Draft version August 27, 2020
Typeset using LaTeX **twocolumn** style in AASTeX63

Assessment of Radio Calibrators using Visibility Properties

Henry A. Prager[1, 2] and Loránt Sjouwerman[2]

[1]*Department of Physics, New Mexico Tech, Socorro, NM 87801*
[2]*National Radio Astronomy Observatory, Socorro, NM 87801*

(Dated: August 27, 2020)

ABSTRACT

Before a radio image can be reconstructed from interferometric data, it must undergo calibration. The initial calibration of the data is done using a known radio source, chosen by eye based on location and shape of the source. We can avoid this introduction of human error into the calibration process by analyzing the quality of a calibrating source mathematically. To do so, we introduce a scoring system for the quality of a calibrator based on its divergence from an ideal point source. This divergence is calculated as the distance in a coordinate space of made of average calculated errors in the visibilities of a source, giving a value for how point-like the source is. To further refine this into a calibrator score, this is combined with a system of weights and cut-offs to enhance the differentiation between known good calibrators and poor calibrators. The system tested analyzes potential calibrators in the VLA S-band, many of which are known calibrators in the VLA L-band and C-band. Using the known calibrator classifications to predict the likely classification in S-band, we demonstrate that this scoring system can effectively classify calibrators through direct analysis of observations of a source.

1. BACKGROUND

Through basic optics, we know the resolution of a telescope is related to the wavelength λ being observed and the diameter D of aperture. Initially determined using visible light, this relation extends to all wavelengths.

$$\theta \approx 1.22\frac{\lambda}{D} \tag{1}$$

The telescope's resolution is not an impractical limiting issue with optical telescopes—a two meter aperture has roughly $0.07''$ of resolution for green light. This is not the case at radio wavelengths. For that same resolution at a wavelength of 2 mm, you would need a telescope 7.2 km in diameter. Building a physical telescope of that size is a practical and engineering impossibility. However, we can build a *virtual* telescope of that size by combining the observations of many smaller telescopes using interferometry. This process does come with its own challenges; here, we will be examining calibration.

1.1. *Interferometry*

The most basic interferometer is made up of two apertures. In the case of radio astronomy, that would be two focused antennas, a variant of the Michelson interferometer. In this interferometer, two telescopes are separated in space, resulting in slightly different path lengths from the source to the antenna. Because of this differing path length, interference occurs, and combining the signal from the two antennas results in a diffraction pattern. The interferometer measures a projection of the intensity from the sky into one dimension. By using an inverse Fourier transform, the image can be reconstructed from the signal detected by the interferometer.

The basic Michelson interferometer is most useful now as an explanatory case. In theory, you can repeat this baseline process for as many antennas as you can connect together; in practice, this is limited by a combination of cost, computational capability, and physical distance between the telescopes. This combination of antennas into many interferometric baselines is the basis for many radio observatories today, such as the Jansky Very Large Array in Socorro County, New Mexico; the Atacama Large Millimeter Array in Chile; and the planned Square Kilometer Array in South Africa. To properly create images from sources with these observa-

henry.prager@student.nmt.edu

lsjouwer@nrao.edu

tories, we must calibrate the measurement with a known source (Thompson et al. 2017).

1.2. *Calibration*

When observing, our telescope cannot simply measure radiation from a target as it was emitted. The telescope has to measure the radiation on Earth with real materials, with all of the obstruction and distance that comes with it. The goal of calibration is to address interference that comes from the atmosphere and from the radio antennas themselves (Thompson et al. 2017). Our atmosphere is complicated; it is a mixture of various gasses that vary in temperature and density throughout–not to mention the dust and clouds–that is constantly experiencing turbulent motion. The antennas themselves are also a source of interference; there is of course their own blackbody radiation, but more importantly the antennas are electronic. By simply changing the currents flowing within the antenna, radio waves are produced by the antenna (Jackson 1998). The exact interference recorded is depends on the locations of the antennas, the antennas themselves, and on how they are paired, so we need to apply calibration to the visibility data, not the assembled image (Thompson et al. 2017).

1.2.1. *Calibration Process*

When calibrating, we can break the process into two time domains. The errors from long-term stable sources are not the focus here; these errors can be corrected using any unresolved radio source, near our target or not, when the baselines are arranged. This includes things such as antenna baseline positions, antenna pointing corrections, and electronic delays (Thompson et al. 2017). We are more concerned with the errors resulting from short-term changes. These can either calculated–such as atmospheric attenuation or gravitational deformation of the antenna–or need to monitored. Some of these effects can be directly monitored with instruments on the antenna, such as system noise or atmospheric delay, and can be corrected with that data. Our main concern is the short-term variable errors that cannot be monitored from the antenna–the routine corrections. These routine corrections account for sources of error such as atmospheric conditions and the phase and amplitude variations due to the antenna's electronics (Fomalont & Perley 1999). These require the observation of a calibration source, a known radio source near our target that we can use to determine (or at least get a good estimate of) the interference from indirectly monitored conditions Thompson et al. (2017).

