All scripts support option -h for help.

1. **find_baseband_runs.py**
   A script that trolls through logs directory and prints contents of log files between two specified dates. Needs a directory that contains following folders:

   albatros_config_fpga albatros_dump_baseband albatros_dump_spectra

   -s start_date -S stop_date (YYYYMMDD)

   ```
   python find_baseband_runs.py ~/logs/snap1/ -s 20210729 -S 20210730
   ```

2. **quick_spectra.py**
   A script to plot direct spectra products (pol0 mag, pol1 mag, pol0xpol1 mag and phase) for any given direct data folder.

   Needs a directory that contains the following files:

   pol00.scio.bz2 pol01i.scio.bz2 pol01r.scio.bz2 pol11.scio.bz2

   Ideally, the directory name should be a timestamp (as is the convention for albatros data dumps), so that the output image has the timestamp in its name. This timestamp indicates the starting time for this direct data file. FPGA code is set to create a new file roughly every hour. Usual directory structure is something like:

   ```
   ~/data_auto_cross/snap1/16272/1627202093/
   ```

   Use -l (ell) for logscale plots, which are more useful. -o for specifying output dir. Default output dir is ./ (dir from which code is called).

   ```
   python quick_spectra.py ~/1627202093/ -o ~/outputs/ -l
   ```

3. **plot_overnight_new.py**
   A script to plot direct spectra products over several hours (combining multiple files). It will generate a single plot showing what the data looks like for an entire night's (or several nights') run. It takes a start timestamp and end timestamp and automatically finds all files that lie between these timestamps. (Timestamp format YYYYMMDD_hhmmss)

   Needs a top-level directory that contains sub-directories with "short" 5 digit timestamps. (Each of these sub-dirs has sub-dirs with full 10 digit timestamps.) E.g.

   ```
   ~/data_auto_cross/snap1/
   ```

In a particular direct data file, each data point (for, say, pol00) is averaged over roughly 6 seconds by the FPGA. This may be too high a resolution if you're plotting for several hours. Use **-a [avglen]** to specify how many points to average over.

```
python plot_overnight_new.py ~/data_auto_cross/snap3/ 20210730_000000
20210730_070000 -o ~/outputs/ -a 50 -l
```

The above example calculates direct data products averaged over 5 minutes (50 points) for data collected between Jul-30 12 AM to Jul-30 7 AM. And once again, use -l for logscale plots.

It is the user's responsibility to make sure that a contiguous chunk of data is available between the two timestamps. If there's some discontinuity (e.g. data wasn't collected for 2 hours in the 7 hour window), it'll be obvious from the plots. In this sense, this is a useful diagnostic tool.

4. **Plot_hist.py**

First switch to branch **newcode.**

```
git checkout newcode
```

If there are no compiled ".so" files in the correlations directory, run:

```
python ./correlations/setup.py
```

(You should still run this if you have some old version of .so files lying around.)

This script allows you to histogram the whole baseband file. You can specify if you want to histogram pol0, pol1, or both (entire data) using option **-m [mode].** (mode=0 for pol0, 1 for pol1, -1 for both. Default is -1.)

Default output plot will have counts for each bit value, e.g. 0 - 15 for 4 bit data. You can shift the bins and their values to reflect the positive and negative levels using option **-r.** E.g. -7 to 7 to 4 bit data.

```
python plot_hist.py ~/1627202093.raw -m -1 -o ~/outputs/ -r
```