Nuclear Physics Computing: Using ROOT to Speed Up Data Analysis

Sebastian Alvis
University of Washington
Majorana
16 August 2018

Outline

- ROOT
- The Majorana Experiment
- My Work

Outline

- **ROOT**
- The Majorana Experiment
- My Work

ROOT: What is it?

- A modular scientific software framework
- Written by and for CERN, but available publicly
- Written in C++, but usable in other languages, such as Python, R, Mathematica, Javascript, and Spark
- Used by most experimental nuclear / particle physicists

ROOT: What is it good for?

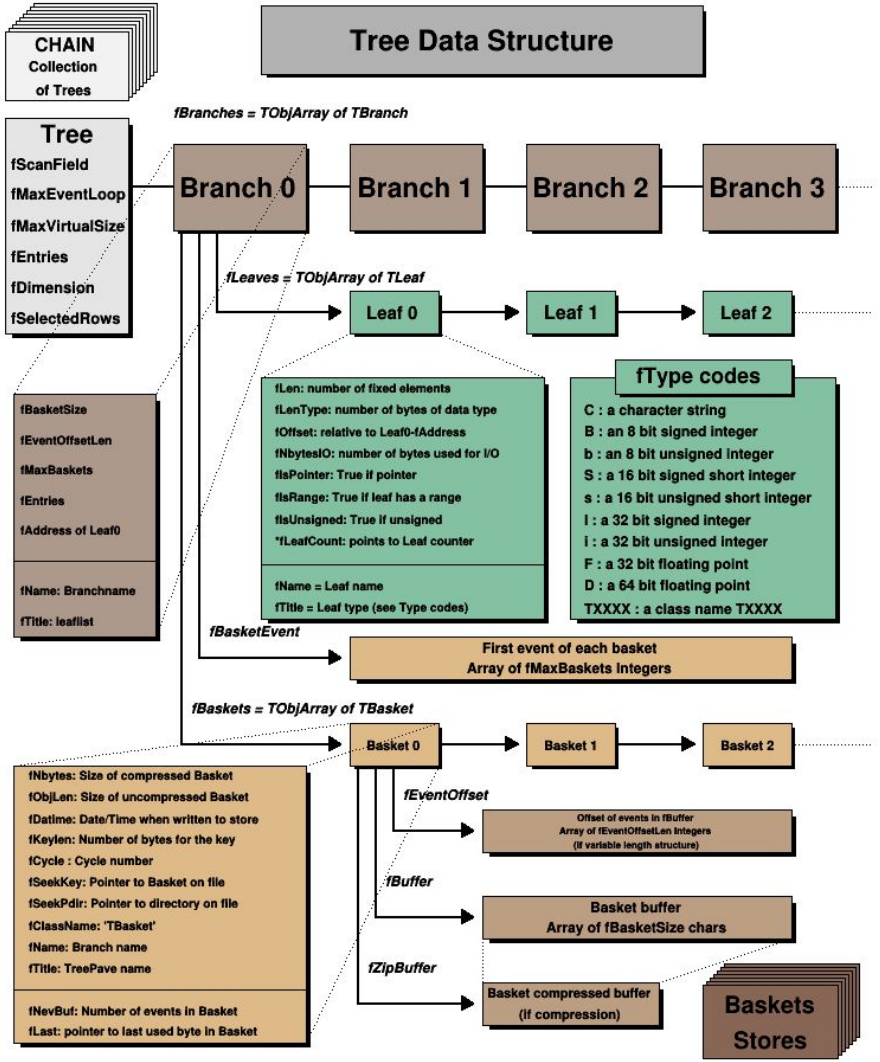
- Data Storage
  - .root - Compressed Binary Files
- Run Simulations
  - Lots of statistical distributions built-in
- Data Analysis
  - Heavy use of parallel processing (esp. with C++)
- Data Visualization
  - Many histogram / graph classes, fitting functions, GUIs
- All of these are supported by...