To determine the correction needed to our target source, we will have to make a comparison to the calibrator source of known flux density. Assume we have some uncalibrated visibility from antennas m and n, with a corresponding gain $G_{mn}(t)$ and the true visibility $\mathcal{V}(u,v)$ we need to determine,

$$[\mathcal{V}(u,v)]_{\text{uncal}} = G_{mn}(t)\mathcal{V}(u,v) \tag{2}$$

and a calibrator visibility with its own gain $[G_{mn}]_c(t)$ and source visibility S_c

$$\mathcal{V}_c(u,v) = [G_{mn}]_c(t)S_c. \tag{3}$$

In theory, the gains should depend on the baseline distances u and v, but if the target is small enough we can assume atmospheric interference is more-or-less uniform, making this dependence negligible. To calibrate the target, we need to determine the antenna gain function $G_{mn}(t)$. While not exactly the same, $G_{mn}(t)$ well approximates the gain for the calibrator as long as it is close to our target.

$$\mathcal{V}_c(u,v) \approxeq G_{mn}(t)S_c. \tag{4}$$

If we know the calibrator flux S_c independently, we can determine the antenna gain; determining S_c confidently is easier with stronger sources as the calibration target will dominate the signal. With some quick algebra, the calibrated visibility is

$$\mathcal{V}(u,v) = \frac{[\mathcal{V}(u,v)]_{\text{uncal}}}{G_{mn}(t)} \approxeq [\mathcal{V}(u,v)]_{\text{uncal}} \left[\frac{S_c}{\mathcal{V}_c}\right]. \tag{5}$$

The gain $G_{mn}(t)$ is a complex-valued function, so we must be careful with our algebra; more detail will follow in the next paragraph. To reiterate, this comes with two approximations–we assume the calibrator is close the target, and that it is a sufficiently strong source. To ensure that approximation is valid, we will have to pick an appropriate calibrator when reducing data (Thompson et al. 2017).

In general, we can write the gain G_{mn} as a product of the complex gains of the individual baselines and antennas

$$G_{mn}(t) = g_m(t)g_n^*(t)g_{mn}(t) \tag{6}$$

where each individual gain is a complex function described by an amplitude and a phase

$$g_j(t) = a_j(t)e^{i\phi_j(t)}. \tag{7}$$

and we note that an asterisk '$*$' denotes the complex conjugate of a function. By calibrating the antenna gain $G_{mn}(t)$, we will simultaneously correct the amplitude and phase variations of the antenna. With the definition given in Eq. 7, we can define an amplitude and phase term for the total gain $G_{mn} \equiv A_{mn} e^{-i\Phi_{mn}(t)}$.

$$A_{mn} = a_m(t) a_n(t) a_{mn}(t) \tag{8}$$
$$\Phi_{mn} = \phi_m(t) - \phi_n(t) + \phi_{mn}(t) \tag{9}$$

Combining these definitions of the amplitude and phase with the true visibility from Eq. 5, we can find the true amplitude and flux in terms of the uncalibrated amplitude and flux.

$$A_{mn} = \frac{[A_{mn}]_{\text{uncal}}}{S_c} \tag{10}$$
$$\Phi_{mn} = [\Phi_{mn}]_{\text{uncal}} = \phi_m - \phi_n + \phi_{mn} \tag{11}$$

We are still missing one constraint, but we can solve that by introducing 'closure'. As long as our source is sufficiently point-like, the baseline phase ϕ_{mn} will approach zero, and the baseline amplitude a_{mn} will approach 1. This reduces our number of unknowns and makes it possible to calibrate the combination of antennas m and n (Fomalont & Perley 1999).

1.2.2. *Selecting Calibrating Sources*

When selecting a calibrator we want a source that is strong enough to have low uncertainty in the calibrator flux and close enough to the target that the gain function is well approximated. Close to the target is fairly self explanatory; a calibrator is closer to a target if their angular separation is smaller[1]. Lowering uncertainty in the calibrator flux has two approaches–we can either have a source that is closer to be unresolved (a point source) or we can have a source that is simply brighter. Point sources are ideal because their theoretical appearance is very well known, so correspondingly our confidence in the flux distribution from any point goes down as we are able to measure structure (Fomalont & Perley 1999).

When selecting a calibrator we try to find a good compromise between distance from the target and our confidence in the calibrator flux. In practice, potential calibrators are categorized based on their estimated closure error, the divergence from a complex baseline gain of 1 (see eq. 6, in the case of a baseline $j = mn$); this divergence effectively characterizes how point-like a source is. These categories can be seen in Table 1. With that, the flux of the calibrator source, and the position in the sky we can roughly judge how good of a calibrator something will be.

[1] Close may be self explanatory, but sufficiently close is more complicated. This depends on factors such as telescope resolution and bandwidth choice.

2. SCORING OF CALIBRATORS

When an observer is choosing a calibrator for their observations, they know the category, flux at chosen wavelength, and the location in the sky. This gives a good idea of the quality of a calibrator target, but it leaves choosing between targets and what to prioritize up to the observer because there are no longer hard numbers associated with the closure error at this point. That makes choosing a calibrator target subjective. As well, with no numbers for comparison, calibrator selection has to be done manually.

