HERA Memo #125
Recieved Oct 2 2023

HERA Team Memo: H6C Internal Data Release 2.2

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

October 2, 2023

1 Executive Summary

In this memo, we explain the updates in data analysis from H6C IDR 2.1, documented in Dillon and Murray (2023), to H6C IDR 2.2. The underlying data set is the same. This is a relatively minor revision and most of what was written there remains true, including the location of most data products on `lustre`. As such, this memo is not an exhaustive description of the IDR and relies on that memo for context. The biggest changes are an additional round of RFI identification (with a new algorithm) and a change to how inpainted data products are produced and LST-binned.

2 Per-Night Analysis

In this section, we detail all the changes since H6C IDR 2.1 to per-night analysis—i.e. everything before LST-binning.

2.1 Pipeline Updates

As before, 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`. The new pipeline (see Figure 1) is restructured to accommodate a second round of RFI-flagging, as we discuss in subsection 2.3.

2.2 Updates to Per-File Calibration

During redundant calibration, antennas with high-χ^2 per antenna are removed iteratively, stopping only after all such antennas are flagged. Previously this was done by imposing an absolute limit on the median value of χ^2 over the band. Now, we are doing this using the mean of over unflagged channels. This will hopefully expose and flag antennas that are particularly bad over certain sub-bands and would have been useful to catching the error mode where data packets were mislabeled for certain X-engines in either the even or odd data (a failure mode we flag on separately).

2.3 RFI Flagging on Redundantly-Averaged, Delay-Filtered Cross-Correlations

The most important update in this IDR is an additional RFI flagging step. Previously, RFI excision was performed in two steps using only autocorrelations, as explained in Dillon and Murray (2023). The first was done per-file: autocorrelations for antennas passing basic checks were filtered using 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.

The second step happens on all autocorrelations at once. Using those flags as a starting point, we develop a 2D DPSS model on 5 MHz and 450 s scales of the average autocorrelation of the best-behaved antennas. This second step is specifically designed to catch broadband RFI that is particularly hard to catch on a per-file basis. It also has the ability to remove prior flags, since it uses the prior flags as weights for DPSS fitting but ultimately develops its own z-score to cut on—both for individual waterfall pixels as well as whole times and frequencies.

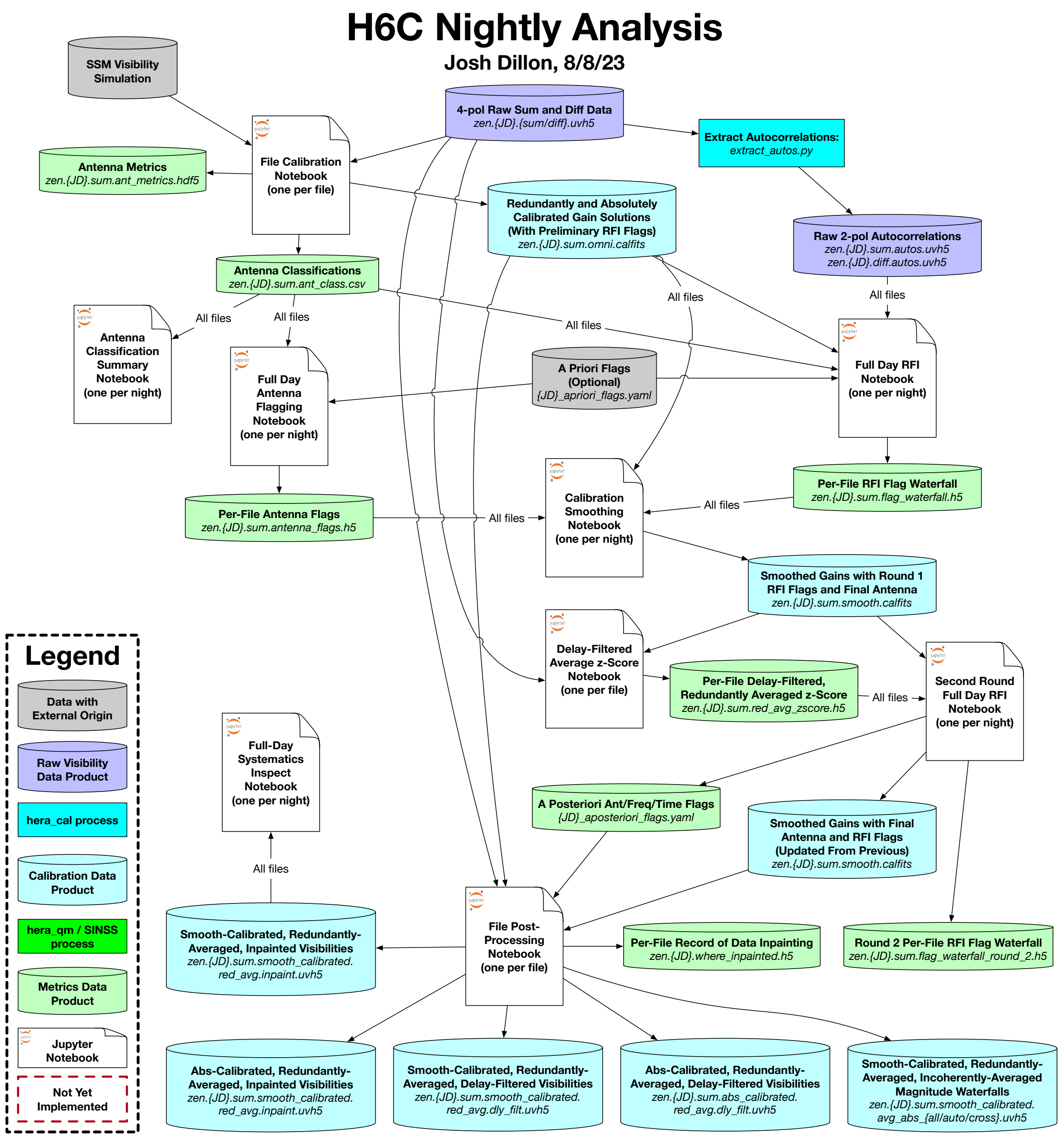
H6C Nightly Analysis
Josh Dillon, 8/8/23
SSM Visibility Simulation
4-pol Raw Sum and Diff Data zen.{JD}.{sum/diff}.uvh5
Extract Autocorrelations: extract_autos.py
Antenna Metrics zen.{JD}.sum.ant_metrics.hdf5
File Calibration Notebook (one per file)
Redundantly and Absolutely Calibrated Gain Solutions (With Preliminary RFI Flags) zen.{JD}.sum.omni.calfits
Raw 2-pol Autocorrelations zen.{JD}.sum.autos.uvh5 zen.{JD}.diff.autos.uvh5
Antenna Classifications zen.{JD}.sum.ant_class.csv
All files
Antenna Classification Summary Notebook (one per night)
Full Day Antenna Flagging Notebook (one per night)
A Priori Flags (Optional) {JD}_apriori_flags.yaml
Full Day RFI Notebook (one per night)
Per-File RFI Flag Waterfall zen.{JD}.sum.flag_waterfall.h5
Per-File Antenna Flags zen.{JD}.sum.antenna_flags.h5
Calibration Smoothing Notebook (one per night)
Smoothed Gains with Round 1 RFI Flags and Final Antenna zen.{JD}.sum.smooth.calfits
Delay-Filtered Average z-Score Notebook (one per file)
Per-File Delay-Filtered, Redundantly Averaged z-Score zen.{JD}.sum.red_avg_zscore.h5
Second Round Full Day RFI Notebook (one per night)
Full-Day Systematics Inspect Notebook (one per night)
A Posteriori Ant/Freq/Time Flags {JD}_aposteriori_flags.yaml
Smoothed Gains with Final Antenna and RFI Flags (Updated From Previous) zen.{JD}.sum.smooth.calfits
Smooth-Calibrated, Redundantly-Averaged, Inpainted Visibilities zen.{JD}.sum.smooth_calibrated.red_avg.inpaint.uvh5
File Post-Processing Notebook (one per file)
Per-File Record of Data Inpainting zen.{JD}.where_inpainted.h5
Round 2 Per-File RFI Flag Waterfall zen.{JD}.sum.flag_waterfall_round_2.h5
Abs-Calibrated, Redundantly-Averaged, Inpainted Visibilities zen.{JD}.sum.smooth_calibrated.red_avg.inpaint.uvh5
Smooth-Calibrated, Redundantly-Averaged, Delay-Filtered Visibilities zen.{JD}.sum.smooth_calibrated.red_avg.dly_filt.uvh5
Abs-Calibrated, Redundantly-Averaged, Delay-Filtered Visibilities zen.{JD}.sum.abs_calibrated.red_avg.dly_filt.uvh5
Smooth-Calibrated, Redundantly-Averaged, Incoherently-Averaged Magnitude Waterfalls zen.{JD}.sum.smooth_calibrated.avg_abs_{all/auto/cross}.uvh5
Legend
Data with External Origin
Raw Visibility Data Product
hera_cal process
Calibration Data Product
hera_qm / SINSS process
Metrics Data Product
Jupyter Notebook
Not Yet Implemented

Figure 1: The update pipeline for H6C IDR 2.2. Just like in IDR 2.1 (Dillon and Murray, 2023), 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). The Delay-Filtered Average z-Score and Second Round Full Day RFI Flagging notebooks are new in H6C IDR 2.2.

This IDR adds a new RFI flagging routine spiritually similar to the first two. It begins with data with `smooth_cal` applied (and thus all previous antenna and RFI flags). On a per-file basis, this data is coherently

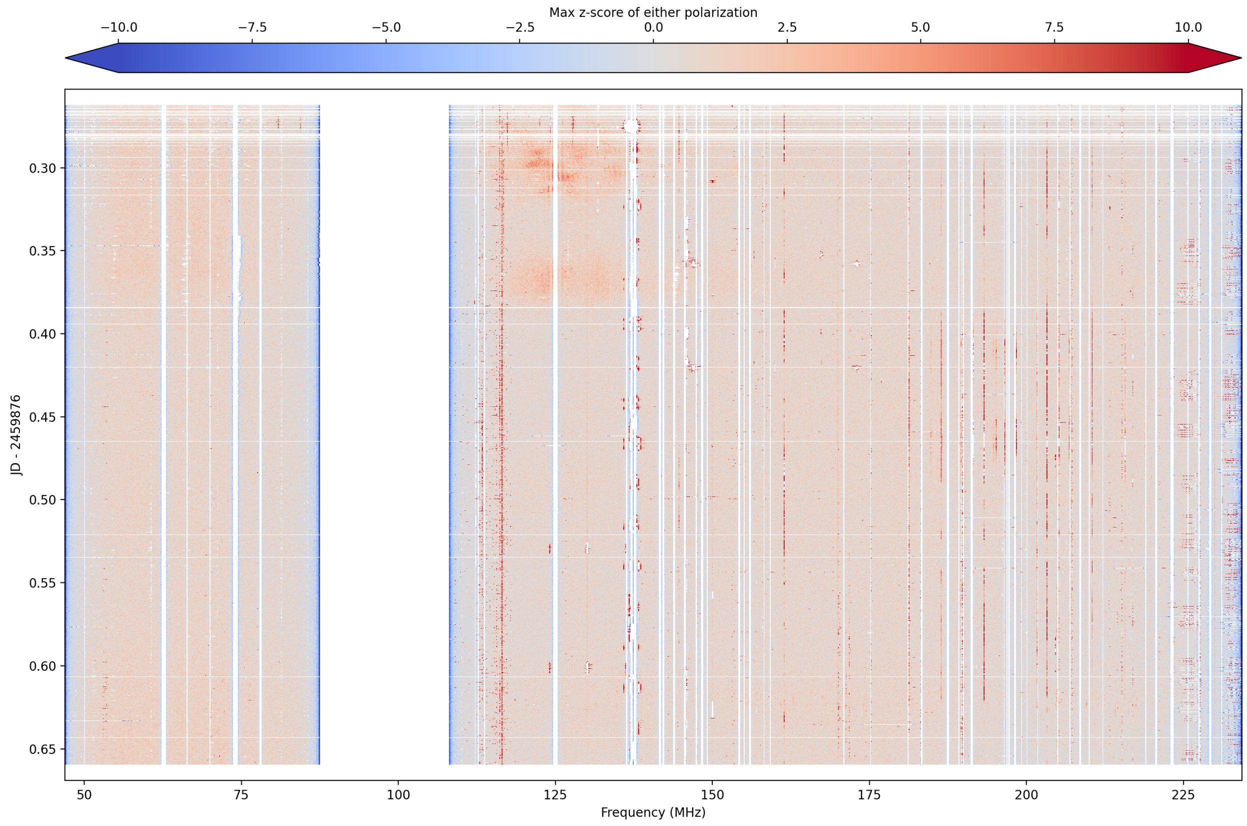

Figure 2: Here we show the z-score metric used in an additional round of RFI flagging, the biggest change in this IDR from H6C IDR 2.1. Specifically, we show the larger of the two polarizations' z-scores for each pixel. By redundantly-averaging, delay-filtering, and then incoherently averaging across baseline groups, we can get a very sensitive metric of residual structure in frequency-space. It highlights a wide variety of RFI types. Most of it is narrowband, especially digital audio above ∼180 MHz, though there also some quite "speckled" RFI. We also see some of what looks like overflow from ORBCOMM satellite passes. Interestingly, the low band (below FM) is pretty clean. The negative z-scores near prior flags comes from the fact that the DPSS filter overfits noise near prior flags, since it is entirely unconstrained in flagged regions.

averaged within each redundant-baseline group. We next remove redundantly-averaged groups with lengths greater than 150 m (500 ns light travel time) and groups whose number of unflagged baselines is less than 5% of the most-redundant group. The remainder are DPSS filtered at 500 ns. The absolute values of the residuals are inverse-variance-weighted-averaged (i.e. inverse N_{samples} weighted) across baselines. This result is Rayleigh distributed, so we can turn it into a z-score by subtracting the expected mean and dividing by the expected noise. We also account for the fact that a 500 ns filter removes a significant fraction of the noise, so we multiply the predicted variance by the fraction of DPSS modes excluded from the filter. In delay_filtered_average_zscore this is all done for each polarization individually and then saved as a waterfall metric-type `UVFlag` file.

Next, in full_day_rfi_round_2 we bring together all those `UVFlag` files. Since this z-score tends not to peak at 0 (the underlying data are not Gaussian), we subtract a single constant from each polarization's waterfall to set the median value to 0. The result for 2459876 is shown in Figure 2. Using this waterfall, we identify pixels to flag with z-scores, as well as whole times and frequencies to flag. The algorithm is very similar to that used in the older full_day_rfi notebooks: first we flag 5σ outliers, then 4σ outliers that neighbor current flags, then any channel or time that averages a z-score greater than 1. This is done on each polarization independently, and then the final flags are OR'ed together. Finally, we flag any channels that are more than 25% flagged and any times that are more than 10% flagged.

2.4 Updates to Inpainting

In H6C IDR 2.1, we produced redundantly-averaged inpainted and delay-filtered data products for every file. This filtering was done at the horizon delay or 150 ns, whichever was larger. In this IDR, we are now separately performing delay filtering (still with a minimum delay of 150 ns) and inpainting, now with a minimum delay of 500 ns, which was picked to allow inpainting of systematics like mutual coupling and cross-talk.

Expanding the range of delays inpainted, combined with an the additional flags described in subsection 2.3, led to another challenge: large flag gaps are difficult to inpaint over. They often result in "pop-ups" where over-fitting noise at the edges creates very large amplitude inpainting results in the gap. We addressed this concern in two ways. First, we imposed a maximum gap size of twice the frequency scale set by the inverse of the inpainting delay. For most baselines, that delay is 500 ns, so that maximum inpaintable gap size 32 channels, just under 4 MHz. For redundantly-averaged baselines groups with gaps larger than that, the entire affected sub-band (either above or below FM, or both) is flagged. This happens relatively rarely, except on 2459866, which had a couple of very bright, rather wide RFI events around 190 MHz. This additional flagging is not applied to the delay-filtered data products.

Second, we added a Tikhonov regularization (also known as ridge regression) to the least-squares fitting of DPSS modes that penalizes large amplitude coefficients. By adding $10^{-3}\mathbf{I}$ to the diagonal of DPSS fitting matrix (which was already normalized to be all 1s on the diagonal). This prevents "pop-ups" and thus minimizes the impact of large channel gaps in inpainted data. However, that ad hoc parameter choice, combined with the fact that the data are very poorly constrained in the widest gaps, leaves us in a position where the inpainted data may be worse than just not including that day in the LST-average. At some point, perhaps it is wiser to accept some discontinuity in the sampling from day to day? This remains a major open question if we want to move forward with inpainting. Perhaps constrained realizations, motivated by the statistics of real data, may be a better approach?

Separately, we have condensed the outputs of the inpainted with the redundantly-averaged file that was neither inpainted nor flagged. This inpainted output file retains flags where inpainting was performed, but we separately output a `UVFlag` describing where inpainting was performed, as we describe in section 4.

3 LST-Binning

Before summarizing the updates to LST-binning, we note that the full python environment specifications in which the LST-binning data products were produced can be found in `/lustre/aoc/projects/hera/h6c-analysis/IDR2/lstbin-outputs/<CASE>/environment.yaml`. Some of the more important package versions are: `pyuvdata==2.4.0`, `hera-calibration==3.4.0`, `hera_qm==2.1.2`.

3.1 Improved Accounting of Inpainted Data

This release contains some updates to LST-binning, most importantly related to properly propagating statistics for inpainted data. Previously, every datum was considered to be in one of two states: flagged or unflagged, i.e. $f_i \in \{0, 1\}$. Letting V_i be *either* a redundantly-averaged visibility *or* an individual visibility, and letting n_i be the number of visibility samples in that visibility (i.e. $n_i \geq 1$ for the redundantly-averaged case, and $n_i = 1$ for the non-averaged case)[1], the LST-binned visibility was computed as

$$\bar{V} = \frac{\sum_i V_i n_i f_i}{\sum_i n_i f_i}, \tag{1}$$

which is a minimum-variance statistic if we assume that the intrinsic variance of each (non-redundantly-averaged) visibility is constant over all nights i. In this case, the variance of the LST-binned visibility is estimated as

$$S^2 = \frac{\sum_i (V_i - \bar{V})^2 n_i f_i}{\sum_i n_i f_i}. \tag{2}$$

[1]Note that n_i is never zero for either case, which follows from the constraint that n is uniform over frequency. In practice, baseline-pol-times that are fully flagged *before* redundant averaging can just be treated as if they do not exist. If all baselines in a group are fully flagged before averaging, that group is treated as if it does not exist in the final product

These equations are presented in Murray and Dillon, 2023b, along with a distribution for the *excess variance*, $\gamma = S^2/\langle S^2 \rangle$. Notice, however, that while n_i is constrained to be spectrally uniform, f_i is not. This can result in spectral structure in the LST-binned visibility. In IDR2.2, for *redundantly-averaged visibilities*, we enable an 'inpaint-mode', in which (most of) the flagged visibilities are *replaced* by inpainted smoothly-modeled visibilities. Let $p_i \in \{0, 1\}$ represent the 'inpainted state' of the visibility, i.e. $p_i = 0$ if the visibility is not inpainted and 1 if it is. Note that by definition all inpainted data has $f_i = 0$. In IDR2.2, we have a single per-night file (with extension `.inpaint.uvh5`, cf. section 4) that contains the *measured* data where $f_i = 1$ and the inpainted data where $f_i = 0$. In this file the `flag_array` represents this f_i. Thus, we can compute the traditional LST-binned average (as in IDR2.1, and presented above) using just this file – the inpainted data are ignored as they are flagged anyway. However, we also use a supplemental `UVFlag` file (again, cf. section 4) specifying p_i. If these files are provided to the LST-binner, it can use the new statistics:

$$\bar{V}_{\rm inpt} = \frac{\sum_i V_i n_i (f_i + p_i)}{\sum_i n_i (f_i + p_i)}, \tag{3}$$

and

$$S^2_{\rm inpt} = \frac{\sum_i (V_i - \bar{V})^2 n_i (f_i + p_i)}{\sum_i n_i (f_i + p_i)}. \tag{4}$$

However, note that the *expectation* of S^2 (again, under the assumption of uniform intrinsic variance per-visibility) is

$$\langle S^2_{\rm inpt} \rangle = \sigma^2 \frac{(\sum_i f_i) - 1}{\sum_i n_i f_i}, \tag{5}$$

which is equivalent to the expected variance of the non-inpainted data alone. Thus, we propagate an $N_{\rm samples} = \sum_i n_i f_i$, *not* $\sum_i n_i (f_i + p_i)$.