ROOT: Data Structures

- TTree: most common data structure
  - Stores data in parallel with TBranches
  - Can store other objects (histograms, canvases, other trees)
  - TChain: group of TTrees with the same layout
    - Allows for easier data processing
- TTreeReader: Reads tree values quickly
  - Store in TTreeReaderValue or TTreeReaderArray<>
- TH1 and TGraph: Histograms and graphs: data visualization in up to 3D
  - Options for granularity / bin size, data size (int / float / double), errors, overlaying
  - Fitting with TF1 on Hists and Graphs
- TCanvas / TPad: organization of plots
- TF1 and TMath: functions and mathematics
- ClassDef/ClassImp: generic object storage, streaming, reflection

ROOT: TTrees

- Image from the ROOT website
- Many features that I don't use
- TLeaf
  - Carries information about the branch data (data type, num elements, isUnsigned, isPointer, etc)
- TBasket
  - Manages TBranch buffers
  - Holds compressed and uncompressed branch size, time of data storage, pointers to directory, etc



ROOT: TTrees but Easier

TTree* myTree			
Name of TBranch*	fEnergy (double)	fWaveforms (vector <double>)	detectorName (TString)
event0	1000.0	&	P34A21
event1	2614.5	&	B2930
event2	1320.1	&	P26A01
event3	4020.0	&	P34A21

- Store data branch by branch (increases compression)
- Loop through indices with TTreeReader

ROOT: TTrees but Easier II

TTree* myTree			
Name of TBranch*	fEnergy (double)	fWaveforms (vector <double>)	channel (int)
event0	1000.0	&	660
event1	2614.5	&	660
event2	1320.1	&	1124
event3	4020.0	&	598

- Draw to histograms with TStrings

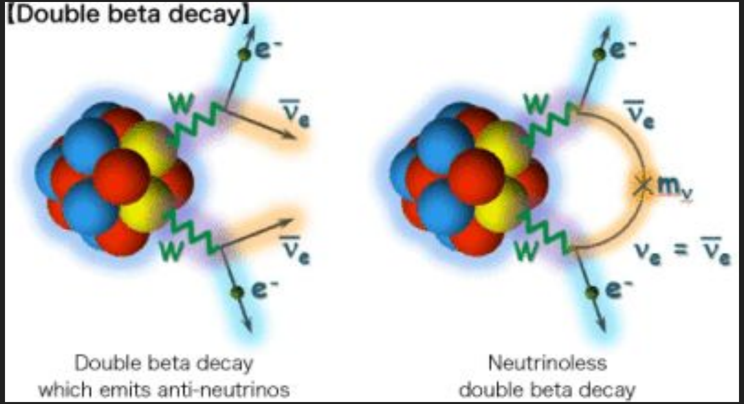
```
cutStr.Form("fEnergy > 1500 && channel == 660");
drawStr.Form("fEnergy>>hist0");
myTree->Draw(drawStr, cutStr);
```

Outline

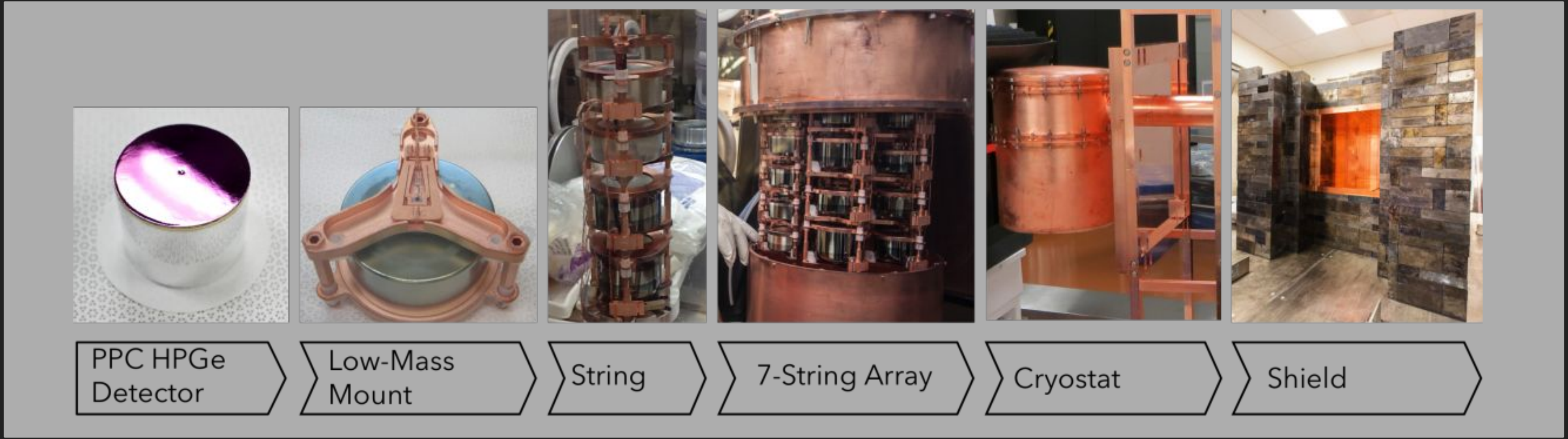
- ROOT
- **The Majorana Experiment**
- My Work

Majorana: What is it?

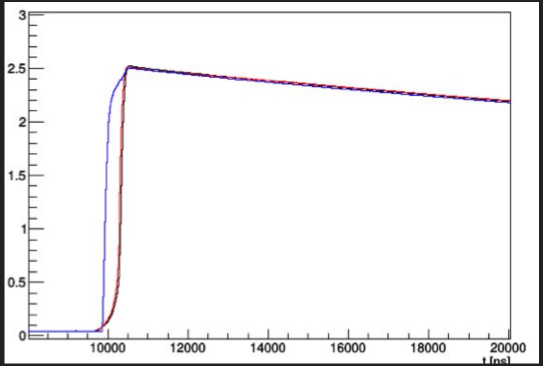
- Search for a matter creation process
  - Humans have never observed the creation of matter without balancing antimatter
  - We live in a matter-dominated universe, why are we here?
- Potential candidate: neutrinoless double beta decay ($0\nu\beta\beta$)
  - 2ν double beta decay is an observed phenomenon
  - “Leptogenesis:” Big Bang model predicts version with no neutrinos
  - ^{76}Ge (Majorana), ^{136}Xe (EXO), ^{130}Te (CUORE), et al.
  - First group to find it ---> Nobel Prize!!!
    - If it exists



Majorana: What do we do?



- Array of enriched and natural Ge detectors
- Wait for Ge to decay: creates pulses in detector
- Analyze data



Outline

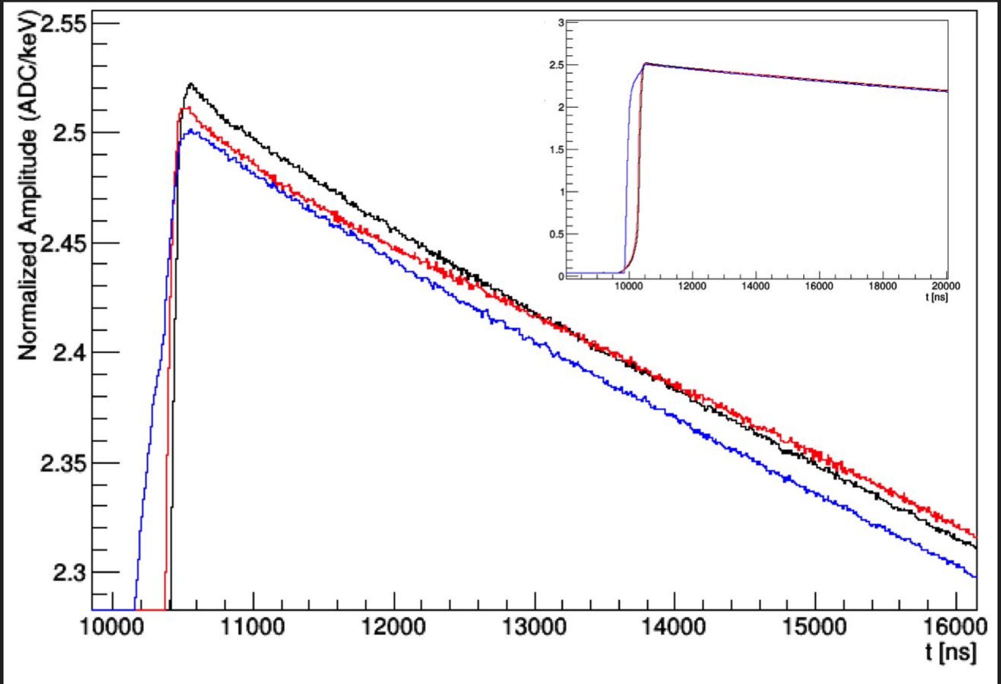
- ROOT
- The Majorana Experiment
- **My Work**

My Work: What's the problem?

- One surface of the detectors is susceptible to alpha particles
- The interaction between them causes charge trapping and re-release
  - Known as Delayed Charge Recovery (DCR)
  - Degraded energy
  - Right on our region-of-interest (ROI)
- Need a way to identify / reject these DCR events

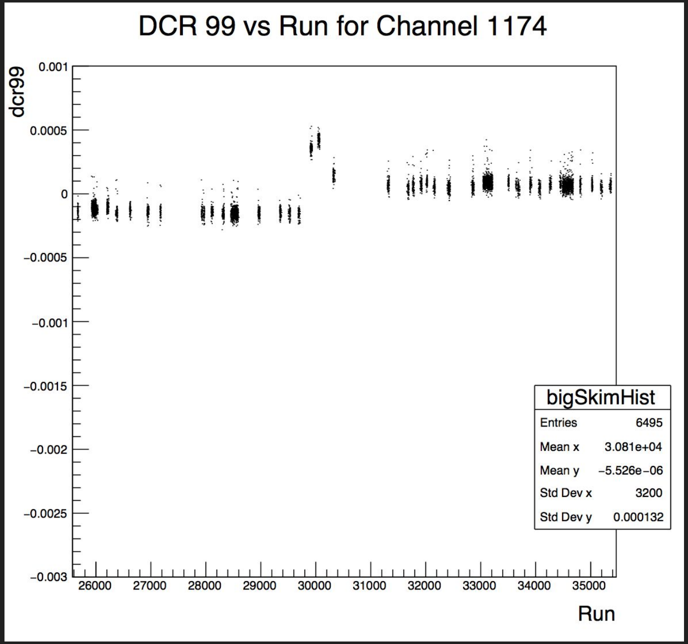
My Work: Others' Solution

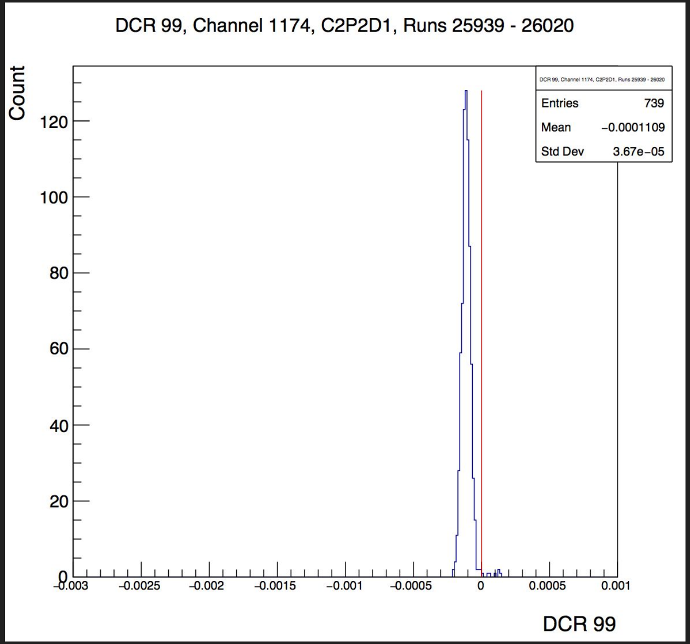
- The charge re-release has an effect on the waveform
  - Higher tail slope
- Characterize the tail slope for each detector
- Convert the tail slope into a universal parameter -> dcr99
  - The 1% cut value is set via the rate of calibration events and expected rate of alphas
- Reject events with dcr99 > 0



My Work: Monitoring DCR

- Easiest way to watch DCR stability is to plot dcr99 over time for each detector
- Also plot each 1D hist of dcr99 for calibration runs





My Algorithm: Collect Existing Data

- Use nested vectors to keep track of the runs I need
  ```
  vector < vector < unsigned int > > runRanges(52, vector<unsigned int>(2));
  ```

- Add runs to large TChain
  ```
  TChain* skimChain = new TChain("skimTree");
  for (int i = 0; i < numRuns; i++) {
          for (int j = runRanges[i][0]; j < runRanges[i][1]; j++) {
                  skimFile.Form("%s/skimRun_%d.root", folder, j);
                  skimChain->Add(skimFile);
          }
  }
  ```

My Algorithm: Trim the TTree

TTree* myTree						
Name of TBranch*	fEnergy	channel	isGood	dcr99	run	EventDC1Bits
event0	1000.0	660	1	0.01	36000	0
event1	2614.5	660	1	-0.015	36000	0
event2	1320.1	660	0	-0.021	36000	1
event3	4020.0	1124	1	-0.034	36000	2

- Ex. Don’t need “EventDC1Bits,” turn it off to make the TTree smaller and faster
- I turn off all 69 branches, and then turn on the 12 I need

My Algorithm: Trim the TTree

TTree* myTree						
Name of TBranch*	fEnergy	channel	isGood	dcr99	run	EventDC1Bits
event0	1000.0	660	1	0.01	36000	0
event1	2614.5	660	1	-0.015	36000	0
event2	1320.1	660	0	-0.021	36000	1
event3	4020.0	1124	1	-0.034	36000	2

- Ex. Don’t need “EventDC1Bits,” turn it off to make the TTree smaller and faster
- I turn off all 69 branches, and then turn on the 12 I need

My Algorithm: Draw to Hist

TTree* myTree

Name of TBranch*	fEnergy	channel	isGood	dcr99	run	EventDC1Bits
event0	1000.0	660	1	0.01	36000	0
event1	2614.5	660	1	-0.015	36000	0
event2	1320.1	660	0	-0.021	36000	1
event3	4020.0	1124	1	-0.034	36000	2

- myTree->Draw(“dcr99:run>>hist0”,”isGood && fEnergy > 2605 && fEnergy < 2635 && channel == 660”);
- hist0->ProjectionY(bin1, bin2);

My Work: Lots of Data

- The code runs on each detector independently
  - ~40 detectors (high and low gain on each)
  - The main dataset (~year) has ~12000 runs
    - I use ~2000 runs, or 2.5 TB of raw data
  - Each run’s TTree has 69 TBranches
    - I use 12 branches for results or cuts
  - Each run’s TTree can have 50k - 150k events
- Rerun every time I want to check a new set of calibration runs
  - Or any time someone else changes something at a lower level
- Runtime (if all jobs can get on in parallel) is about an hour
  - Each detector has 4k - 20k events that pass all the cuts

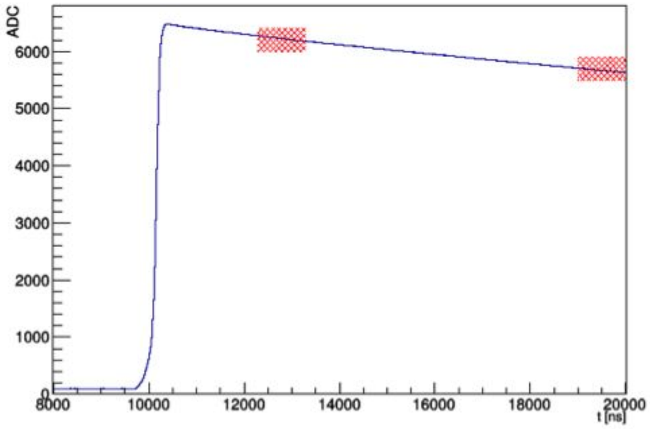
Summary

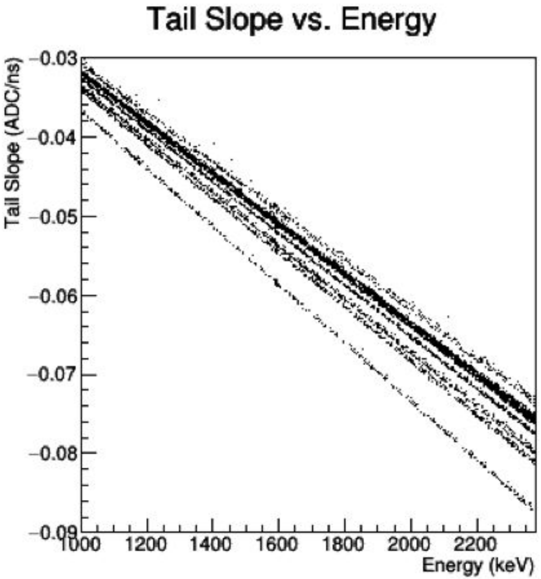
- ROOT is a data science toolkit used by nuclear/particle physicists
- Majorana is a $0\nu\beta\beta$ experiment in ^{76}Ge
- I worked on monitoring DCR for the interaction between a detector surface and alpha particles
- The script I made is highly reusable and efficient