The new system we present here is a semi-relative scoring system for calibrators. Instead of being categorized based on how point-like they are through closure error, we calculate how similar the calibrator is to a point source from the visibility observations. This similarity score is then modified based on the relative flux of the calibrators an observer wants scored and the number of visibilities. A higher flux and greater visibilities than average result in a higher score, while the opposite is true for being below average.

This method has the benefit of being objective–there is no ambiguity in which source has better score. This immediately makes it easier for an observer to pick a calibrator target manually. With a numerical value it is also possible to automate the picking of calibration targets, now that a computer has values to compare. Finally, as telescopes are upgraded, new calibrators have to be found; this scoring system can automate the characterization of new calibrators.

2.1. *Theory*

2.1.1. *Point-like Scoring*

From every observation of a radio source we measure the following: real and imaginary parts of the visibilities, the number of visibilities, and the amplitude of the visibilities. When analyzing these calibrators we have also determined the fraction of those visibilities that have good model solutions. With these measurements we can characterize observed properties of each source by finding the difference between the observed and model values as well as by finding the ratio between observed and model values. For an ideal point source these values should all be either 0 (in the case of the subtracted values) or 1 (in the case of the divided values); so we can characterize everything as their proximity to 0, the

Table 1. Calibrator Closure Error Classification

Calibrator Classification	Estimated Closure Error δ	Recommended Use
P	$\delta < 3\%$	All Uses
S	$3\% < \delta < 10\%$	Phase and amplitude calibration only
W	$\delta > 10\%$ or Unknown	Phase Calibration Only
C	Confused Source	Exceptional Use Only
X	N/A	Do Not Use

Note—This table describes calibrators based on their closure error classifications. This classification must be made for source at a target wavelength and for the resolution of the telescope baseline. For estimated errors under 3% (Type P) we essentially have point sources, for estimated errors between 3 and 10% (Type S) we have minor structure, and for estimated errors over 10% (Type W) we have significant structure in the source. Confused sources (Type C) are not predictable enough for regular use and some sources are so unreliable at certain resolutions and wavelengths they are classified as 'Do Not Use'.

divided values are subtracted by 1. These are:

$$\mathrm{SRE} \equiv \langle [\mathrm{Re}(\mathcal{V})]_{\mathrm{obsv.}} - [\mathrm{Re}(\mathcal{V})]_{\mathrm{model}} \rangle \tag{12}$$

$$\mathrm{SIM} \equiv \langle [\mathrm{Im}(\mathcal{V})]_{\mathrm{obsv.}} - [\mathrm{Im}(\mathcal{V})]_{\mathrm{model}} \rangle \tag{13}$$

$$\mathrm{SAMP} \equiv \langle |\mathcal{V}_{\mathrm{obsv.}}| - |\mathcal{V}_{\mathrm{model}}| \rangle \tag{14}$$

$$\mathrm{SFRAC} \equiv \left(\left[\frac{\text{Found Soln.}}{\text{Prdct. Soln.}} \right]_{\text{Subtr. Data}} - 1 \right) \tag{15}$$

$$\mathrm{DRE} \equiv \left(\langle \frac{[\mathrm{Re}(\mathcal{V})]_{\mathrm{obsv.}}}{[\mathrm{Re}(\mathcal{V})]_{\mathrm{model}}} \rangle - 1 \right) \tag{16}$$

$$\mathrm{DIM} \equiv \left(\langle \frac{[\mathrm{Im}(\mathcal{V})]_{\mathrm{obsv.}}}{[\mathrm{Im}(\mathcal{V})]_{\mathrm{model}}} \rangle - 1 \right) \tag{17}$$

$$\mathrm{DAMP} \equiv \left(\langle \frac{|\mathcal{V}_{\mathrm{obsv.}}|}{|\mathcal{V}_{\mathrm{model}}|} \rangle - 1 \right) \tag{18}$$

$$\mathrm{DFRAC} \equiv \left(\left[\frac{\text{Found Soln.}}{\text{Prdct. Soln.}} \right]_{\text{Dvd. Data}} - 1 \right). \tag{19}$$

We want to mathematically characterize how these values of a real calibrator diverge from the ideal point source, in multiple ways. First, knowing what their ideal values should be, we can measure how close a calibrator is to the ideal point source by calculating a distance in an appropriate 'error space', with locations on axes defined by equations 12 through 19. Note that the 'subtracted values' are sensitive to the flux of the source, so to avoid weighting our score incorrectly, we will scale these values by dividing by the average flux of the source, $\langle S \rangle$. Also, some of these values may be more important than others in determining the quality of a source, so we will multiply each term by a weighting value β_i, which we must determine experimentally. We will call this the 'characteristic sub-score'.

$$\mathrm{CSS} \equiv \frac{1}{\sum_{i=1}^{8} \beta_i} \Bigg[\beta_1 \left(\frac{\mathrm{SRE}}{\langle S \rangle} \right)^2 + \beta_2 \left(\frac{\mathrm{SIM}}{\langle S \rangle} \right)^2 + \beta_3 \left(\frac{\mathrm{SAMP}}{\langle S \rangle} \right)^2 + \beta_4 \mathrm{SFRAC}^2 + \beta_5 \mathrm{DRE}^2 + \beta_6 \mathrm{DIM}^2 + \beta_7 \mathrm{DAMP}^2 + \beta_8 \mathrm{DFRAC}^2 \Bigg]^{1/2} \tag{20}$$

Second, we can determine how reliable a given observation is. An ideal calibrator would have no error, so we want the error on our real calibrators to be as low as possible. That means we want as low of root-mean-square or average values as possible for the values found in equations 12 through 19. We will call this the 'reliability sub-score'. Again, the 'subtracted values' are sensitive to the flux of the source, so they are scaled based on the average flux. Just as an equations 12–19, each value can be weighted to change how much they influence the final score, noted here by γ_i.

$$\mathrm{RSS} \equiv \frac{1}{\sum_{i=1}^{8} \gamma_i} \Bigg[\gamma_1 \left(\frac{\mathrm{SRE_{RMS}}}{\langle S \rangle} \right)^2 + \gamma_2 \left(\frac{\mathrm{SIM_{RMS}}}{\langle S \rangle} \right)^2 + \gamma_3 \left(\frac{\mathrm{SAMP_{RMS}}}{\langle S \rangle} \right)^2 + \beta_4 \mathrm{SFRAC}^2_{\mathrm{Avg.}} + \gamma_5 \mathrm{DRE}^2_{\mathrm{RMS}} + \gamma_6 \mathrm{DIM}^2_{\mathrm{RMS}} + \gamma_7 \mathrm{DAMP}^2_{\mathrm{RMS}} + \gamma_8 \mathrm{DFRAC}^2_{\mathrm{Avg.}} \Bigg]^{1/2} \tag{21}$$

For ease of understanding, we want a higher scored calibrator to be considered a better calibrator. We will calculate the final score is calculated as the inverse of the sum of the sub-scores, the inverse of the weighted flux of the source, and one. A stronger flux inherently improves a calibrator, so this must be taken into account. The one establishes the score range between zero and one, as opposed to zero and infinity. We introduce three more scaling weights: δ_1, δ_2, and δ_3 acting on the CSS, RSS, and flux, respectively. And finally, we can establish binary cut-off values α_i for all of the components of the score, where if that value does not reach the defined minimum or exceeds the defined maximum, the source is automatically discarded by setting the score to zero.

$$\text{Score} \equiv \frac{\left(\prod_{i=1}^{18} \alpha_i\right)}{\left(1/\sum_{i=1}^{3} \delta_i\right)\left(\delta_1 \text{CSS} + \delta_2 \text{RSS} + \delta_3/\langle S\rangle\right) + 1} \tag{22}$$

The values of each α_i depend on the data set, and for our data they are defined in the next section.

2.2. *Implementation*

While the score is in theory applicable to any source, in practice we can determine if sources are not worth the computational effort to calculate. By default, these cut-off values are set so that all sources have scores calculated, but values can be declared by the user. For more information and the implementation of this system, please see appendix A. For the data set being examined (see section 2.3), we have determined fourteen cut-offs for usable calibrators, and four values with no effective cut-off:

$$\alpha_1 = \begin{cases} 0, & |\frac{\text{SRE}}{\langle S\rangle}| > 0.32 \\ 1, & |\frac{\text{SRE}}{\langle S\rangle}| \leq 0.32 \end{cases} \qquad \alpha_2 = \begin{cases} 0, & \frac{\text{SRE}_{\text{RMS}}}{\langle S\rangle} > 2.75 \\ 1, & \frac{\text{SRE}_{\text{RMS}}}{\langle S\rangle} \leq 2.75 \end{cases}$$

$$\alpha_3 = \begin{cases} 0, & |\frac{\text{SIM}}{\langle S\rangle}| > 0.25 \\ 1, & |\frac{\text{SIM}}{\langle S\rangle}| \leq 0.25 \end{cases} \qquad \alpha_4 = \begin{cases} 0, & \frac{\text{SIM}_{\text{RMS}}}{\langle S\rangle} > 0.88 \\ 1, & \frac{\text{SIM}_{\text{RMS}}}{\langle S\rangle} \leq 0.88 \end{cases}$$

$$\alpha_5 = \begin{cases} 0, & |\frac{\text{SAMP}}{\langle S\rangle}| > 0.32 \\ 1, & |\frac{\text{SAMP}}{\langle S\rangle}| \leq 0.32 \end{cases} \qquad \alpha_6 = \begin{cases} 0, & \frac{\text{SAMP}_{\text{RMS}}}{\langle S\rangle} > 1.33 \\ 1, & \frac{\text{SAMP}_{\text{RMS}}}{\langle S\rangle} \leq 1.33 \end{cases}$$

$$\alpha_7 = \begin{cases} 0, & \text{SFRAC} > \infty \\ 1, & \text{SFRAC} < \infty \end{cases} \qquad \alpha_8 = \begin{cases} 0, & \text{SFRAC}_{\text{AVG}} > \infty \\ 1, & \text{SFRAC}_{\text{AVG}} < \infty \end{cases}$$

$$\alpha_9 = \begin{cases} 0, & |\text{DRE}| > 0.32 \\ 1, & |\text{DRE}| \leq 0.32 \end{cases} \qquad \alpha_{10} = \begin{cases} 0, & \text{DRE}_{\text{RMS}} > 2.75 \\ 1, & \text{DRE}_{\text{RMS}} \leq 2.75 \end{cases}$$

