smalldata_tools

Supporting LCLS data analysis

Silke Nelson, SLAC

With contributions from Henrik Lemke, Chris O'Grady, TJ Lane, Diling Zhu, Tim van Driel, Clemens Weninger and many more.

Outline

- Analysis at an FEL: boundary conditions
- Lessons from the past years of operation
- smalldata_tools
  - Default Data
  - Detector Treatment
  - Event binning
- Application Example
  - Timetool calibration
- Conclusions

Analysis at LCLS - Variety in many different ways

SLAC

- A big machine with a small number instruments, sending photons to one or two end stations at a time
- Large variations between experiments, even in ‘standard config’
- Data contains many different sources of data (60+ different data sources for LCLS-I, 10-20 per experiment)
- Optimal detector treatment can depend on physics signal
- Raw data typically on the order of 10s of TB
- User groups vary in size & (computing) expertise
- LCLS Computing resources are limited

What we have learned: Starting point

- File format based on system developed for HEP is used to achieve necessary data writing speed
- Analysis code had to be written in a (C++) framework
- Parallel offer: translation to hdf5 files
- Tension between ease of use, flexibility, efficiency and effort
- User written ‘frameworks’ have costs when devices/detectors are upgraded, a different instrument is used or the SLAC code stack changes
- Fast turnaround analysis
  - Code should be efficient and be used efficiently
- Consistency: same data, same name; same method, same code

What we have learned: Frameworks

- File format based on system developed for HEP is used to achieve necessary data writing speed
- Analysis code had to be written in a (C++) framework
- Parallel offer: translation to hdf5 files
- Tension between ease of use, flexibility, efficiency and effort
- User written ‘frameworks’ have costs when devices/detectors are upgraded, a different instrument is used or the SLAC code stack changes
- Fast turnaround analysis
  - Code should be efficient and be used efficiently
- Consistency: same data, same name; same method, same code

What we have learned: Starting point

- File format based on system developed for HEP is used to achieve necessary data writing speed
- Analysis code had to be written in a (C++) framework
- Parallel offer: translation to hdf5 files
- Tension between ease of use, flexibility, efficiency and effort
- User written ‘frameworks’ have costs when devices/detectors are upgraded, a different instrument is used or the SLAC code stack changes
- Fast turnaround analysis
  - Code should be efficient and be used efficiently
- Consistency: same data, same name; same method, same code

Requirements

- Users want:
  - Portable data and choice of analysis tools
  - Access to data quickly after a run
  - Simple, instrument agnostic setup

- Staff wants:
  - Conventions for data names
  - Standard methods
  - Predictable code behavior
  - Convenient way to analyze calibration runs

smalldata_tools: Overview

SLAC

Production

Event-based hdf5 production (smallData)

- Instrument-based default data
- Detector & physics-based data reduction

Detector-based hdf5 production (cube)

