HERA Memo #131
Posted 4 Sept 2024

HERA Team Memo: H6C Internal Data Release 2.3

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

August 28, 2024

1 Executive Summary

In this memo, we explain the updates in data analysis from H6C IDR 2.2, documented in Dillon and Murray (2023b), to H6C IDR 2.3. H6C IDR 2.2 was itself an update from 2.1 (Dillon and Murray, 2023a), the first release of reduced data products from 14 nights of data from October 2022. All three IDRs have the same underlying data. Much of what was written in those two memos 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 those memos for context. The biggest analysis changes are:

- improved antenna flagging;
- relative phase calibration of e- and n-polarized antennas;
- inclusion of redundantly-averaged en and ne visibilities;
- change to using the `"sum"` pseudo-Stokes polarization convention
- explicit recording of which baselines actually contributed to each redundant average;
- LST-stacking as a notebook with plotting;
- re-calibration (and frequency smoothing) of abscal degrees of freedom during LST-stacking (i.e. `lst_cal`); and
- simultaneous inpainting and LST-stacking, using other nights to inform the inpainting of each individual night.

2 Per-Night Analysis

In this section, we detail all the changes since H6C IDR 2.2 to per-night analysis—i.e. everything before LST-stacking.

2.1 Pipeline Updates

As before, the pipeline (see Figure 1) is defined by a single `.toml` file, which interfaces with shell scripts that run the various analysis steps using `hera_opm` and `makeflow`. The only real change to the architecture of files and notebooks is the change to the outputs of `file_postprocessing` (see subsection 2.5 and section 5 for more details. As before, 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/`. Older notebooks are still available at `https://data.nrao.edu/hera/Notebooks/H6C_IDR2/IDR2_2/` and `https://data.nrao.edu/hera/Notebooks/H6C_IDR2/IDR2_1/`.

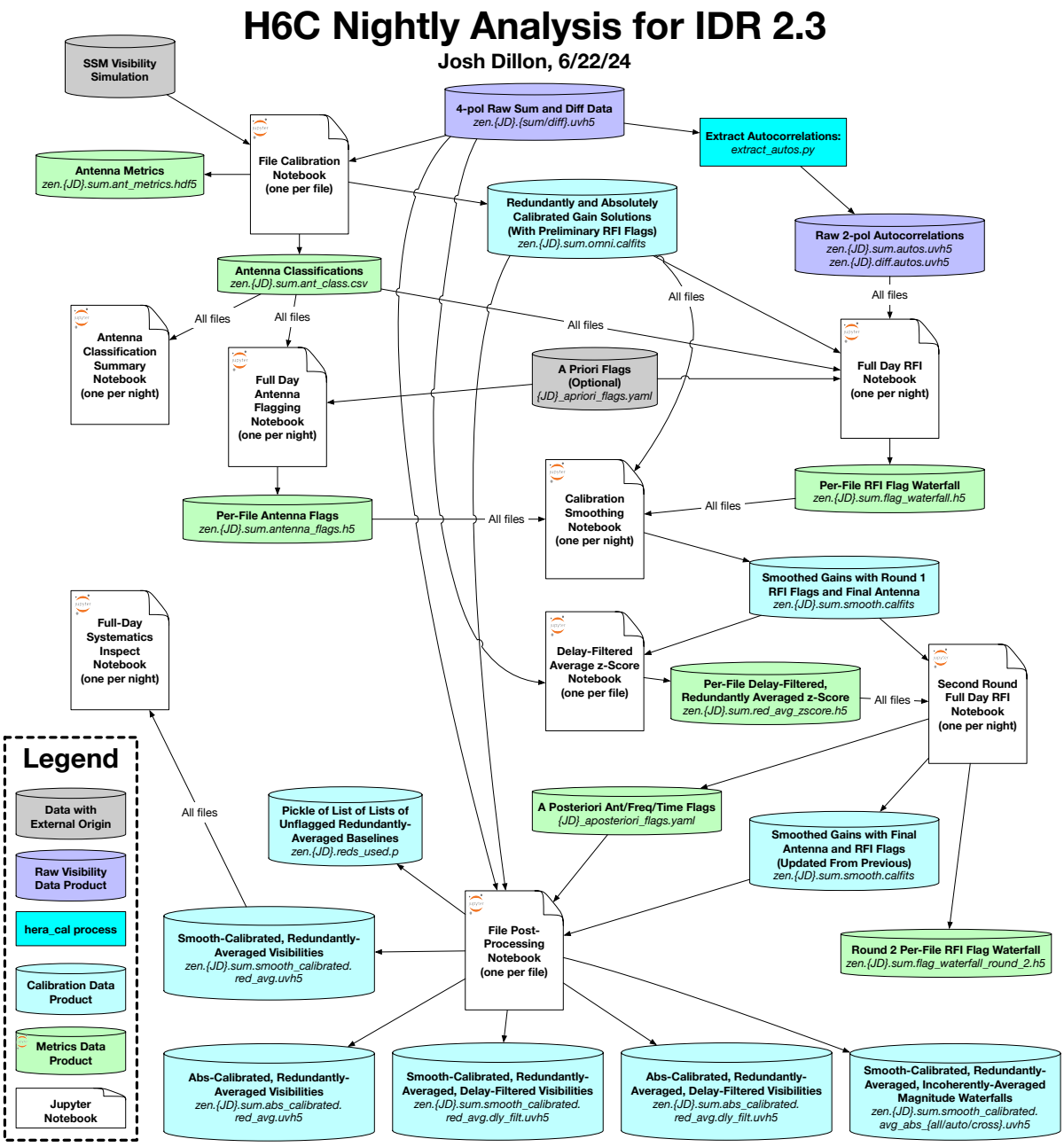

Figure 1: The updated pipeline for H6C IDR 2.3. Just like in IDRs 2.1 and 2.2 (Dillon and Murray, 2023a,b), the per-day analysis is centered on a series of notebooks, either one per file or just one per night, each of which both performs an analysis task and visualizes representative results in plots 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 (2023).

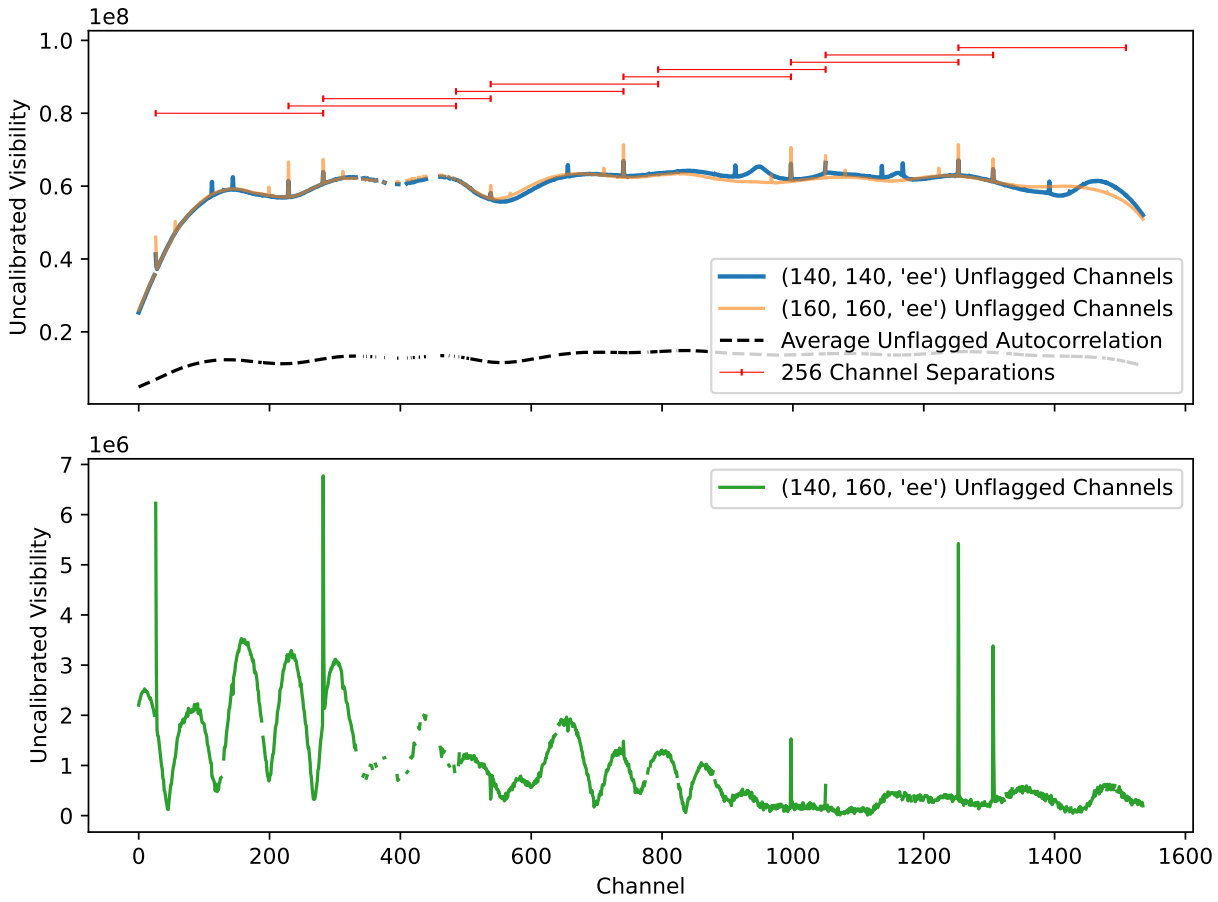

Figure 2: On 2459861, two antennas passing previous checks exhibited strong RFI features in their autocorrelation (top panel) that were not being flagged because RFI was not seen in those channels on almost any other antenna. In some of the channels where these spikes were coincident, the cross-correlation (bottom panel) also showed clear RFI that was missed by previous flagging routines and was clearly discernible in even in deeply-integrated data products. The RFI appears to be due to aliased out-of-band RFI which appears on these antennas in part because their gains are both quite high—note the amplitude of the averaged unflagged autocorrelation (black dashed line). The key piece of evidence supporting this hypothesis is the precise 256 channel separation between pairs of RFI spikes (red lines).

2.2 Updates to Per-File Calibration and Antenna Flagging

2.2.1 Antenna Flagging for Excess RFI

There have been a number of updates to how antennas are flagged on a per-file basis in the `file_calibration` notebook. These were largely made in response to the discovery of RFI in autocorrelations also being discovered in cross-correlations. This effect was initially found in highly-reduced data on a particular baseline separation, and then traced back to a handful of channels on a specific baseline (`140,160,"ee"`) on a specific day (2459861). We believe this is aliased RFI from out of band appearing on antennas with relatively high amplitudes. When those RFI spikes are coincident between antennas, the spikes can also be seen in cross-correlations. See Figure 2 for more details.

As the pipeline was designed, it was previously believed that spikes in autocorrelations that were not visible across the majority of the array were likely not affecting cross-correlations. Now it appears that they can, in rare cases. Previously, antenna flagging for excess RFI (or really any spectral structure) in autocorrelations was based on DPSS filtering each individual auto and then flagging antennas based on

the number of additional channels they would want flagged, regardless of the magnitude of that deviation from smoothness. That is why the large RFI spikes seen in Figure 2 did not result in those antennas being flagged—they were good other than in those few channels were they were very bad.

The difficulty of taking RFI amplitude into account is that now the metric on which individual antennas are flagged is rather sensitive to the RFI flagging mask applied to the array as a whole. The reverse is also true: the overall RFI flagging mask depends on which antennas get included in the average of unflagged autocorrelations. We therefore use an iterative process for determining the overall RFI flagging mask and the antenna classification based on both excess RFI and autocorrelation shape:

1. Perform all previous antenna flagging based on `ant_metrics`, excess zeros in even or odd visibilities, non-noise-like diffs, and autocorrelation power and slope.
2. Identify very strong (20σ) outlier antennas using neighboring channel differences.
3. Use a 300 ns DPSS filter on the average unflagged autocorrelation to identify 4σ outliers that define the initial global RFI mask.
4. Use that mask to DPSS filter each autocorrelation individually, dividing each result by the noise on the autocorrelation, and then taking the RMS over time and frequency to get a per-antenna z-score.
5. In stages, take the set of A) best half of the unflagged antennas (in terms of z-score), B) the best half of unflagged antennas or all those labeled "good" ($z < 1.5$), whichever is more antennas, or C) all antennas labeled "good" or "suspect" ($z < 2$). Use them to compute a new set of global RFI flags using averaged unflagged autos.
6. Perform antenna classification for excess RFI (or other spectral structure) and for shape deviation from mean unflagged autocorrelation (after a scalar amplitude correction).
7. If not converged on RFI and antenna flags, go back to step 3
8. When convergence of antenna flags and the RFI mask is reached, advance from stage A to B to C as detailed in step 5 and resume iteration at step 3.

The result of this somehwat convoluted process is that a handful of antennas that were not (consistently) flagged in IDR 2.2 are now flagged (see Figure 3). While a bit convoluted, this algorithm relatively quickly determines which antennas were outliers in unique ways (and should be flagged) and which times and channels were actually contaminated with low-level RFI across the array (and should be excluded from consideration as to whether an antenna ought to be excluded).

2.2.2 Antenna Flagging for Excess χ^2

We have also noticed that some antennas that are still calibratable were being flagged for high χ^2 per antenna, often as a consequence of being near antennas with even higher χ^2 per antenna. This problem got worse due to a change in IDR 2.2 where the χ^2 per antenna summary statistic went from being a median over the band to a mean over unflagged channels. We have addressed this issue by introducing a `OC_MAX_CHISQ_FLAGGING_DYNAMIC_RANGE` parameter, which we've set to 1.5. That means that antennas whose χ^2 is less than the worst antenna's χ^2 are set to "suspect" rather than "bad," meaning that they are kept for the next set of `omnical` iterations. If they are truly bad, they'll be caught in future and flagged. But if even worse neighboring antennas dominate their χ^2, they may end up meeting the overall normalized χ^2 per antenna cut of 3.0 and thus be kept.

2.2.3 Relative Phase Calibration of East and North Polarizations

Following antenna flagging and both redundant and absolute calibration, we now also calibrate the relative phase between the e- and n-polarized antennas. This is a degeneracy of "2-pol" redundant-calibration schemes (Dillon et al., 2018) and since only ee- and nn-polarized visibilities are used for `redcal`, it needs absolute calibration with reference to a sky model.

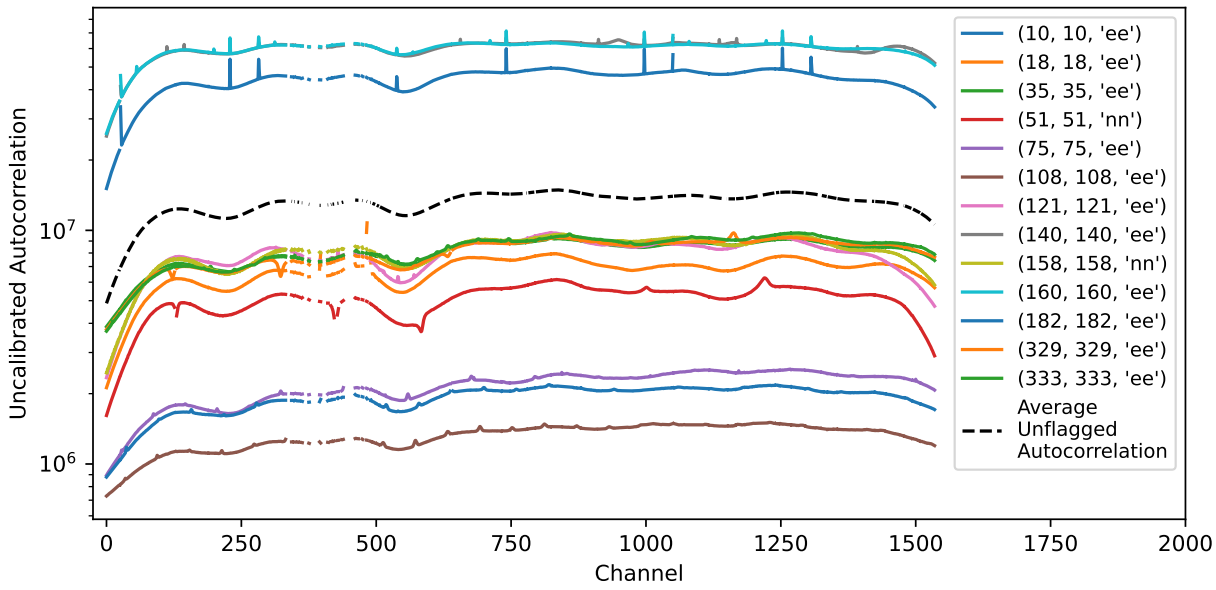

Figure 3: These autocorrelations are flagged due to excess spectral structure (sometimes RFI, sometimes something else) or due to a shape too dissimilar to the averaged unflagged autocorrelation. With the exception of the RFI seen in Figure 2, none of the spectral structure seen here has been definitiviely identified in the cross-correlations. Flagging these antennas is therefore a conservative choice, though one that makes subsequent RFI flagging on autocorrelations (or array-averaged autocorrelations) easier because they simply have less spectral structure.

While the sky model we have used thus far (Dillon and Murray, 2023a) is unpolarized (Stokes Q, U, and V are all identically zero), the *visibility* model has non-zero en and ne components because the beam model is faithful to the intrinsic polarization of an antenna's response. As long as the average polarization fraction, defined as $\sqrt{Q^2+U^2+V^2}/I$ is relatively small ($\sim$1%), then en and ne visibilities will have comparatively large contributions from the imprint of Stokes-I on the polarized beam responses. While we might expect our visibility model to be less accurate for en and ne visibilities than for ee and nn visibilities, it may be accurate enough to calibrate this one degree of freedom.

More precisely, let us begin by defining V_{ij}^{ab} to be the per-pol calibrated data, i.e. redundantly calibrated and absolutely calibrated against the sky model for each of ee and nn individually. Here a and b are polarization indices that run over East-West (e) and North-South (n), and i, j are antenna indices. Likewise, denote the abscal model by $\overline{V}_{ij}^{ab}$. If previous calibration is done well and if the sky is sufficiently unpolarized, then the remaining phase differences between the data and the model should be, on average, due to the final phase degeneracy. In other words, the difference between the data and the model (after calibrating each polarization separately) is roughly

$$V_{ij}^{ab} \approx e^{i(\phi_a - \phi_b)} \overline{V}_{ij}^{ab} \tag{1}$$

Recall that we only need to fix the relative phase between en and ne, so we define the difference parameter $\Delta = \phi_e - \phi_n$. To set Δ we define a least-squares objective function simply by summing the differences in Equation 1 over baselines ij and cross-polarizations en and ne

$$\mathcal{L}(\Delta) = \sum_{ij} w_{ij} \left(\left| V_{ij}^{en} - e^{i\Delta} \overline{V}_{ij}^{en} \right|^2 + \left| V_{ij}^{ne} - e^{-i\Delta} \overline{V}_{ij}^{ne} \right|^2 \right), \tag{2}$$

where w_{ij} is an inverse noise variance weight. Currently the baseline set is the same as is used in the per-pol abscal steps which excludes both the cross-feed correlations (i.e., $i = j$) as well as baselines longer than 140 m.

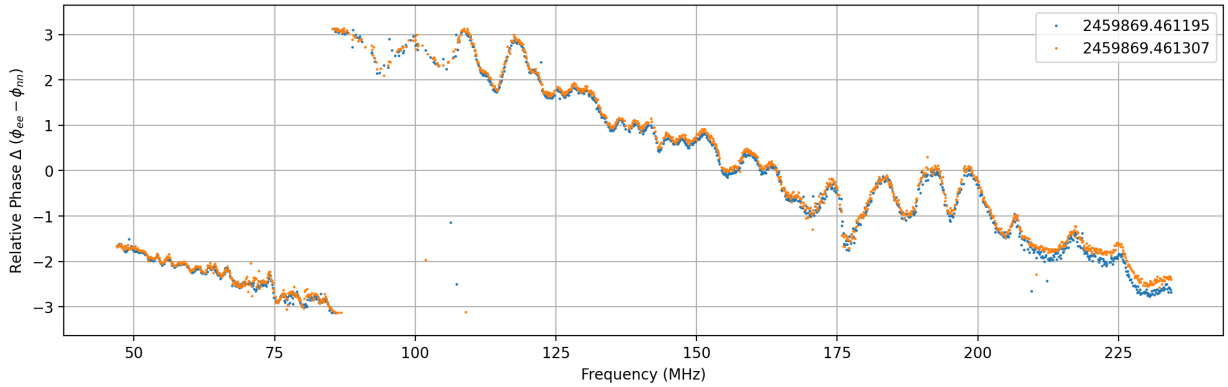

Figure 4: An example of the solved-for relative phase of the e and n polarizations, using an abscal model with an unpolarized sky. Likely the clear delay slope between the polarizations is real; how much of the fine-scale spectral structure between the polarizations is real and how much is a calibration error due to model insufficiency or other factors remains an open question. This figure was produced in this notebook.

The solution here can be computed analytically by evaluating the partial derivative as

$$\frac{\partial \mathcal{L}}{\partial \Delta} = ie^{-i\Delta} \sum_{ij} w_{ij} \left(V_{ij}^{en} \overline{V}_{ij}^{en,*} + \overline{V}_{ij}^{ne} V_{ij}^{ne,*} \right) - ie^{i\Delta} \sum_{ij} w_{ij} \left(\overline{V}_{ij}^{en} V_{ij}^{en,*} + V_{ij}^{ne} \overline{V}_{ij}^{ne,*} \right). \tag{3}$$

Now setting $\frac{\partial \mathcal{L}}{\partial \Delta} = 0$ produces

$$e^{2i\Delta} = \frac{\sum_{ij} w_{ij} \left(V_{ij}^{en} \overline{V}_{ij}^{en,*} + \overline{V}_{ij}^{ne} V_{ij}^{ne,*} \right)}{\sum_{ij} w_{ij} \left(\overline{V}_{ij}^{en} V_{ij}^{en,*} + V_{ij}^{ne} \overline{V}_{ij}^{ne,*} \right)}. \tag{4}$$

Observing that this ratio is of the form

$$e^{2i\Delta} = \frac{Z}{Z^*} \tag{5}$$

where $Z = |Z| \, e^{i\Delta}$, we can see that the principle value of the angle Δ is thus

$$\Delta = \mathrm{Arg} \left(\sum_{ij} w_{ij} \left(V_{ij}^{en} \overline{V}_{ij}^{en,*} + V_{ij}^{ne,*} \overline{V}_{ij}^{ne} \right) \right). \tag{6}$$

This is essentially a weighted average of the phase differences between the data V_{ij}^{en} and the model $\overline{V}_{ij}^{en,*}$, and likewise for the reverse polarization.[1]

In Figure 4 we show an example result for a single `file_calibration` notebook. To be clear, during calibration *en*- and *ne*-polarized visibilities are *only* used to calibrate this degree of freedom and care is taken throughout the pipeline to prevent calibration errors at this stage from impacting any calibrated *ee*- or *nn*-polarized visibilities (see subsection 2.4 for a related issue).

2.2.4 Updated Polarization Convention

In addition to introducing relative polarization calibration, we have also made the decision at the 2024 HERA Annual Meeting at MIT to switch from the `"avg"` polarization convention, where psuedo-Stokes I visibilities are given by $V^I = (V^{ee} + V^{nn})/2$, to the `"sum"` polarization convention, where $V^I = (V^{ee} + V^{nn})$.

We are now documenting this explicitly using the new `pol_convention` attribute for `UVData` and `UVCal` objects in `pyuvdata`. The abscal model files were divided by two from the previous ones and written to

[1] Note that this solution was originally presented in a memo which contained a conjugation error in the final expression for Δ which has been corrected here in Equation 6.

/lustre/aoc/projects/hera/h6c-analysis/abscal_models/h6c_abscal_files_unique_baselines (as well as in the Karoo at /mnt/sn1/data1/abscal_models/H6C). The autocorrelation amplitude model used in the file_calibration notebooks was also divided by two. Both now have the "sum" written explicitly in the files and the pipeline now respects and propagates through that polarization convention to derivative gain and visibility data products.

While we were better documenting the physical meaning of our data products, we also updated the pipeline to propagate through the physical units of the abscal model (Jy) through to gains (via the gain_scale parameter) and visibilities (via the vis_units parameter).

2.3 Updates to Full-Day RFI Flagging

Previously, when we did round 1 full day RFI flagging using 2D DPSS filtering, we only used the "best" autocorrelations to search for RFI in the array-averaged autocorrelations, since we were looking only for real, low-level RFI, which should be relatively consistent across the array. This means that we could potentially unflag RFI that was caught in the file_calibration notebook but not on the set of "best" antennas used in the full-day flagger. Because of the more stringent flagging of antennas with excess RFI (or any spectral structure in their autos), we wanted to make it harder to undo RFI flags, so we lowered the REPEAT_FLAG_Z_THRESH from 2.0 to 0.0, meaning that any per-file flags that still have positive z-scores after DPSS filtering are still flagged.

2.4 Updates to Calibration Smoothing

Because we do not want relative polarization phase calibration to affect calibrated ee or nn visibilities, it is important to keep that term isolated during smoothing. Since smoothing fits real and imaginary parts of gains with DPSS modes separately, an antenna with extra spectral structure in its gain (see Figure 4) could see significant modifications to its phase—and even amplitude.

Instead, we continue to rephase both sets of polarizations to their own reference antennas before smoothing, as we have always done in smooth_cal. However, we now also preserve the waterfall of the relative phase between those two reference antennas (which is the relative phase degeneracy itself). Casting this phase term as a gain with amplitude 1, we apply the same smoothing algorithm to it as any other antenna, and then enforce that the smoothed amplitude remains 1. This term is then applied one of the two polarizations' gains after smoothing. In this way, only en and ne calibrated visibilities are affected.

2.5 Updates to File Post-Processing Notebook

Because we are now performing inpainting and LST-averaging simultaneously (see subsection 3.2), there is no need to perform per-file inpainting every night. This functionality has been removed from file_postprocessing, as have the where_inpainted.h5 files. See section 5 for more details on the changes to data products.

We have also added a new data product, reds_used.p, which is a Python pickle of a list of lists of *unflagged* redundant baseline tuples, as is generally used in hera_cal.redcal. This should allow better bookkeeping of which antenna pairs contributed to which redundant baseline groups on which nights.

3 LST-Stacking

LST-stacking[2] has been significantly overhauled in this IDR, both algorithmically and in the code itself. It now includes relative calibration of the abscal degeneracies (subsection 3.1), a new method for inpainting that uses information from other nights (subsection 3.2), and a new notebook-based pipeline that includes better summary statistics and plotting (subsection 3.3).

[2]Until recently, we have called this step "LST-binning," which many people have found confusing. Binning is part of the procedure, but does not clearly imply that we're doing an average over nights, which is key. In reality, one first picks a set of LST-bins, stacks nights into those bins, and then averages over nights. We settled on "LST-stacking" because it more clearly implies that averaging is going on, like a "stacked" analysis of faint objects in other fields of astronomy, while "LST-averaging" might have implied that different LSTs were being averaged together.

3.1 LST-Stacked Calibration (LST-Cal)

A significant change in IDR 2.3 LST-stacking stage is the introduction of `lst_cal`, a final calibration step preceding LST-stacking and averaging. Previous iterations of H6C IDR2 (Dillon and Murray, 2023b) revealed that the variance within a baseline-LST-channel exceeded the expected thermal noise variance predicted by per-night auto-correlations and N_{samples}. This "excess variance" was hypothesized to at least partially result from day-to-day variations in per-antenna calibration solutions.

The `lst_cal` algorithm partially addresses this issue by comparing each night's visibilities to the average across all nights within a given LST-bin,

$$V_{ij}^{\text{avg}}(\nu) = \left(\sum_{n}^{N_{\text{nights}}} V_{ij,n}(\nu)\, N_{ij,n}(\nu)\right)\left(\sum_{n}^{N_{\text{nights}}} N_{ij,n}(\nu)\right)^{-1}. \tag{7}$$

We then perform a calibration of each night's visibilities to the averaged model solving for the per-frequency and polarization amplitude degeneracy, per-frequency and polarization tip-tilt phase-gradient, and per-frequency relative cross-polarized phase for each night. In order to prevent `lst_cal` from introducing spurious spectral structure into our calibrated visibilities, we smooth the gains with a set of DPSS basis functions at 10 MHz scales before applying them to the data. However, no time-smoothing of `lst_cal` solutions is currently being done, which means that `lst_cal` has the potential both to introduce spurious temporal structure into the calibration solutions as well as to help calibrate out real temporal structure that the per-night analysis did not calibrate because of `smooth_cal`. Both possibilities need further investigation.

Figure 5 demonstrates the impact of this final calibration on two baselines within an LST-bin, clearly reducing the nightly variance of the calibrated visibilities. Figure 8 provides a more quantitative view of `lst_cal` using a summary statistic described in subsection 3.3. While this calibration may not be strictly necessary for a delay spectrum analysis if flagged channels are inpainted on a per-night basis, this final calibration step ensures that we are being consistent in our calibration from night-to-night without risk of introducing spurious spectral structure in our visibilities. Additionally, aligning the data via `lst_cal` improves our ability to inpaint visibilities using information across nights within a single LST-bin (further described in subsection 3.2 below)

The reduction in excess variance achieved by only modifying the `abscal` degrees of freedom is currently not fully understood. In theory, each day's independent calibration to the same `abscal` model should result in consistent agreement in the calibrated visibilities across days. However, these gain solutions are ultimately smoothed over in the `smooth_cal` step by fitting a set of 2D DPSS basis functions to each antenna's gain waterfall, which helps to filter out spurious gain structure. The quality of the `smooth_cal` fits can easily be influenced by day-to-day variations in the flagging pattern, as well as small discontinuities in the amplitude and phase of the gains. Additionally, potential issues may arise from the exact time and frequency scales that are used for smoothing. While these scales are somewhat well motivated by previous work (Dillon et al., 2020; Kern et al., 2020), they may not perfectly align with the true time and frequency variation in the gains.

These factors may introduce smooth gain errors at the few-percent level, which `lst_cal` partially corrects for. Because `lst_cal` only operates on the redundant-calibration degeneracies, we are only able to fix the component of the smooth gain errors that is consistent with these degeneracies, meaning that there will always likely be some excess variance that `lst_cal` is unable to be solved for due to gain errors introduced by `smooth_cal` on the per-antenna level. This may explain the clear non-noise-like nightly variance of `lst_cal`'d visibilities, especially at low-frequencies, in the right-hand column of Figure 5. It is also possible that the change in antenna flags from night to night, combined with antenna-to-antenna variation, plays a role in this variance, even after redundant averaging. The extent to which `smooth_cal` contributes to these calibration errors, both in the `abscal` degrees of freedom and on the per-antenna level, warrants further investigation.

3.2 Simultaneous Inpainting

One of the biggest upgrades in this IDR is the introduction of simultaneous inpainting and LST-stacking. The idea here is that while we have in memory every night's visibility at a given LST for a given baseline that

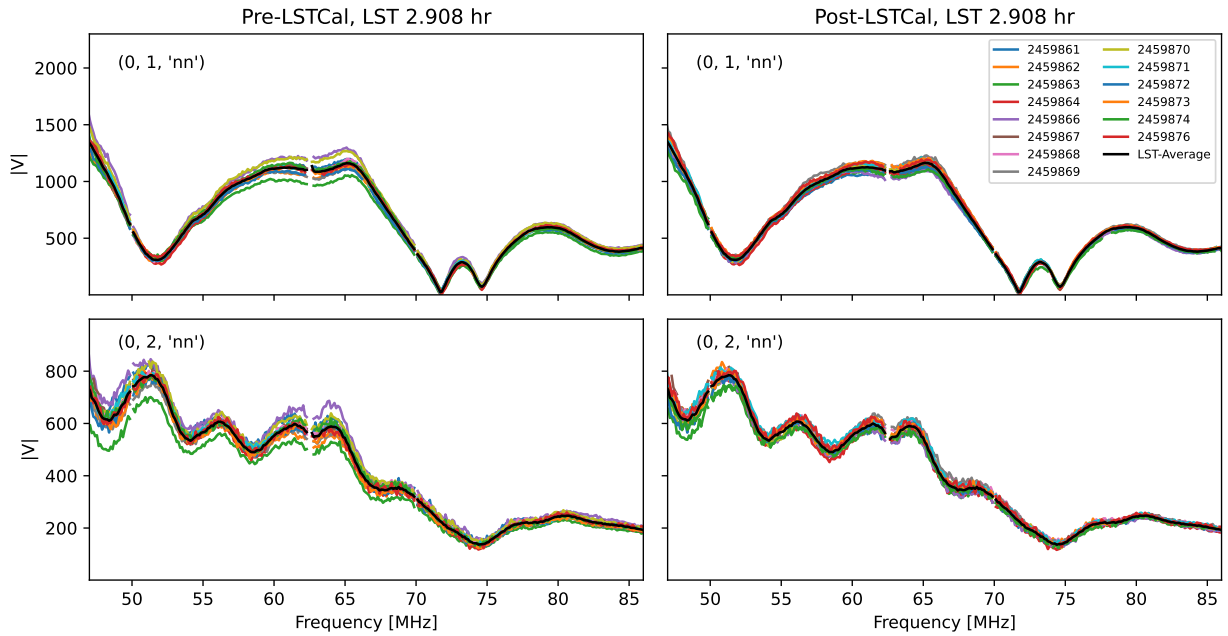

Figure 5: A demonstration of the effect of `lst_cal` on two redundantly-averaged baselines within a single LST-bin (LST = 2.908 hr). The left column shows the redundantly averaged visibilities for two baseline types across all nights within the LST-bin. The N_{samples} weighted average visibility across nights is shown in black. The selected baselines highlight a night-to-night variance in visibilities that clearly exceeds the noise variance for any individual night. This variance can be partially corrected by solving for the per-night and channel amplitude and tip-tilt abscal degeneracies, which bring the nightly data into alignment with the average across nights. The right column displays the results after applying these calibration solutions, which show a clear reduction in the variance in the visibilities across nights. While `lst_cal` is able to reduce the nightly variance, we still excess variance, especially at lower frequencies, likely due to nightly differences in the per-antenna gains.

we intend to stack and average together, we can use information from other nights to inform the inpainting of each individual night.

To inpaint missing data in a given baseline's visibility on the nth night, $d_n(\nu)$, we want to fit a smooth model made of x_n DPSS coefficients, where $\mathbf{A}x_n$ is the model transformed to frequency space. What makes a good fit? Well, we should expect that the smooth model for a given night should both match the data—up to some level of expected noise variation—and should match the average model over nights—up to some level of disagreement between nights. In other words, we would like to minimize a χ^2_n for each night, summed over frequency and DPSS mode number m,

$$\chi^2_n = \sum_{\nu} \left(d_n - \mathbf{A}x_n\right)^\dagger \mathbf{N}_n^{-1} \left(d_n - \mathbf{A}x_n\right) + \sum_{m} \left(x_n - \mu\right)^\dagger \mathbf{C}^{-1} \left(x_n - \mu\right) \tag{8}$$

where $\mu \equiv \langle x_n \rangle_n$, $\mathbf{C} \equiv \mathrm{Cov}(x_n)$, and where we've dropped the explicit dependence of d_n on frequency. $\mathbf{N}_n$ here is the diagonal noise covariance of the data on the nth night, which formally has infinite variance for any flagged channels.

Another way of arriving at the same equation is via a Bayesian model. The assumptions required are firstly that the data on any given night, $d_n(\nu)$, has Gaussian-distributed thermal noise on top of an underlying model that is well-described by a smooth DPSS model. This is a fairly safe assumption, with the possible imperfection that the 21 cm component itself may not be well-described by such a model. The coefficients of this DPSS model, per-night, are unknown parameters to be inferred, which means they must be drawn from some prior. For that prior, we may take a Gaussian distribution in the space of the coefficients, with some mean μ and covariance $\mathbf{C}$. This is equivalent to describing the true, underlying sky model that is invariant

over nights as some smooth DPSS model with coefficients $\vec{\mu}$, and each night having some smooth additive bias with respect to this true model, given by $\mathbf{A}(x_n - \mu)$, where the deviations are drawn from a Gaussian distribution (both on the parameters and the visibilities values per-channel). The posterior distribution of a Gaussian likelihood and Gaussian prior is itself Gaussian, and the *maximum a posteriori* (MAP) point-estimate of this distribution is the same as the estimator that minimizes the above χ^2_n, which is described below.

By taking the partial derivative with respect to x_n and setting it to zero, it can be shown that the estimator that minimizes χ^2_n in this case is

$$\hat{x}_n = \left(\mathbf{A}^\dagger \mathbf{N}^{-1} \mathbf{A} + \mathbf{C}^{-1}\right)^{-1} \left(\mathbf{A}^\dagger \mathbf{N}^{-1} d_n + \mathbf{C}^{-1}\mu\right). \tag{9}$$

If we accept the assumption that the model can be treated as a multivariate Gaussian, as the second part of Equation 8 does, then a key question remains: what should we use for μ and $\mathbf{C}$? Defining the noise covariance between DPSS modes

$$\mathbf{D}_n \equiv \left(\mathbf{A}^\dagger \mathbf{N}_n^{-1} \mathbf{A}\right)^{-1}, \tag{10}$$

we can recast the estimation of a DPSS model for each night independently as

$$\hat{x}_n^{\text{indep}} = \mathbf{D}\mathbf{A}^\dagger \mathbf{N}_n^{-1} d_n. \tag{11}$$

Given that we know the noise covariance between DPSS modes of this estimator, $\mathbf{D}$, we can combine nightly estimates in a inverse-covariance weighted way to get

$$\hat{\mu} = \left(\sum_n \mathbf{D}_n^{-1}\right)^{-1} \left(\sum_n \mathbf{D}_n^{-1} \hat{x}_n^{\text{indep}}\right). \tag{12}$$

This estimator naturally handles the case where large flagging gaps on individual days creates strong correlations between DPSS modes—the estimator $\hat{\mu}$ simply gives more weight to the estimate of those modes on other days in order to break degeneracies.

In practice, if a given integration has too large of a flagging gap on all days, we simply throw out the LST for the whole subband (below or above FM). This is only if the local density of channels flagged on all days gets too high. We compute this by convolving the AND of flags over all nights with a triangular kernel of half-width $2\tau/\Delta\nu$ where τ is the inpainting filter width and $\Delta\nu$ is the channel width. The 2 is the critical value above which we find large excursions—the so-called "pop-ups" and "pop-downs"—to be more likely. If this convolved and normalized flagging pattern ever rises above a 2/3 flagging density, we flag the whole LST. This is quite rare, since wide flagging gaps are relatively uncorrelated between nights.

If our goal were simply to make a single best-fit DPSS model for all the nights, $\hat{\mu}$ would be it. But that is not actually the goal of inpainting. Because the inpainted model is only used in flagged channels, we still need a per-night $\hat{x}_n$ that closely matches the data were we have it and does "reasonable" things where we're missing data data so as to avoid introducing discontinuities in the inpainted result. This is a challenge, for in a sense inpainted data is inherently discontinuous. In certain channels, the data comes from the measurement, while in neighboring channels, it comes indirectly from other channels (and other nights) via a relatively small number of fit model parameters.

This then highlights the importance of choosing our estimator of the night-to-night model $\hat{\mathbf{C}}$, the term used to penalize nightly DPSS fits that are too far from the weighted average over nights. In the presence of medium-to-large flagging gaps, over-fitting to the unflagged data creates pop-ups while under-fitting to the unflagged data risks discontinuities in the inpainted result, especially for complex flagging patterns. One reasonable choice would be:

$$\hat{\mathbf{C}} = \frac{N^2}{N-1}\left[\left(\sum_n \mathbf{D}_n^{-1}\right)^{-1}\left(\sum_n \mathbf{D}_n^{-1}\left(\hat{x}_n^{\text{indep}} - \mu\right)^\dagger \left(\hat{x}_n^{\text{indep}} - \mu\right)\mathbf{D}_n^{-1}\right)\left(\sum_n \mathbf{D}_n^{-1}\right)^{-1} - \left(\sum_n \mathbf{D}_n^{-1}\right)^{-1}\right], \tag{13}$$

which is, in theory, an unbiased sample covariance estimator. However, we have found through some limited experimentation, better success in inpainting with a much simpler form, namely,

$$\hat{\mathbf{C}}' = \text{diag}\left[\frac{N^2}{N-1}\left(\sum_n \mathbf{D}_n^{-1}\right)^{-1}\left(\sum_n \mathbf{D}_n^{-1}\left(\hat{x}_n^{\text{indep}} - \mu\right)^\dagger \left(\hat{x}_n^{\text{indep}} - \mu\right)\mathbf{D}_n^{-1}\right)\left(\sum_n \mathbf{D}_n^{-1}\right)^{-1}\right]. \tag{14}$$

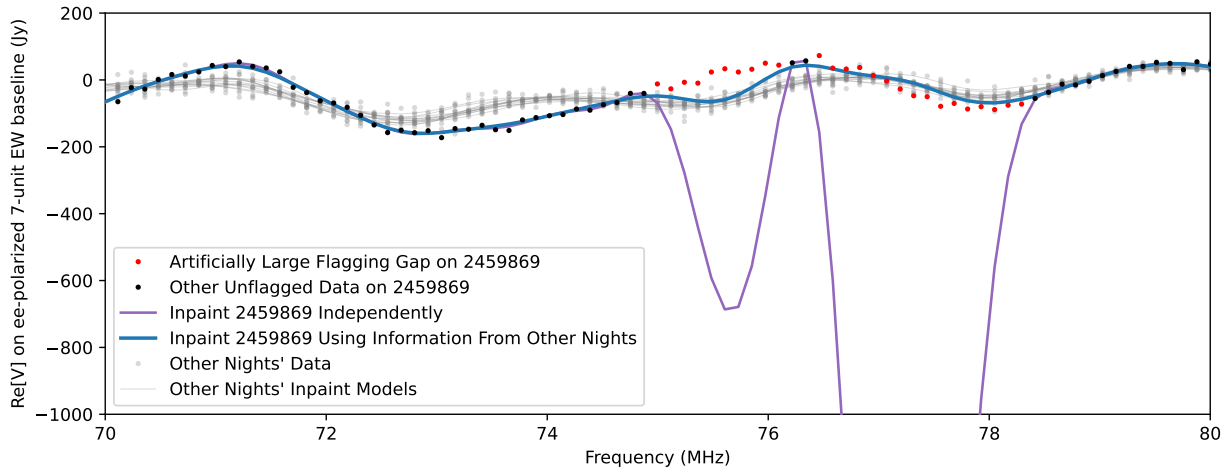

Figure 6: Simultaneous inpainting and LST-stacking uses information from other nights to inform inpainting—especially over large flagging gaps. In this example with real data, we've added two artificial flagging gaps (red points) to 2459869, separated by only two unflagged points. Both are relatively large compared to the 1000 ns delay limit of the inpainter. As a result, inpainting only the black points results in large excursions (purple) produced by strongly correlated DPSS modes. By contrast, using information from other nights (gray points) and their DPSS fits (gray lines), we can produce a much more reasonable result (blue), even though this night is something of an outlier compared to the others.

Here, we've thrown away all the off-diagonal elements in Equation 13 and we've also taken off the noise bias subtraction term, which can cause those DPSS parameter variances to go negative (which is unphysical).

While the forms of $\hat{\mathbf{C}}$ and $\hat{\mathbf{C}}'$ make some intuitive sense—we're forming a sample covariance estimator using the differences between the sample mean and each night's estimate, weighted by the noise covariance of that night's estimate—the math here remains unsatisfyingly ad hoc. In particular, the well-motivated Bayesian model we outlined above is only strictly valid when μ and $\mathbf{C}$ are either known or estimated *without reference to the data*, which is not the case for $\hat{\mu}$ and $\hat{\mathbf{C}}'$. Another aspect of this computation that makes it difficult to properly interpret is that in the end we are not adopting the average of $\mathbf{A}\hat{x}_n$ over nights n as our averaged data, but rather we are only inpainting the *flagged* nightly data with this model before performing standard averaging. Inherent in this procedure is the assumption that this DPSS fit is *not* the best estimator of the underlying true model of the data (e.g. it doesn't include the 21 cm signal itself), and yet it's "good enough" within the flagging gaps to represent that data for our purposes. But these assumptions are not at all baked into the Bayesian model itself.

In the end, the precise normalization, the unbiasing procedure, the choice to only take the variance—these all need to be put on firmer statistical footing. We hope that some of the work now being done to understand the statistics and behavior of per-night DPSS inpainting (Chen et al., 2024, in prep.) can be adapted to study the impact of picking different estimators and inform our ultimate choice of inpainting technique. For now, we are using Equation 9 with $\mu = \hat{\mu}$ as defined in Equation 12 and $\mathbf{C} = \hat{\mathbf{C}}'$ as defined in Equation 14. Note that for non-redundantly-averaged data, we use $\mathbf{N}_n$ given by the *redundantly averaged* autos, rather than each particular auto-pair. This is primarily to improve caching performance and should have little difference on the final result.

Regardless, of the ambiguity and hand-waviness here, the fact is that this technique works quite well and largely eliminates the pop-ups we that had problematized previous inpainting attempts, especially as we move toward higher and higher inpaint delays in an attempt to completely inpaint the mutual coupling signal (which can extend to 750 ns or more). In Figure 6 we show how much better simultaneous inpainting can be than per-night inpainting. And in Figure 7 we begin to examine the subtle impact of the choice of estimator for $\mathbf{C}$, motivating our choice for $\mathbf{C} = \hat{\mathbf{C}}'$ as defined in Equation 14. This is not a systematic treatment of the question, which merits further study.

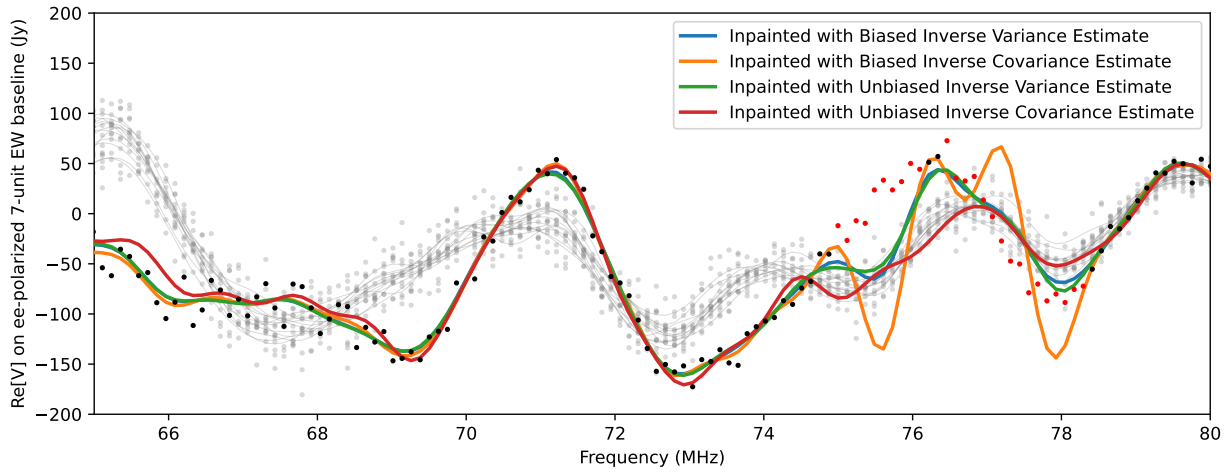

Figure 7: Here we examine the exact same inpainting problem as in Figure 6, with the same artificially added flags on 2459869, zoomed-in to highlight the differences between methods. The inpainting solution in blue, which uses $\hat{\mathbf{C}}'$ as defined in Equation 14, is the same in both figures. In red, we show the result using $\hat{\mathbf{C}}$ as defined in Equation 13. We also show in orange the result using Equation 14 but without taking the diagonal, and in green the result using Equation 13 but then taking the diagonal only of the whole matrix. Both inverse variance methods give similar results, though in this particular case the unbiased version did not suffer from any of the negative variances we had seen elsewhere. The inverse covariance methods perform quite differently. The biased covariance estimator seems to overfit the data, producing minor pop-ups. The unbiased covariance estimator, by contrast, skews much harder toward the sample mean. This causes it to significantly miss the two unflagged points around 76 MHz. When inpainting, one keeps those points, leading to significant spectral structure. That said, the effect is most prominent because 2459869 is an outlier—an problem likely mitigated by `lst_cal` (see subsection 3.1).

3.3 The Overhauled LST-Stacking Code-Base

Apart from the addition of `lst_cal` and simultaneous inpainting, LST-stacking is essentially the same as in previous IDRs. However, its implementation is quite different.

The code for stacking and averaging of the data within LST-bins has been refactored into its own sub-package of `hera_cal`, called `lst_stack`. While much of the functionality remains the same, the code has been significantly modularized to simplify future development.

One aspect of the code-base that is new is that the determination of which nightly files (and times within those files) goes into which LST bin is determined up-front by code in a dedicated `config.py` module. The methods in this module are generally called by `hera_opm`, that is, the determination of which data goes into which bin is done once when *setting up* the Makeflow jobs (this necessitated some updates to `hera_opm` as well). Much of the underlying code for determining the bins into which each file's data falls is the same as in IDR2.2, however one bug was caught that misaligned bins at this step by half a bin-width.

A larger-scale change in this IDR is that we have changed the workflow to center around a Jupyter notebook, like the rest of the pipeline for this season.[3] One difficulty with using notebooks is that loops over large sections of processing (where several figures are produced during each loop) are infeasible. In previous iterations of the LST-stacking there were two major loops: one over chunks of LST-bins and the other over chunks of baselines (which helped to reduce peak memory usage). We previously managed this by parallelizing over the chunks of LST-bins, i.e. performing one job per chunk of two bins, where each job wrote out a single file containing those two bins. Internal to the job was the loop over baseline chunks. However, in the notebook format, this 'inner loop' is not possible, so each job—i.e. each notebook—deals with one LST-bin chunk *and* one baseline chunk, and the output files are indexed by both their LST-bin and their baseline

[3]In fact, we plan to have *three* notebooks for LST-stacking and averaging: the first to stack data and determine metrics, a second to collate the metrics and decide on data to cut, and a third to perform final averaging with these cuts being used. So far, only the first has been implemented.

chunk. Generally, the redundantly-averaged products require only one baseline chunk in this IDR, so they all have `.000.` in their filename indicating the filename (or `.autos.` which are written out separately, since they are need for estimation noise on all baseline chunks). However, non-redundantly-averaged products require more than one chunk.

The broad format of the LST-stacking notebook is the following:

1. Configuration/Imports
2. Read Data
3. LST-Cal
4. Simultaneous Inpaint of Autos
5. Simultaneous Inpaint of Crosses
6. Write LST-averaged data
7. Determine Metrics and Make Plots
8. Write Metrics.

The configuration of the notebook uses a new feature in the hera_notebook_templates package, utilizing the `papermill` package to pass along configuration options (instead of using `nbconvert` and needing to define environment variables).

Reading the data has been made very simple using the new `lst_stack` package. Using the pre-created LST-stacking configuration file, which can be read as a Python object, the data for a particular bin can be read with a single function. We hope this will be useful for others looking to experiment with LST-stacked data.

The metrics are determined for all data in this IDR, however, the plots are rather resource-hungry, and we only make them for an LST-bin every ∼hour (every 180^{th} output file). These notebooks with plots can be viewed at `https://data.nrao.edu/hera/Notebooks/H6C_IDR2/lstbin/{{CASENAME}}/`.

The metrics computed in the notebook are written to disk in order that they can be read by a follow-up notebook that makes decisions about flagging data in a more coherent way. Currently, no flagging is performed by this notebook, and there is no follow-up notebook in this IDR, so these metrics do not affect any of the results. However, we here describe the metrics that are computed for future reference.

The primary metric we compute is a Z^2-score, where Z is a complex number whose real and imaginary parts are independent and normally-distributed (if they are systematics-free). To be specific, they are:

$$Z_i \equiv \sqrt{\frac{2n_i}{\sigma^2}\frac{M}{M-n_i}}(V_i - \bar{V}), \tag{15}$$

where V_i is the visibility on night i, $\bar{V}$ is the weighted average of visibilities over nights, where the weights are n_i, i.e. the number of samples per night. We take σ^2 to be the expected noise variance obtained from the autos. Finally, M is the sum of n_i over nights. We show in Murray (2024) that this metric is normally distributed with unity variance (if indeed the true noise variance is given by the autos). The absolute square of Z is a metric of deviation from the mean, and has well-understood statistical properties, both as a direct quantity, and also as a mean over channels, nights or baselines. We write out the following:

- The indices (baseline, LST, channel) of *all* data that has $|Z| > 3$.
- The mean Z^2 over baselines within baseline selections (e.g. EW/NS, long/short) for all nights and channels
- The mean Z^2 over nights for all channels and baselines
- The mean Z^2 over channels within certain bands for all baselines and nights
- The mean Z^2 over baselines within groups and nights for all channels

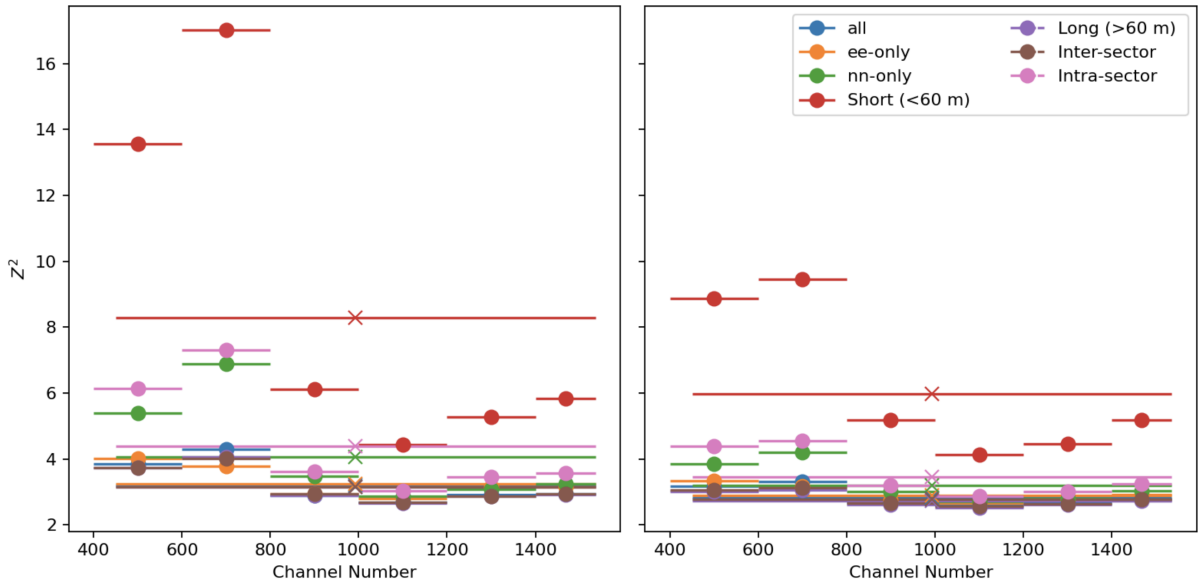

Figure 8: Comparison of the mean Z^2-score (Equation 15) computed over different frequency bands above FM for different subsets of the data, before and after applying `lst_cal`. Assuming that the data are systematics-free, the expectation value of this quantity should be of order unity. This makes the Z^2-score a useful metric for diagnosing non-thermal noise-like deviations (calibration errors, unflagged RFI, etc.) from the average visibility across nights. In the left-hand panel of the figure, we plot the average Z^2-score for various subsets of the data. Here, filled circle markers denote an average across 22.3 MHz chunks of the band, while "x" markers are an average of the Z^2 score over all frequency channels above FM. Clearly, all subsets of the data have mean Z^2-scores greater than 1, indicating some nightly-varying systematic that affects all baselines within this LST-bin. In the right panel, we plot the average Z^2-scores for the same subsets of data after applying `lst_cal`, demonstrating that some portion of this excess variance can be accounted for by performing a nightly calibration of the `redcal` degeneracies to the mean of the visibilities across nights.

- The mean Z^2 over baselines within groups and channels for all nights
- The mean Z^2 over nights and channels within bands for all baselines
- The mean Z^2 over nights, channels within bands, and baselines within groups.

These are all written to a single HDF5 file, named with the same naming scheme as the LST-stacked data, but with the kind `LSTBIN-METRICS`.

4 Post-LST-Stacking

While the primary data products in this IDR are the per-night visibilities, flags, and calibration solutions as well as the LST-stacked visibilities, we have also produced a number of useful data products for subsequent analyses. These are made available to the wider collaboration, with the caveat they are still experimental, not as well documented or validated as most of the rest of this IDR, and potentially subject to change.

4.1 Corner-Turned Datasets

After LST-stacking, which produces thousands of files, it is useful to produce additional datasets that enable faster I/O in subsequent analyses. The first such dataset we produce is simple: take all the redundantly averaged visibilities for a given baseline for all times, frequencies, and polarizations, and then write them to disk. We discuss where to find these data products in subsection 5.3.

The second data post-LST-stacking visibility data product available is a "mini-dataset." These visibilities are redundantly averaged, then we average together 10 integrations (with rephasing) and 4 channels, to produce a dataset 1/40th the size of the full data. The result is grouped into a set of files that includes all baselines and polarizations, each spanning approximately 1 hour of LST and each only about 1 GB.

The averaging of visibilities is done *without* any kind of weighting by N_{samples}. Flags are ORed across all 40 times and frequencies that go into each bin, though since this is only done on inpainted data, the only flags come from the FM band or from entirely flagged integrations. N_{samples} is simply the sum of unflagged N_{samples} before time- and frequency-averaging. To ensure a constant Δt across this mini-dataset, the last few integrations are simply dropped if the total number of integrations in the LST-stacked dataset is not divisible by 10. All of these analysis decisions could be revisited in the future if this dataset proves useful. A discussion of where to find these data products in also in subsection 5.3.

Both of these processes are run through `makeflow` with this `.toml` file. The `corner_turn.py` script produces the single-baseline visibilities. The `make_mini_dataset.py` script performs time and frequency averaging in chunks.

4.2 Single-Baseline Post-Processing and Power Spectra

The current plan for further data reduction and power spectrum estimation is a single notebook that operates on full-day, 4-pol, single-baseline files. The current version of that notebook is here. The basic set of steps run by the notebook is the following:

- Define power spectrum bands (currently 3 below FM and 9 above, though that is subject to change).
- Optionally inpaint and/or delay filter.
- Split the data into four time-interleaved datasets.
- Notch filter out the fringe rate 0 mode.
- Top-hat filter to keep only the fringe rate modes consistent with the main beam (determined per band).
- Form pseudo-Stokes visibilities.
- Coherently average in ∼300 s chunks.
- Estimate power spectra of all cross-interleaved datasets.
- Estimate noise on power spectrum, attempting to account.
- Write out power spectra and error bars, as well as meta-data related to filtering useful in further downstream error bar and signal loss calculations.

Along the way, there are a number of plots that help assess the performance of these various steps.

This notebook can be run interactively, as well as via `makeflow` with this `toml`. The method, as well as the various parameters, are subject to change and still need substantial review, code-graduation, validation, and general battle-testing. However, we include it here as a possible entry point for interested data analysts to look at highly-reduced data, apply some of algorithms that have become standard in the HERA codebase, and begin exploring the subtle systematics that undoubtedly remain in the data.

5 Summary of New, Modified, or Removed Data Products

5.1 Per-Night Data Products

Per-night data products can be found in `/lustre/aoc/projects/hera/h6c-analysis/IDR2/24598??/`. Due to the changes in the pipeline, the following output files were added:

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

- zen.24598??.?????.reds_used.p

The first two are calibrated, redundantly-averaged, visibility files with all the accompanying flags and N_{samples}. New to this release is that they are now include the *en* and *ne* cross-polarized visibilities. No inpainting was performed. The last is a record of the unflagged baselines used in the redundant average for a particular file. It is a pickle file that contains a list of lists of redundant baseline tuples of the form `(0, 1, "ee")`, as is commonly produced and used by `hera_cal` submodules like `redcal`.

The following output files were removed due to changes in the post-processing notebook and the moving of inpainting to the LST-stacking step:

- zen.24598??.?????.sum.smooth_calibrated.red_avg.inpaint.uvh5
- zen.24598??.?????.sum.abs_calibrated.red_avg.inpaint.uvh5
- zen.24598??.?????.where_inpainted.h5

5.2 LST-Stacked Data Products

The output LST-stacked products, which are available in `/lustre/aoc/projects/hera/h6c-analysis/IDR2/lstbin-outputs/<CASE>`, are largely the same. To briefly recap, all outputs are written in the format `<INPAINT_MODE>/zen.<KIND>.<LST:7.5f>.<BLCHUNK:03d>.sum.uvh5`. The `INPAINT_MODE` is either `flagged` or `inpaint`. The `KIND` parameter can take several different values: "LST", "STD", "MEDIAN" and "MAD" for the respective data-based statistics ("LST" refers to the lst-stacked mean, for historical reasons), as well as "HIGHZ" for indices of all data with $|Z| > 3$ and "LSTBIN-METRICS" for the averaged $|Z|^2$ over different axes. The value of `BLCHUNK` is either "autos" or a three-digit integer with leading zeros indicating which chunk of baselines is in the file. Examples of such files are:

```
inpaint/zen.LST.8.12345.000.sum.uvh5
flagged/zen.STD.8.12345.autos.sum.uvh5.
```

The LST-stacking cases we have produced in this PR are:

- redavg-smoothcal-inpaint-500ns-lstcal
- redavg-abscal-inpaint-500ns-lstcal
- redavg-smoothcal-inpaint-500ns-nolstcal
- redavg-smoothcal-inpaint-1000ns-lstcal
- redavg-smoothcal-dlyfilt-500ns-lstcal
- nonavg-smoothcal-inpaint-500ns-lstcal.

A final data product is written to `<INPAINT_MODE>/zen.LST.<LST:7.5f>.<BLCHUNK:03d>.pkl`, which is a pickle file that contains the `abscal` degenerate solutions computed during the `lst_cal` step. The contents of the file are a dictionary for each LST-bin in the corresponding LST-stacked `uvh5` file, which contains the numpy arrays of the amplitude degeneracy for each polarization (with dictionary key `A_{pol}`), the tip-tilt degeneracy for each polarization and night (`T_{pol}`), and the cross-polarized relative phase degeneracy (`delta`). These dictionaries also contain various bits of useful metadata, such as the array-averaged flagging pattern for each night and Julian dates for each night in the LST-stack.

5.3 Post-LST-Stacking Data Products

Single-baseline corner-turned files are available in folders inside the folders that contain the LST-stacked outputs. The general format is:

- `<INPAINT_MODE>/single_baseline_files/zen.LST.baseline.<ANT_1>_<ANT_2>.sum.uvh5`

For example, baseline (0, 11) for all LSTs, frequencies, and polarizations in the redundantly averaged case with smoothed calibration solutions, `lst_cal`, and 500 ns inpainting would be at `/lustre/aoc/projects/hera/h6c-analysis/IDR2/lstbin-outputs/redavg-smoothcal-inpaint-500ns-lstcal/inpaint/zen.LST.baseline.0_11.sum.uvh5` Likewise, frequency and time averaged mini-datasets similarly available in

- `<INPAINT_MODE>/mini_dataset/zen.LST.<LST:.2f>_hours.mini_dataset.sum.uvh5`

Only the `redavg-smoothcal-inpaint-500ns-lstcal`, `redavg-smoothcal-inpaint-500ns-nolstcal`, and `redavg-smoothcal-inpaint-1000ns-lstcal` cases were run through the corner turner.

Power spectrum data products—which are experimental and subject to revision—are available in `/lustre/aoc/projects/hera/h6c-analysis/IDR2/pspec/<CASE>`. TThese include:

- `zen.LST.baseline.<ANT_1>_<ANT_2>.sum.pspec.h5`: `hdf5` files that store `hera_pspec` objects.
- `zen.LST.baseline.<ANT_1>_<ANT_2>.single_baseline_postprocessing_and_pspec.html`: executed notebooks that produce those files, containing a variety of useful plots documenting its processing and filtering and the subsequent power spectra and power spectrum statistics.

Those `html` files are also available to view here. While single baseline power spectra are probably not "spoilers" from the perspective of blinding, loading up many of these and averaging them together would be.

References

Chen, Kai-Feng, Michael J. Wilensky, Adrian Liu, et al. (2024, in prep.). "Impacts and Statistical Mitigation of Radio Frequency Interference on 21cm Power Spectrum: A Case Study with the Hydrogen Epoch of Reionization Array". In.

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

— (2023b). "HERA Memo #125: H6C Internal Data Release 2.2". 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].

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].

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

Murray, S. G. (2024). "HERA Memo #130: Excess Variance of Visibilities and the Z-squared Statistic". reionization.org/memos.