$$\alpha_{11} = \begin{cases} 0, & |\text{DIM}| > 0.06 \\ 1, & |\text{DIM}| \leq 0.06 \end{cases} \qquad \alpha_{12} = \begin{cases} 0, & \text{DIM}_{\text{RMS}} > 0.94 \\ 1, & \text{DIM}_{\text{RMS}} \leq 0.94 \end{cases}$$

$$\alpha_{13} = \begin{cases} 0, & \text{DAMP} > 0.786 \\ 1, & \text{DAMP} \leq 0.786 \end{cases} \qquad \alpha_{14} = \begin{cases} 0, & \text{DAMP}_{\text{RMS}} > 1.13 \\ 1, & \text{DAMP}_{\text{RMS}} \leq 1.13 \end{cases}$$

$$\alpha_{15} = \begin{cases} 0, & |\text{DFRAC}| > \infty \\ 1, & |\text{DFRAC}| < \infty \end{cases} \qquad \alpha_{16} = \begin{cases} 0, & \text{DFRAC}_{\text{AVG}} > \infty \\ 1, & \text{DFRAC}_{\text{AVG}} < \infty \end{cases}$$

$$\alpha_{17} = \begin{cases} 0, & N_{\mathcal{V}} < 100 \\ 1, & N_{\mathcal{V}} > 100 \end{cases} \qquad \alpha_{18} = \begin{cases} 0, & \langle S\rangle < 0.05 \text{ Jy/bm} \\ 1, & \langle S\rangle \geq 0.05 \text{ Jy/bm} \end{cases}.$$

For α_7, α_8, α_{15}, and α_{16}, we could not determine a value above or below which all sources were poor calibrators, so they are set such that the expression is always true for any value of SFRAC, $\text{SFRAC}_{\text{AVG}}$, DFRAC, or $\text{DFRAC}_{\text{AVG}}$, respectively.

2.3. *Data Set*

The calibrators the VLA uses are documented on the NRAO Science website, sorted based on the right ascension of the source (Van Moorsel & Sjouwerman 2019). Data for the sources was provided by my advisor, Loránt Sjouwerman, which were fitted to models using the Astronomical Image Processing System (AIPS) (Bridle 2020). The calibrators used here are distributed across the entire VLA sky, so there should no bias based on location.

The data set at hand serves to verify that the scoring system established in section 2.1 functions as intended. That is, high flux, point-like sources should be sorted near the top while low flux, resolved, or noise-polluted sources should be sorted toward the bottom. 3358 sources are being ranked and sorted, so it would be impractical to present them all in the order they appear here. The bulk properties of the data set can be seen in the histograms given in Figures 3–6. For more details on the calculation of these scores, see Appendix A.

2.4. *Execution*

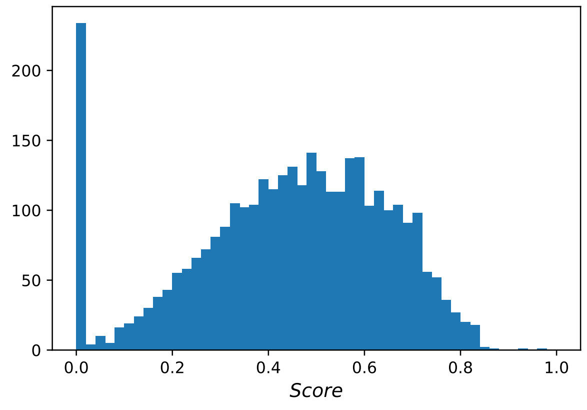
Note, that due to the structure of the scoring system, determining weightings is not straight-forward. Emphasizing a score over others is an exercise in adjusting all values, as a straight forward increase in the weight will simply decrease the final score, and decreasing it to increase the score can hide its impact. With nineteen different values to adjust, this problem is too complex to be solved without simulation. For now, the weights used in this run are as follows:

$$\begin{aligned}
\beta_1 &= \frac{1}{30} & \beta_2 &= \frac{1}{8} \\
\beta_3 &= \frac{1}{128} & \beta_4 &= \frac{1}{6} \\
\beta_5 &= \frac{1}{5} & \beta_6 &= \frac{4}{5} \\
\beta_7 &= \frac{1}{20} & \beta_8 &= 10 \\
\gamma_1 &= \frac{1}{10} & \gamma_2 &= \frac{1}{40} \\
\gamma_3 &= \frac{1}{64} & \gamma_4 &= 15 \\
\gamma_5 &= \frac{3}{5} & \gamma_6 &= \frac{2}{5} \\
\gamma_7 &= \frac{1}{16} & \gamma_8 &= 20 \\
\delta_1 &= 1.2 & \delta_2 &= 0.8 \\
\delta_3 &= 0.1.
\end{aligned}$$

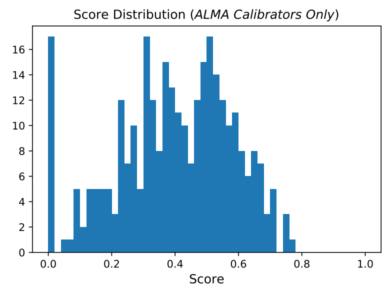
These β and γ weights were decided by ranking qualitatively how well each of the 16 contributing values (eqs. 12–19 and their respective statistical qualities) discriminated a likely-P (PP) classified calibrator from a likely-S (SS) classified calibrator, with δ_1 and δ_2 chosen to make the contributions of the CSS and RSS of similar magnitude. δ_3 is special and simply scales the score; it was adjusted so that the scores usefully filled the space from zero to one. The results of this run can be seen in figure 1.

What then is a likely-P or likely-S calibrator? To assess the system, we are ultimately looking to see how well it distinguishes a good calibrator from a poor calibrator. With over 3000 sources in the S-band, a band with no pre-assessed calibrators to examine, it would be impractical to assess each source individually. Instead, a likely classification was assigned using the existing VLA calibrator database (Van Moorsel & Sjouwerman 2019) for L-band and C-band sources, in the VLA's A configuration, where all of our measurements were taken. The assumption we are making is that if a calibrator is of equal quality in the L- and C-bands, it will probably be of the same quality in S-band, an intermediate frequency.

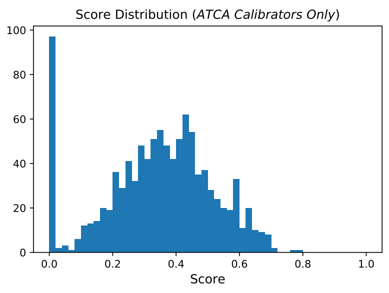
An effective system should show separation between the calibrator classes, with P calibrators peaking closest to one, S calibrators following, then W calibrators, followed finally by C and X. The order of P, S, and W is easy to establish, but C and X will have unknown placement and spread; without more information, we are unable to tell the reason for that classification, a notable drawback of the closure phase error method. In this case, we are ignoring likely-W calibrators due to a lack of sources—of over 3000 sources, 6 are likely-W (WW) calibrators in the VLA A configuration in S band, far too few to draw conclusions from. In figure 2, we can see the results of this run of the system with the weights above. We do indeed find that the likely-P sources are more highly ranked on average than any other. However, there is significant overlap and as demonstrated in figures 7 and 8 significant work that must be done to refine the weights used in the system.



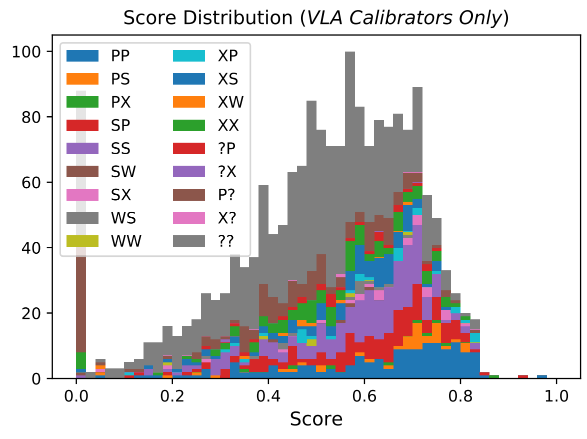
(a)



(b)



(c)



(d)

Figure 1. In subfigure (a), we can see the distribution of all S-band sources used for this study. For ranked calibrators, most sources are of average quality, but a few are sorted out to being of exceptionally good or of poor quality. In subfigures (b), (c), and (d) we can see the breakdown of scores for sources that make up our data set. In subfigures (b) and (c), we have sources that are observed by ALMA or ATCA and are known calibrators at higher frequencies than S band. In subfigure (d), we have sources that are known calibrators in L-band and C-band. These are color-coded by the classification in each band, of which 18 of 36 have entries. As can be seen here, sources that are of higher quality in either of these bands are more likely to be scored highly in our calculation.

REFERENCES

Bridle, A. 2020, Astronomical Image Processing System (AIPS) Cookbook, Socorro: National Radio Astronomy Observatory. http://www.aips.nrao.edu/cook.html

Fomalont, E. B., & Perley, R. A. 1999, in Astronomical Society of the Pacific Conference Series, Vol. 180, Synthesis Imaging in Radio Astronomy II, ed. G. B. Taylor, C. L. Carilli, & R. A. Perley, 79

Hunter, J. D. 2007, Computing in science & engineering, 9, 90

Jackson, J. D. 1998, Classical Electrodynamics, 3rd Edition

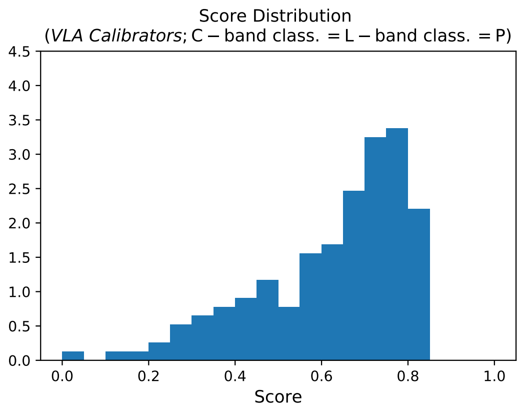
Kluyver, T., Ragan-Kelley, B., Pérez, F., et al. 2016, in Positioning and Power in Academic Publishing: Players, Agents and Agendas, ed. F. Loizides & B. Schmidt, IOS Press, 87 – 90

