2021/11/18

How-to guide for analysis of analogue output sensors in Lab and with TAF

Jerome Baudot

- Basic concepts in TAF
- RAW analysis
- Final analysis

IPHC Institut Pluridisciplinaire Hubert CURIEN STRASBOURG
cnrs Université de Strasbourg

Basic TAF

Install

- 2 GITs repositories synchronised:
  https://github.com/jeromebaudot/taf https://gitlab.cern.ch/bjerome/taf

- **Note for CE65:** Already installed on sbgat497 under **/home/alice/Software/taf/**

Documentation

- In distribution: **doc/taf_shortDoc.pdf**
- General page: http://www.iphc.cnrs.fr/TAF.html

Some concepts

- TAF identifies a sequence of data with a run-number
  - When running TAF, the run-number is associated with a given data-file (keep track of this matching)
- TAF behaviour is driven by a configuration file + some limited parameters
  - The same configuration file can be used for several runs
- There are two ways to analyse your data
  - **"RAW" analysis**: 1 click, only histos saved on ROOT file => output directory **Results/run_number**
  - **"Final" analysis**: first generate and store Tree of hits, then analyse the Tree info => output directory **results_ana_XX**

Preparing to run with TAF

- Steps are explained in the README file, reminded below
  (always assuming commands are launched from taf directory)

- Load environment variables once you start working with TAF
  - source Scripts/thistaf.sh
- Compile
  - maketaf
- Check TAF works
  - taf -h
- Check the config file
  - Usually, an expert has prepared a generic file for your sensor in directory **config_TEST**
  - User copies the generic file (from **config_TEST**) in directory **config** and needs to pay attention to:
    - Path to the data with key: **DataPath**
    - Number of events used to initialise noise with key: **InitialNoise**
    - Cuts to find hits with keys: **ThreshSeedSN**, **ThreshNeighbourSN**
    - Additional cuts used for the final analysis, see the section **Parameter for Analysis**
      - Especially the geometrical definition of submatrices

- Plot styles
  - Edit the rootlogon.C files under taf directory, however some styles are overridden in MRaw.cxx and Mpost.cxx

RAW analysis – menu based

Launching TAF

- If the datafile is known from the config file:
  taf –run [my-run-number] –cfg ./config/[my-config-file]
- If you want to specify a datafile different from the config (the path is still taken from the config file):
  taf –run [my-run-number] –cfg ./config/[my-config-file] –datafile [my-data-file] –gui
- **Note for CE65:** taf –run 2 –cfg ./config/ce65abc.cfg –datafile 2021-11-16-B1_source.root –gui
  - This links run-number 2 to data in file 2021-11-16-B1_source.root
  - There is also a ce65d.cfg config file if you read a sensor of type D (only one submatrix)

Menu based analysis

- The –gui option generates a clickable menu
  - You don't control the nb of events or the submatrix analysed, everything is hard-coded
  - You can change this by editing function PreparePost() in source code: code/src/Mraw.cxx
  - If you forgot the –gui option, you can generate the menu within TAF with: gTAF->GetRaw()
- Noise check: DISPLAY NOISE
- Looking at frames event by event: RAW CHANNELS 2D
  - Re-click the same menu choice to move to next event
- Displaying the raw spectrum: RAW SPECTRUM
- Investigating the hit properties: CUMULATE HITS 2D

