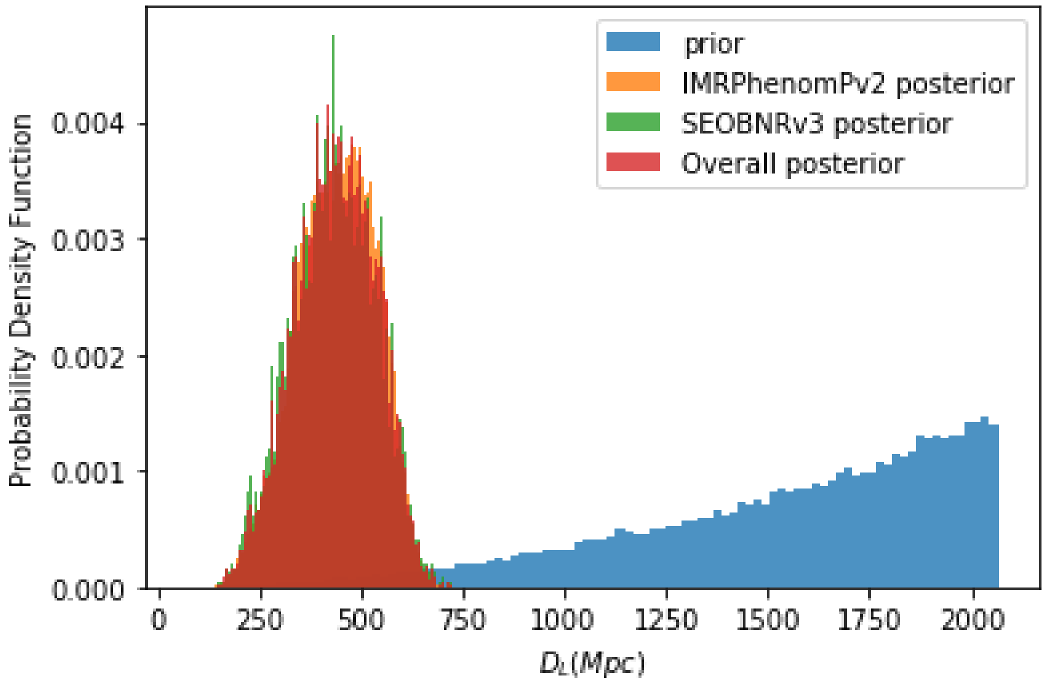
[9]:
```
corner.corner(samples,labels=['costhetajn',
                                'distance [Mpc]',
                                'ra',
                                'dec',
                                'mass1 [Msun]',
                                'mass2 [Msun]',
                                'spin1',
                                'spin2',
                                'costilt1',
                                'costilt2']);
```



Each one and two dimentional histogram are *marginalised* probabilby density functions. We can manualy select one parameter, say `luminosity distance`, and plot the four different marginalised distributions:

[10]:
```
plt.hist(posterior['prior']['luminosity_distance_Mpc'], bins = 100,␣
 ↪label='prior', alpha=0.8, density=True)
plt.hist(posterior['IMRPhenomPv2_posterior']['luminosity_distance_Mpc'], bins =␣
 ↪100, label='IMRPhenomPv2 posterior', alpha=0.8, density=True)
plt.hist(posterior['SEOBNRv3_posterior']['luminosity_distance_Mpc'], bins = 100,␣
 ↪label='SEOBNRv3 posterior', alpha=0.8, density=True)
plt.hist(posterior['Overall_posterior']['luminosity_distance_Mpc'], bins = 100,␣
 ↪label='Overall posterior', alpha=0.8, density=True)
plt.xlabel(r'$D_L (Mpc)$')
```

```
plt.ylabel('Probability Density Function')
plt.legend()
plt.show()
```



1.3.3 Computing new quantities

The masses given are the ones seens by the detector, in the "detector frame". To get the masses of the source black holes, we need to correct for the gravitational-wave redshifting. This forces us to assume a cosmology:

[11]:
```
import astropy.units as u
from astropy.cosmology import Planck15, z_at_value
```