In practice the LST-binner can output both flagged-mode and inpainted-mode visibilities in a single run (requiring only one read of the data). These LST-binner updates were implemented in `hera-cal` PRs 907 and 912.

3.2 Other LST-Binning Updates

Another addition to the LST-binner is that it now has the capability to write out the median (and MAD) of the LST-bin (in addition to the mean and standard deviation). This could be useful for investigating the effects of one-shot sigma-clipping in future analyses.

Finally, the LST-binner now attempts to automatically choose the best LST branch-cut for writing out files. To do this, it sorts the non-empty LST-bins, and finds the largest gap, and uses the LST just after this gap as the branch cut. This LST becomes the lowest used in output filenames, such that lower LSTs are wrapped around 2π to become higher. Also, the JD's used in the `UVData` files to represent each LST bin start at this branch cut.

4 Summary of New, Modified, or Removed Data Products

4.1 Per-Night Data Products

Due to the changes in the pipeline, the following output files were added:

- `zen.24598??.?????.sum.flag_waterfall_round_2.h5`: This contains the flags waterfall generated from RFI flagging on redundantly-averaged, delay-filtered, incoherently averaged data. It will also contain all previous flags.

- `zen.24598??.?????.where_inpainted.h5`: This baseline-type `UVFlag` file says which times and frequencies inpainting was performed for in the redundantly-averaged files. This file enables us to use the `red_avg.inpaint.uvh5` files for LST-binning both with and without inpainting.

The following output files have had their meaning changed:

- `zen.24598??.?????.sum.smooth.calfits`: This file gets updated in place with the round 2 flag waterfall.
- `zen.24598??.?????.sum.smooth_calibrated.red_avg.inpaint.uvh5`: This file has has inpainted data replacing flagged channels wherever the `where_inpainted.h5` indicates (generally everywhere except FM or completely flagged integrations/baselines). Those channels remain flagged. N_{samples} contains the number of baselines in each redundant group and will be spectrally constant.
- `zen.24598??.?????.sum.abs_calibrated.red_avg.inpaint.uvh5`: Same as above.

The following output files were removed:

- `zen.24598??.?????.sum.smooth_calibrated.red_avg.uvh5`
- `zen.24598??.?????.sum.abs_calibrated.red_avg.uvh5`

4.2 LST-Binned Data Products

Recall that all data products are available in `/lustre/aoc/projects/hera/h6c-analysis/IDR2/lstbin-outputs/<CASE>/`. The following `CASE`s have been **removed**:

- `redavg-smoothcal-sigclip`: it is not clear how to apply sigma-clipping in a way that doesn't jeopardize spectral smoothness, so we have removed this case for this IDR.

The following `CASE`s have been **moved**:

- `redavg-smoothcal` → `redavg-smoothcal/flagged`: this represents the case in which flagged data is not inpainted.
- `redavg-smoothcal-inpaint` → `redavg-smoothcal/inpaint`: the case in which inpainted data is included in the average (but not N_{samples}, cf. §3)
- `redavg-abscal` → `redavg-abscal/flagged`: this represents the case in which flagged data is not inpainted.
- `redavg-abs-inpaint` → `redavg-abscal/inpaint`: the case in which inpainted data is included in the average (but not N_{samples}, cf. §3)

In each redundantly-averaged `CASE` directory, the following new files have been **added**:

- `zen.MED.<LSTRADIAN>.sum.uvh5` – standard UVH5 file containing the median of LST-bins for the two LST-bins starting at `LSTRADIAN`. Not added for case `nonavg` (due to data volume considerations).
- `zen.MAD.<LSTRADIAN>.sum.uvh5` – standard UVH5 file containing the median-absolute-deviation of LST-bins for the two LST-bins starting at `LSTRADIAN`. Not added for case `nonavg` (due to data volume considerations).

References

Dillon, J. S. and S. Murray (2023). "HERA Memo #124: H6C Internal Data Release 2.1". reionization.org/memos.

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]`.

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.