Effects of Different Data Quality Veto Methods in the PyCBC Search for Compact Binary Coalescences

LIGO SURF 2020
August 5, 2020

Brina Martinez
University of Texas Rio Grande Valley
Mentor: Dr. Derek Davis
LIGO Laboratory, California Institute of Technology

Topics we covered last time

- How the PyCBC search pipeline works
- DQ veto analysis
- Current status of PyCBC veto methods
- How to correctly choose flags
- Downranking time around signals
- Improving the search background

Topics we will cover today

- Goals of the project
- Current veto methods
- New methods
- Current results

Goals

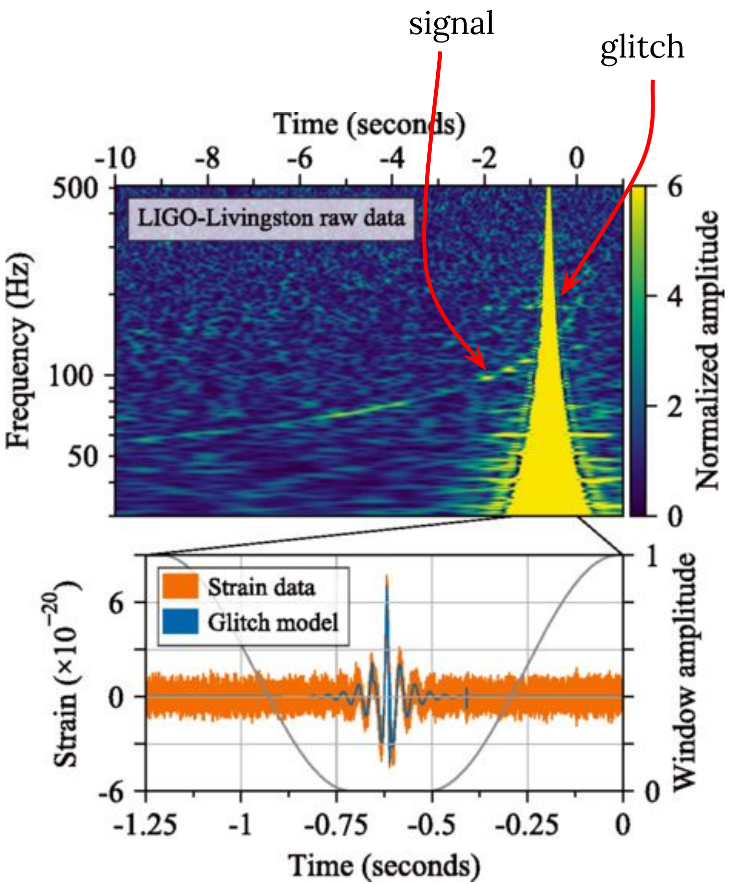
We want to confidently mitigate noisy data in the detector's data and finely tune our machine to prevent a decrease in search sensitivity and detect more signals!

- What can cause a decrease in search sensitivity?
  - Keeping loud glitches
  - Removing too much data (time)
  - Using ineffective flags

Current DQ Veto Methods

- A few problems we can see with current methods of veto analysis in PyCBC are:
  - Not removing enough glitches can decrease the search sensitivity
  - The possible removal of a signal if it occurs the same time as a glitch

- Our method shows an effective glitch veto that increases the significance of signals and the overall number of detectable signals without removing data.
  - Use the likelihood of our glitches to re-rank them against the original background
  - Increase the search sensitivity without risking the removal of a signal



B.P. Abbott *et al.* Phys. Rev. Lett. 119, 161101

Old Method vs New Method

Previous Method

- Removes glitches and flagged times completely
- If flags are not as efficient, they do not highlight enough glitches
- Uses chi-square consistency test to analyze glitches and downrank

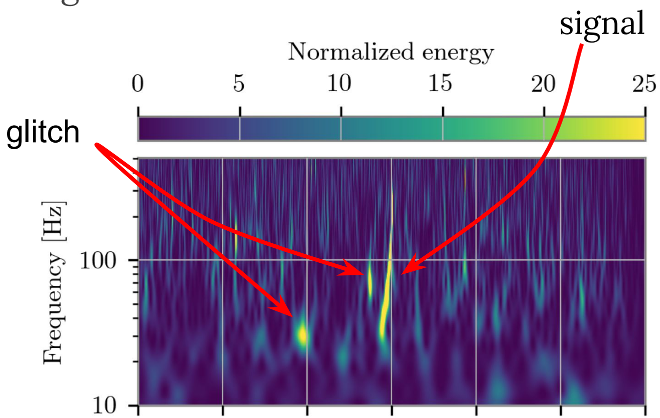


Fig from GstLAL, LIGO Document T2000349-v2

New Method

- Keeps glitches that are flagged, removing no data
- Uses chi-square consistency test and re-ranking of the glitch statistic

How is this done?

- Uses CAT2 data quality vetoes
- Uses Likelihood of glitches that fall into flags to re-rank data

Likelihood in the New Method

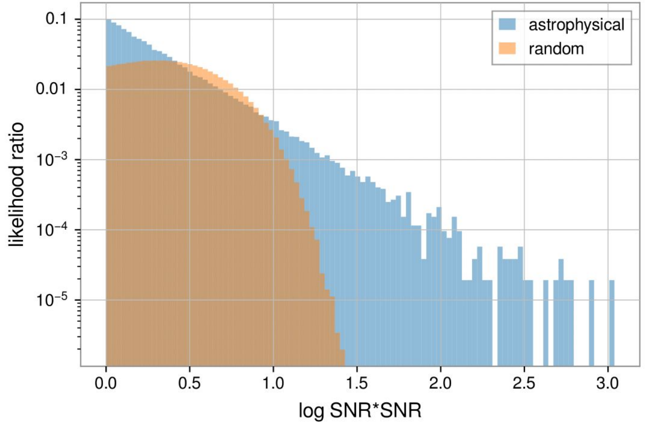
Likelihood Ratio

$$\Lambda(\theta_1 : \theta_2 \mid x) = \frac{\mathcal{L}(\theta_1 \mid x)}{\mathcal{L}(\theta_2 \mid x)} \longrightarrow \frac{\mathcal{L}_n(\widetilde{\rho}) = Ce^{-\frac{\widetilde{\rho}^2}{2}}}{\mathcal{L}_n(\rho) = Ce^{-\frac{\rho^2}{2}}}$$

- How much more likely is a trigger to show up during a flag vs all time?

$$\mathcal{L}(flag) = \frac{\mathcal{L}(flagtime)}{\mathcal{L}(totaltime)}$$

- We want our likelihood ratio ≥ 1




```
Number of triggers: 10621
Number of flagged times: 20
Total known time: 711183
Total active time of flags: 211.0
Likelihood of total time: 0.000001406
Likelihood of flags: 8.924463784749073e-06
Likelihood ratio: 6.347083926
```

```
Number of triggers: 10621
Number of flagged times: 31
Total known time: 711183
Total active time of flags: 115.0
Likelihood of total time: 0.000001406
Likelihood of flags: 2.5380398963497253e-05
Likelihood ratio: 18.051209104
```

Ranking in New Method

Re-ranking glitches

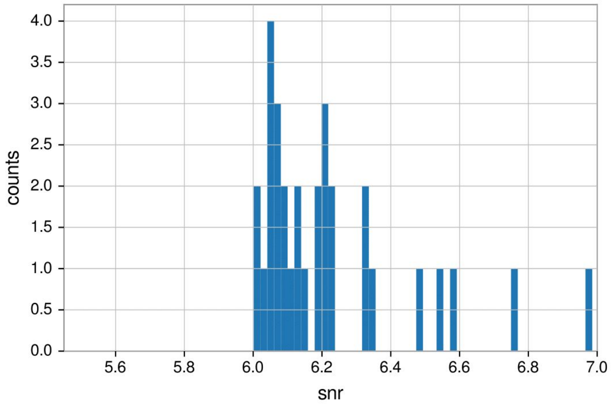
$$\widetilde{\rho} = \sqrt{\rho^2 - 2ln\mathcal{L}}$$

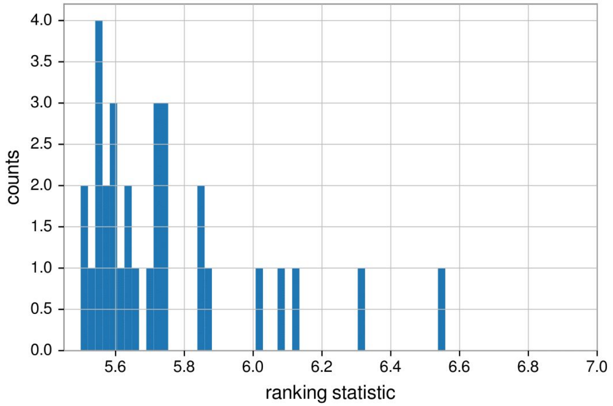
New statistic values of noise

Original snr values of noise

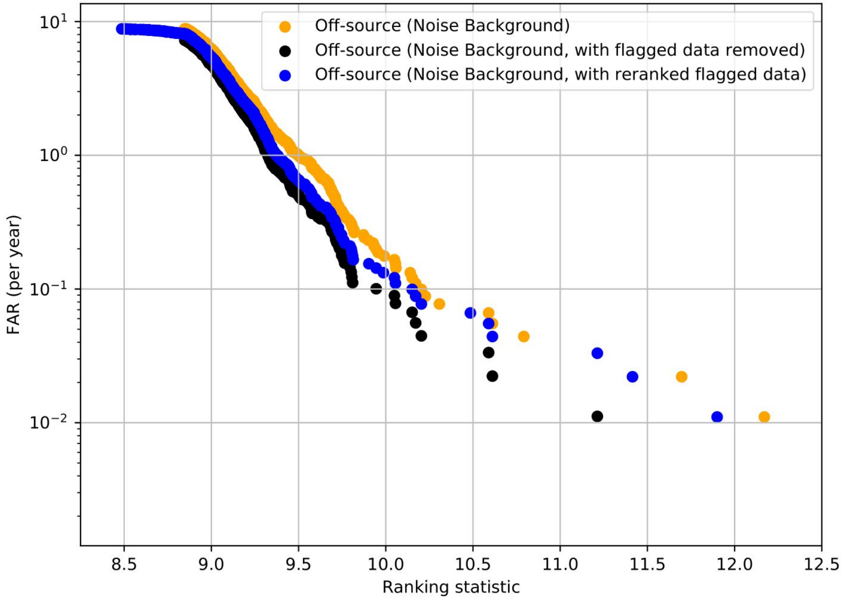
Likelihood ratio of noise

- Glitches are updated by ranking statistic





Results




```
original vs flagged comparison
The ratio of distance:          1.09
The ratio of time:              0.99
The ratio of volume * time:     1.27
original vs reranked comparison
The ratio of distance:          1.02
The ratio of time:              1.00
The ratio of volume * time:     1.07
flagged vs reranked comparison
The ratio of distance:          0.94
The ratio of time:              1.01
The ratio of volume * time:     0.84
```

Next Steps:

- Expand on parameters of templates
- Expand amount of flags applied

Thank you! Questions?