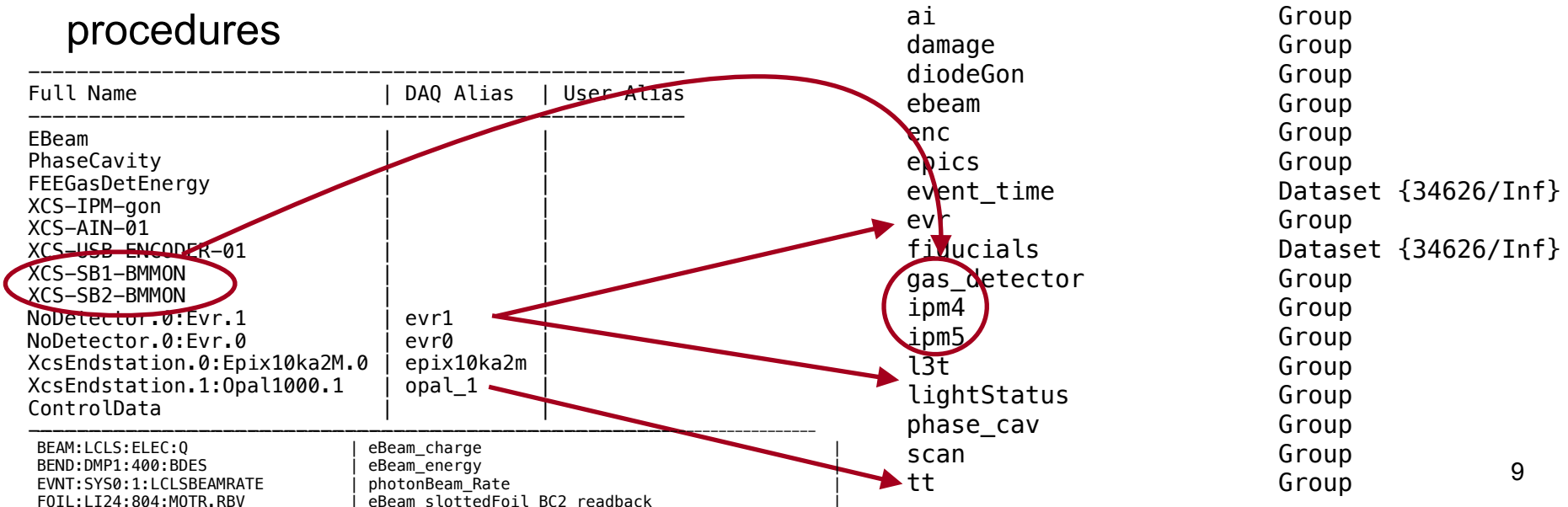
- Sum over selected events

Analysis

- Quick plots
- Tools to setup data reduction
- Multidimensional binning w/ filtering based on small data
  - Addition of big data
- Notebooks for standard procedures

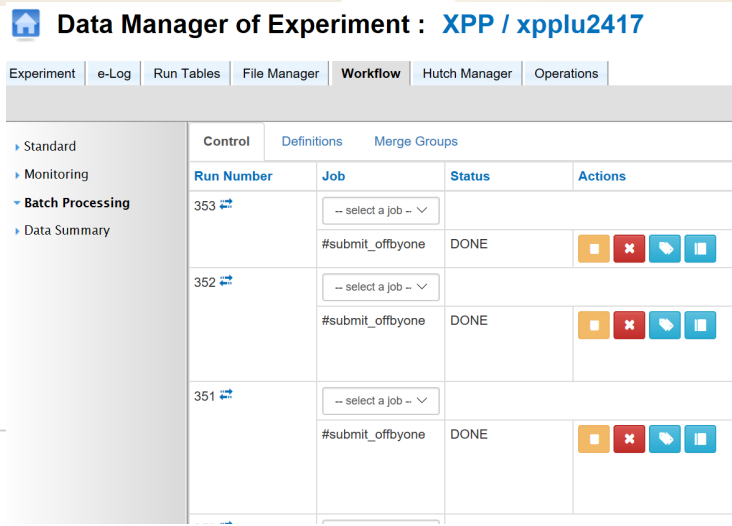
Instrument Based Default Data

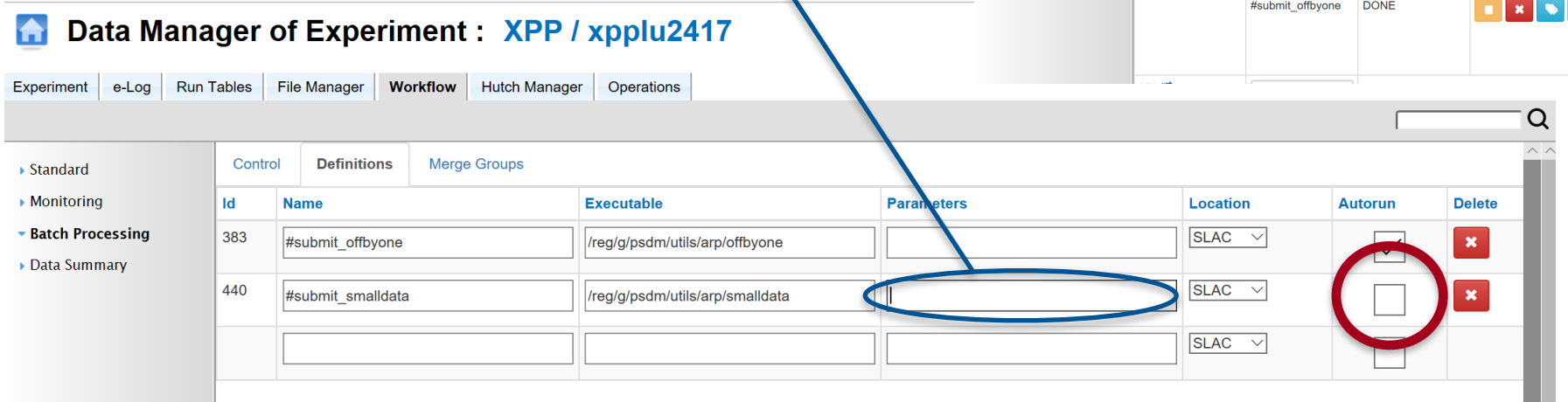
- Names in data files can be cryptic and often differ from names used by staff in discussions while data taking
- Only few variables of a few different detector types are important
- Extract meaningful variables from technical implementation of procedures



DataManager and Automatic Run Processing

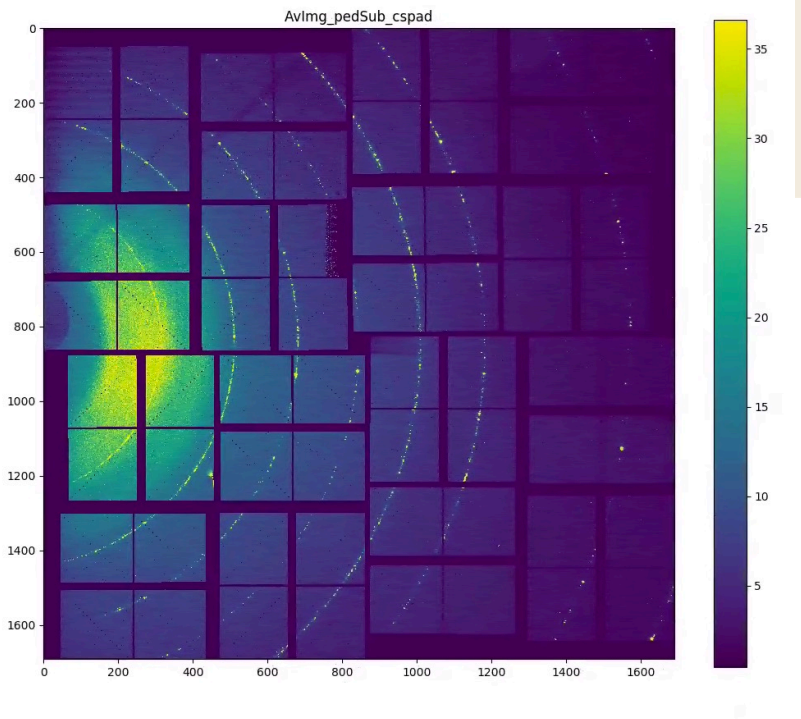
- Main interface of the user to the data
  - eLog for experiments
  - Run Tables
  - Definition of scripts to be executed for all runs
    - Pass potential user variation as parameter



