

Anomaly Detection and Stability Measurement in the CMS Pixel Luminosity Telescope

Katie Ream

Advisors: Andres Guillermo Delannoy Sotomayor, Francesco Romeo, and on behalf of the BRIL group

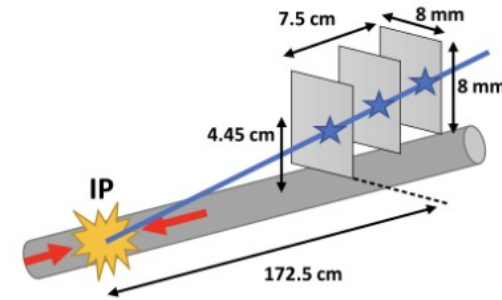
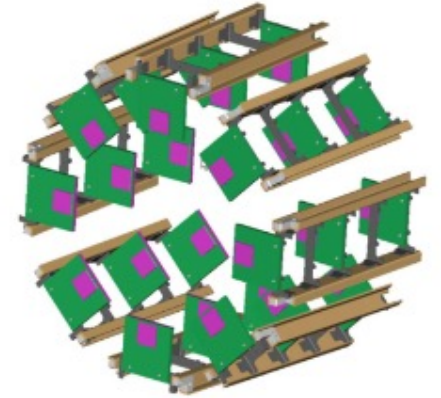
August 7, 2024

Talk Overview

1. Background
2. Detecting anomalies in data
3. Discussing ML, or the lack thereof
4. Example data
5. Constructing the algorithm
6. Applying to data
7. Results
8. Conclusion

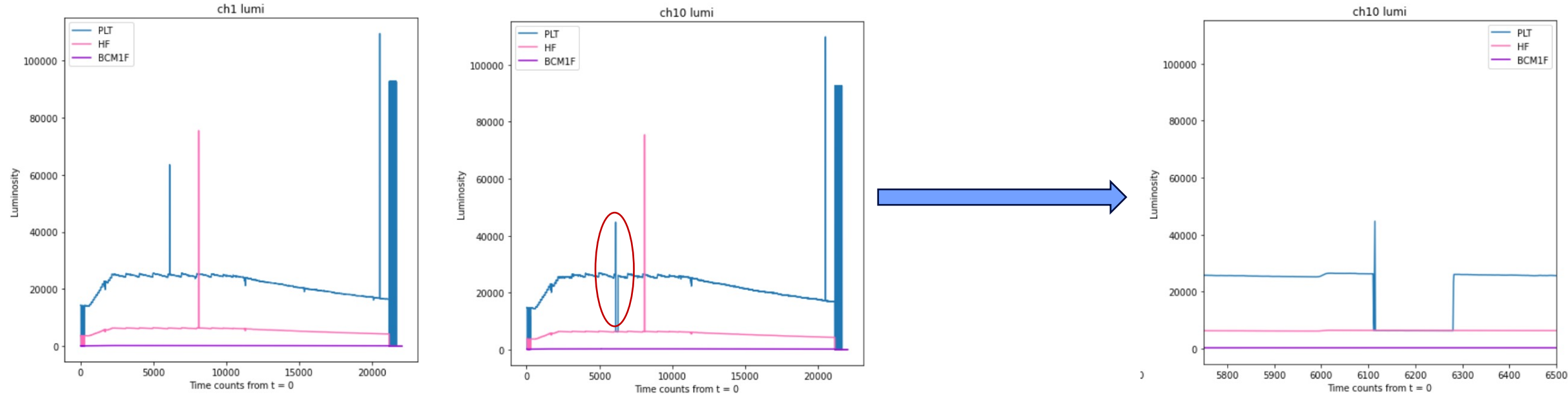
Background

- The BRIL (beam, radiation, intensity and luminosity) group delivers luminosity measurements to CMS using the following detectors:
 - **PLT**: *pixel luminosity telescope*
 - **BCM1F**: *fast beams condition monitor*
 - **HF**: *hadronic forward calorimeter*
 - **PCC**: *pixel clustering counting*
 - **RAMSES**: *radiation monitoring system for the environment and safety*
- PLT consists of 16 channels arranged shown to the right and consist of three silicon sensor planes
 - Luminosity hits are recorded from particles that hit all three planes on that channel
- Want to deliver accurate data to limit uncertainty on integrated luminosity measurement for CMS



Detecting anomalies in data

- We need to be able to detect outliers in the data, so they are removed for the final integrated luminosity calculation
- Also important to track overall shift of luminosity data per channel over longer time interval

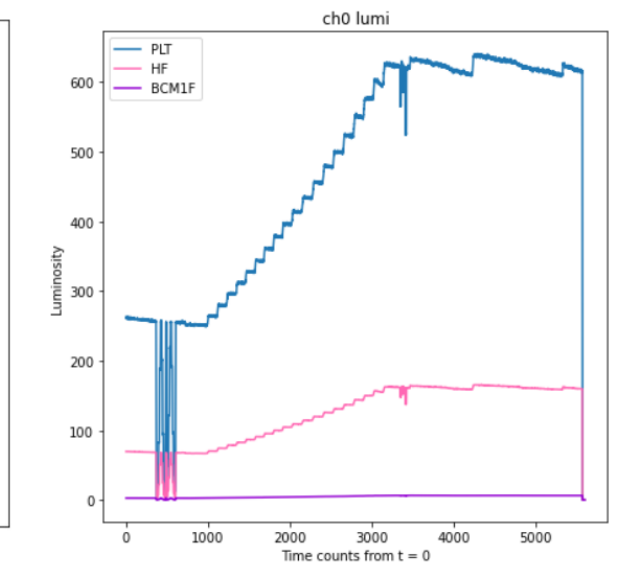
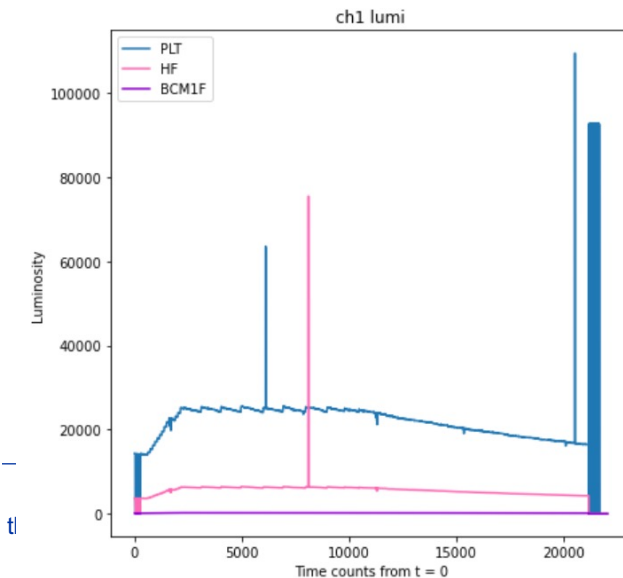
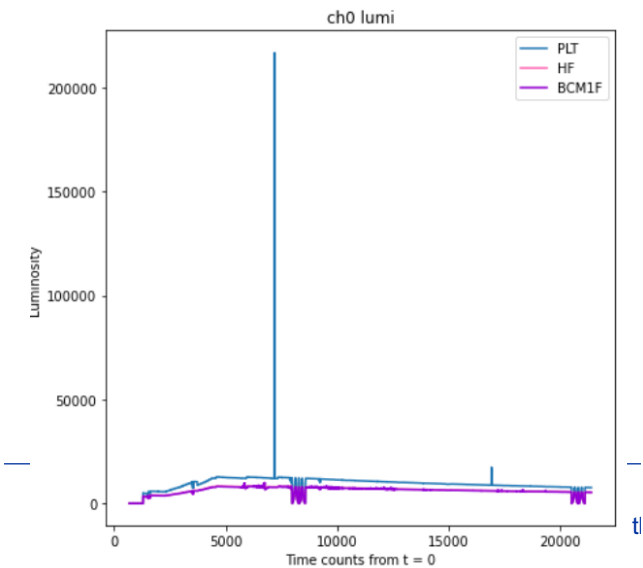
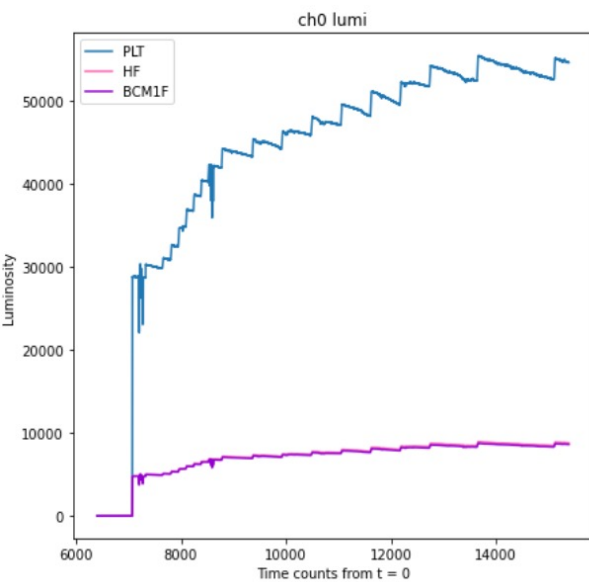
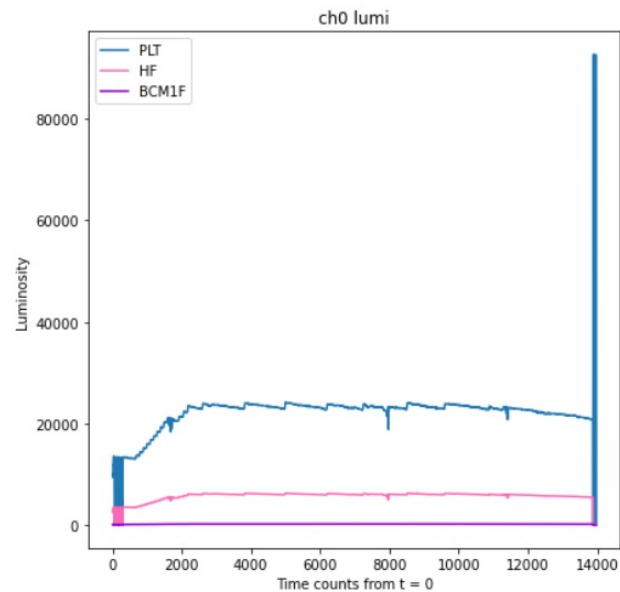
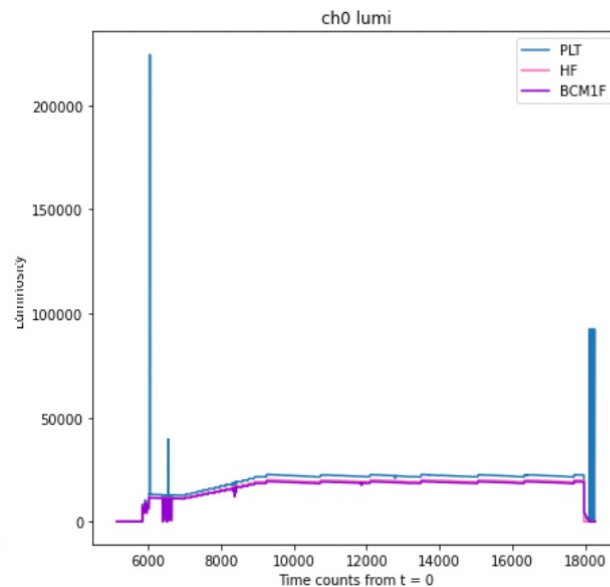
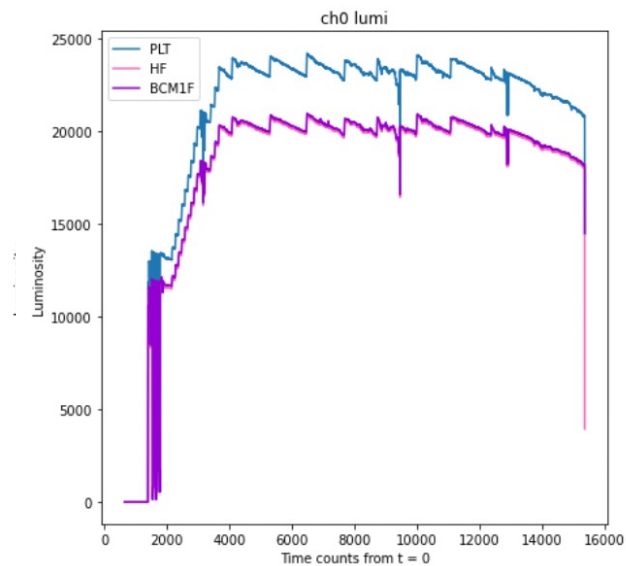
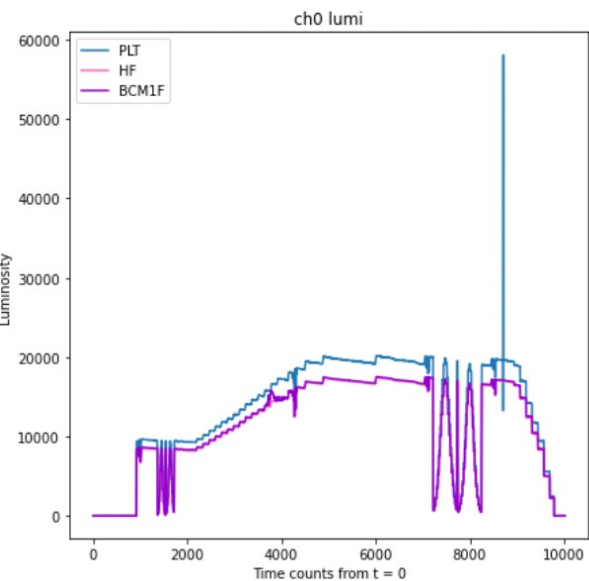


Discussing ML, or the lack thereof

- Detecting anomalies in data over multiple channels lends itself towards using ML
- Issues with this:
 1. The data is non-cyclical, meaning its not easily predicted
 2. The data varies in shape by fill
 3. Different data fills vary in time length, from five minutes to two days in length
- Thus, alternative route was explored: ***adtk python package***
 - Works on any time-series based data and is one of the few options that doesn't require data to be cyclical
 - Contains variety of different detector types that need to be tuned
 - To read more: [adtk documentation](#)

- Detector
 - ThresholdAD
 - QuantileAD
 - InterQuartileRangeAD
 - GeneralizedESDTestAD
 - PersistAD
 - LevelShiftAD
 - VolatilityShiftAD
 - SeasonalAD
 - AutoregressionAD
 - MinClusterDetector
 - OutlierDetector
 - RegressionAD
 - PcaAD
 - CustomizedDetector
- Transformer
 - RollingAggregate
 - DoubleRollingAggregate
 - ClassicSeasonalDecomposition
 - Retrospect
 - RegressionResidual
 - PcaProjection
 - PcaReconstruction
 - PcaReconstructionError
 - CustomizedTransformer

Example data



Constructing the algorithm

- The algorithm consists of a combination of detectors in adtk, each tuned to detect anomalies that reside above or below certain threshold values or pick up values due to unexpected spikes
- Additionally, BRIL needed a channel selection tool, specifically for 2023 PLT data – created a more generalized version
 - Channel selection tool counts the number of anomalies flagged per channel
 - If enough anomalies are flagged, then it checks the distribution of these anomalies
 - If one condition is not met, the channel fails and needs to be inspected further
 - Multiple scenarios are accounted for to account for variety in data

Applying to data

- Example of loading a data file into the algorithm to check if channels have passed or failed inspection
- Second code cell shows further inspection of a failed channel
- Following this, fills are appended to histograms to keep track of the channel shapes as time evolves

```
Begin Channel Selection
In [91]: reference = ch_data[closest_channel]['lumi']
reference_dt = dt_data[closest_channel]
percent = 0.20
levelshift = 300
window = 2
bad_channels = [6, 8, 9]
for channel in channels:
    if channel in bad_channels:
        print(f"Skipping bad channel {channel}. \n")
        continue
    select_channel = channel
    if not (len(reference) == len(ch_data[select_channel]['lumi']) == len(reference - ch_data[select_channel]['lumi'])):
        print(f"Length of reference: {len(reference)}")
        print(f"Length of select_channel: {len(ch_data[select_channel]['lumi'])}")
        print(f"Length of channels subtracted: {len(reference - ch_data[select_channel]['lumi'])}")
        continue
    ch_data[select_channel]['lumi'].replace([np.inf, -np.inf], np.nan, inplace=True)
    ch_data[select_channel]['lumi'].dropna(inplace=True)
    difference = reference - ch_data[select_channel]['lumi']
    difference.index = reference_dt
    difference.index = pd.to_datetime(difference.index).floor('s')
    ratio = (reference / ch_data[select_channel]['lumi'])
    ratio.index = reference_dt
    ratio.index = pd.to_datetime(ratio.index).floor('s')
    ratio = ratio.replace([np.inf, -np.inf], np.nan).dropna()
    difference.mean = np.mean(difference)
    ration.mean = np.mean(ratio)
    threshold_high2 = ration.mean + percent + ration.mean
    threshold_low2 = ration.mean - percent + ration.mean
    if difference.mean < 0:
        threshold_high_diff = difference.mean - percent + difference.mean
        threshold_low_diff = difference.mean + percent + difference.mean
    else:
        threshold_high_diff = difference.mean + percent + difference.mean
        threshold_low_diff = difference.mean - percent + difference.mean
    try:
        channel_selection(difference, threshold_high1-threshold_high1_diff, threshold_low1-threshold_low1_diff, levelshift_c_value=levelshift, levelshift_window=window, channel_number=select_channel,
            diff_ratio, threshold_high2-threshold_high2, threshold_low2-threshold_low2,
            title1=f"Reference - {select_channel} Anomaly Detection, Percent = +- {(percent * 100)}%", title2=f"Reference/{select_channel} Ratio Anomaly Detection, Percent = +- {(percent * 100)}%",
            title3=f"Reference - {select_channel} Common Anomaly Detection, Percent = +- {(percent * 100)}%", title4=f"Reference / {select_channel} Ratio Common Anomaly Detection, Percent = +- {(percent * 100)}%")
    except Exception as e:
        print(f"Channel {select_channel} failed inspection with error: {e}. \n")
print(f"({np.random.choice(happy_message)}) You just verified fill number {fill_number}.")
Number of common anomalies in Channel 0: 7
Difference1: 488.4927978745, Difference2: 1.0; Difference3: nan, Difference4: nan
Channel 0 has passed inspection.
Number of common anomalies in Channel 1: 5
Difference1: 5243.9379918100, Difference2: 1.0; Difference3: nan, Difference4: nan
Channel 1 has passed inspection.
Number of common anomalies in Channel 2: 4
Difference1: 282.63196888226, Difference2: 0.6666666666666666; Difference3: nan, Difference4: nan
Channel 2 has passed inspection.
Number of common anomalies in Channel 3: 7
Difference1: 2251.2371294346897, Difference2: 4.0; Difference3: nan, Difference4: nan
Channel 3 has passed inspection.
Number of common anomalies in Channel 4: 0
```

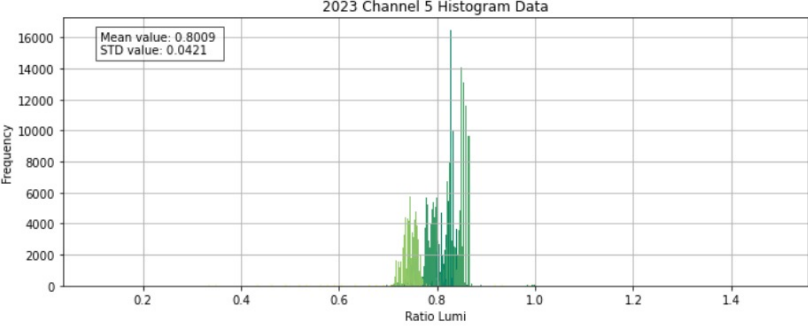
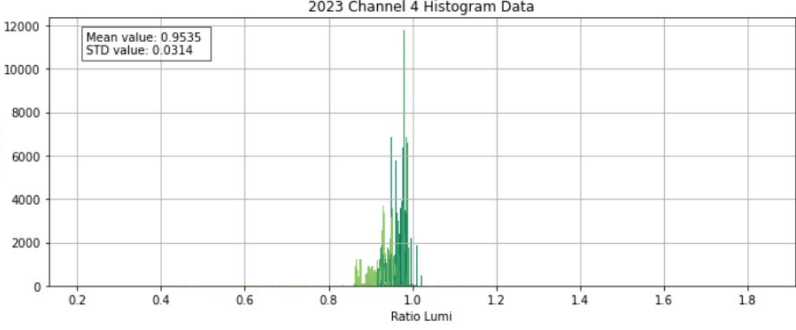
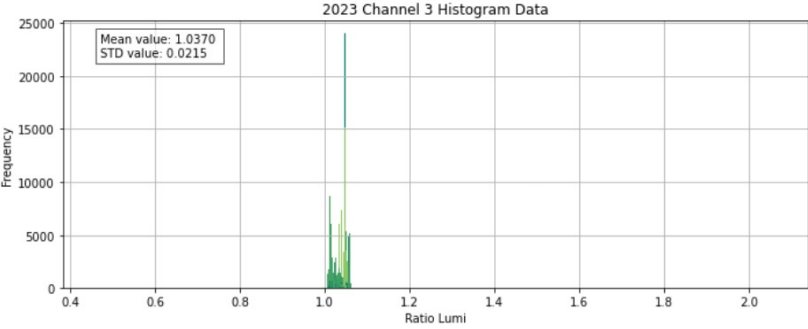
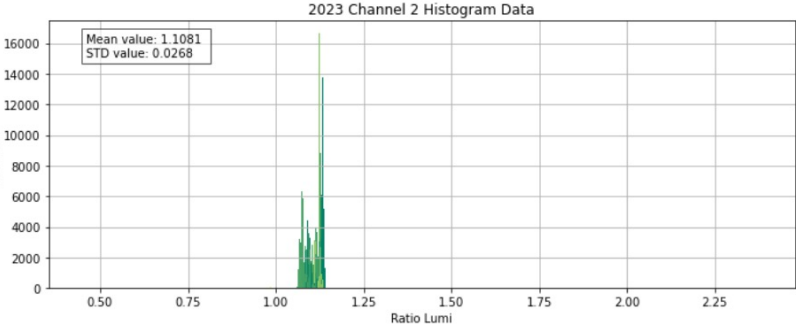
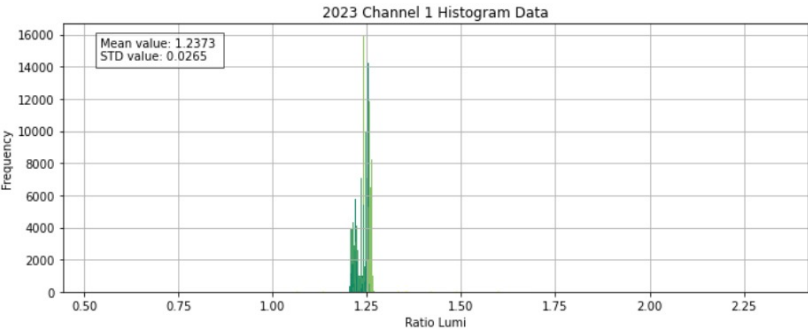
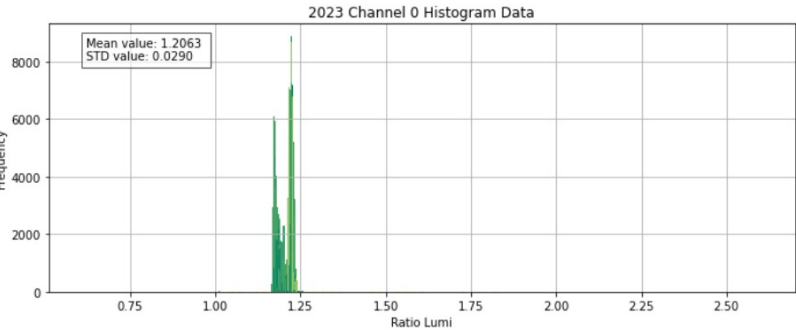

Results

- Look at the overall histograms (next slides) for each channel and in addition information about all fills examined
- Note: channels 6, 8, 9 are known to be bad so we skip looking at them
- In total 40 fills have been verified for 2023 PLT data

PLT Channel Histogram Data

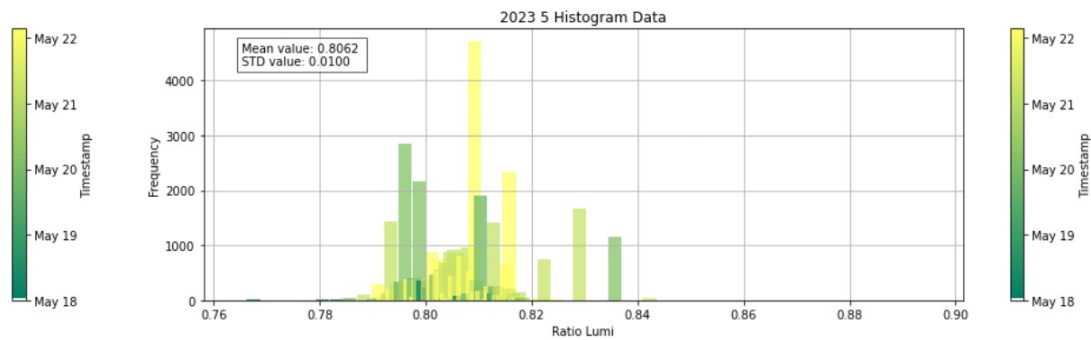
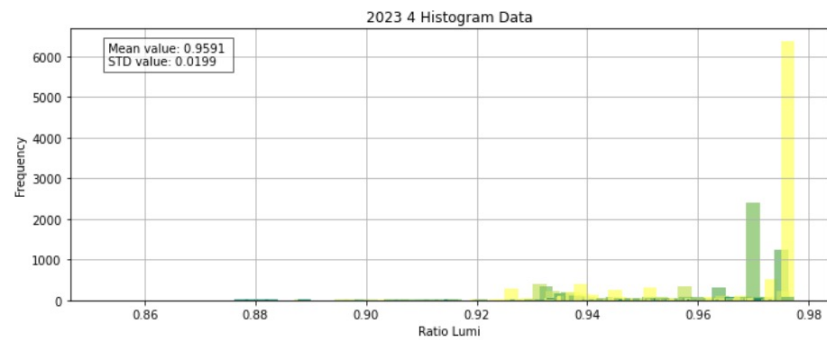
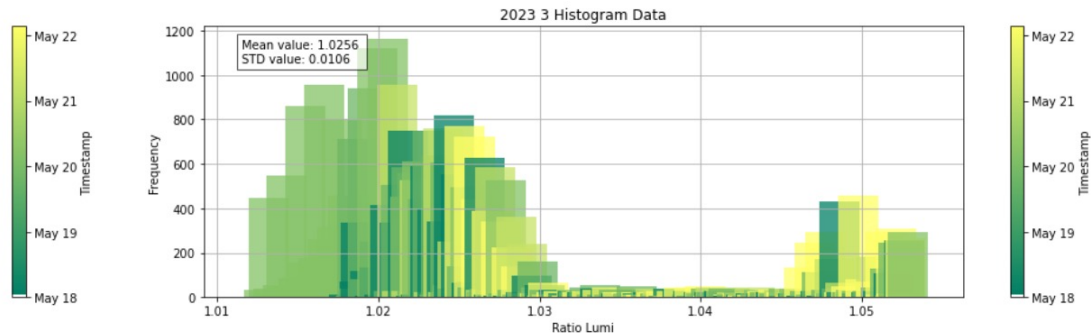
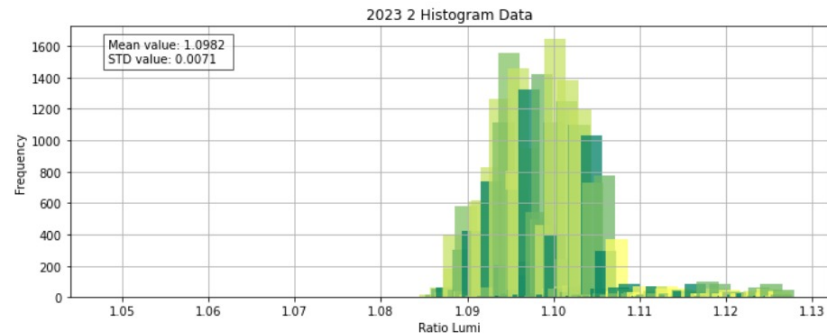
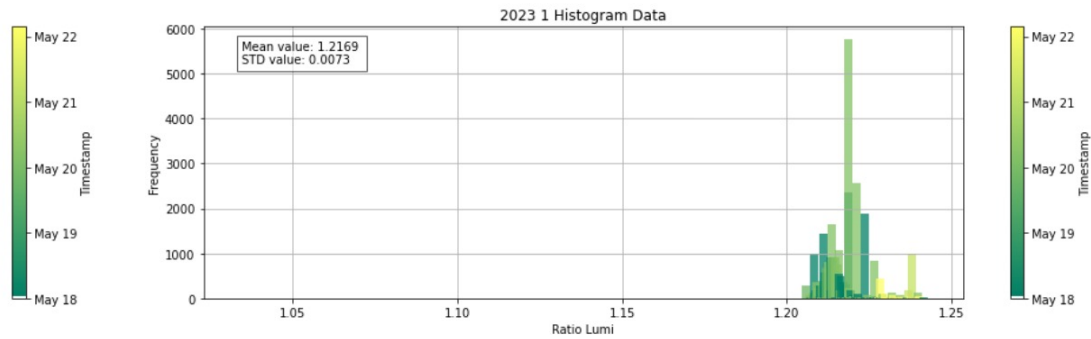
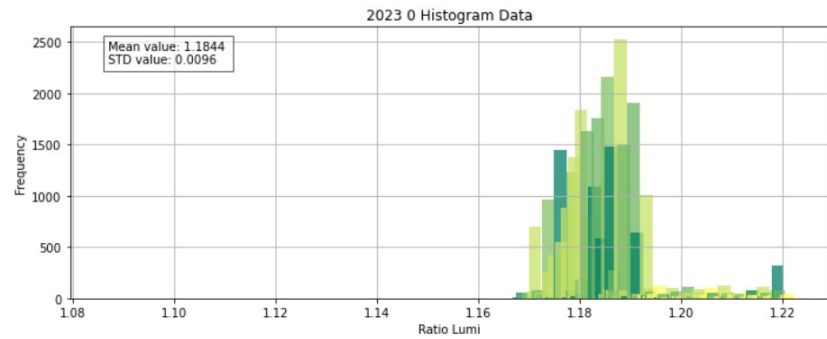
	Mean	Std	Spread	Std Low	Std High	Points Above 1 Sigma	Points Below 1 Sigma	Percent Error	Meets 1 Sigma Expectation
0	1.206005	0.022000	0.099217	1.184004	1.228005	68231	128258	40.190101	No
1	1.236973	0.018575	0.082032	1.218399	1.255548	84810	119969	41.883948	No
2	1.107874	0.020301	0.086400	1.087573	1.128175	86577	94595	37.052747	No
3	1.036991	0.014791	0.065520	1.022200	1.051782	72784	123076	40.049239	No
4	0.953811	0.027954	0.225678	0.925857	0.981766	59834	70306	26.634140	Yes
5	0.801124	0.039704	0.202342	0.761420	0.840828	106145	112757	44.800874	No
7	0.595441	0.030815	0.177896	0.564626	0.626256	122052	84340	42.229752	No
10	1.295180	0.016851	0.100340	1.278329	1.312031	88079	70424	32.405882	No
11	1.278451	0.020782	0.081375	1.257668	1.299233	96179	84794	37.009172	No
12	1.180223	0.021358	0.086059	1.158865	1.201581	94841	116781	43.274447	No
13	1.207884	0.027760	0.165793	1.180124	1.235645	83452	88160	36.043249	No
14	0.947960	0.006992	0.068808	0.940968	0.954952	32105	16632	9.966096	Yes
15	0.948098	0.012585	0.078030	0.935513	0.960683	39203	6176	9.279428	Yes

Results



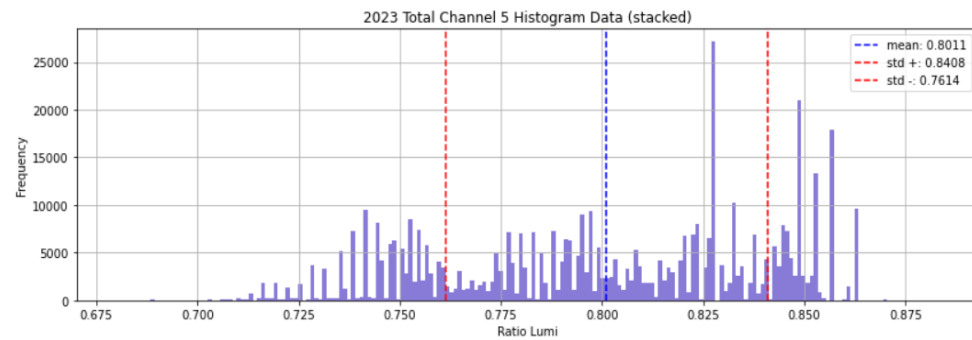
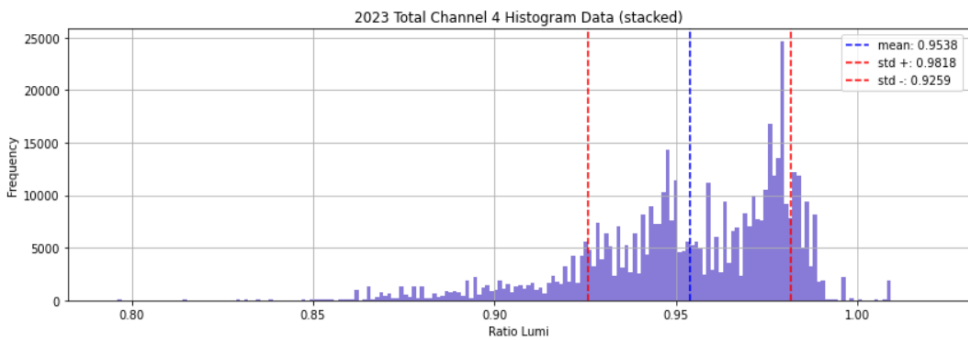
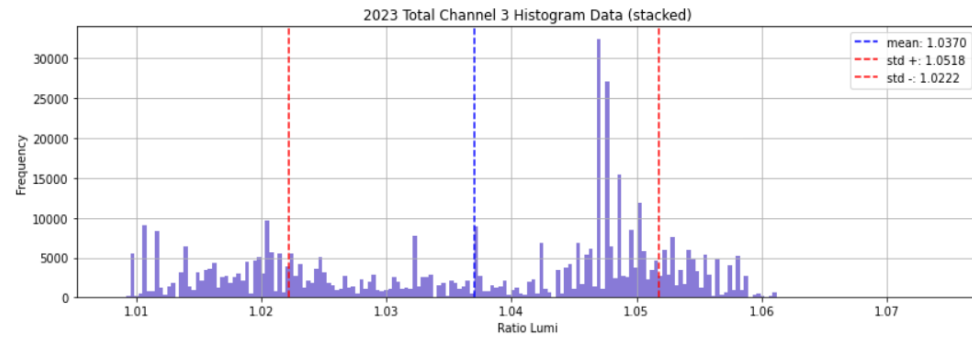
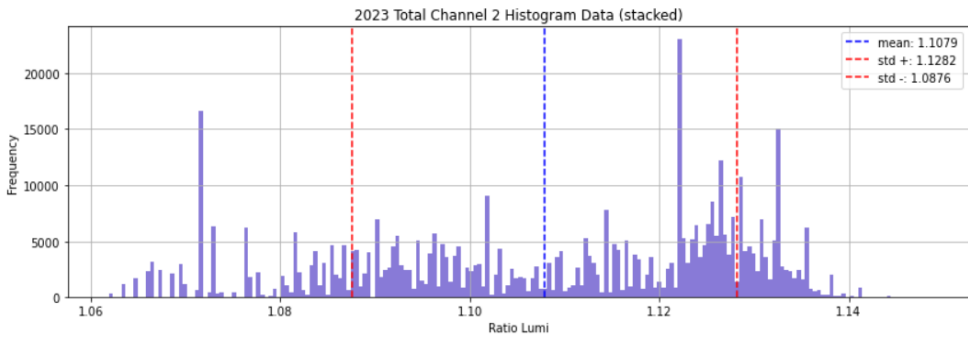
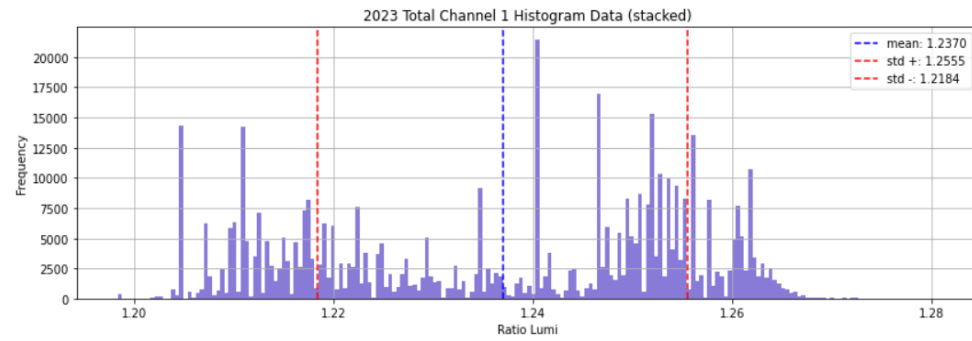
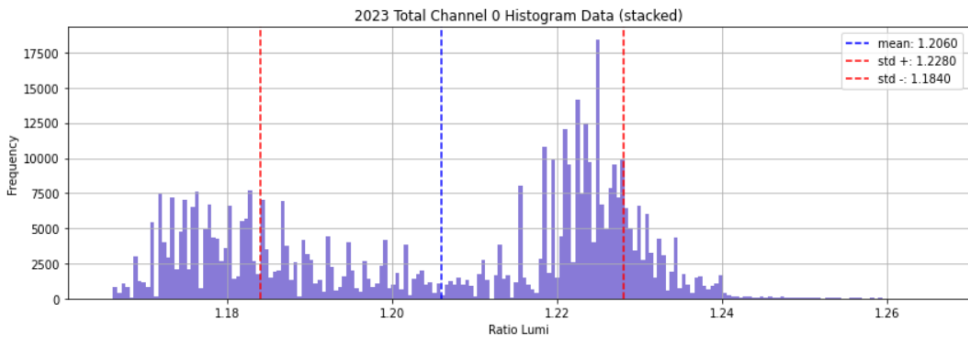
note only channels 0-5 shown, rest are in backup slides

Results



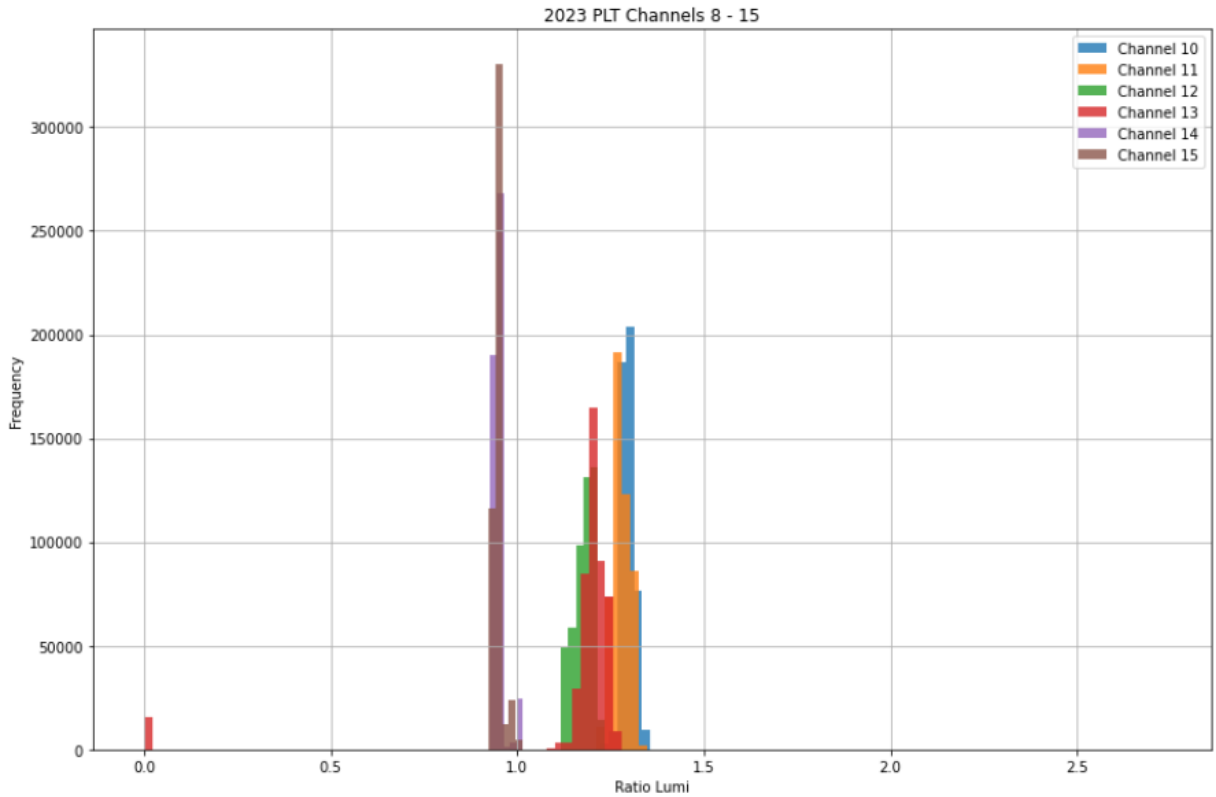
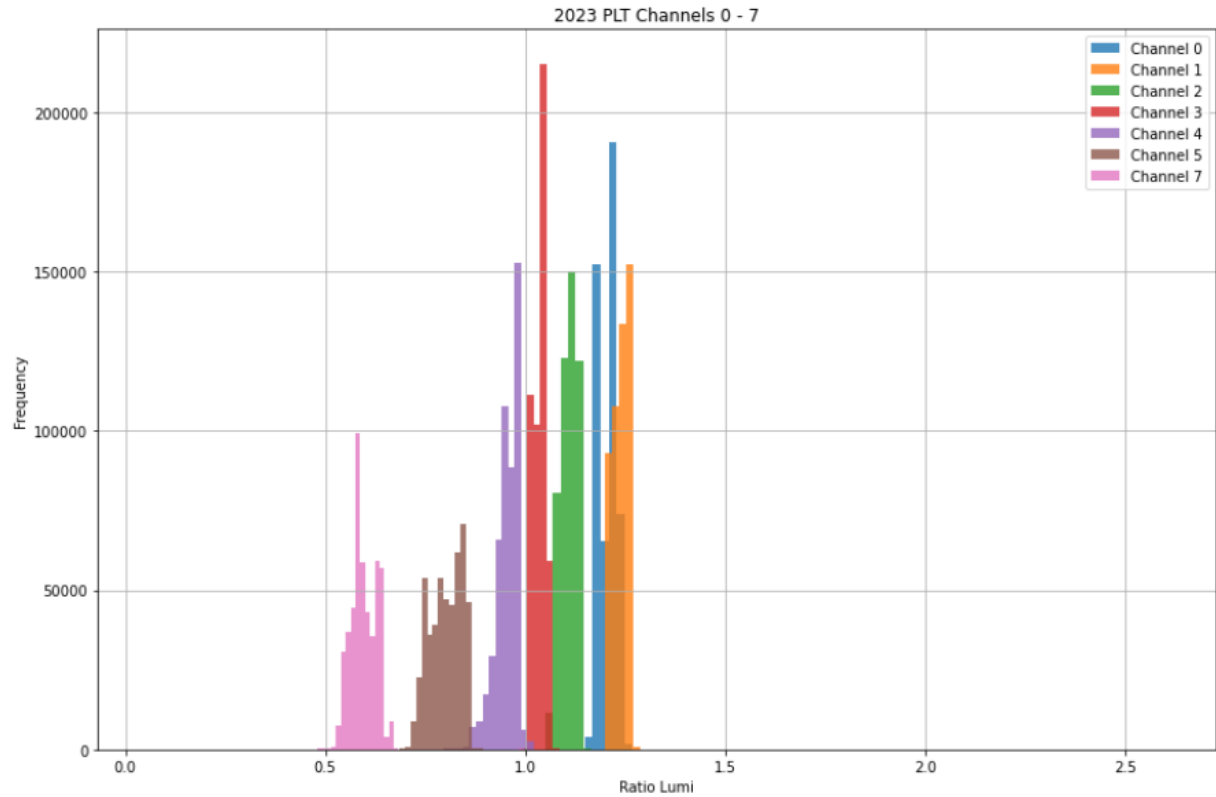
displaying the 5 most recent fills around the just studied fill to examine how data has shifted in a shorter time scale

Results



note only channels 0-5 shown, rest are in backup slides

Results

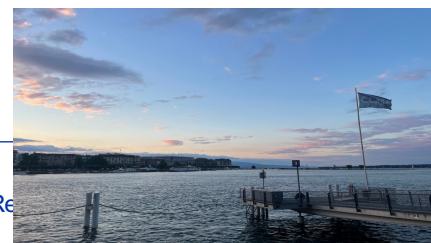
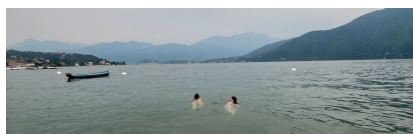


Conclusion

- The histograms match what is expected for PLT behavior
 - As a note, PLT channels shift around much more by comparison to HF or BCM1F luminosity data, thus fitting a gaussian is nearly impossible on the histograms shown previously
- The algorithms accurately detect anomalies in the data for channel selection and will aid in improving integrated luminosity measurements for CMS
- A similar algorithm for BCM1F has been created and is ready to work on BCM1F data
- GitHub page for more specific information: <https://github.com/katieream/byebyebyeBRIL>

Thank you to the NSF grant number 2243608 and University of Michigan for organizing the CERN REU program, Andres, Francesco, and the entire BRIL group I've worked with!

summer in a summary :)



Backup

