CHALLENGE4

May 15, 2021

[1]:
```
from gwpy.timeseries import TimeSeries
data1= TimeSeries.read('challenge3.gwf', 'H1:CHALLENGE3')
data2=TimeSeries.read('challenge3.gwf', 'L1:CHALLENGE3')
```

[2]:
```
import numpy as np
signal1=np.array(data1)
signal2=np.array(data2)
```

[3]:
```
from pycbc.frame import read_frame

ts_H1 = read_frame("challenge3.gwf", "H1:CHALLENGE3")

print("Duration_H1: {}s delta_t_H1: {}s sampling rate_H1: {}Hz Start_H1: {}␣
 ↪End_H1: {}".format(ts_H1.duration, ts_H1.delta_t, 1/ts_H1.delta_t,
                                                    int(ts_H1.start_time),
                                                    int(ts_H1.end_time)))

ts_L1 = read_frame("challenge3.gwf", "L1:CHALLENGE3")

print("Duration_L1: {}s delta_t_L1: {}s sampling rate_L1: {}Hz Start_L1: {}␣
 ↪End_L1: {}".format(ts_L1.duration, ts_L1.delta_t, 1/ts_L1.delta_t,
                                                    int(ts_L1.start_time),
                                                    int(ts_L1.end_time)))
```

```
Duration_H1: 4096.0s delta_t_H1: 0.000244140625s sampling rate_H1: 4096.0Hz
Start_H1: 0 End_H1: 4096
Duration_L1: 4096.0s delta_t_L1: 0.000244140625s sampling rate_L1: 4096.0Hz
Start_L1: 0 End_L1: 4096
```

[4]:
```
from pycbc.filter import resample_to_delta_t, highpass
from pycbc.catalog import Merger
from pycbc.psd import interpolate, inverse_spectrum_truncation

data_H1 = resample_to_delta_t(ts_H1, 1.0/2048).crop(2, 2)

p_H1 = data_H1.psd(2)
p_H1 = interpolate(p_H1, data_H1.delta_f)
```

```
p_H1 = inverse_spectrum_truncation(p_H1, 2 * data_H1.sample_rate,␣
 ↪low_frequency_cutoff=15.0)
psd_H1 = p_H1

data_L1 = resample_to_delta_t(ts_L1, 1.0/2048).crop(2, 2)

p_L1 = data_L1.psd(2)
p_L1 = interpolate(p_L1, data_L1.delta_f)
p_L1 = inverse_spectrum_truncation(p_L1, 2 * data_L1.sample_rate,␣
 ↪low_frequency_cutoff=15.0)
psd_L1 = p_L1
```

[10]:
```
class color:
   PURPLE = '\033[95m'
   CYAN = '\033[96m'
   DARKCYAN = '\033[36m'
   BLUE = '\033[94m'
   GREEN = '\033[92m'
   YELLOW = '\033[93m'
   RED = '\033[91m'
   BOLD = '\033[1m'
   UNDERLINE = '\033[4m'
   END = '\033[0m'
```

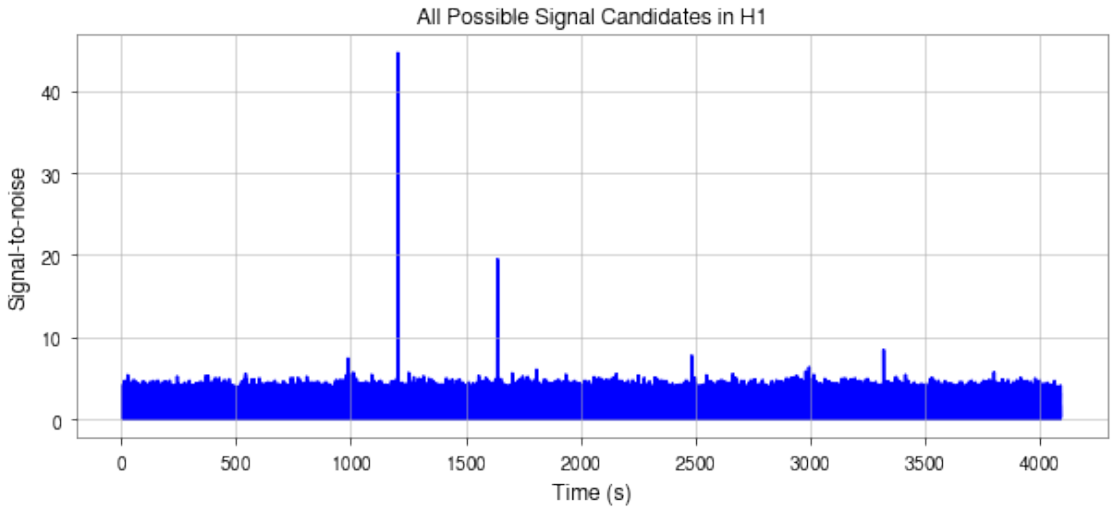
[16]:
```
masses=[]
import pylab
from colorama import Fore, Back, Style
from pycbc.waveform import get_fd_waveform
from pycbc.filter import matched_filter
import numpy
from pycbc.vetoes import power_chisq
from pycbc.events.ranking import newsnr

for x in range(22,23):
  print(color.BOLD + "Individual masses of the BBHs -- ", x, "solar masses" +␣
 ↪color.END)
  print(Style.RESET_ALL)

  hp1, _ = get_fd_waveform(approximant="IMRPhenomD",
                       mass1=x,
                       mass2=x,
                       delta_f=data_H1.delta_f,
                       f_lower=20.0)


  hp1.resize(len(psd_H1))
```