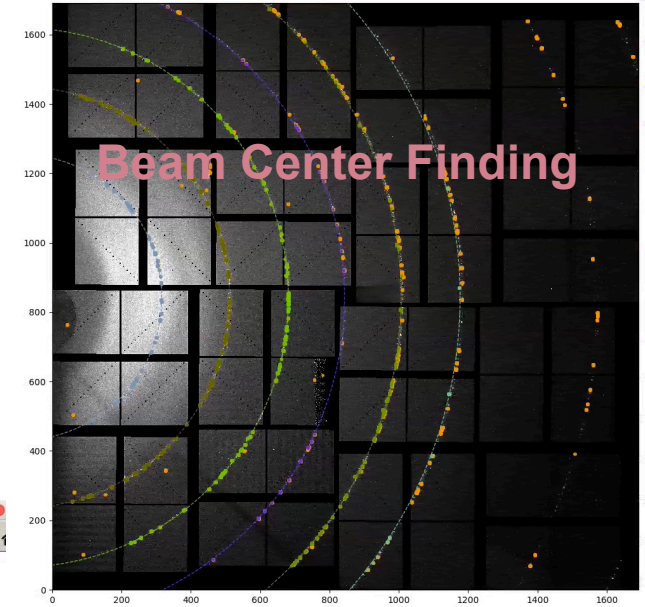


Data Reduction: DetObject

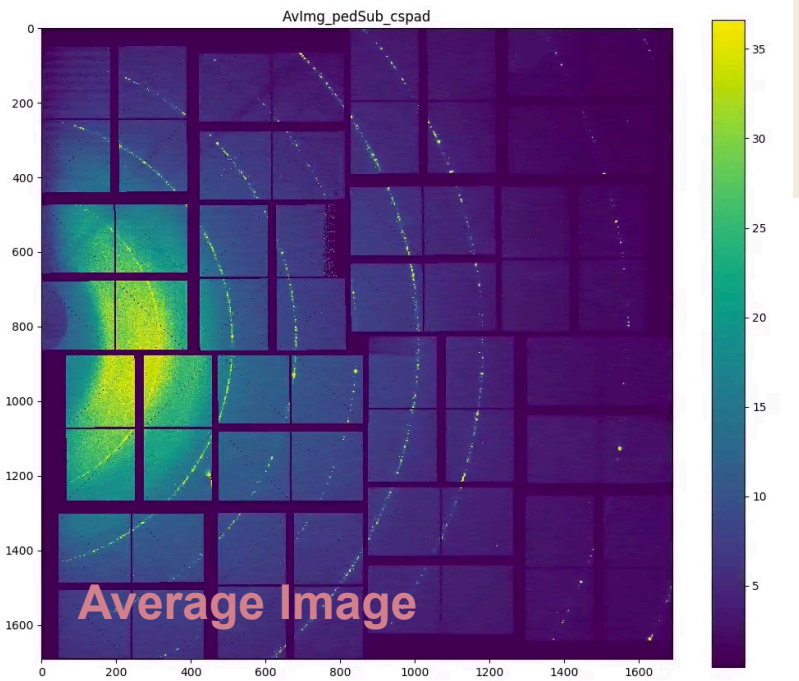
- Define detector object based on its name
- Add methods to be performed
  - ROI (sum, max, center-of-mass, area)
  - Binning, projection, images
  - Azimuthal averaging
  - Droplets (need sparse data)
  - Photon finding (need energy as input)
- Methods can be chained
- Save configuration of feature extraction and detector calibration information hdf5



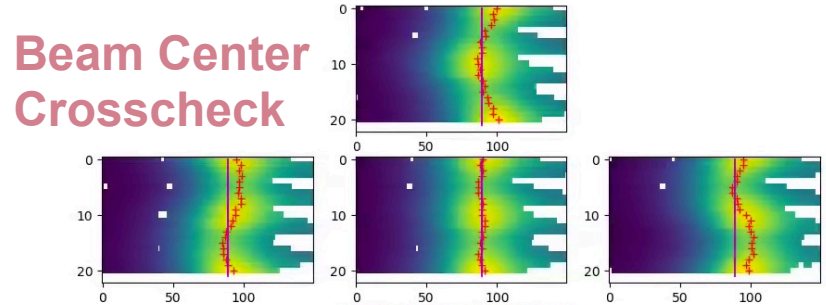
Data Reduction: Tools

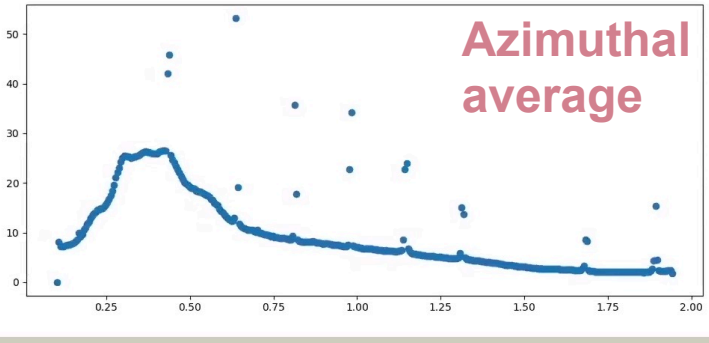


- Tools to make parameter setting easy



Beam Center Crosscheck

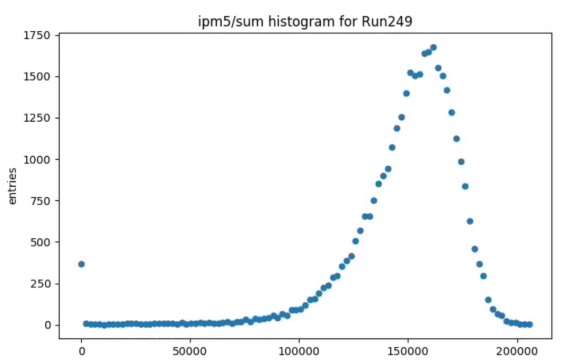


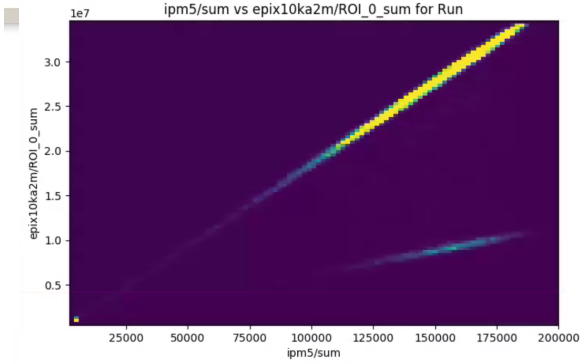


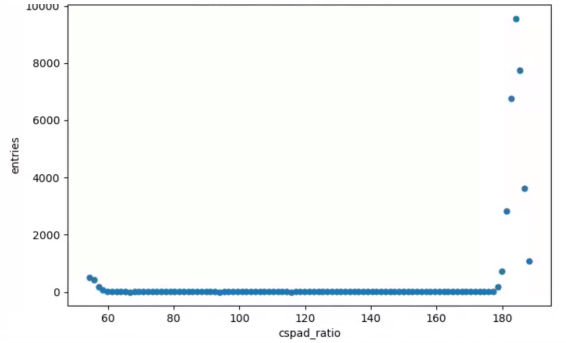
Plotting & Filtering

- Access to all variables in hdf5 file
- Simple histogram plots and correlation plots
- Filters as sets of square cuts
- Create derived variables and add them to interface for further use

```
SDAna In [2]: ana.addVar('det_i0_ratio', ana.getVar('epix10ka2m/ROI_0_sum') / ana.getVar('ipm5/sum'))
/reg/g/psdm/sw/conda/inst/miniconda2-prod-rhel7/envs/ana-1.4.16/bin/ipython:1: RuntimeWarning: divide
  #!/reg/g/psdm/sw/conda/inst/miniconda2-prod-rhel7/envs/ana-1.4.16/bin/python

SDAna In [3]: ana.addCut('ipm5/sum',50000,1e6,'good')

SDAna In [4]: h2 = ana.plotVar('det_i0_ratio', useFilter='good')
plot det_i0_ratio from 54.4157 to 189.505

SDAna In [5]: ana.addCut('det_i0_ratio',120,1e6,'good')
```



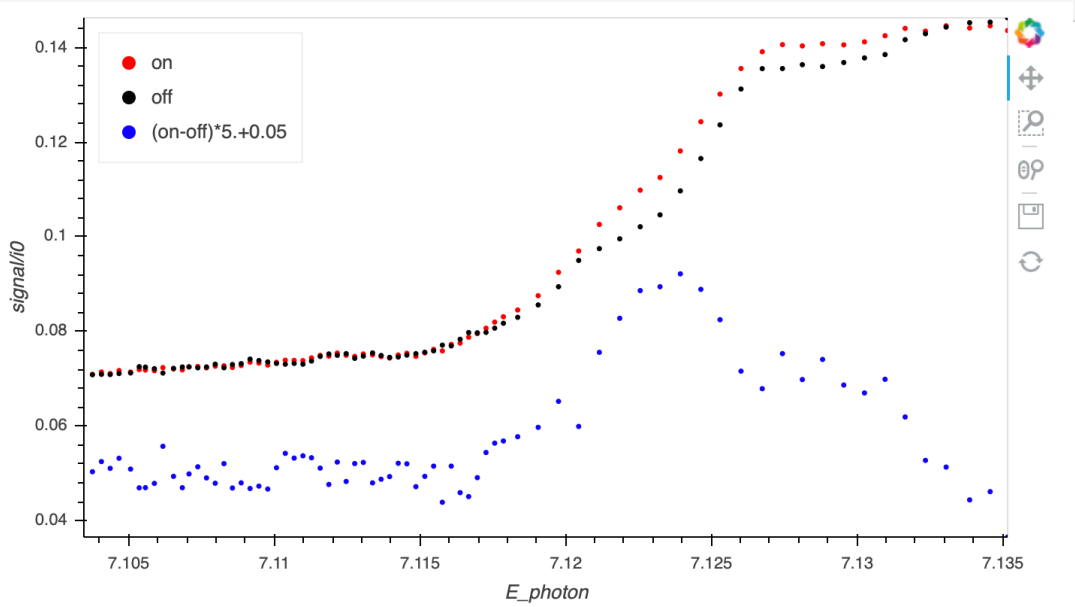




Event Binning

```
In [2]: ana.addCut('lightStatus/xray',0.5,1.5,'ipm')
        ana.addCut('ipm3/sum',0.1,1.5,'ipm')

        scanName, scanValues = ana.getScanValues()
        scanSteps = np.unique(scanValues)
        ana.addCube('cube3d','scan/%s'%scanName,scanSteps,useFilter='ipm',addBinVars={'lightStatus/laser':[-0.5,1.5,2]})
        ana.addToCube('cube3d',['ipm3/sum','diodeU/channels'])
        cubeData3d = ana.makeCubeData('cube3d')
```

- Instructions to bin data (variable[s] & bin boundaries)
- A filter
- Variables to be binned



smalldata_tools: Overview

SLAC

Production

Event-based hdf5 production (smallData)

- Instrument-based default data
- Detector & physics-based data reduction

Detector-based hdf5 production (cube)

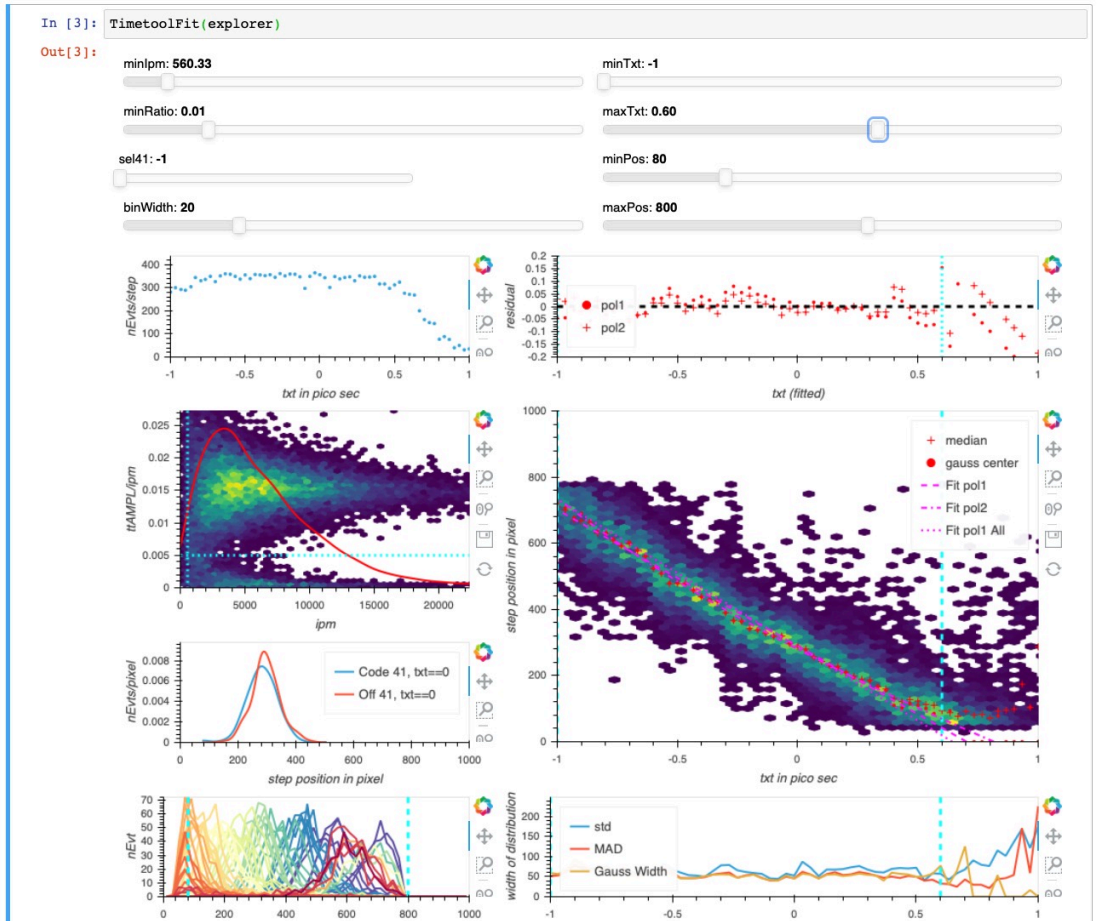
- Sum over selected events

Analysis

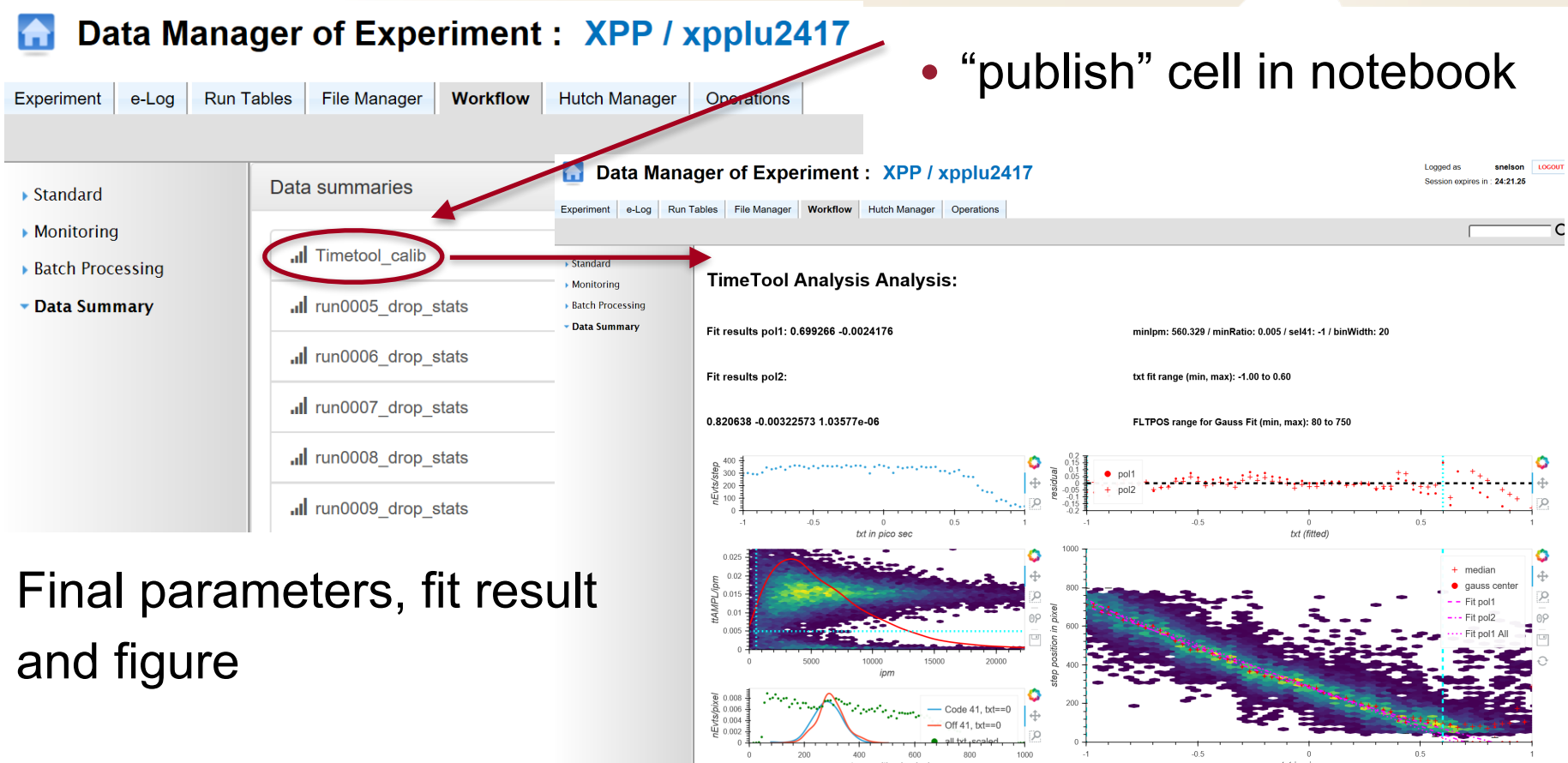
- Quick plots
- Tools to setup data reduction
- Multidimensional binning w/ filtering based on small data
  - Addition of big data
- Notebooks for standard procedures

Standard Notebooks: Timetool Calibration

- Documentation/ explanations within notebook
- Main cell with all ‘knobs’ which are pre-set to an optimal value
- Follow-up cell that will create an html file for eLog



Timetool Calibration Result in Data Manager



- "publish" cell in notebook
- Final parameters, fit result and figure

Summary & Brief Outlook

SLAC

- Challenge at FEL is the wide variety of analyses/detectors/user expertise
- smalldata_tools has been developed to be
  - Simple to set for ‘easy’ analyses
  - Extendable for more complicated cases
  - Base for detector monitoring, calibration procedures
- Easy-to-produce hdf5 files w/ hidden MPI integrated to LCLS analysis
- Production via DataManager, true workflow in the future
- Pre-prepared Jupyter notebooks
- ‘small’ hdf5 production and detector interface also to be used for LCLS-2
- Concepts of event-based small data and binned data present in data access and DRP (data reduction pipeline) for LCLS-2


Questions?