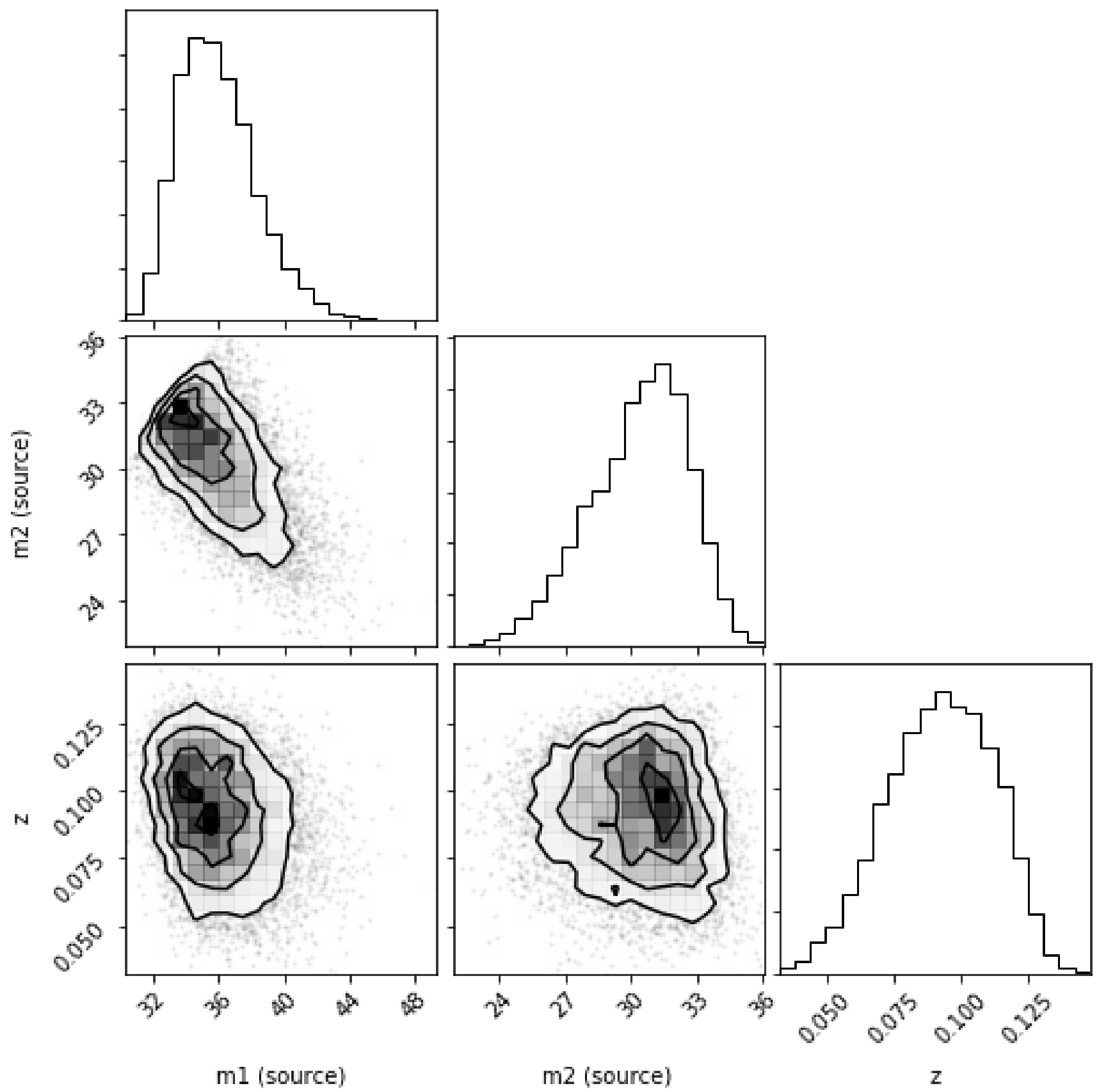
We now compute the redshift value for all the samples (using only their distance value). See astropy.cosmology for implementation details, in particular how to make the following more efficient:

[12]:
```
z = np.array([z_at_value(Planck15.luminosity_distance, dist * u.Mpc) for dist in␣
↪samples['luminosity_distance_Mpc']])
```

[13]:
```
samples['m1_source_frame_Msun']=samples['m1_detector_frame_Msun']/(1.0+z)
samples['m2_source_frame_Msun']=samples['m2_detector_frame_Msun']/(1.0+z)
samples['redshift']=z
```

And we can plot the marginalised probability density functions:

[14]:
```
corner.
↪corner(samples[['m1_source_frame_Msun','m2_source_frame_Msun','redshift']],labels=['m1␣
↪(source)',
                                                                                        ␣
↪           'm2 (source)',
                                                                                        ␣
↪           'z']);
```



1.4 Calculating credible intervals

Let's see how we can use bilby to calcuate summary statistics for the posterior like the median and 90% credible level.