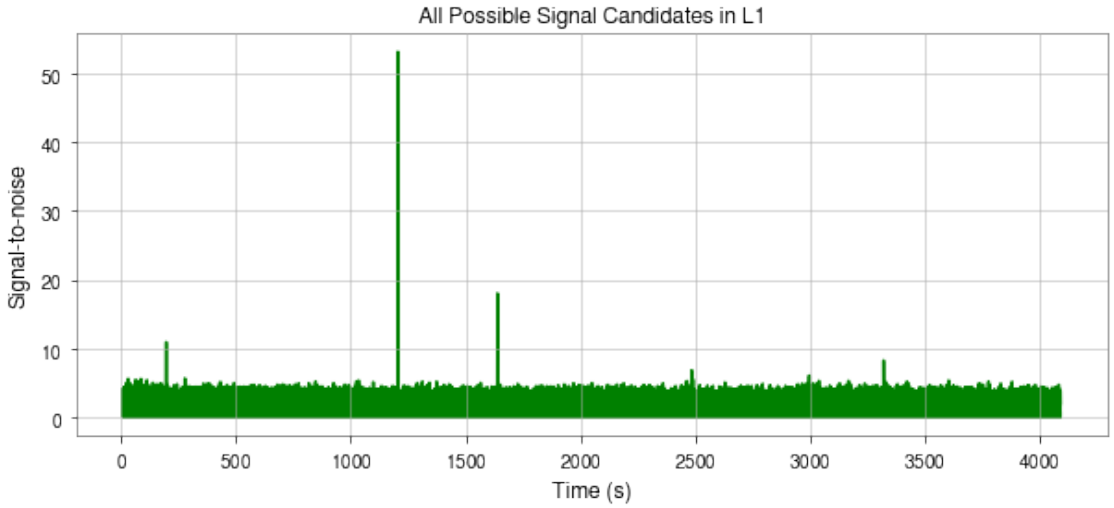
```
snr1 = matched_filter(hp1, data_H1,
                      psd=psd_H1, low_frequency_cutoff=20)

snr1 = snr1.crop(5, 4)

peak1 = abs(snr1).numpy().argmax()
snrp1 = snr1[peak1]
time1 = snr1.sample_times[peak1]


#print("We found a possible signal candidate at {}s with SNR {} in H1".
↪format(time1,
#                                                    abs(snrp1)))
hp2, _ = get_fd_waveform(approximant="IMRPhenomD",
                      mass1=x,
                      mass2=x,
                      delta_f=data_L1.delta_f,
                      f_lower=20.0)

# We will resize the vector to match our data
hp2.resize(len(psd_L1))

snr2 = matched_filter(hp2, data_L1,
                      psd=psd_L1, low_frequency_cutoff=20)


snr2 = snr2.crop(5, 4)
peak2 = abs(snr2).numpy().argmax()
snrp2 = snr2[peak2]
time2 = snr2.sample_times[peak2]


#print("We found a possible signal candidate at {}s with SNR {} in L1".
↪format(time2,
#                                                    abs(snrp2)))
#if time1==time2:
 # print("Peak Time is coincident in both detectors")

pylab.figure(figsize=[10, 4])
pylab.plot(snr1.sample_times, abs(snr1), '-b')
pylab.title('All Possible Signal Candidates in H1')
pylab.ylabel('Signal-to-noise')
pylab.xlabel('Time (s)')
pylab.show()

pylab.figure(figsize=[10, 4])
```

```
pylab.plot(snr2.sample_times, abs(snr2), '-g')
pylab.title('All Possible Signal Candidates in L1')
pylab.ylabel('Signal-to-noise')
pylab.xlabel('Time (s)')
pylab.show()

nbins = 26
dof = nbins * 2 - 2
chisq_H1 = power_chisq(hp1, data_H1, nbins, psd_H1, low_frequency_cutoff=20.0)
chisq_H1 = chisq_H1.crop(5, 4)
chisq_H1 /= dof

chisq_L1 = power_chisq(hp2, data_L1, nbins, psd_L1, low_frequency_cutoff=20.0)
chisq_L1 = chisq_L1.crop(5, 4)
chisq_L1 /= dof

nsnr1 = newsnr(abs(snr1), chisq_H1)
nsnr2 = newsnr(abs(snr2), chisq_L1)

peak_H1 = nsnr1.argmax()
snrp_H1 = nsnr1[peak_H1]
time_H1 = snr1.sample_times[peak_H1]



if snrp_H1>8:
  print("For the Hanford data we found a confirmed signal at {}s with SNR {}␣
↪after ruling out the other candidates as glitches".format(time_H1,␣
↪abs(snrp_H1)))
  print(Style.RESET_ALL)
  pylab.figure(figsize=[10, 4])
  pylab.plot(snr1.sample_times, nsnr1, '-b')
  pylab.title('Remaining Signal Candidate in H1 after reweighing data')
  pylab.xlabel('Time (s)')
  pylab.ylabel('Re-weighted Signal-to-noise')
  pylab.show()

peak_L1 = nsnr2.argmax()
snrp_L1 = nsnr2[peak_L1]
time_L1 = snr2.sample_times[peak_L1]

if snrp_L1>8:
  print("For the Livingston data we found a confirmed signal at {}s with SNR␣
↪{} after ruling out the other candidates as glitches".format(time_L1,␣
↪abs(snrp_L1)))
  print(Style.RESET_ALL)
  pylab.figure(figsize=[10, 4])
```

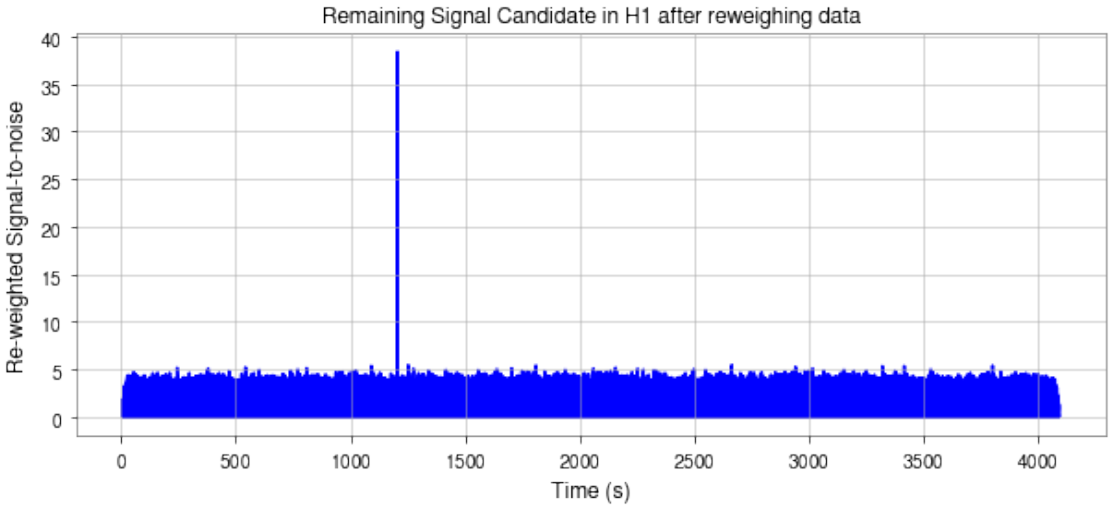
```
    pylab.plot(snr2.sample_times, nsnr2, '-g')
    pylab.title('Remaining Signal Candidate in L1 after reweighing data')
    pylab.xlabel('Time (s)')
    pylab.ylabel('Re-weighted Signal-to-noise')
    pylab.show()

  if ((snrp_L1>8 and snrp_H1>8) and (time_H1==time_L1)):
    print(Fore.BLUE + "There is a confirmed signal detection at {}s for both␣
↪the detectors".format(time_L1))
    print(Style.RESET_ALL)
  else:
    print("There was no simultaneous detection in H1 and L1 hence we rule out␣
↪any authentic GW detection")
    print(Style.RESET_ALL)
  masses.append(x)
```