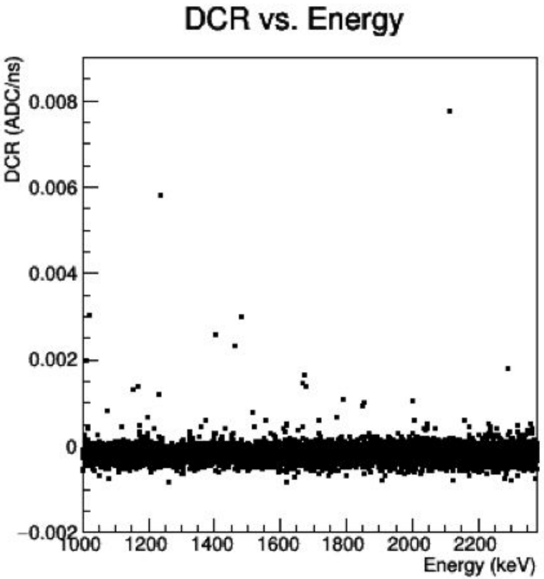
Backup: DCR Explanation

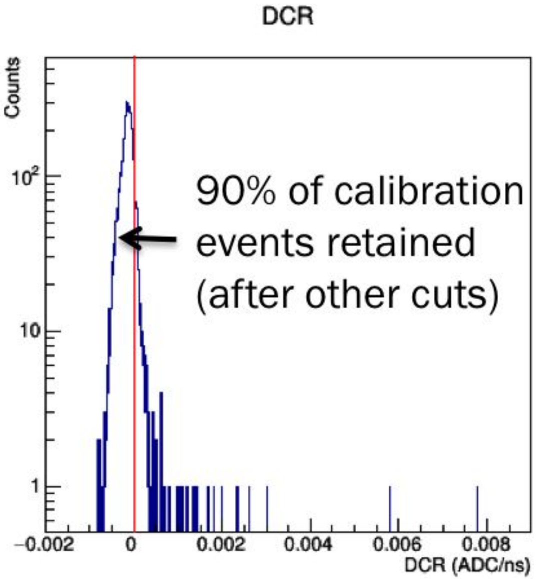
Calculate tail slope -> plot tail slope v energy ->

Remove energy scaling -> shift to cut a certain %



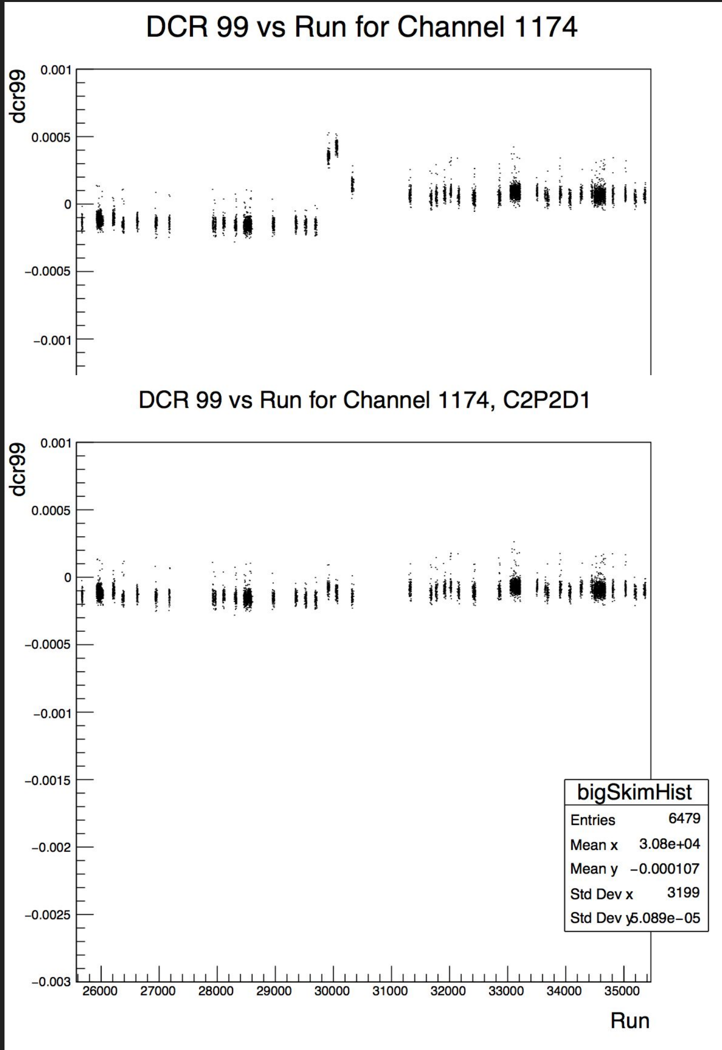






Backup: Retuning DCR

- Original tuning happens on the first long calibration run of the dataset, for highest statistics
- "Retuning" is the process of re-running the DCR analysis on new runs to realign distribution



Backup: DCR Efficiency

- Most DCR cuts weren't modified after the first long calibration run
- 3 detectors had been rebiased, changing their electronics' responses
- Retuning these worked perfectly
- Very accurate and stable cut level