[15]:
```
import bilby
# calculate the detector frame chirp mass
mchirp = ((samples['m1_detector_frame_Msun'] *
 ↪samples['m2_detector_frame_Msun'])**(3./5))/\
         (samples['m1_detector_frame_Msun'] +
 ↪samples['m2_detector_frame_Msun'])**(1./5)
# initialize a SampleSummary object to describe the chirp mass posterior samples
chirp_mass_samples_summary = bilby.core.utils.SamplesSummary(samples=mchirp,
 ↪average='median')
print('The median chirp mass = {} Msun'.format(chirp_mass_samples_summary.
 ↪median))
print('The 90% confidence interval for the chirp mass is {} - {} Msun'.
 ↪format(chirp_mass_samples_summary.lower_absolute_credible_interval,

 ↪chirp_mass_samples_summary.upper_absolute_credible_interval))
```

```
The median chirp mass = 31.23055308109465 Msun
The 90% confidence interval for the chirp mass is 29.655877108464615 -
32.97324559242388 Msun
```

1.5 Challenge question

Calculate the posterior for the effective spin, which is the mass-weighted component of the binary spin aligned to the orbital angular momentum. It is given by Eqn. 3 of https://journals.aps.org/prx/pdf/10.1103/PhysRevX.9.011001. The z-component of each component spin is defined as $\chi_{1z} = \chi_1 \cos\theta_1$ Then initialize a `SamplesSummary` object for the chi_eff posterior and calculate the mean and the lower and upper absolute credible interval.

1.5.1 $\chi_{eff} = \dfrac{m_1\chi_{1z} + m_2\chi_{2z}}{m_1 + m_2}$

$$\chi_{eff} = \frac{m_1\chi_1 \cos\theta_1 + m_2\chi_2 \cos\theta_2}{m_1 + m_2}$$

[26]:
```
X_det = ((samples['m1_detector_frame_Msun'] * samples['spin1'] *
 ↪samples['costilt1']) + (samples['m2_detector_frame_Msun'] * samples['spin2'] *
 ↪samples['costilt2']))/(samples['m1_detector_frame_Msun'] +
 ↪samples['m2_detector_frame_Msun'])
```

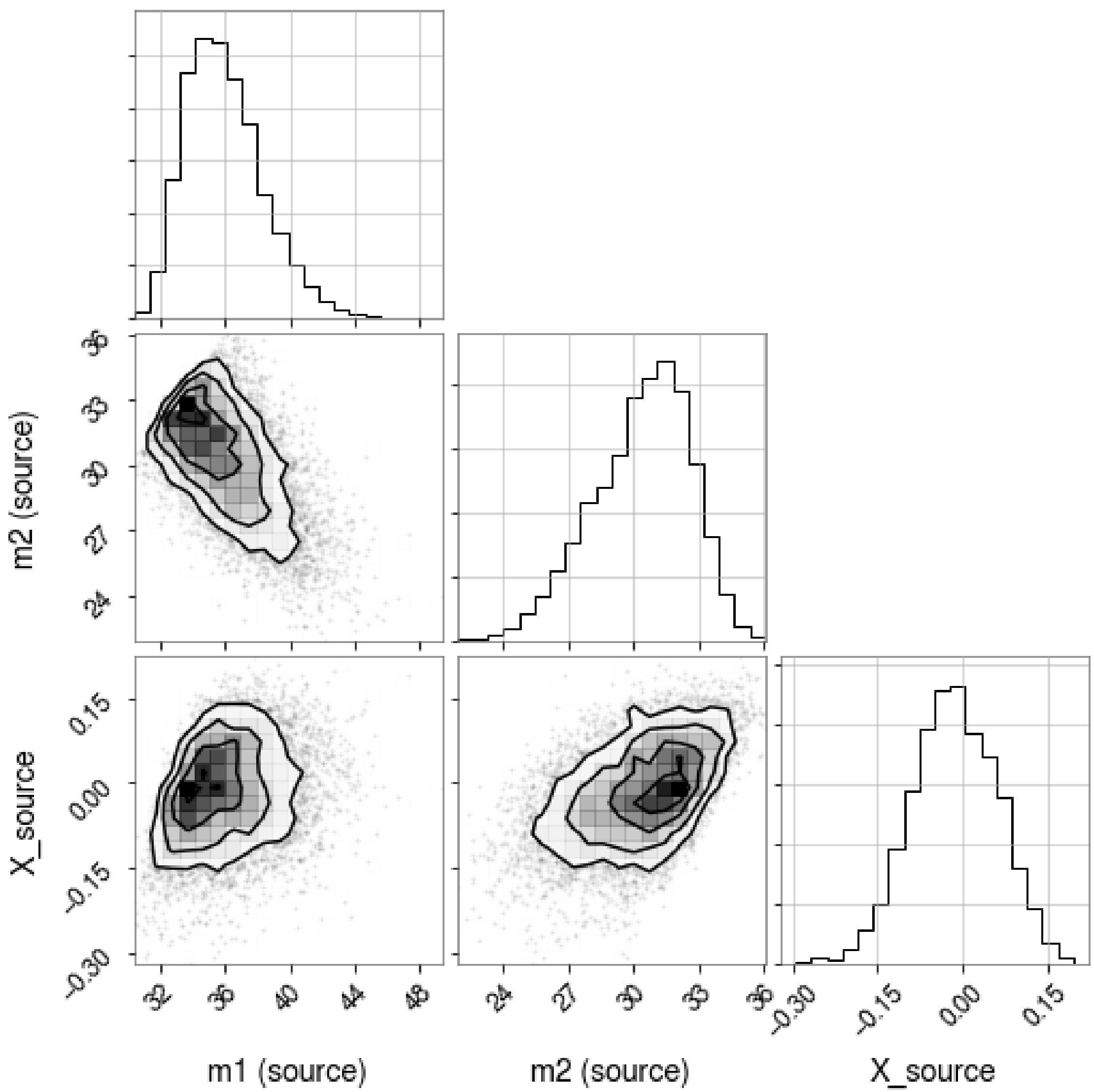
[28]:

```
X_source = ((samples['m1_source_frame_Msun'] * samples['spin1'] *␣
↪samples['costilt1']) + (samples['m2_source_frame_Msun'] * samples['spin2'] *␣
↪samples['costilt2']))/(samples['m1_source_frame_Msun'] +␣
↪samples['m2_source_frame_Msun'])
```

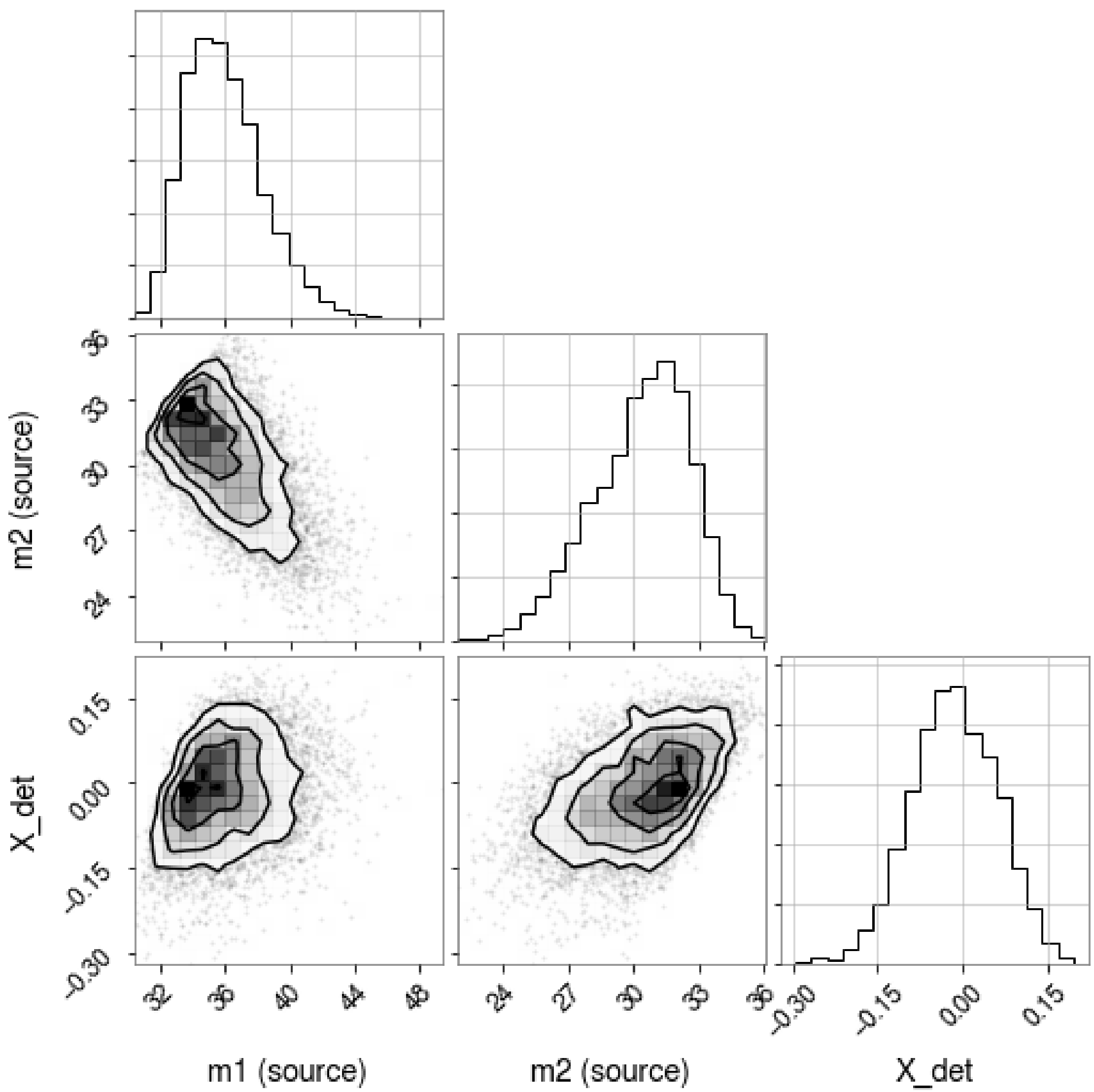
[31]:
```
samples['chi_eff_det']=X_det
```

[32]:
```
samples['chi_eff_source']=X_source
```

[33]:
```
corner.
↪corner(samples[['m1_source_frame_Msun','m2_source_frame_Msun','chi_eff_source']],labels=['m1␣
↪(source)',
                                                                                        ␣
↪         'm2 (source)',
                                                                                        ␣
↪         'X_source']);
```



[34]:
```
corner.
↪corner(samples[['m1_source_frame_Msun','m2_source_frame_Msun','chi_eff_det']],labels=['m1␣
↪(source)',
                                                                                    ␣
↪           'm2 (source)',
                                                                                    ␣
↪           'X_det']);
```



[38]:
```
chi_eff_det_samples_summary = bilby.core.utils.SamplesSummary(samples=X_det,␣
    ↪average='median')
print('The median chi_eff_det = {} '.format(chi_eff_det_samples_summary.median))
print('The 90% confidence interval for the chi_eff_det is {} - {} '.
    ↪format(chi_eff_det_samples_summary.lower_absolute_credible_interval,
                                                                              ␣
    ↪chi_eff_det_samples_summary.upper_absolute_credible_interval))
```

```
The median chi_eff_det = -0.014932627380503358
The 90% confidence interval for the chi_eff_det is -0.14138898362283844 -
0.10932272798368625
```

[39]:

```
chi_eff_source_samples_summary = bilby.core.utils.
 ↪SamplesSummary(samples=X_source, average='median')
print('The median chi_eff_source = {} '.format(chi_eff_source_samples_summary.
 ↪median))
print('The 90% confidence interval for the chi_eff_source is {} - {} '.
 ↪format(chi_eff_source_samples_summary.lower_absolute_credible_interval,
                                                                              ␣
 ↪chi_eff_source_samples_summary.upper_absolute_credible_interval))
```

```
The median chi_eff_source = -0.014932627380503357
The 90% confidence interval for the chi_eff_source is -0.14138898362283844 -
0.10932272798368625
```

[ ]:

```

```