Individual masses of the BBHs -- 22 solar masses

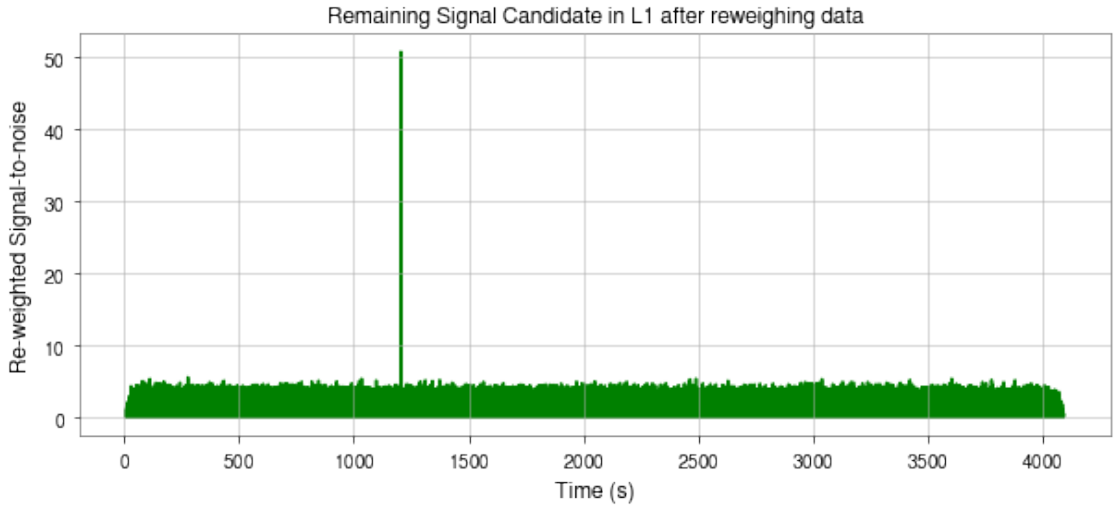




For the Hanford data we found a confirmed signal at 1204.71044921875s with SNR 38.386718619181224 after ruling out the other candidates as glitches



For the Livingston data we found a confirmed signal at 1204.71044921875s with SNR 50.73312219066904 after ruling out the other candidates as glitches



There is a confirmed signal detection at 1204.71044921875s for both the detectors

[ ]:

[ ]:

[ ]:

[5]:
```
from __future__ import division, print_function
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt

import bilby
from bilby.core.prior import Uniform
from bilby.gw.conversion import convert_to_lal_binary_black_hole_parameters,␣
 ↪generate_all_bbh_parameters

from gwpy.timeseries import TimeSeries

sampling_rate=2048 #needs to be high enough for the signals found in steps above
duration=8 #needs to be long enough for the signals found in steps above
start_time=100 #needs to be set so that the segment defined by␣
 ↪[start_time,start_time+duration] contains the signal

interferometers = bilby.gw.detector.InterferometerList([])
```

```
for ifo_name in ['H1','L1']:
    ifo=bilby.gw.detector.get_empty_interferometer(ifo_name)
    ifo.set_strain_data_from_frame_file('challenge3.gwf',sampling_rate,␣
 ↪duration, start_time=start_time ,channel=ifo_name+':CHALLENGE3')
    interferometers.append(ifo)
```

```
17:48 bilby INFO    : Reading data from frame file challenge3.gwf
17:48 bilby INFO    : Successfully loaded H1:CHALLENGE3.
17:48 bilby INFO    : Reading data from frame file challenge3.gwf
17:48 bilby INFO    : Successfully loaded L1:CHALLENGE3.
```

[ ]: