Weekly Meeting

Feb 14th 2018

Meeting with Amy

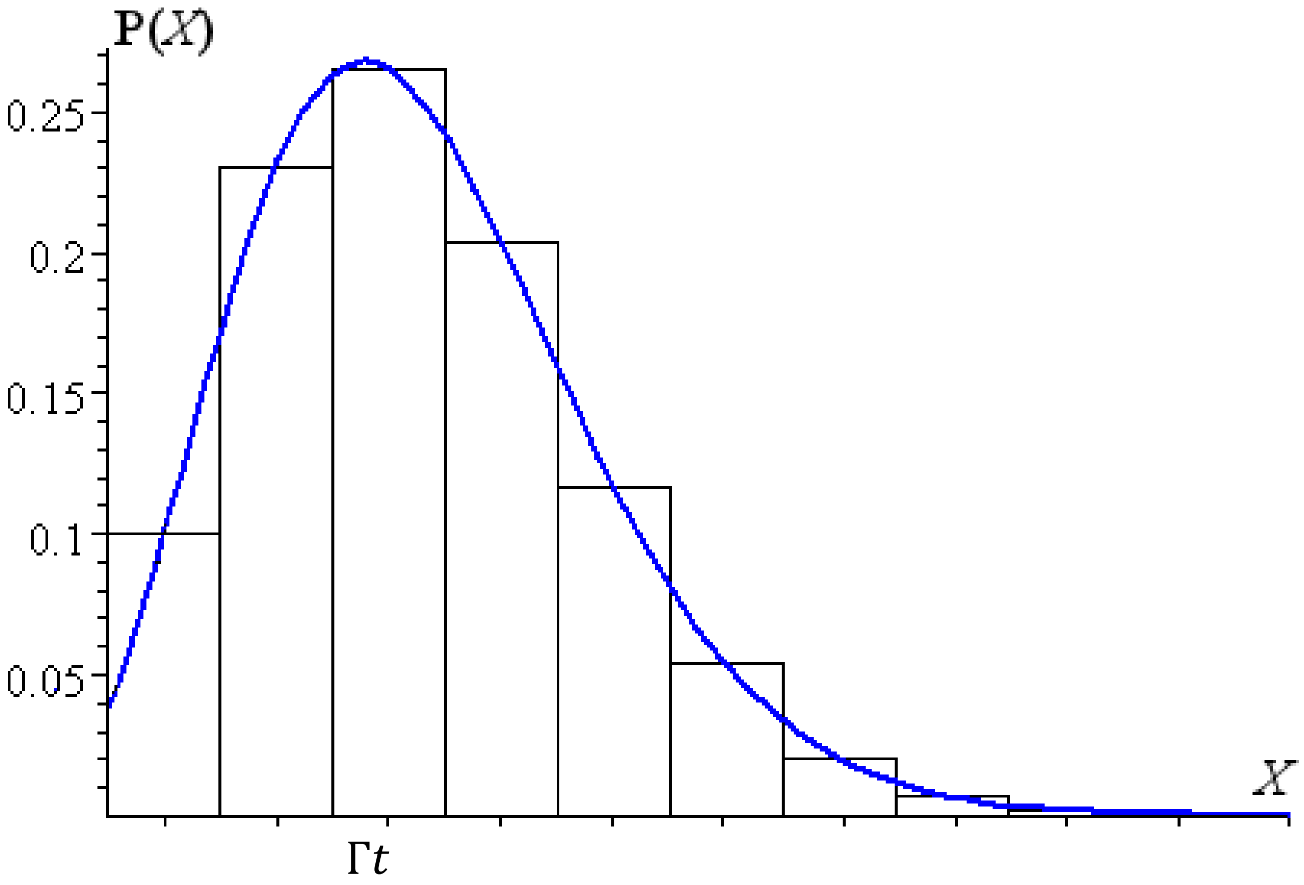
- Goal is do have a rough estimate of the time it will take to produce noisePSDs from noise traces
- There is currently a limit set to 5 min, check to see if code (in serial) can produce noisePSDs within this time. If not, may need to parallelize the code
- There are 3 things that need to be accounted for:
  - Trace length (need SNOLAB trace length)
  - Number of channels for SNOLAB detectors
  - Adjusting for processor speed

Meeting with Amy

- I can use traces from UMN or SLAC, but they may not be the right length (check with Scott/Bill)
  - Even if they aren’t the right length, it wouldn’t be a bad idea to see how runtime scales with trace length
- Amy suggested using a program called Valgrind, which can isolate the code from the rest of the programs running on the processor
  - They may already be a compiled version of CDMSBats on UMN computers with Valgrind enabled.
- Timescale is mid-March

Trigger Burst Cut

- After talking with Noah, we have come up with a new way to apply the trigger bursts cut – making use of **expected** trigger rates and Poisson distribution.
- If the expected trigger rate is Γ, and I take a sample over a time of t, then I would expect Γt events. If the data was a perfect Poisson distribution, then I would have a distribution of mean Γt and standard deviation $\sqrt{\Gamma t}$.



Trigger Burst Cut

- In terms of trigger rate, this would mean we expect a mean trigger rate of Γ and a standard deviation of $\sigma = \frac{\sqrt{\Gamma t}}{t}$. We can then apply a 3σ cut.
- To determine the length of t (bin size), we use the following constraint: we require $m * (\sigma t) = n * \Gamma t$. This leads to: $t = \frac{m^2}{n^2 \Gamma}$. For $m = 7$ and $n = 2$ and $\Gamma = 5$, $t = 2.45\ s$

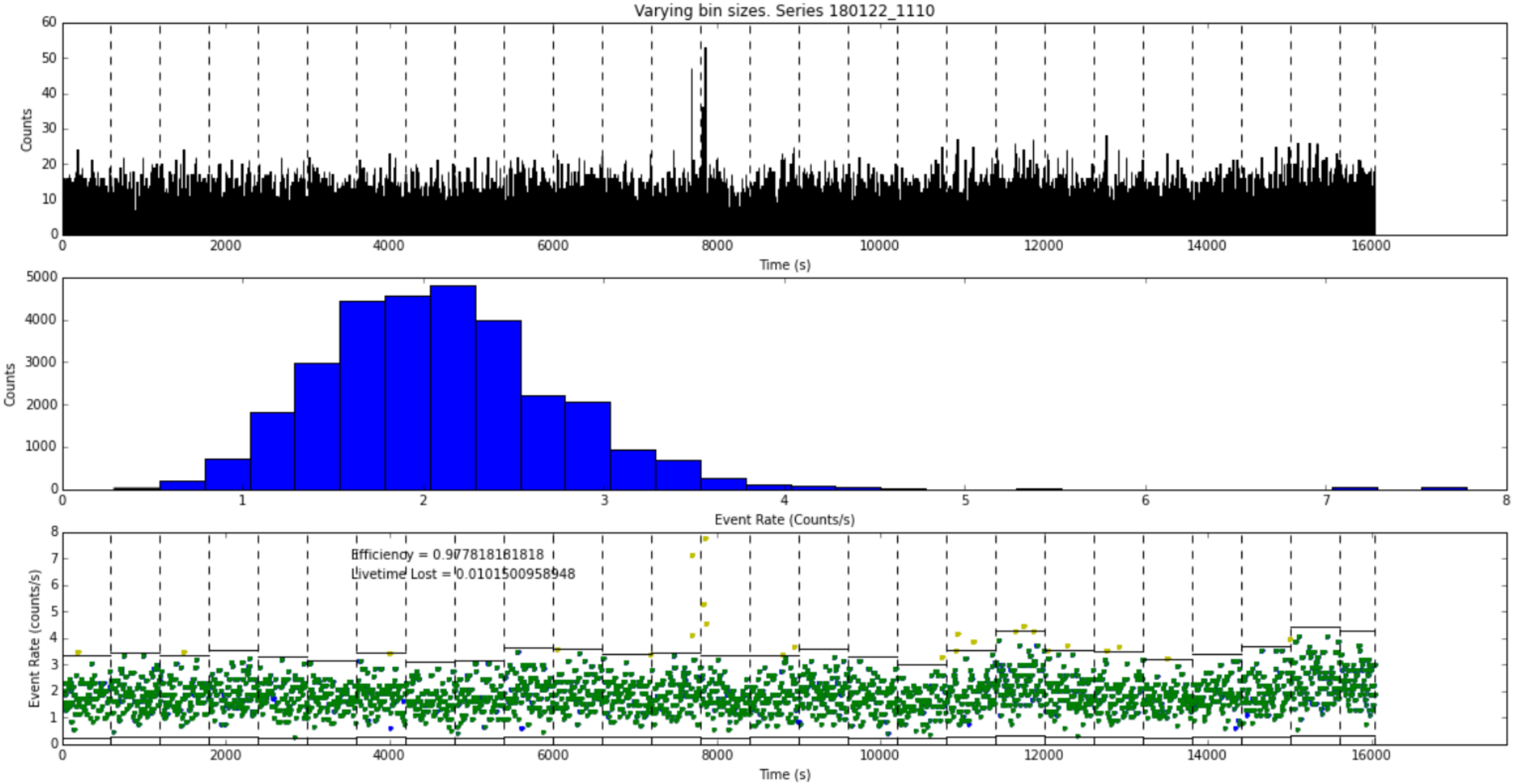
Trigger Burst Cut

- We extended this method further by allowing for variation in mean trigger rate → using mean trigger rate μ instead of Γ.
- We do this by using the method previously described to determine trigger rates, and then remove obvious outliers (8σ). We also split the data into time segments ~10 min. We then calculate the mean trigger rate μ, and use that value to determine bin size t^* and σ^*. The cuts are then $\mu \pm 3\sigma^*$.
- This way, the bin sizes and cuts are recalculated for every 10 min of data.
- There is also a 10Hz hard cut applied.

IP[y]: Notebook TriggerBurstCut Last Checkpoint: Feb 12 17:07 (autosaved)

File Edit View Insert Cell Kernel Help

Code Cell Toolbar: None



In [19]:

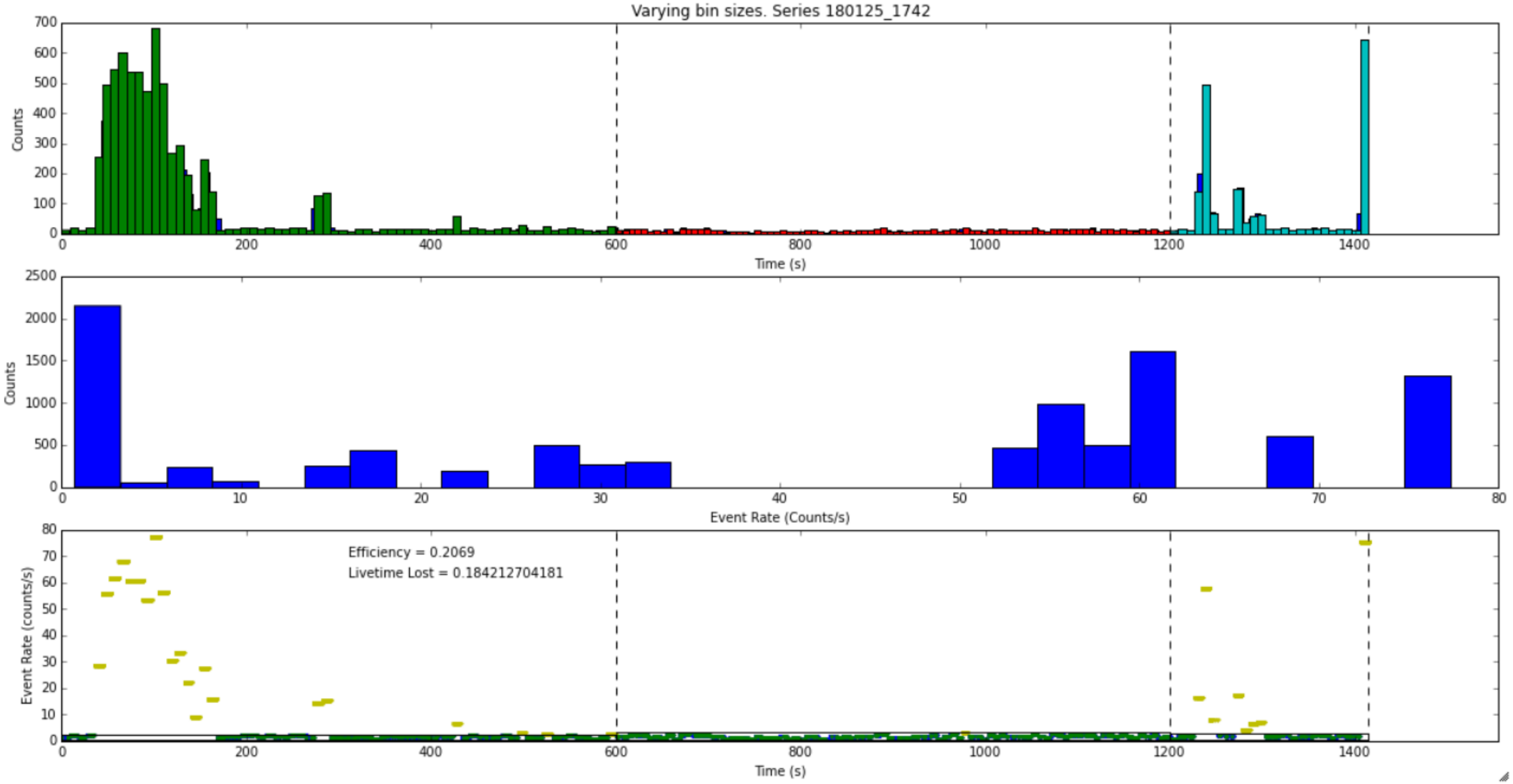
IP[y]: Notebook TriggerBurstCut Last Checkpoint: Feb 12 17:07 (autosaved)

File Edit View Insert Cell Kernel Help

Code Cell Toolbar: None

```
    ax.text(0.2*xplotmax,0.8*max(event_rate),'Livetime Lost = %s' %livetime)

plt.show()
```




```
In [19]:

In [ ]:
```

IP[y]: Notebook TriggerBurstCut Last Checkpoint: Feb 12 17:07 (autosaved)

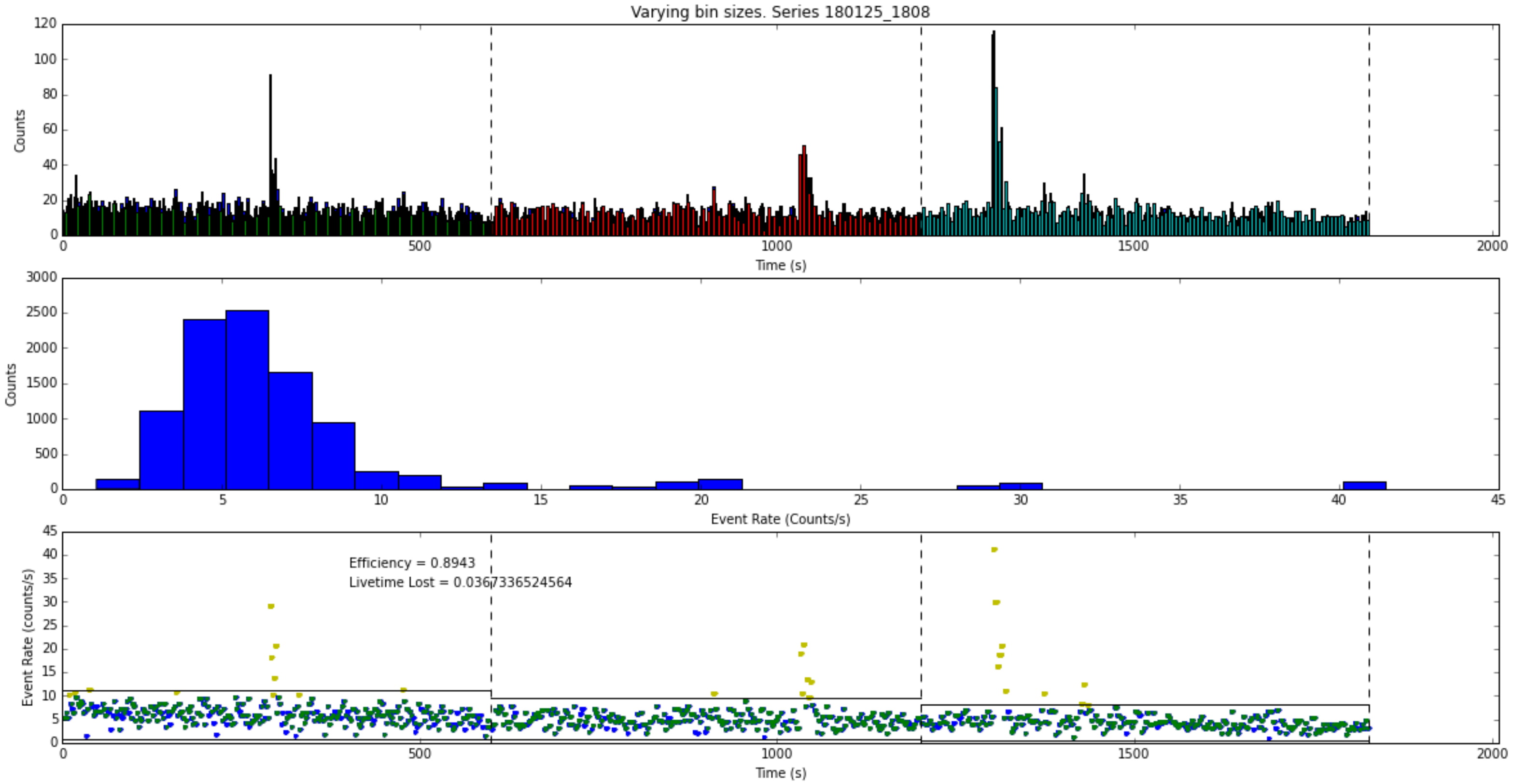
File Edit View Insert Cell Kernel Help

Code Cell Toolbar: None

```
    ax.set_ylabel('Event Rate (counts/s)')

    ax.text(0.2*xplotmax,0.9*max(event_rate),"Efficiency = %s" %eff)
    ax.text(0.2*xplotmax,0.8*max(event_rate),"Livetime Lost = %s" %livetime)


plt.show()
```



In [19]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

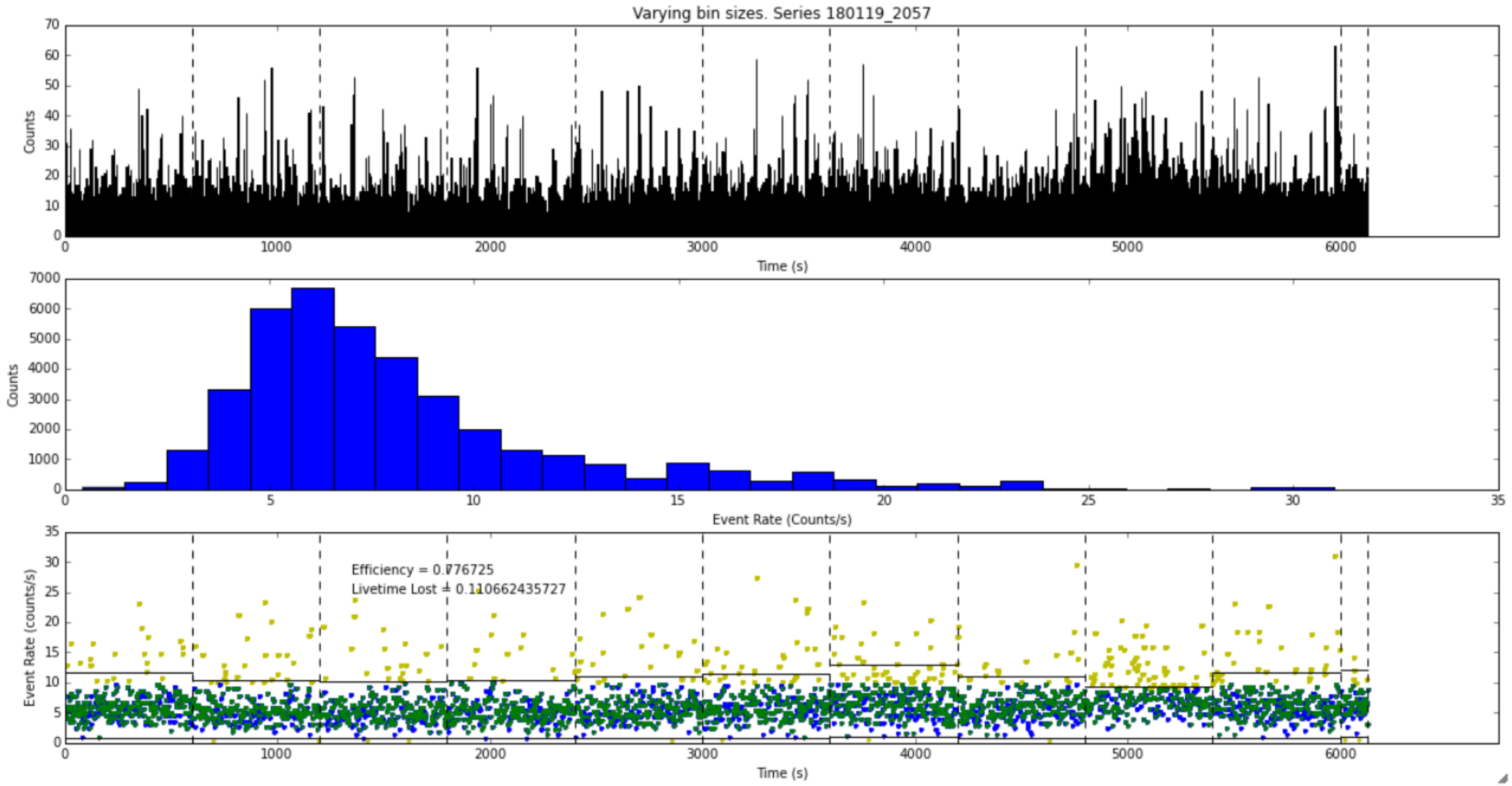
IP[y]: Notebook TriggerBurstCut Last Checkpoint: Feb 12 17:07 (unsaved changes)

File Edit View Insert Cell Kernel Help

Code Cell Toolbar: None

```
    ax.text(0.2*xplotmax,0.8*max(event_rate),'Livetime Lost = %s' %livetime)

plt.show()
```




```
In [19]:

In [ ]:
```

IP[y]: Notebook TriggerBurstCut Last Checkpoint: Feb 12 17:07 (unsaved changes)

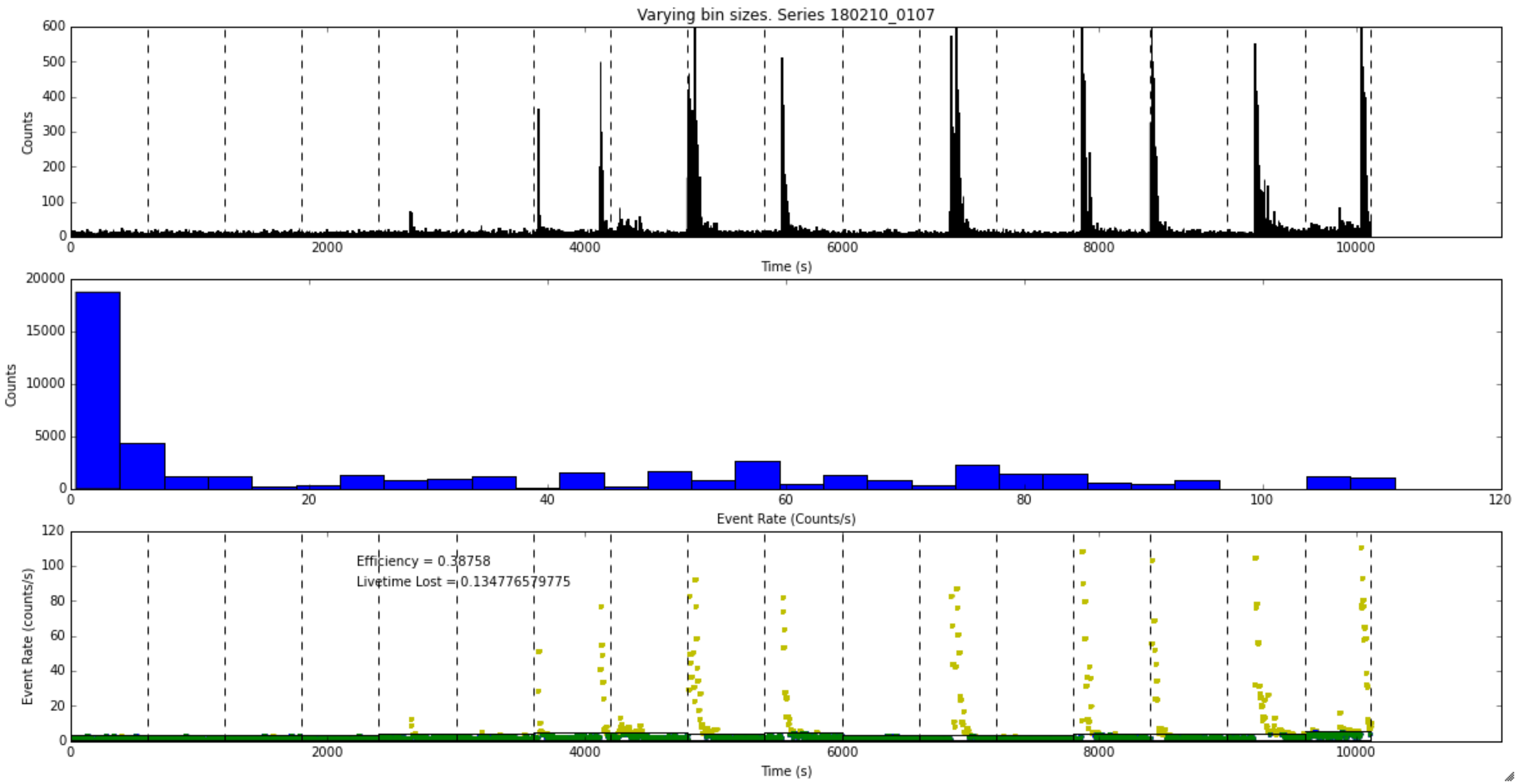
File Edit View Insert Cell Kernel Help

Code Cell Toolbar: None

```
    ax.set_ylabel('Event Rate (counts/s)')

    ax.text(0.2*xplotmax,0.9*max(event_rate),"Efficiency = %s" %eff)
    ax.text(0.2*xplotmax,0.8*max(event_rate),"Livetime Lost = %s" %livetime)


plt.show()
```



In [19]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: