HERA Memo #124
Version 3
18 Sept 2023

HERA Team Memo: H6C Internal Data Release 2.1

Joshua S. Dillon, Steven G. Murray, Zachary E. Martinot, and the HERA Analysis Team

August 30, 2023

1 Executive Summary

In this memo, we lay out the first release of data products from a significant subset (14 nights) of the 6th season of HERA observing. We explain the available data products, where to find them, how they were made, and how the analysis has changed since H1C and H4C—most notably the shift to notebook-based processing and the overhaul of the LST-binner. The purpose of this IDR is to explore various analysis choices and their impact on systematics as we narrow in on a final analysis and begin power spectrum estimation and comparison to astrophysical theory.

2 Observing Epoch and Data Selection

This IDR includes 14 nights starting with 2459861 (October 8-9, 2022) and ending with 2459876 (October 23-24, 2022). These nights were picked because they were an early set of approximately two weeks of continuous data taken with no major issues noted by observers—a similar length of time that went into H1C IDR2 (Dillon, 2019), the data set underlying the first hera limits (Abdurashidova et al., 2022a) and their interpretation (Abdurashidova et al., 2022b).

2459865 and 2459875 were excluded due to large quantities of broadband RFI (Murray and Dillon, 2023a), currently believed to be attributable to lightning (Heligenstein and Jacos, 2023). Frequent, bright broadband RFI is basically impossible to detect in our per-file RFI flagging and difficult to detect in our full-day RFI flagger, especially low-level broadband RFI between brighter events. When the density of RFI gets bad enough, the choice was made to flag the whole night. Additionally, bright RFI events (likely lightning storms) that persist for only a modest fraction of the night resulted in partial-night flags. These include:

- 2459863.2525197007 – 2459863.3724208707
- 2459869.2528783116 – 2459869.3774771024
- 2459869.6649267366 – 2459869.6694006610
- 2459872.2527171504 – 2459872.2930104310
- 2459876.2527275416 – 2459876.2624583268
- 2459876.6598546500 – 2459876.6692498910

Finally, small quantities of data we flagged because the sun was above the horizon (generally at the end of the night).

All these JD-based flags were developed by looking at just autocorrelations as part of a "data-scouting" exercise using the `full_day_auto_checker` notebook, with which we produced one "a priori" flag YAML file per night. The evaluated notebooks are available at NRAO and final a priori YAMLs are also in `hera_pipelines`. The full season has roughly 1,300 hours over 147 nights currently believed to be good quality data. For more details, see Murray and Dillon (2023a).

The LST-coverage of this dataset, after all of that flagging and the per-night RFI flagging discussed in subsection 3.4 below, is shown in Figure 1.

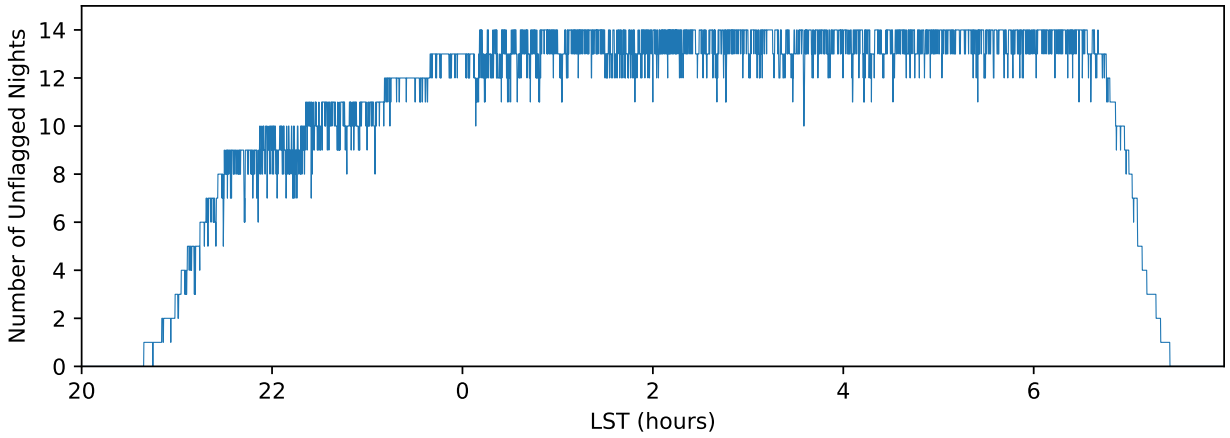

Figure 1: Number of nights observing each LST bin, after flagging. More time is flagged earlier in the IDR in part because lightning was more common during the first part of the night than later into the night.

3 Per-Night Analysis

The goal of nightly analysis is to prepare the data for LST-binning by performing calibration and flagging. Calibration includes redundant-baseline calibration (Dillon et al., 2020), fixing of the degeneracies of redundant-baseline calibration (Dillon et al., 2018) with absolute (sky-based) calibration (Kern et al., 2020), and calibration smoothing. Flagging includes both the generation of a single RFI flagging mask applied to all baselines, as well as flagging of whole antennas over either the whole night or part of the night.

The pipeline also produces a number of post-processed data products, which incorporate all of the flagging and calibration mentioned above. These include averaging over redundant groups and delay-filtering. These allow for the LST-binning of data products with favorable properties, like file size or foreground contamination.

3.1 Pipeline

As in previous IDRs, the pipeline is defined by a single `TOML` file, which interfaces with shell scripts that run the various analysis steps using `hera_opm` and `makeflow`. Unlike in previous seasons, the processing has largely replaced python scripts with Jupyter Notebooks. These are either run one-per-file (most nights have 1,862 sum files, each with two integrations) or one per-night, where the per-night processing steps tend to be the bottleneck. These notebooks—all of which are available as unprocessed "templates" in `hera_notebook_templates`—are meant to both be a clear, well-annotated presentation of the analysis steps performed accompanying a set of representative plots and tables that might allow the analyst to quickly identify issues in the analysis before performing LST-binning. In Figure 2 we show the pipeline with the input and output data products and the notebooks that perform the processing. In the rest of this section, we explain the various processing steps, focusing on what is new for this season.

The biggest single change is the introduction of the `file_calibration` notebook, which is an attempt to carry a single file as far as possible without reference to external information or other times. It therefore includes antenna flagging, RFI-flagging, and both redundant-baseline and absolute calibration. It is not the final word on flagging or calibration—later steps allow information from other times to impact the results.

3.2 Antenna Flagging

Antenna flagging has been largely overhauled, most antenna flagging takes place on a per-file basis in the `file_calibration` notebook. Unlike in previous seasons, antennas can be flagged per-polarization and per-file. Antenna flagging always includes both times and all frequencies in the file. In the context of the `file_calibration` notebook, an antenna is labeled good, bad, or suspect on a number of criteria. Bad

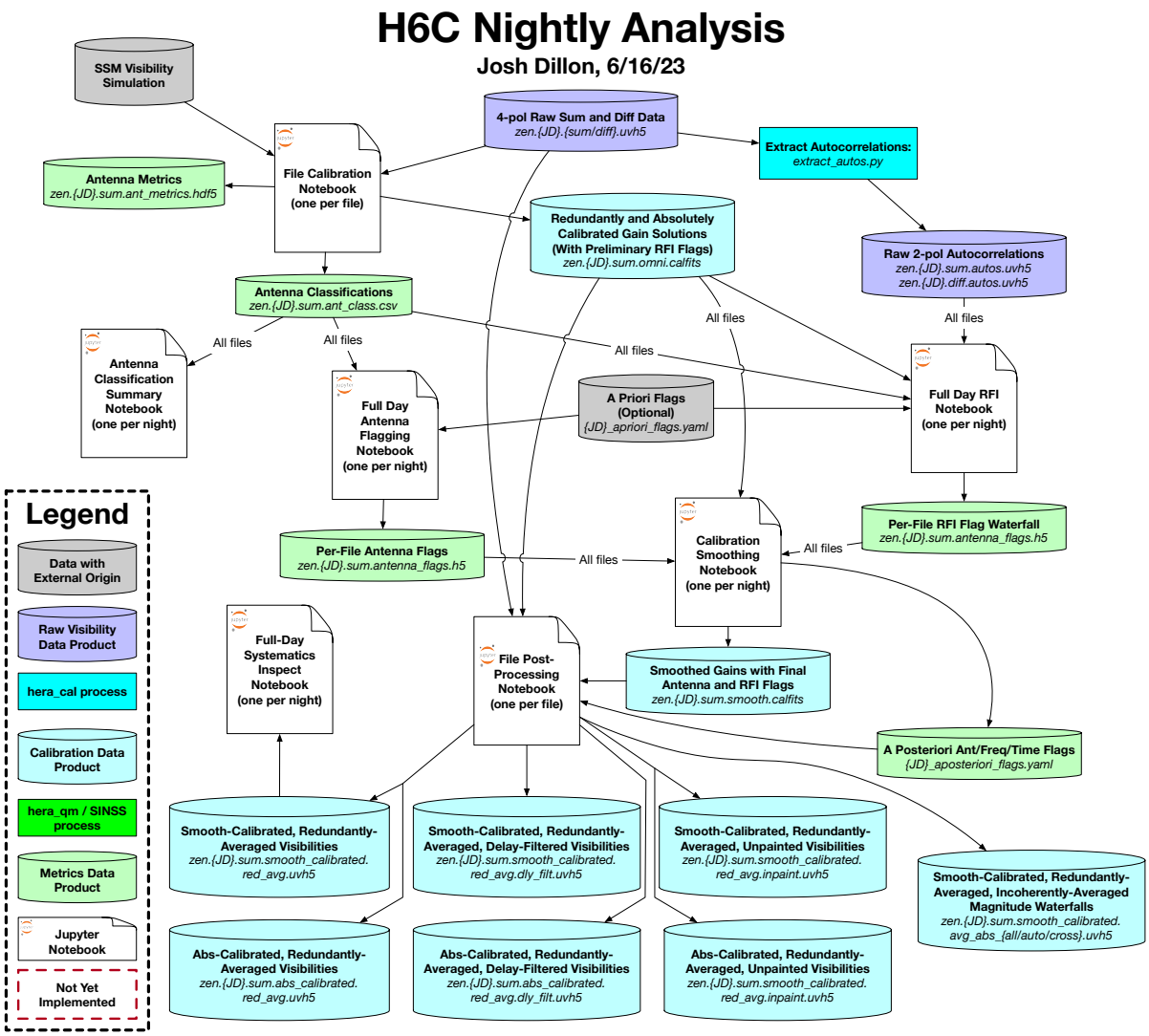

Figure 2: The per-day analysis is centered on a series of notebooks, either one per file or just one per night, each of which both performs an analysis task and visualizes representative results in plots and tables. These notebooks are all saved as `.html` files with embedded images, serving as rich logs of the processing. All the per-night notebooks, as well the per-file notebook corresponding to the middle file of each night, are available at https://data.nrao.edu/hera/Notebooks/H6C_IDR2/. The a priori flags come from the "scouting" discussed in Murray and Dillon (2023a).

antennas for any reason are bad and flagged. Suspect antennas for any reason are suspect, but not necessarily flagged. Antenna classification and the reasons behind it are reported in Table 1 of the `file_calibration` notebook.

Antennas can be flagged for any of the following reasons:

- **Dead**: Antennas with visibilities that are more than half zeros are considered dead. This is a binary classification; the rest can be good, bad, or suspect.
- **Low-correlation**: Antennas with low correlation coefficients are flagged—usually because of clock distribution issues that affect whole nodes (Storer et al., 2022).
- **Cross-polarized**: Antennas with larger visibility amplitudes when correlated with putatively opposite-

polarized antennas are likely cross-polarized (Storer et al., 2022) and are thus flagged.

- **Excess even/odd zeros**: While the visibility data products from the correlator are sums and diffs, these can be rearranged into even and odd samples. An excess of zeros in either the evens or the odds is generally a sign of packet loss. While this happens on a per-baseline basis, we have generally found that packet loss tends to affect a few antennas disproportionately because of the ordering of visibility information. So we pick the smallest number of antennas to flag for a given file that then flags all visibilities with signs of packet loss. This overflagging has the benefit of allowing us to carry per-antenna flags with calibration solutions—something we could not do with per-visibility flags.

- **High or low autocorrelation power**: Autocorrelations with median autocorrelations that are too high or too low are flagged.

- **Excess RFI in autocorrelations**: Each autocorrelation is DPSS-filtered (Ewall-Wice et al., 2021) and then divided by the expected noise on an autocorrelation to form a z-score. This allows for detection of channels with RFI or other non-smooth structure. Channels flagged on more than half of antennas are considered "consensus flags." Antennas are flagged if their autocorrelations have too many anomalous channels that are not among the "consensus flags." These appear to have some sort of internal RFI, a comb of small spikes, large reflections, or similar, though the exact nature of the problems caught by this check are not fully understood.

- **Anomalous autocorrelation slope**: After a linear fit to median-filtered autocorrelations (to remove RFI), antennas with autocorrelations that tilt too far from flatness in either direction are flagged.

- **Anomalous autocorrelation shape**: After performing the above checks, antennas with autocorrelations too dissimilar to the average (previously) unflagged antenna are themselves flagged. This check is done after applying a consistent set of RFI flags and dividing out by a single overall normalization per antenna.

- **Excess per-X-engine power in `diff` visibilities:** After performing the above checks, we next see whether the diff visibilities are consistent with noise predicted by the autocorrelations. This catches issues due to stale or otherwise mis-written packets, which had previously led to autos getting written in either the even or odd subset of the visibility, but not the other. We suspect that wrong crosses were also leaking into crosses. We do this check by X-engine group (i.e. every 96 channels), since that's the observed failure mode, looking for 10σ outliers. Just as with the excess even/odd zeros, this per-baseline failure is attributed to the smallest number of individual antennas that explain all such issues. We have noticed that this check also sometimes catches broadband RFI (e.g. lightning), perhaps because it creates temporal discontinuities in the actual signal on the even/odd timescale.

- **High `redcal` χ^2 per antenna**: Antennas with high χ^2 per antenna are flagged iteratively; after identifying bad antennas, a further round of redundant calibration is performed until no more antennas are flagged. This can find issues like index-swapped antennas, open dish doors, and feed position/orientation issues.

- Outrigger antennas are all flagged for ease of calibration and for reducing the size of redundantly-averaged visibility files.

While all this per-file antenna flagging enabled us to carry raw data all the way though basic calibration and RFI flagging, it resulted in often-inconsistent antenna-flagging where an antenna would be flagged for much but not all of the night, sometimes going in and out repeatedly. Since much of systematics rejection depends on temporal smoothness, we decided to apply an addition round of antenna flagging harmonization. This was performed in one notebook, `full_day_antenna_flagging`, which ran after calibration but before calibration smoothing.

Without going into too much specifics, the basic idea of this notebook is to harmonize flags by looking for periods of time when various metrics which produce the above flags consistently exceed the thresholds for being identified as "bad." No more than 30 files (60 integrations) are allowed to be flagged in a row, otherwise, the whole night before or after those files would also be flagged (since large flag gaps cause

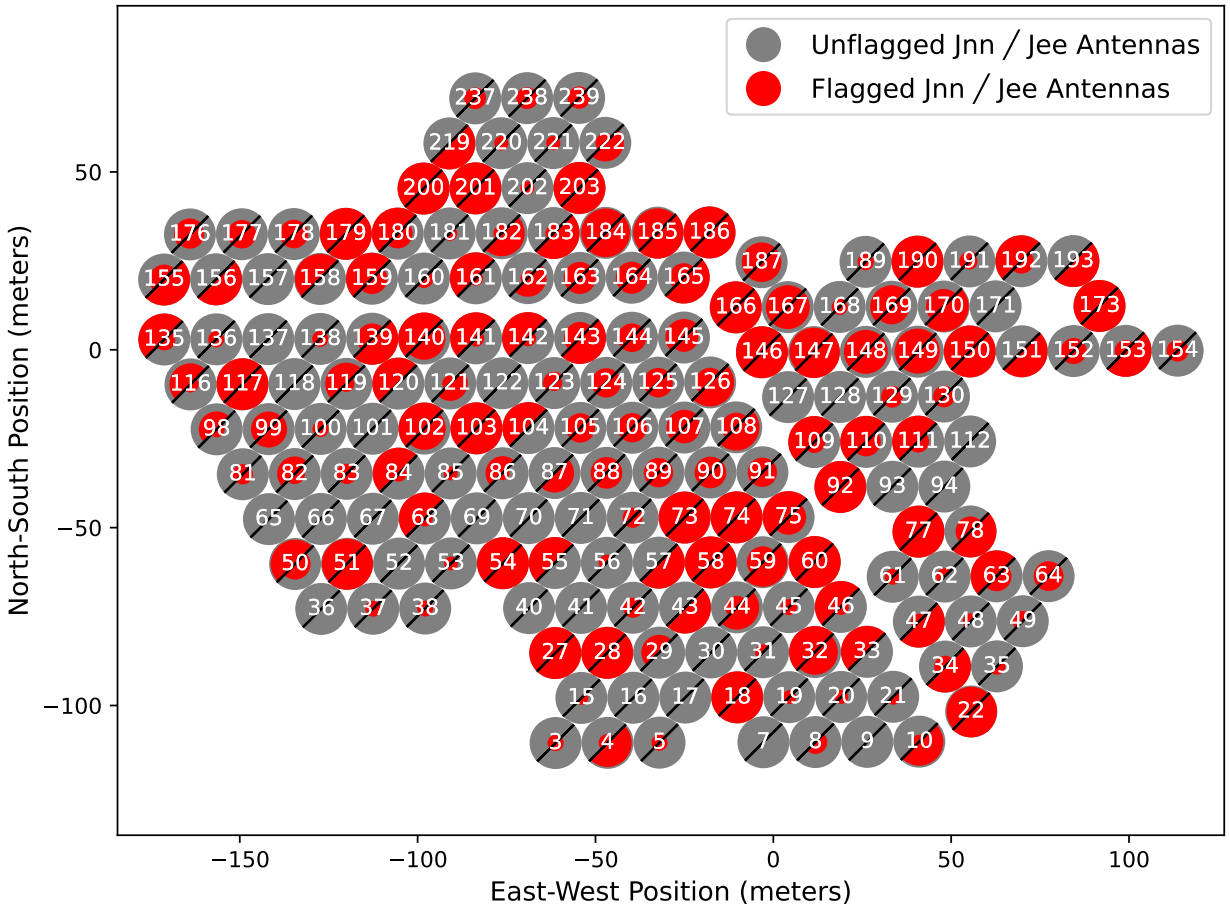

Figure 3: Antenna flagging fractions over the IDR, where the red area is proportional to fraction of total observing time flagged (excluding times when the whole array was flagged). The top-left semicircles indicate north-polarized antennas, the bottom right semicircles indicates east-polarized antennas.

problems with smoothing and fringe-rate filtering). Finally, antennas flagged for more than half the night (or more than a quarter of the night for certain specific metrics) were also simply flagged for the whole night. In Figure 3, we summarize the antenna flag fractions over the whole season.

3.3 Calibration

The basic structure of our calibration algorithm is very similar to what was done for H1C and H4C. Redundant-baseline calibration is performed as in H1C (Dillon et al., 2020) and H4C. The degeneracies of `redcal` are then calibrated using a simulated visibility model. Like in H4C, the visibility model is computed with the RIMEz visibility simulation code, but in H6C the sky model used is Zac Martinot's Southern Sky Model (Martinot, 2022) which blends the GLEAM point-source catalog with a diffuse component. By contrast, in H4C the sky model used consisted of only the GLEAM sources plus a set of very bright sources that are not included in the GLEAM catalog (Dillon and Martinot, 2020), while H1C was calibrated against calibrated data; see Kern et al. 2020. Finally, these per-time-per-frequency calibration solutions are smoothed in both time and frequency using least-squares fits to a basis of DPSS functions (as opposed to H1C, which used CLEAN).

Redundant-baseline calibration is only modestly changed. We have eliminated `logcal`, instead solving for starting visibilities for `omnical` with averaged visibilities calibrated with `firstcal`. After some experimentation, we have dramatically reduced the number of `omnical` iterations to a minimum of 100. Each time

antennas are thrown out for high χ^2, an extra 50 iterations are run. Our experiments indicate that this does not change the gain solutions appreciably compared to the noise and has sub-percent-level effects on χ^2.

Absolute calibration has been more substantially changed.

The process begins in the `file_calibration` notebook with a very simple delay-slope calibration. The delay calibration uses the `omnical` visibility solutions at the channels of known transmitters (with known headings) to remove overall delay slopes—a subset of the phase slope degeneracy of `redcal`. This step is no longer needed for absolute phase calibration with the new phase calibration algorithm (described below), but has been left in place in the `file_calibration` notebook since it is cheap to run and can still provide a rough phase calibration if the visibility model is not available.

Next, we calibrate the overall gain amplitude spectrum using the model autocorrelations simulated with RIMEz and the Southern Sky Model. This autocorrelation model includes a model of the receiver temperature spectrum adapted from Fagnoni et al. (2021) - the autocorrelation model is a cubic spline interpolation of the five temperature values specified at the end of section III-A. Cross-correlation visibilities are no longer used to perform absolute calibration of gain amplitudes, which means we no longer need to worry about the "`abscal` bias" that resulted from low-SNR complex calibration (Aguirre et al., 2021).

Next, we use the amplitude and delay-slope calibrated `omnical` visibility solutions to calibrate out the per-frequency generalized phase gradient across the array. The number of components of the generalized phase gradient depends on the level of redundancy of the array. While full HERA has only two such degrees of freedom, which correspond to EW and NS phase slopes, the split-core configuration can produce more degrees of freedom if enough antennas are flagged or not included in one of the three sectors.

The algorithm for this phase calibration has been overhauled to produce much more accurate phase solutions when model errors are π-radians or more on some visibilities. This level of error is generally present even at the best-modeled LST's (e.g. LST 2hr) and the old algorithm solved a linearized approximation problem which was generally not valid in the presence of errors of that magnitude. The new technique solves the non-linear least-squares problem exactly. Roughly, the new technique takes the product $V_{\rm data}V^*_{\rm model}$, puts it on a grid, FFTs, and finds the per-frequency generalized phase gradient corresponding to the maximum of the real part of that quantity. This initial estimate is then refined with Newton's method to fined the exact maximum of the objective function. There are still occasional calibration failures at some LST's where the data and model are quite dissimilar, but the new method does substantially better than the old `TT_phs_logcal` approach and reasonable calibration solutions can now be obtained for most LST's where the solutions were previously collapsing, necessitating smoothing over.

This new algorithm was implemented in this pull request as a new set of functions in `hera_cal.abscal`:

```
abscal.complex_phase_abscal
abscal._put_transformed_array_on_integer_grid
abscal._phase_gradient_solution
abscal._newton_solve
abscal._grad_and_hess
abscal._eval_Z
```

The entry point to the new algorithm within the `file_calibration` notebook is the function `complex_phase_abscal`.

The results of `redcal` and `abscal` can be seen in the `file_calibration` notebooks. These are created per file in parallel and the middle one per day is hosted at that link. The rest sit on `lustre` with the other data products.

Calibration smoothing was performed jointly in two dimensions with filter scales picked to fit time evolution with 3 DPSS modes and frequency evolution with 51 DPSS modes. Unlike in H4C, no LSTs were "blacklisted" (i.e. given 0 weight in the fit, but not necessarily flagged). The `calibration_smoothing` notebooks that performed the calculation are all available at that link. They also plot the results of smoothing for a few representative antennas. One algorithmic change was the addition of the detection of phase flips in the gains, which were exhibited by two antennas (121-East and 144-North) occasionally. These flips are taken out before smoothing and then put back in after smoothing, just like an overall delay. Integrations before and after phase flips were flagged since the flip could have happened in the middle of an integration.

3.4 RFI Excision

RFI excision has been completely overhauled and quite simplified. The basic idea is to try to identify RFI using array-averaged autocorrelations, which have very low noise and are thus very sensitive to RFI. This happens in two stages.

In the first, which is part of the `file_calibration` notebook, flagging is performed on a single spectrum created from averaging over all non-flagged antennas' autocorrelations and over both integrations in the file. First the extreme outlier channels are flagged with a simple differencing method for identifying outliers. Next, the autocorrelation spectrum is fit with DPSS (Ewall-Wice et al., 2021) modes out to 300 ns. The residual is then turned into a z-score by dividing by the expected noise on that array-averaged autocorrelation. Outliers greater than 6σ are flagged.

After all `file_calibration` notebooks are complete but before `calibration_smoothing` is performed, we run a full_day_RFI excision notebook. The key idea of this notebook is to look for RFI using unsmoothness in both time and frequency, which is particularly important for detecting broadband RFI like that created by (we believe) lightning storms. This notebook, which runs once per day, takes the initial set of flags as a starting point, but it is allowed to add or remove flags as appropriate.

The basic idea is as follows: using the starting set of flags, along with the range from 87.5–108 MHz and any times identified by the season scouting (Murray and Dillon, 2023a), antennas are individually filtered in 2D with DPSS on 5 MHz and 450 s scales. Ignoring antennas with substantial residual structure (or the worst 75% of antennas, whichever is fewer), the autocorrelations are then averaged and filtered. Again, we turn the filtered autocorrelations into a z-score by dividing by the expected noise. We add flags to 5σ outliers and to 4σ outliers that neighbor other flags (iteratively, using the watershed algorithm). Next, we iteratively flag whole times and/or channels whose average z-score is greater than 1.5σ. After this, any integration more than 10% flagged and any channel more than 25% flagged is completely flagged.

With this "round 1" flagging mask created by expanding the initial flags, the averaged autocorrelation is DPSS filtered again. After re-flagging any a priori flagged data, any 2σ outliers that were flagged in round 1 are flagged again. We now repeat the above procedure, starting with the 5σ outliers, the 4σ neighbors, and the per-integration and per-channel thresholding. In this way, times and frequencies flagged by the individual notebooks but not showing significant outliers from the 2D DPSS model are allowed to remain unflagged. This might be because they were due to spectral structure on individual antennas not present in all antennas, though this remains an open question. Likewise, there is an open question about using `redcal` χ^2 as another source for identifying low-level RFI and indeed some unflagged spikes in χ^2 are apparent in the `calibration_smoothing` notebooks.

3.5 Post-Processing

In theory, one could be done at this point; you can take the calibration solutions and flags you want from the above steps and proceed to LST-binning. However, there is one final step in the per-night data analysis pipeline designed to reduce the data volume before LST-binning. This "post-processing" step is run for all files in parallel. We also post the file_postprocessing notebook posted from each night as a (hopefully) representative sample, since it is not feasible to manually inspect thousands of such notebooks. It consists of three main tasks.

The first is the coherent averaging of visibilities in redundant baseline groups. This is performed both for visibilities with smoothed calibration solutions and for visibilities with all the same flags but no calibration smoothing.[1] The notebook also includes fitting of all redundantly-averaged baselines (with both smoothed and not smoothed gains) using DPSS at the horizon delay or 150 ns, whichever is larger. This enables both inpainting and delay-filtering. Finally, we produce waterfalls of incoherently baseline-averaged magnitudes of autocorrelations, cross-correlations, and both, which may be useful for detecting transients like FRBs.

After post-processing is done, we perform a full-day look at systematics on a handful of redundantly-averaged baselines using the full_day_systematics_inspect notebook. This is examining full-day waterfalls in frequency/time space, delay/time space, and delay/fringe-rate space before and after delay and fringe-rate filtering, as well as coherent and incoherent averaging and after forming pseudo-Stokes I and Q. Though this

[1] Likely one of these two will be removed in future analyses, but at this point we are trying to explore multiple possible ways forward.

notebook does not examine all of the data, it is the deepest possible look at a subset of baselines on a single day, enabling the quick detection of low-level systematics like the characteristic "X" in delay/fringe-rate due to mutual coupling (Josaitis et al., 2022).

4 LST-Binning

The goal of LST-binning is to coherently average all the nights in the data together, within LST-aligned bins. This takes the total data volume from $N_{\rm nights} \times N_{\rm intg} \times N_{\rm bl} \times N_{\rm freq} \times N_{\rm pol}$ down to $N_{\rm LST} \times N_{\rm bl} \times N_{\rm freq} \times N_{\rm pol}$. The challenges associates with LST-binning include performance (especially I/O and Memory usage), book-keeping (ensuring baselines/pols between nightly files are treated properly) and quality assurance (i.e. ensuring that flagged or otherwise bad data doesn't get averaged into the final product).

For H6C, the LST-binner was completely overhauled[2] to improve performance and add some new features to enable the averaging of redundantly-averaged visibilities. We outline the basic algorithm in the next section, and highlight substantive changes following that.

4.1 Outline of LST-binning Procedure

The following attempts to summarize the new LST-binning procedure at a high-level:

1. **Determine the LST grid in which to bin the observations.** For H6C IDR2, this is a grid starting with its first *edge* at zero hours, its last edge at 24 hours, and a bin size as closely matching the raw integration time (∼9.6 s) as possible (while allowing for the exact match of the endpoints to 0 and 24).

2. **Determine which input files correspond to which output LST bin files.** We write out LST bin files that include 2 LST bins each (and thus are similar in data volume to the input files). Before running the LST-binner proper, we roughly configure which LST-bin file each input file corresponds to. We do this based on the JD encoded in the *filename* of each input file, and use the knowledge that (most) input files are regularly spaced in time. From this we obtain a conservative list of input files matching an output LST file (i.e., it includes *at least* all the input files required for each output bin). This configuration is run by `hera_opm` and saved to a YAML file in the lstbin-output directory.

3. **Run the LST-binner for each particular output file separately.** `hera_opm` creates individual tasks for each LST file, which are run via SLURM in parallel. For each, we:

   (a) **Read all input metadata**. We obtain the (conservative) list of matching input files for the two bins in this output file from the YAML configuration reference above, and access *only* the parts of their metadata that we require using the new `FastUVH5Meta` class in `pyuvdata`.

   (b) **Determine if this is redundantly-averaged input data.** We do this by checking if all baselines included in one of the input files are non-redundant.

   (c) **Collate all baselines across input files.** We go through each input file, read which antenna-pairs are included, and take a union of all of them. We do *not* consider whether they are fully flagged. If we have redundantly averaged data, we ensure that a change of representative baseline within the same group across different files is handled correctly. Note that since we do this collation of baselines at the per-output-file level, different output files may include different baselines.

   (d) **Initialize output files**. Now that we know the final size of the output data (baselines, polarizations, frequencies and 2 LSTs), we initialize the output UVH5 file with metadata and allocated space (on disk, not memory) for the LST-binned data. This is done for all output file *kinds*: the LST-binned data, the STD file (measured night-to-night standard deviations), and potentially a "GOLDEN" file, which has data shape $(N_{\rm nights}, N_{\rm bls}, N_{\rm freqs}, N_{\rm pols})$, and includes all the data for a particular LST-bin. This is only written if the user requests a GOLDEN LST that falls within the bins of the current output file. For H6C IDR2, we write one GOLDEN file for every LST-hour.

[2]Details of the changes in the code are in https://github.com/HERA-Team/hera_cal/pull/846.

(e) **Filter input files based on the LSTs in the output file.** Since the YAML configuration is potentially conservative, doing a rough calculation based on the filename of each input file, we now do a more accurate input-file selection based on the metadata itself.

(f) **Split the collated baselines into chunks with a size set by the user.** Then for each each chunk:

i. **Read the data**. Read the data (and $N_{\rm samples}$ and flags) associated with any of the baselines in the current chunk from all of the input files. We read these into contiguous arrays of shape $(N_{\rm times}, N_{\rm bls}, N_{\rm freqs}, N_{\rm pols})$, where the times may be sourced from multiple nights.

ii. **Calibrate.** If calibration files are specified by the user, read them and calibrate the data. This is impossible for redundantly-averaged data, which therefore already has calibration applied before averaging.

iii. **Rephase data to central LST grid.** In order to alleviate decoherence when averaging complex visibilities from across a full LST bin, we rephase the visibilities to the central LST of each bin.

iv. **LST-align.** We assign each time in the data to its appropriate LST-bin.

v. **Optionally Sigma-Clip**. We optionally perform a single-iteration sigma-clip within each LST-bin, where we obtain Z-Scores for each baseline and frequency (real and imaginary components treated separately) based on the (unweighted) median-absolute-deviation (MAD). There is an option to either flag data points with too few samples, or ignore sigma-clipping for those points.

vi. **LST-average.** For each LST bin, we average the data within the bin. We use weighted averages where the weights are $\xi N_{\rm samples}$ for each observation (where ξ is an indicator function for whether the data is flagged or not). These are typically binary for data that has not been redundantly averaged, but can be more general for redundantly-averaged data.

vii. **Find the standard deviation of the bin.** Using the weighted mean, we also determine the weighted sample standard deviation of the data (for both real and imaginary components separately), representing the night-to-night variance of each baseline and frequency channel.

viii. **Write the chunk to file.** We write the data generate for this baseline chunk to the initialized files, freeing up the memory for the next chunk.

Table 1 summarizes the options and parameters used in the LST-binning for the H6C IDR2 dataset.

4.2 Changes from H1C

Substantive changes to the algorithm are as follows:

1. The rough pre-matching of input files to output LST-bin files, and the saving of that configuration into a YAML file, was introduced. This improves performance significantly, and also makes it easier to do post-facto exploration of the LST-binning by reading in the configuration file. In terms of performance, the reason this is much faster is that it can use the JD's defined in the filenames of the input files to get a rough match. Parsing filenames is *significantly* faster than creating an open HDF5 File object (not to mention *reading* that file). The previous code was reading all files' metadata at the configuration step, taking about 40min for H6C IDR2, and then each task for each output LST bin file had to perform the *same match* again, because that configuration was never saved. Now, the quick rough-match on a single head node takes of order 1 minute, and the more accurate matching that happens for each particular LST file is quick because it only has to read a small number of files.

2. Instead of looping over baselines and appending each LST-binned result to a growing list, we now read a baseline 'chunk' from each constituent file, and perform LST-binning in a vectorized manner. This improves both CPU performance (due to the vectorization and keeping data contiguous in memory), and RAM usage.

3. We have added proper handling of redundantly-averaged input data, so that baseline groups are properly matched together (even if labeled by a different key for different files).

Option	Description	Value
`rephase`	Whether to rephase visibilities to the central LST of each bin	True
`golden_lsts`	LSTs for which to output full data for later exploration.	Every LST hour
`save_channels`	Channels to save full data over all LST bins	150, 750, 1250
`only_last_file_per_night`	Whether to use only the last file on any given night to determine the observed antenna pairs.	True
`sigma_clip_thresh`	The number of sigma before clipping data (zero means no clipping)	0, 4
`sigma_clip_min_N`	The number of data points required in an LST bin to do sigma-clipping	4
`flag_below_min_N`	Whether to flag data with fewer than `sigma_clip_min_N` points, or just to ignore sigma-clipping there.	False
`flag_thresh`	If the fraction of data in an LST bin (for a baseline-channel) that is flagged is higher than this value, flag the whole bin for that baseline-channel. Applied *before* sigma-clipping.	0.8

Table 1: Table of functional options for the LST-binner (i.e. ignoring paths and performance parameters). Sigma-clipping was in general turned off (i.e. threshold set to zero) but one run using a 4σ threshold was performed.

4. We have added proper handling of non-uniform N_{samples} in the LST-averaging and calculation of the standard deviation across nights (to accommodate redundantly-averaged input data).

5. We now initialize each LST-bin output file at the beginning of the LST-binning process (just setting up the metadata and data size on disk), then progressively write to that file for each baseline 'chunk' to be read. This enables the peak memory usage to be essentially set by the size of the baseline chunk (which can be controlled) and number of nights, without accumulating RAM-usage over the course of the averaging. Previously, RAM usage was difficult to predict and could exceed the available RAM at any point of the program's execution.

6. In addition to the LST-average and Standard Deviation (STD) files, we now make it possible to output 'GOLDEN' files that include all the calibrated input data for a given selection of LST bins, which is helpful for post-facto examination. Furthermore, we allow the output of 'REDUCEDCHAN' files, which contain all the data for all LST-bins, but for a reduced number of frequency channels.

4.3 Quick Overview of Results

One of the primary metrics used post-LST-binning to assess the quality of data and processing up til that point is the "excess variance" – that is, the ratio of measured variance (either over nights, or between neighbouring frequency channels) compared to the theoretical prediction based on auto-correlations. We give an example plot of the mean of this quantity over baseline groups, as a function of frequency, in Fig. 4. Note that the night-to-night variance is 43% larger than the theoretical expectation when averaged over all baselines and channels, though this is higher for `ee` polarizations, short baselines (< 60m) and sub-FM channels. Although the night-to-night variance is significantly inflated, the frequency-to-frequency variance, which is perhaps more important for our purposes, is only at a 20% mean excess, and only 8% when restricted to intra-sector baselines.

The use of sigma-clipping at the level of 4σ reduces the baseline-and-frequency-averaged night-to-night excess variance of redundantly-averaged data to 32%, though interestingly *increases* the channel-to-channel variance to 30%.

In Fig. 5 we show the equivalent plot for *non*-redundantly-averaged data. This indicates that the mean excess night-to-night variance above FM is stable at around 18%, with spikes around Orbcomm and the

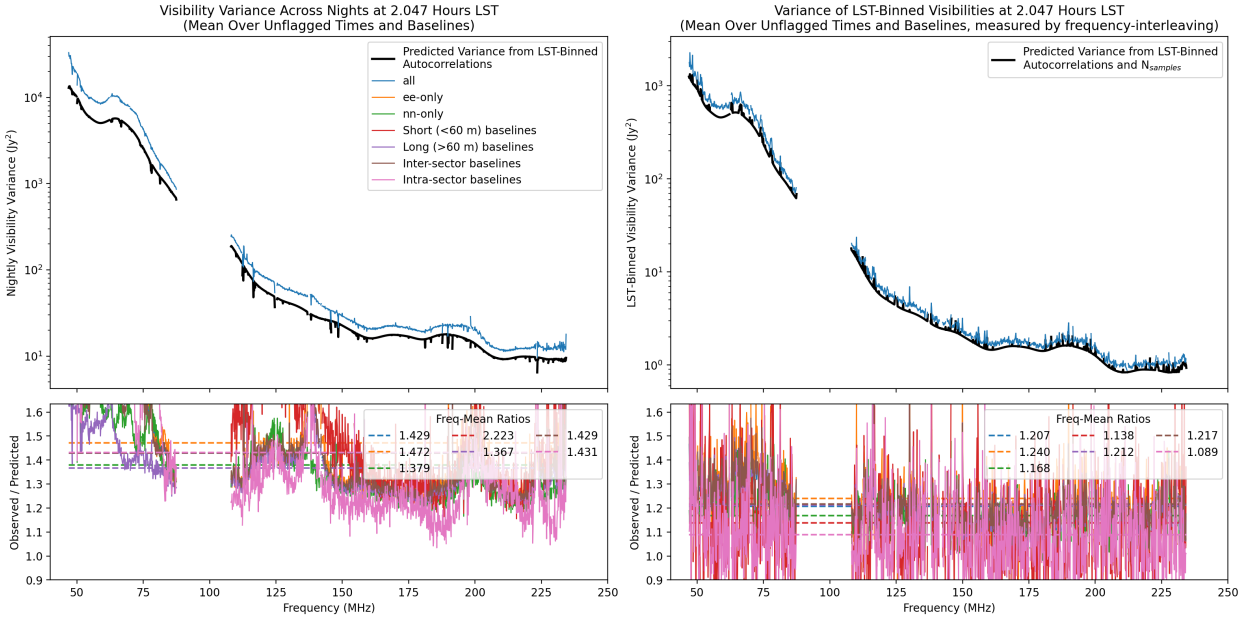

Figure 4: Mean excess variance (over baselines) as a function of frequency at LST∼2 hr, for the smooth-calibrated, redundantly-averaged data.

Run	Peak RAM	BL Chunk Size	Wall-time Per File	Storage Size
Red-averaged No Sigma-Clip	5.3 GB	880	3.9 min	350 GB
Red-averaged w/ Sigma-Clip	5.4 GB	880	3.2 min	350 GB
Non-Red-averaged No Sigma-Clip	19.7 GB	1500	49.5 min	6300 GB

Table 2: Gross performance summary for different kinds of LST-bin runs in this IDR. Wall-time per file is the average, obtained using `pipeline_status.py` from `hera_opm`. Peak RAM was obtained from in-script logging information (see https://github.com/hera-team/hera-cli-utils for the implementation) using `tracemalloc`. Peak RAM is mainly controlled by the baseline chunk size (and number of days binned). This run was not well-optimized, as it was allocated 40GB of RAM, and used only half.

known RFI channels at ∼ 200 MHz. The width of these spikes *may* indicate low-level RFI around the known channels, or some effect of smoothing. The channel-to-channel variance is at an excess of about 10% across the full band. The fact that the non-redundantly-averaged data has significantly lower excess variance probably indicates the presence of non-redundancies that should be identified prior to redundant averaging.

4.4 Performance and Bottlenecks

The new LST-binner improves performance (both wall-time and peak RAM usage) significantly. The primary cause of the speedup is the pre-configuration of the matched files, and the vectorization over baselines, with some speedup also coming from being able to partially read metadata on the input files (using `pyuvdata`'s new `FastUVH5Meta` class). The major improvement in peak RAM usage comes from the use of numpy arrays (i.e. vectorization) along with the pre-initialization and on-the-fly writing to the output files.

The major sink of time in the LST-binner is file I/O. Essentially the entire raw dataset needs to be read to perform LST-binning. This could be improved by using faster data readers, but these are not employed yet (due to the fact that they don't support reading partial times yet).

Table 2 summarizes the performance aspects of the LST-binner for H6C IDR2, while Table 3 breaks down the performance into components of the processing.

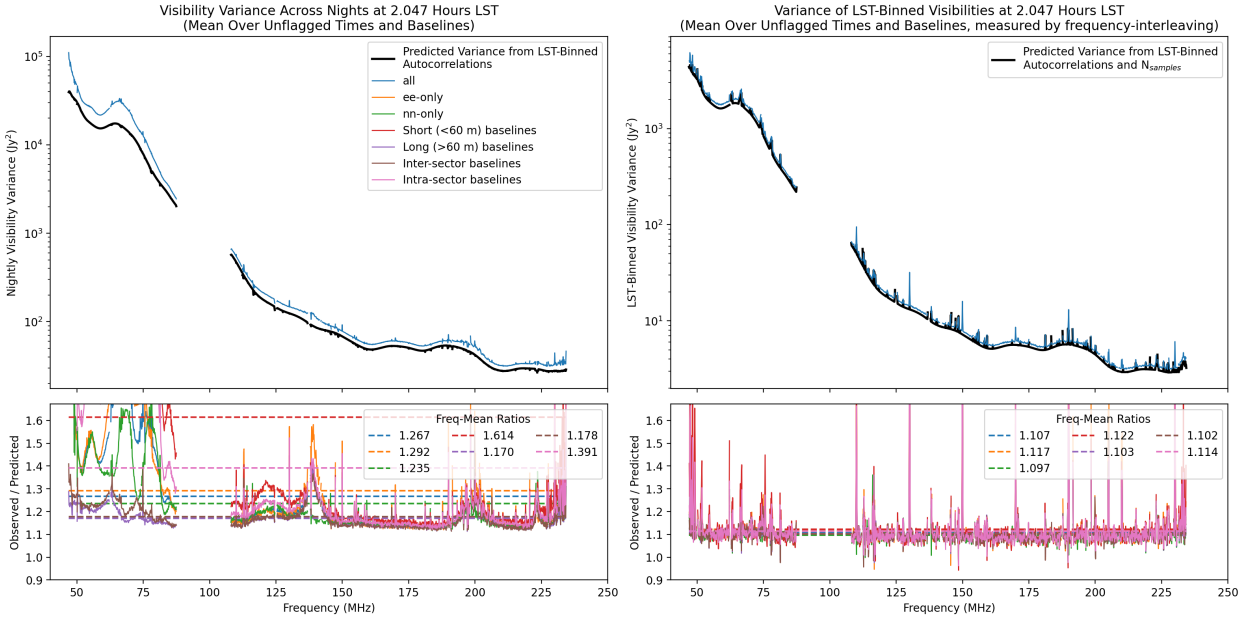

Figure 5: Mean excess variance (over baselines) as a function of frequency at LST∼2 hr, for the smooth-calibrated, *non*-redundantly-averaged data.

5 Summary of Data Products

Here we list as a quick reference all the data products available as part of this IDR.

5.1 Per-Night Data Products

All of the following are available either in `/lustre/aoc/projects/hera/h6c-analysis/IDR2/24598??` or in `/lustre/aoc/projects/hera/h6c-analysis/IDR2/notebooks`, which is linked here.

- `zen.24598??.?????.sum.calibration_notebook.html`: This is the per-file calibration notebook, which includes plots and tables detailing antenna flagging, preliminary RFI excision, and redundant and absolute calibration results. See subsection 3.2, subsection 3.3, and subsection 3.4 above for more details. It produces the following:
  - `zen.24598??.?????.sum.ant_metrics.hdf5`: Summary of information about flagging produced by `hera_qm.ant_metrics`, which looks for dead antennas, low-correlation, and cross-polarized antennas.
  - `zen.24598??.?????.sum.omni.calfits`: Gain solutions that include both redundant-baseline calibration and absolute calibration, as well as preliminary RFI flagging.[3]
  - `zen.24598??.?????.sum.ant_class.csv`: Summary of statistics used for antenna classification as good, bad, or suspect.
- `antenna_classification_summary_24598??.html`: This notebook summarizes the reasons antennas were flagged over a whole night. It was primarily used for commissioning, but it is trivial to reproduce it as part of the analysis pipeline.
- `full_day_antenna_flagging_24598??.html`: This notebook shows how and why antenna flagging was harmonized and expanded over a whole day, adding flags to often-but-not-always flagged antennas coming out of per-file processing. See subsection 3.2 above for more details. It produces the following files:

[3]It is called "`omni`" for historical reasons, but now includes a full calibration solution.

Stage	Step	red-avg Time [sec]	red-avg %	red-avg %	non-avg Time [sec]	non-avg %	non-avg %
Setup	Loading Config	55.6	17.0	**55.8**	53.8	1.5	**1.5**
	Config Redundancies	126.5	38.8		-	-	
Reading	Data Reading	36.0	11.0	**17.2**	1058.6	29.6	**54.7**
	Build DataContainer	20.2	6.2		931.4	26.1	
	Read Calfiles	-	-		65.5	1.8	
Work	Apply Calibration	-	-	**20.7**	200.5	5.6	**25.4**
	Rephasing/Aligning	5.1	1.6		203.3	5.8	
	Sigma-Clip	53.3	16.3		-	-	
	LST averaging	9.1	2.8		501.2	14.0	
Writing	Writing	10.6	3.3	**3.3**	29.2	0.8	**0.8**

Table 3: Timing of different steps of the LST-binning process, grouped into major stages. Note that the times for the redundantly-averaged case come from a 'golden LST', and so incur some extra time for writing the GOLDEN file. Those for the non-averaged case are for a regular file (which is the more common case). Note that the setup phase for the redundantly averaged data is disproportionately long, and should be the first aspect to be optimized in the future. In the non-averaged case, data reading is the dominant phase, as might be expected. However, the fact that a quarter of the total time is taken in converting the data into a DataContainer can surely be improved. Furthermore, using faster readers in the future that don't require reading nsamples arrays will speed this up.

  - `zen.24598??.?????.sum.antenna_flags.h5`: These per-file flags include whether an individual antenna-polarization was flagged for a given file, but the per-antenna waterfalls are either 100% flagged or 100% unflagged—information about RFI flags is available here.

- `full_day_rfi_24598??.html`: This notebook creates the single-full day RFI waterfall, looking for outliers in 2D-DPSS-filtered averaged autocorrelations. It documents the evolution of the flagging mask. See subsection 3.4 above for more details. It produces the following files:

  - `zen.24598??.?????.sum.flag_waterfall.h5`: This file, which complements the `.antenna_flags.h5` file, contains a single flagging waterfall for each file.

- `calibration_smoothing_24598??.html`: This notebook performs calibration smoothing, showing full-day calibration solutions before and after smoothing. It also looks for phase flips and prints out when it finds them. See subsection 3.3 above for more details. It produces the following files:

  - `zen.24598??.?????.sum.smooth.calfits`: These "final" calibration solutions include the full set of flagging above (`.antenna_flags.h5` and `.flag_waterfall.h5`), as well as calibration solutions constrained to be smooth in time and frequency.
  - `24598??_aposteriori_flags.yaml`: This YAML file contains a list of JD ranges, frequency ranges, and antennas that are 100% flagged over the day. This is used in post-processing to help make sure redundantly averaged data products have the same baseline keys.

- `zen.24598??.?????.sum.postprocessing_notebook.html`: This notebook visualizes redundantly-averaged visibilities and nsamples, including after delay filtering. See subsection 3.5 It also produces the following files:

  - `zen.24598??.?????.sum.abs_calibrated.red_avg.uvh5`: Redundantly averaged visibility sums with `.omni.calfits` gains applied, but with flags from `.antenna_flags.h5` and `.flag_waterfall.h5`.
  - `zen.24598??.?????.sum.smooth_calibrated.red_avg.uvh5`: Redundantly averaged visibility sums with the gains and flagged from `.smooth.calfits` applied.

- `zen.24598??.?????.sum.abs_calibrated.red_avg.dly_filt.uvh5`: Redundantly averaged visibility sums with the gains and flagged from `.omni.calfits` applied and then delay filtered. Autocorrelations are inpainted rather than filtered.
- `zen.24598??.?????.sum.smooth_calibrated.red_avg.dly_filt.uvh5`: Redundantly averaged visibility sums with the gains and flagged from `.smooth.calfits` applied and then delay filtered. Autocorrelations are inpainted rather than filtered.
- `zen.24598??.?????.sum.abs_calibrated.red_avg.inpaint.uvh5`: Redundantly averaged visibility sums with the gains and flagged from `.omni.calfits` applied and with flagged channels inpainted.
- `zen.24598??.?????.sum.smooth_calibrated.red_avg.inpaint.uvh5`: Redundantly averaged visibility sums with the gains and flagged from `.smooth.calfits` applied and with flagged channels inpainted.
- `zen.24598??.?????.sum.smooth_calibrated.avg_abs_all.uvh5`: Redundantly averaged visibility sums with the gains and flagged from `.smooth.calfits` applied, but then all visibility amplitudes are averaged together incoherently.
- `zen.24598??.?????.sum.smooth_calibrated.avg_abs_auto.uvh5`: Same as above, but with just autocorrelations.
- `zen.24598??.?????.sum.smooth_calibrated.avg_abs_cross.uvh5`: Same as above, but with just cross-correlations.

- `full_day_systematics_inspect_245986??.html`: This notebook looks for systematics by examining full-day waterfalls in frequency/time space, delay/time space, and delay/fringe-rate space before and after delay and fringe-rate filtering, as well as coherent and incoherent averaging and after forming pseudo-Stokes I and Q. See subsection 3.5 for details.

5.2 LST-Binned Data Products

All notebooks are available in `/lustre/aoc/projects/hera/h6c-analysis/IDR2/notebooks/lstbin-inspect` which is linked here. All data products are available in `/lustre/aoc/projects/hera/h6c-analysis/IDR2/lstbin-outputs/<CASE>/`, where `CASE` is one of;

- `nonavg` – non-redundantly-averaged sooth-calibrated data run without sigma-clipping.
- `redavg-smoothcal` – redundantly-averaged, absolutely-calibrated, smooth-calibrated, no sigma-clipping.
- `redavg-smoothcal-dlyfilt` – redundantly-averaged, absolutely-calibrated, smooth-calibrated, delay-filtered, no sigma-clipping.
- `redavg-smoothcal-inpaint` – redundantly-averaged, absolutely-calibrated, smooth-calibrated, frequency in-painted, no sigma-clipping.
- `redavg-smoothcal-sigclip` – redundantly-averaged, absolutely-calibrated, smooth-calibrated, with sigma-clipping.
- `redavg-abscal` – redundantly-averaged, absolutely-calibrated (but not smooth-calibrated) data, with no sigma-clipping.
- `redavg-abscal-dlyfilt` – redundantly-averaged, absolutely-calibrated (but not smooth-calibrated), delay-filtered, with no sigma-clipping.
- `redavg-abscal-inpaint` – redundantly-averaged, absolutely-calibrated (but not smooth-calibrated), frequency in-painted, with no sigma-clipping.

Each of the above `CASE` directories has the following files:

- `lstbin-config.toml` – an exact copy of the pipeline configuration file used to create this output data.

- `environment.yaml` – an exact copy of the conda environment used to run this data.
- `file-config.yaml` – the file-matching configuration of the LST-bin outputs, created by `build_makeflow_from_config.py` in `hera_opm`.
- `zen.LST.<LSTRADIAN>.sum.uvh5` – standard UVH5 file containing LST-averaged data for the two LST-bins starting at `LSTRADIAN`.
- `zen.STD.<LSTRADIAN>.sum.uvh5` – standard UVH5 file containing the night-to-night standard deviation (for both real and imag components) for the two LST-bins starting at `LSTRADIAN`.
- `zen.GOLDEN.<LSTRADIAN>.sum.uvh5` – standard UVH5 file containing the calibrated input visibilities for all nights within the LST-bin starting at `LSTRADIAN`.
- `zen.REDUCEDCHAN.<LSTRADIAN>.sum.uvh5` – standard UVH5 file containing the calibrated input visibilities for all nights within the LST-bin starting at `LSTRADIAN`, but only for channels 150, 750 and 1250.

6 Known Issues and Possible Future Improvements

H6C IDR 2.1 is not the final work on H6C analysis; it may not even be the final word on this particular data set. We are tracking a handful of issues for aspects of the analysis that are we currently think need further investigation and perhaps improvement.

- While the new `abscal` method discussed above is substantially more stable than our previous algorithm when calibrating against a simulation for all LST, it is still not perfect. As discussed starting here, there are still a few LSTs for frequencies where solutions go awry, likely due to bright sources near the horizon (especially Cass A and Cyg A). While we refined the algorithm to minimize this effect, an improved sky model might also improve the calibration. For now, calibration smoothing is expected to minimize the effect of these calibration errors.
- With a calibrated, LST-binned data set, we can use frequency-redundancy to form a NUCAL model of visibilities on baselines in certain frequently-sampled orientations. These visibilities are guaranteed to be smooth in a physically-motivated way. These could be useful for solving for (some of) the degeneracies of redundant-baseline calibration. Another possible approach is to take the NUCAL and go back to the original uncalibrated visbilities, starting calibration from scratch in a way designed to produce calibrated visbilities with minimal spectral structure. This technique should also allow us to find RFI.
- Relatedly, some work has been done to try to calibrate nights to one another before LST-binning. This `LSTcal` idea might help mitigate the so-called "excess variance" problem (see below).
- As it stands, either we flag a frequency and time for all antennas, or we flag an antenna for all times and frequencies (in a file, or sometimes over a whole night). We do not have antenna-dependent flag waterfalls. However, we see some evidence in the `full_day_rfi` notebooks that certain antennas have a lot more residual power after DPSS filtering of their autocorrelations than others. Whether this is a problem with the filtering or the antennas themselves, and whether it affects cross-correlation, is a matter than deserves further investigation.
- When plotting `redcal` χ^2 over the whole day (which we do in the `calibration_smoothing` notebooks), we see unflagged spikes in χ^2 that might be due to RFI. Or it could simply be that the calibration solution there was bad, but the visibilities are fine and `smooth_cal` will basically fix the problem. Whether we should be using χ^2 as an additional source of flagging, or whether it simply reveals some other issue that should be caught when inspecting visibilities, remains to be investigated.

- When performing per-night inpainting (which for now is only done on redundantly-averaged baselines), we use the same DPSS model that we subtract to perform delay filtering. This is at the horizon or 150 ns, whichever is larger. This removes much of the foregrounds, but we know residual foregrounds and systematics exist beyond that scale. No attempt is made to inpaint higher delay systematics, like the fringe-rate 0 mode or any of the mutual coupling, which might make those effects harder to filter off in LST-binned data. Perhaps we should be fringe-rate filtering before LST-binning? This remains an open research question.

- After LST-binning, the resulting LST-binned data is inspected with the lstbin-inspect notebook. One of the key metrics investigated by this notebook is that of 'excess variance', which measures the ratio of observed variance on a particular baseline-LST-channel compared to its theoretical variance, as derived from the autocorrelations and N_{samples}. Murray and Dillon, 2023b derived theoretical distributions of this ratio, and the LST-bin inspect notebook explores the relationship of the excess variance with properties of the baselines (eg. length, orientation) and frequency. The mean excess variance over all baselines averages $\sim 50\%$ for redundantly-averaged data, and $\sim 20\%$ for non-redundantly-averaged data, which is higher than we saw in H1C IDR3. The solution is most likely to be found in the problems mentioned above.

References

Abdurashidova, Zara, James E. Aguirre, Paul Alexander, et al. (2022a). "First Results from HERA Phase I: Upper Limits on the Epoch of Reionization 21 cm Power Spectrum". In: *The Astrophysical Journal* 925.2, p. 221. DOI: 10.3847/1538-4357/ac1c78. URL: https://doi.org/10.3847%2F1538-4357%2Fac1c78.

Abdurashidova, Zara, James E. Aguirre, Paul Alexander, et al. (2022b). "HERA Phase I Limits on the Cosmic 21 cm Signal: Constraints on Astrophysics and Cosmology during the Epoch of Reionization". In: *The Astrophysical Journal* 924.2, p. 51. DOI: 10.3847/1538-4357/ac2ffc. URL: https://doi.org/10.3847%2F1538-4357%2Fac2ffc.

Aguirre, J. E., S. G. Murray, R. Pascua, et al. (2021). "Validation of the HERA Phase I Epoch of Reionization 21 cm Power Spectrum Software Pipeline". In: *Ap.J. in review*.

Dillon, J. S. (2019). "HERA Memo #69: H1C Internal Data Release 2.2". reionization.org/memos.

Dillon, J. S., S. A. Kohn, A. R. Parsons, et al. (July 2018). "Polarized redundant-baseline calibration for 21 cm cosmology without adding spectral structure". In: MNRAS 477, pp. 5670–5681. DOI: 10.1093/mnras/sty1060. arXiv: 1712.07212 [astro-ph.IM].

Dillon, J. S. and Z. E. Martinot (2020). "HERA Memo #78: Absolute Calibration of H1C Data with RIMEz Simulations". reionization.org/memos.

Dillon, Joshua S., Max Lee, Zaki S. Ali, et al. (Oct. 2020). "Redundant-Baseline Calibration of the Hydrogen Epoch of Reionization Array". In: MNRAS. DOI: 10.1093/mnras/staa3001. arXiv: 2003.08399 [astro-ph.IM].

Ewall-Wice, Aaron, Nicholas Kern, Joshua S. Dillon, et al. (Jan. 2021). "DAYENU: a simple filter of smooth foregrounds for intensity mapping power spectra". In: MNRAS 500.4, pp. 5195–5213. DOI: 10.1093/mnras/staa3293. arXiv: 2004.11397 [astro-ph.CO].

Fagnoni, Nicolas, Eloy de Lera Acedo, Nick Drought, et al. (2021). "Design of the New Wideband Vivaldi Feed for the HERA Radio-Telescope Phase II". In: *IEEE Transactions on Antennas and Propagation* 69.12, pp. 8143–8157. DOI: 10.1109/tap.2021.3083788. URL: https://doi.org/10.1109%2Ftap.2021.3083788.

Heligenstein, W. and D. Jacos (2023). "HERA Memo #121: How often is lightning to blame for RFI in HERA data?" reionization.org/memos.

Josaitis, Alec T, Aaron Ewall-Wice, Nicolas Fagnoni, et al. (2022). "Array element coupling in radio interferometry I: a semi-analytic approach". In: *Monthly Notices of the Royal Astronomical Society* 514.2, pp. 1804–1827. DOI: 10.1093/mnras/stac916. URL: https://doi.org/10.1093%2Fmnras%2Fstac916.

Kern, Nicholas S., Joshua S. Dillon, Aaron R. Parsons, et al. (Feb. 2020). "Absolute Calibration Strategies for the Hydrogen Epoch of Reionization Array and Their Impact on the 21 cm Power Spectrum". In: ApJ 890.2, 122, p. 122. DOI: 10.3847/1538-4357/ab67bc. arXiv: 1910.12943 [astro-ph.IM].

Martinot, Zachary E (2022). "Improvements in Interferometric Data Modeling for the New Era of Radio Cosmology". In: URL: https://repository.upenn.edu/handle/20.500.14332/32091.

Murray, S. and J. S. Dillon (2023a). "HERA Memo #122: H6C Season A Priori Flag Summary". reionization.org/memos.

— (2023b). "HERA Memo #123: LST-Binning Statistics". reionization.org/memos.

Storer, Dara, Joshua S. Dillon, Daniel C. Jacobs, et al. (2022). "Automated Detection of Antenna Malfunctions in Large-N Interferometers: A Case Study With the Hydrogen Epoch of Reionization Array". In: *Radio Science* 57.1. DOI: 10.1029/2021rs007376. URL: https://doi.org/10.1029%2F2021rs007376.