RAW analysis – command based

Command line analysis

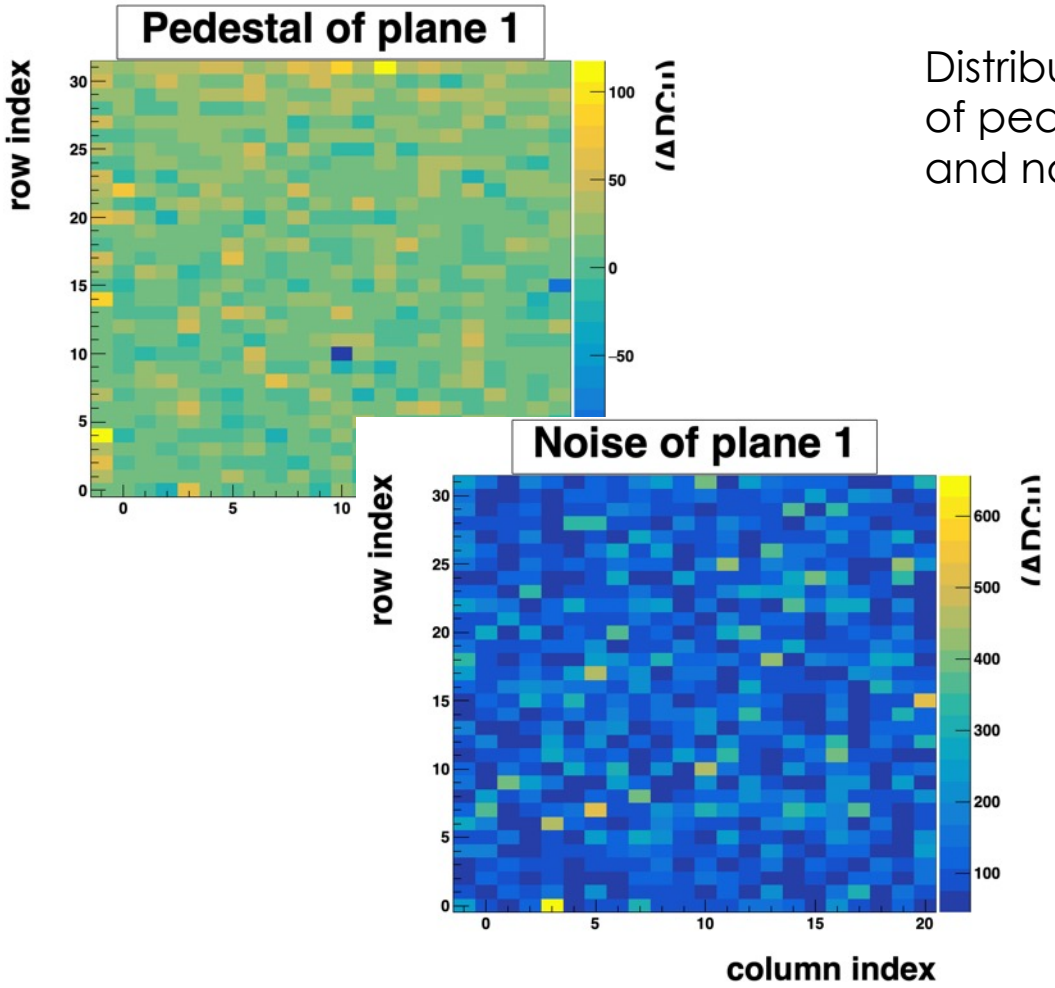
- All functions are coded in class **code/src/Mraw.cxx**
- Command line allows you to control parameters (#events, submatrix, …)
- Noise check:
  - gTAF->GetRaw()->DisplayNoise(colmin, colmax)
  - colmin, colmax (optional) select the submatrix based on column numbers
- Looking at frames event by event:
  - gTAF->GetRaw()->DisplayRawChan()
  - Re-click the same menu choice to move to next event
- Displaying the raw spectrum:
  - gTAF->GetRaw()->DisplaySpectrum(#events, 0, charge_min, charge_max, colmin, colmax)
  - charge_min, charge_max are only for the histo range
  - colmin, colmax select the submatrix based on column numbers
- Investigating the hit properties:
  - gTAF->GetRaw()->DisplayCumulatedHits2D (#events, 0, 1, #bins_position, xmin, xmax, ymin, ymax)
  - xmin, xmax, ymin, ymax select the submatrix based on hit position
  - Range of histo displaying charge with additional option: , 1, #bins_charge, charge_max)

Outputs

- In Menu or Command line all plots are saved in root format
- Output directory is **Results/[run_number]**
- Pay attention to the prints on the screen, usually the exact name of the files saved is indicated

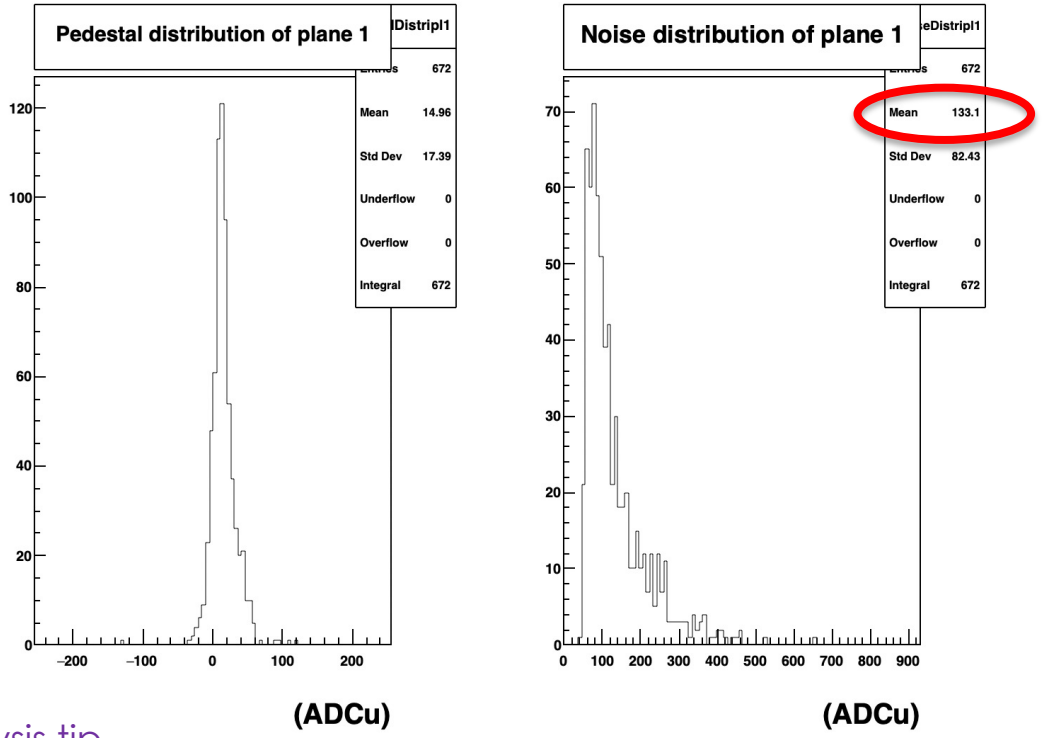
Raw analysis – Display Noise

Pedestal and noise are computed for each pixel individually over **InitialNoise** events



Maps of pedestals and noises

Distributions of pedestals and noises



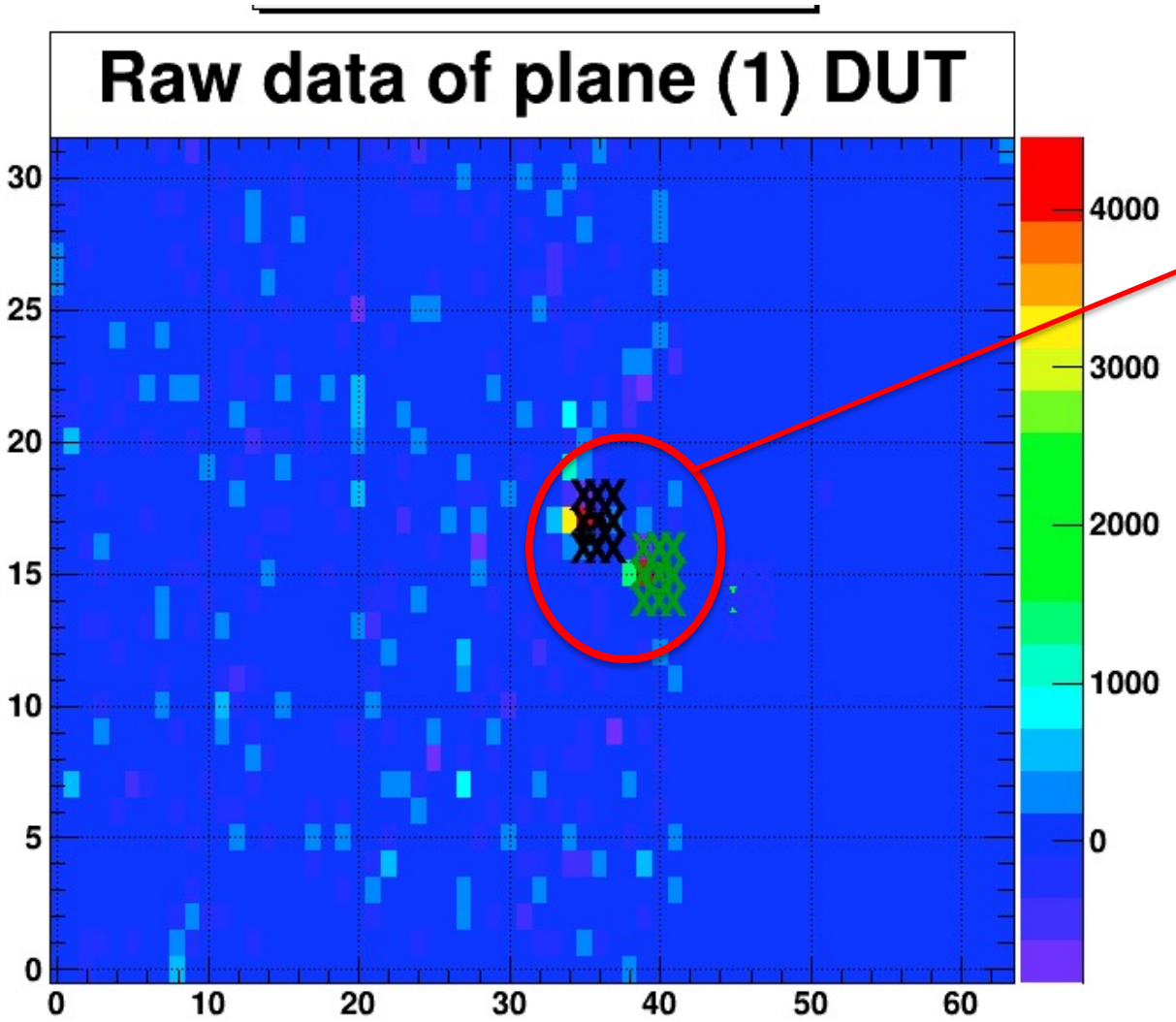
Analysis tip

Once the calibration factor is known,

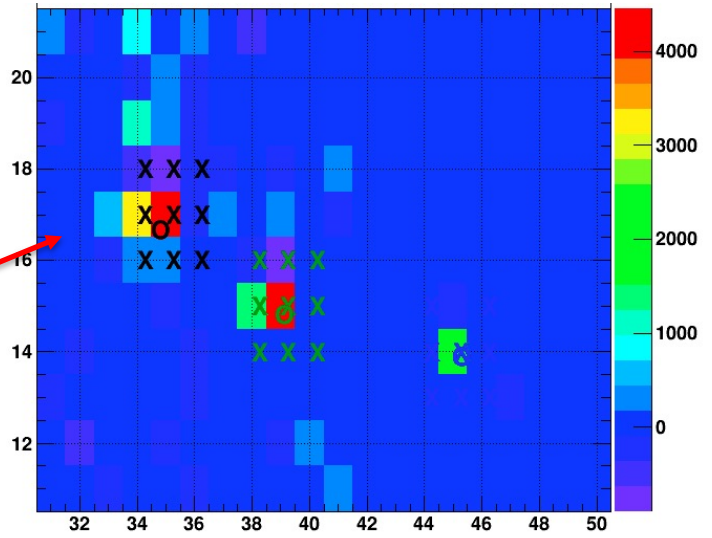
Raw analysis – Event by Event

Display the map of raw data, after CDS. If a hit is found, crosses indicate the pixels associated

Note: need to pass the #events required to initialise noise before starting to observer hits.



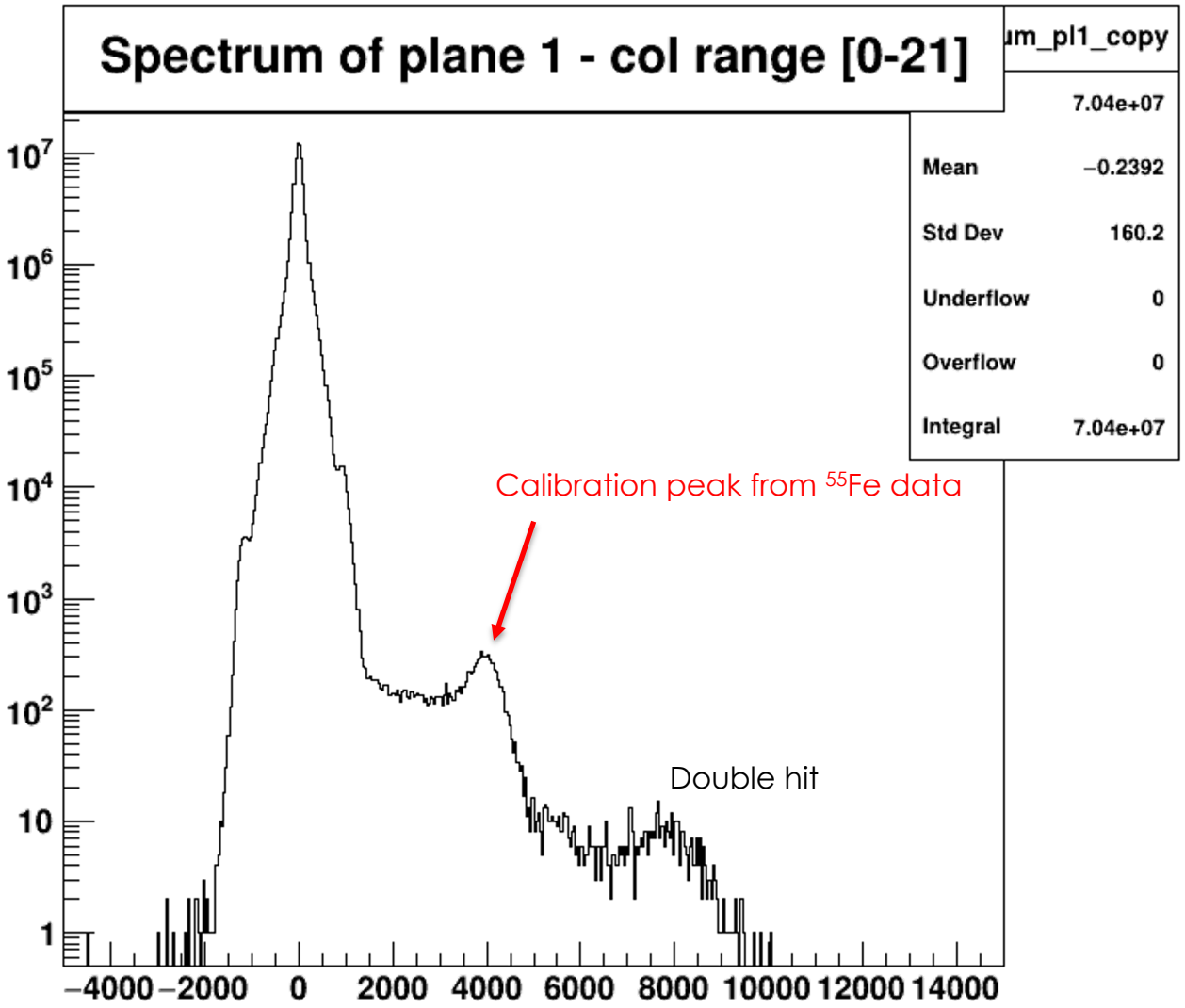
ZOOM



If a hit is found, crosses indicate the pixels associated to the hit. The circle indicates the seed pixel.

Raw analysis – Raw Spectrum

Distribution of raw data (after CDS) for all pixels over all events required.



Raw analysis – Hit Properties

Hit selected using cuts specified in config file:
ThreshSeedSN, ThreshNeighbourSN

Analysis tip

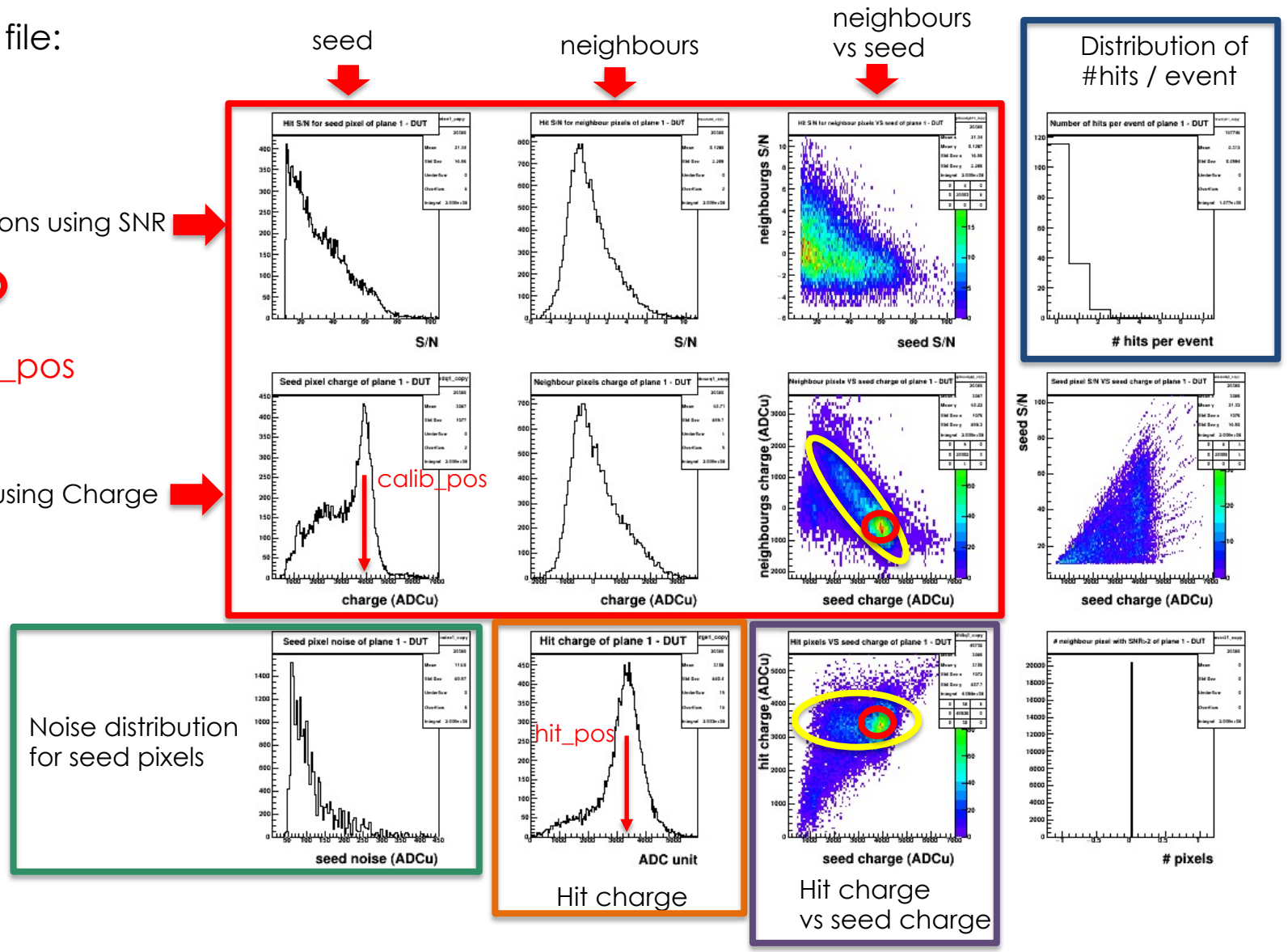
Here ^{55}Fe data are displayed:
- calibration peak position = calib_pos
- Signal of real hit
- Average charge collection per hit = hit_pos

Analysis tip

In case of ^{55}Fe data, we can estimate 2 quantities:

- The calibration factor from the calibration pea position:
e-/ ADCu = 1640 / calib_pos

- the "charge collection efficiency" (CCE), from the ratio of calibtration peak position and average hit charge:
CCE = hit_pos / calib_pos



Final analysis

First step, generate and store the TREE

- Commands:
  taf –run [my-run-number] –cfg ./config/[my-config-file] –datafile [my-data-file]
  gTAF->DSFProduction(#events)
- Hits are selected according to the cuts defined in the config file
- ROOT file stored in: datDSF/run2_01.root
  - TAF increments the file name automatically if you iterate DSFProduction
  - You can change the name to something comprehensible to you
- The TREE is readable with a Tbrowser, leaves definition in class: code/src/Devent.cxx

Second step, the analysis

- Launch TAF with the last automatically named DSF file (like datDSF/run2_03.root):
  taf -run [run_nb] -cfg ./config/[my_config.cfg]
- Or Launch TAF with a specific DSF file:
  taf -run [run_nb] -cfg ./config/[my_config.cfg] -dsffile datDSF/[my_DSFfile.root]
- Start the analysis for ^{55}Fe data used to calibrate with command:
  gTAF->MimosaCalibration(#events, minSNRseed, minSNRneighbour, maxChargeneighbourCalib, #submatrix, #geomatrix)
- Start the analysis with command for beam test data with command:
  gTAF->MimosaCluster(#events, minSNRseed, minSNRneighbour, #submatrix, #geomatrix)

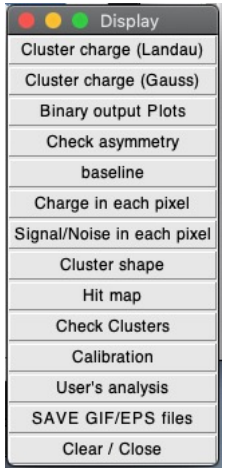
Final analysis

Cuts explanations

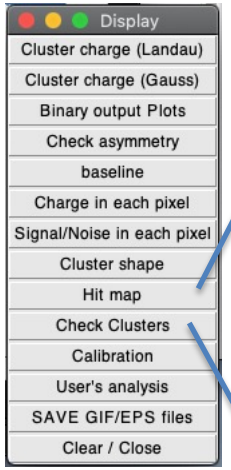
- **minSNRseed_cut**, **minSNRneighbour_cut** are the same type of cuts set in the config file but values can be tighter
- **submatrix#**, **geomatrix#** are the ones defined in the configuration files and set a region of interest
- For calibration plot only, **maxChargeneighbourCalib** is used to focus on the calibration peak
- New OPTIONAL cuts are possible and set in the config file (defined at submatrix level):
  - It might be surprising but the numbers **PixelsInRaw**, **PixelsInColumn** should stick to the size of the whole matrix
  - **MinSeedCol**, **MinSeedRow** allos to define a region of interest
  - **MinSeedCharge**, **MinClusterCharge**, **MinNeighbourCharge** cut on charge (in addition to SNR cuts)

Output

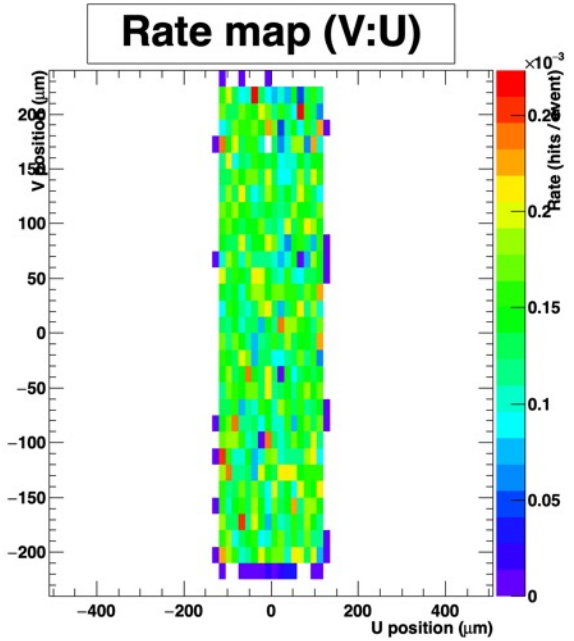
- Once all entries of the DSF file have been analyzed, a menu opens
- Each menu will display a set of plots (see next slides)
- All these plots are then saved in the output file: **results_ana_M[mimosa-type/run[run_nb]PI1_ClCharge.root**
  - **Note for CE65:** mimosa_type = 651 or 652



Final analysis: cuts and basic charges



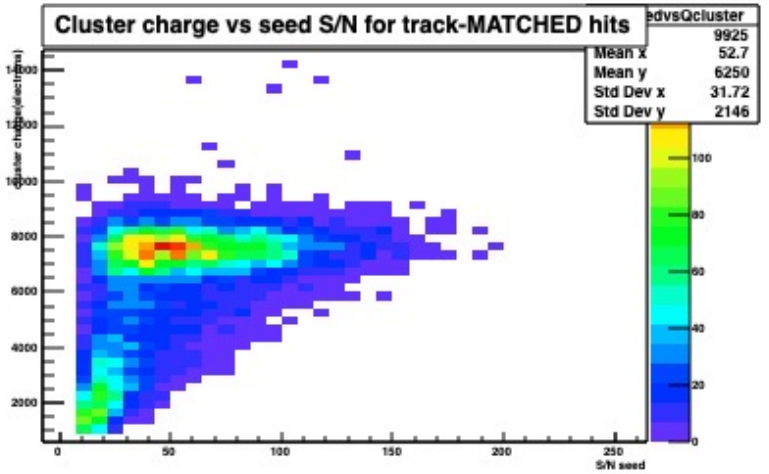
Various hit maps,
where you can check your area selection is effective



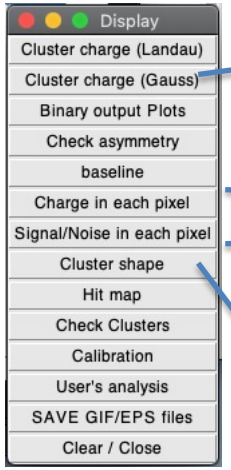
Various charges (seed, neighbours, hit) plots
allow to check signal presence and compute CCE

Analysis tip

Like in the RAW analysis, you will get the distribution of the hit and seed charge, from with you compute the charge collection efficiency:
CCE = hit_pos / peak_pos



Final analysis: cluster shape



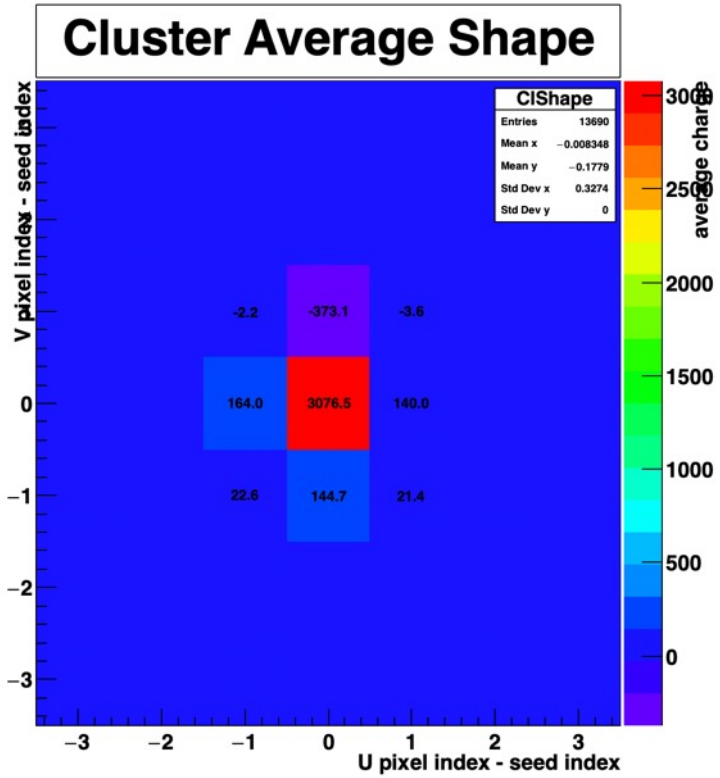
Order pixels in cluster by charge, and sum them in order to estimate how many pixels carry some significant signal.

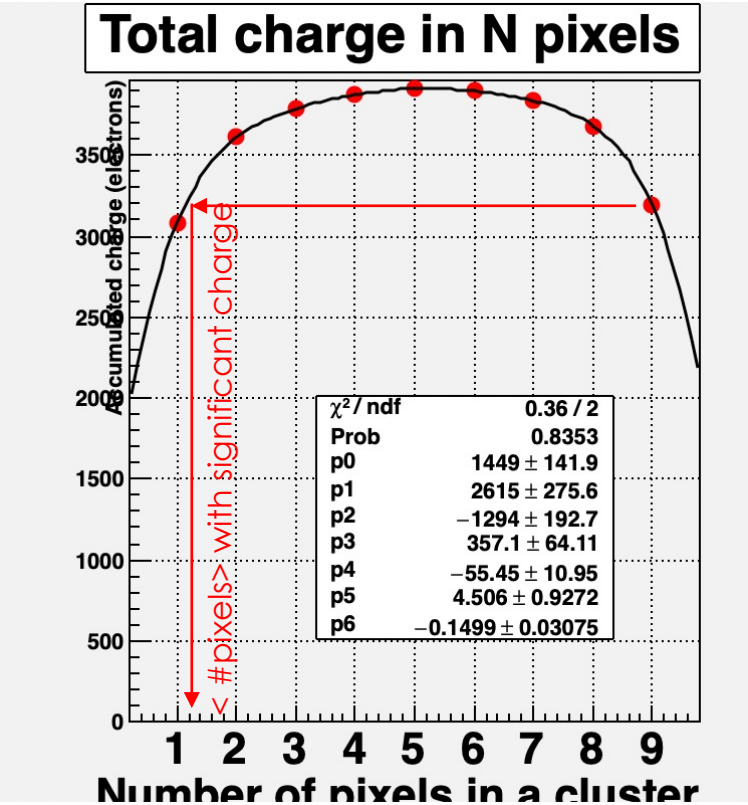
Study charge and SNR of pixels separately within a cluster

Plot average cluster shape =>

...

and also plot #pixels in cluster with a charge or SNR higher than predefined values (hardcoded in code/src/Manalysis.cxx ClusterShape_init())

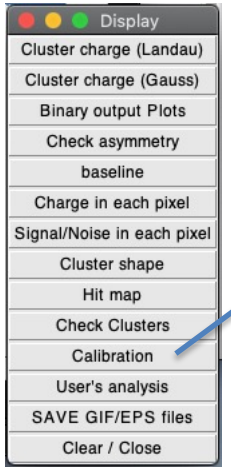




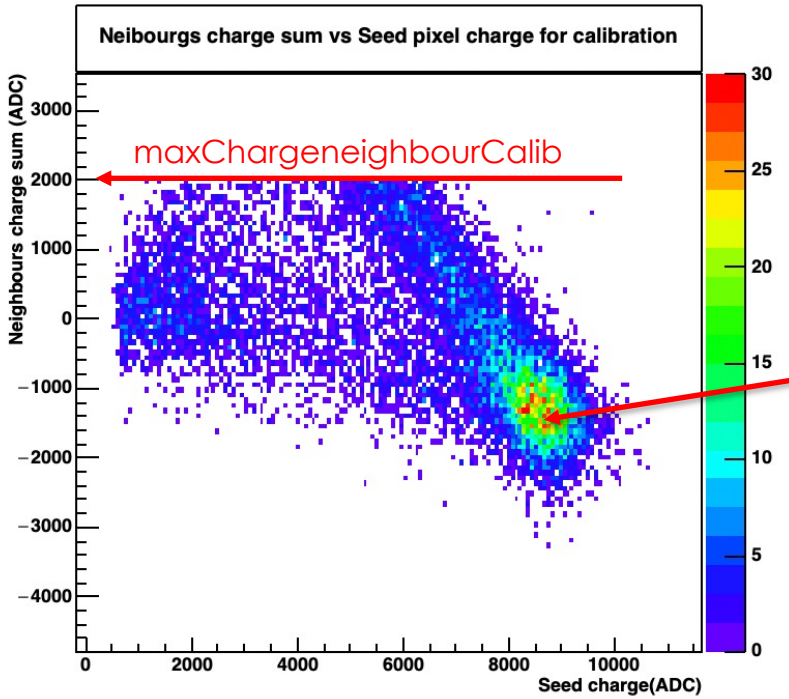
Analysis tip

You should be able to characterize the cluster size from these various plots.

Final analysis: calibration peak



Zoom on charge on seed with specific cut so that the calibration peak is well isolated



Calibration peak from ^{55}Fe data