3. Cube production

This section details how to set up custom binning of the full detector images and save the output to an hdf5 file.

The main cube script is `./producers/letsCube.py`, but this file should generally not be modified. The configuration is done through a hutch-specific file of the type `./producers/cube_config_<hutch>.py`.

Config file

The configuration consists of the following main sections (see example below too):

- Custom bin definition:
  Using a run-based logic, one can define how to bin the data here. There are multiple options:
  - Give the bin_name and the bins. The bin name must be given as a path in the smalldata h5 file. For example `ipm2/sum` or `epics/gon_y`. The bins should be an array defining the bin edges to uses. This is generally what should be use for non-standard scans.
  - If bins are given, but no bin_name, it will try to make a delay scan. This will fail if that is not one.
  - If no bins are given for a run, then it will try to use the unique position of the corresponding scan field. This won't work for scan not taken with a Bluesky plan.
- Filter definition: Default filter name is `filter1`. Any other filter name will lead to the creation of separate cube files ending with `_filtername`. This can be useful in case where an external field is applied on certain shots, or separate data based on an event code marker for example.
- Laser on/off: If True will create _on and `_off` cubed file.
- Detectors: Area detectors dictionaries to be cubed are defined here. Each detector is defined through a dictionary that can have the following keys:
  - `thresADU (float)`: pixel intensity threshold (float)
  - `common_mode (int/None)`: common mode selection (int/None). Is `None` in most cases (take default common mode for that detector). This is generally set by your data analysis PoC and should rarely be changed.
  - `image (bool)`: whether to return the image (True) or the calib (False, individual tiles)
  - `full (bool)`: not-implemented (ignore for now)
  - `det_proc (dict)`: a dictionary containing the name and arguments for post-processing analysis to be applied to the binned detector data. See 3.1 Post-processing functions for details on the different functions available.
- All the variable to add to the cubed file are then added to `varList`. The syntax for each non-area detector variable should follow the h5 tree structure in the smalldata h5 file. Area detectors dictionaries must be added to this list as well.
- Histograms: At the end of the cube, a summary is posted to the elog. A list of variables can be customized here. The IPMs and timing tool are generally considered default. If a variable is also in the filter list, the filter boundaries will be shown in the histogram too. The pixel histograms for the area detector defined are also included in the summary. These plots can help you choose relevant filters and thresholds. Adding event codes to this list might be a good idea in cases where event code-based filters are used.

Example config file for XPP:

```
import numpy as np

# custom bins
def binBoundaries(run):
    if isinstance(run,str):
        run=int(run)
    if run>0:
        return np.arange(-5.,50.,0.2)
    return None


# filters to apply to the data
# format: list of [det (field), low, high, name]
# 'filter1' is the standard name and will not be added to the h5 file name.
filters = [
    ['lightStatus/xray',0.5,1.5,'filter1'],
    ['ipm2/sum',3e2,6e4,'filter1'],
    ['evr/code_41',0.5,1.5,'custom']
]

# Laser on/off.
laser = True

# List detectors to be cubed. Area detector have additional options such as threshold
# For now only full image or calib works. TODO: Add photon maps. And then any detObjectFunc
# Detectors should then be added to varList
detDict = {'source':'jungfrau1M',
           'full':1,
           'image':0,
           'thresADU':6.5,
           'common_mode':0}

varList = ['ipm2/sum','ipm3/sum','diodeU/channels', detDict]


# histogram configuration. Usually does not need to be changed
# field: destination in smd, list: [low,high,n] or None (then default to some percentile)
hist_list = {
    'ipm2/sum': [0, 5e4, 70],
    'ipm3/sum': [0, 5e3, 70],
    'tt/FLTPOS_PS': [-0.5, 0.5, 70],
    'tt/AMPL': [0, 0.2, 70],
    'tt/FLTPOSFWHM': [0, 300, 70]
}


# save as tiff file (ignore unless MEC)
save_tiff = False
```

Cube file content

The cubed data are saved in a h5 files with the following naming convention:

`Cube_<exp>_<run#>_<bin_axis>_<laser_status>_<filter_name>.h5`

When using `laser=False` or `filter1` as the filter name, the last two bits will be omitted.

The datasets are the following:

- `binVar_bins`: bin edges as defined in the config or the default (unique) motor positions. In the latter case, the first value is NaN, as a bin is manually added to catch outliers on the left for the general case.
- `nEntries`: number of shots in each bin
- `binVar`: sum of the binned variable. When binning linearly, binVar/nEntries should return the most accurate position. A field with the binned variable name contains the same information.
- All the binned variables defined in `varList`. These entries are the sum of all data falling in the corresponding bin. For the regular detectors (non-area detectors) the syntax follow that of the varList, where "/" have been replaced by "__" (two underscores). Binned area detectors images or calib are under `<detname>_data`. A `<detname>_nEntries` field should match the `nEntries` field. If not, contact your controls and data PoC. The binned bin-axis variable also appears here. As for the other variables, it is summed and should be divided by nEntries to retrieve a value

close to the original bin settings (if this is something of use).
Example from the config file above:
    - `ipm2__sum`
    - `ipm3__sum`
    - `diodeU__channels`
    - `jungfrau1M_data`
    - `jungfrau1M_nEntries`
- `<detname>_cfg`: a config group for each area detector that contains the detector config (pedestals, gain map, mask, etc.)

An example notebook that load and plots a cube file can found at: `/cds/group/psdm/sw/tools/smalldata_tools/example_notebooks/cube.ipynb`

Please don't modify this file directly, copy it to your home or experiment area to test and explore.

Debug the cube production

Job is done but the report does not show "Cube: Done"

This means the job failed somewhere. Open the log files and look if one of the following cases apply:

1. The most common case is that there is a typo or syntax in the config file. In this case, the log file should be very short, directly pointing at the section in the config file that has issue. As often with typo or syntax errors in python, the error is not always at the exact line mention in the error.
2. Check that the filters do not filter out all the pulses. This can easily spotted by looking for the following section at the end of the log file:

```
did not select any event, quit now!
getFilter: Cut 0.500000 < evr/code_94 < 1.500000 passes 0 events of 11252, total passes up to now: 0
getFilter: Cut 24.464119 < scan/diag_x < 24.496000 passes 11252 events of 11252, total passes up to now:
0
getFilter: Cut 0.500000 < damage/jungfrau1M < 1.500000 passes 11252 events of 11252, total passes up to
now: 0
```

Analyze cube results

An example notebook of a cube file analysis can be found in `/cds/group/psdm/sw/tools/smalldata_tools/example_notebooks/cube.ipynb`.