Oliphant, T. E. 2006, A guide to NumPy, Vol. 1 (Trelgol Publishing USA)

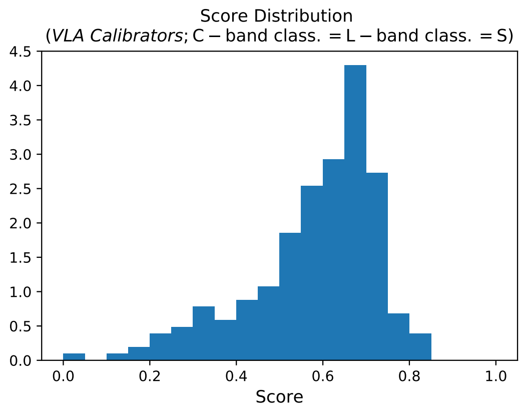
Thompson, A. R., Moran, J. M., & Swenson, George W., J. 2017, Interferometry and Synthesis in Radio Astronomy, 3rd Edition, doi: 10.1007/978-3-319-44431-4

Van Moorsel, G., & Sjouwerman, L. 2019, List of VLA Calibrators, National Radio Astronomy Observatory. https://science.nrao.edu/facilities/vla/observing/callist

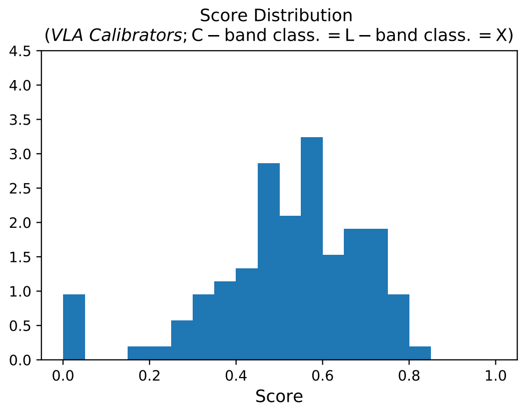
Van Rossum, G., & Drake, F. L. 2009, Python 3 Reference Manual (Scotts Valley, CA: CreateSpace)



(a)



(b)



(c)

Figure 2. In subfigure (a), we can see the distribution of likely-P calibrators, whose score probability peaks around 0.75. In subfigure (b), we can see the distribution of likely-S calibrators, whose score peaks near 0.7. Finally, in subfigure (c), we can see the score of likely-X calibrators peaking around 0.5. As we desired, likely-P and likely-S calibrators peak at different places. Without more information about the reasoning for X-classification, we must simply be satisfied that they are on average worse then likely-P calibrators.

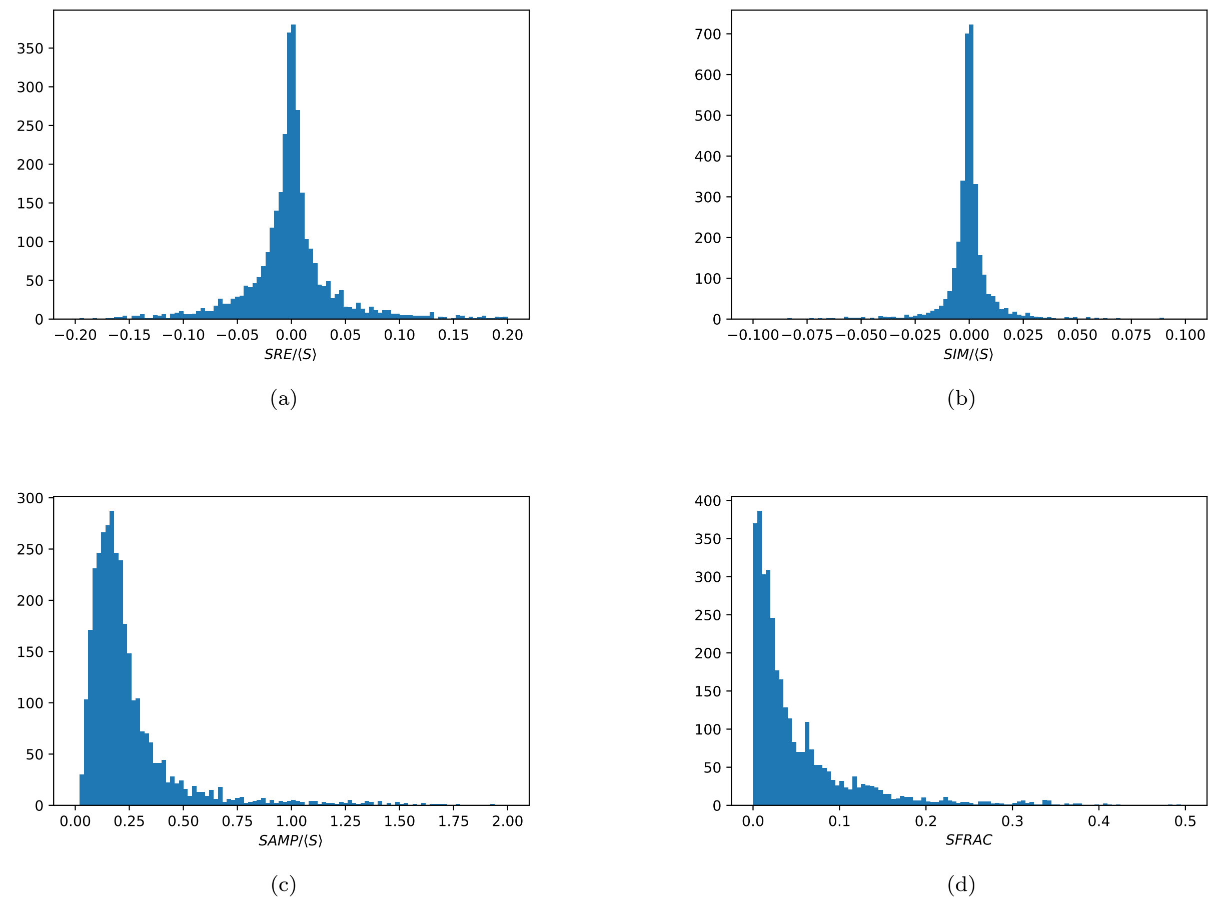


Figure 3. These histograms show the distributions of the 4 'subtracted' scoring components (equations 12–15) we are looking at to construct the score (equation 22) for a calibrator. A good point source should be closer to 0 for all of these properties.

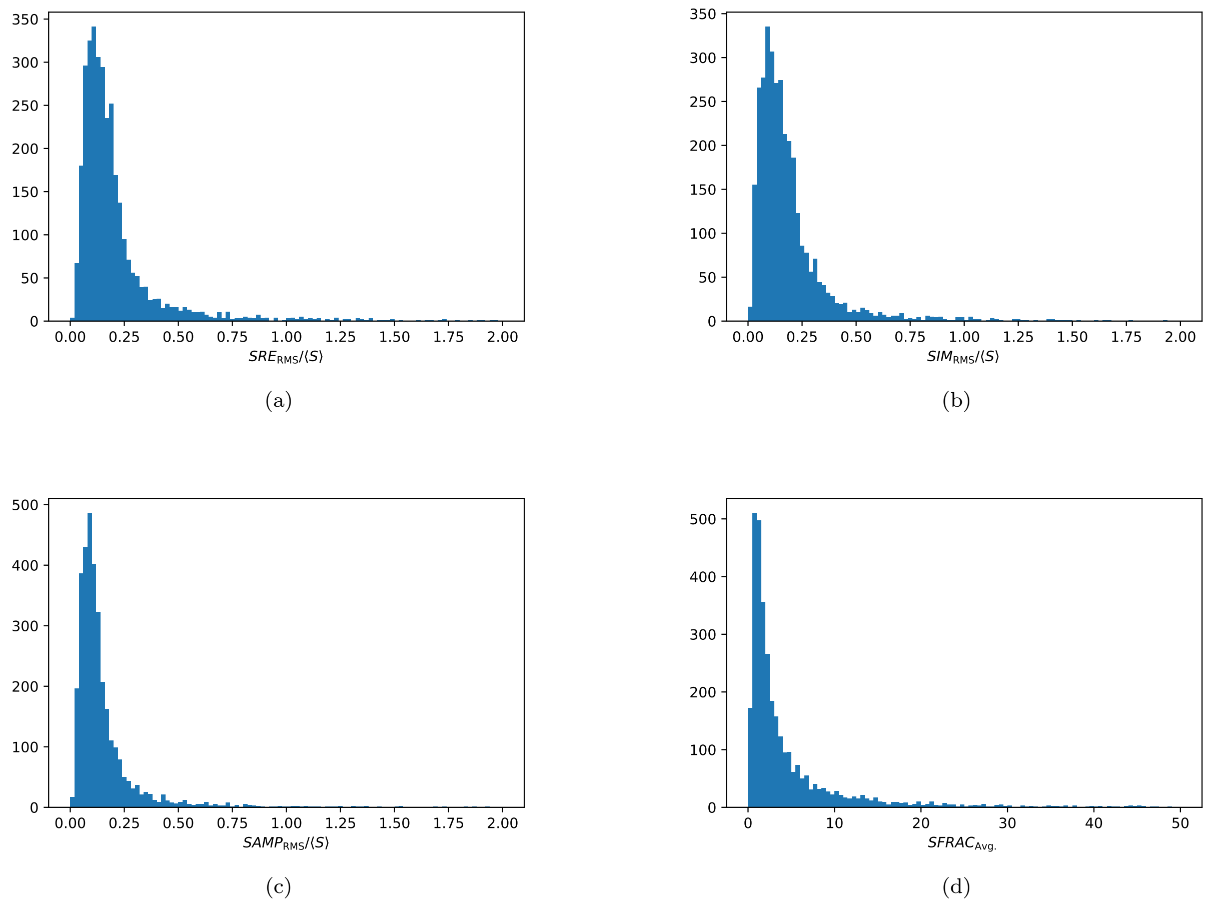
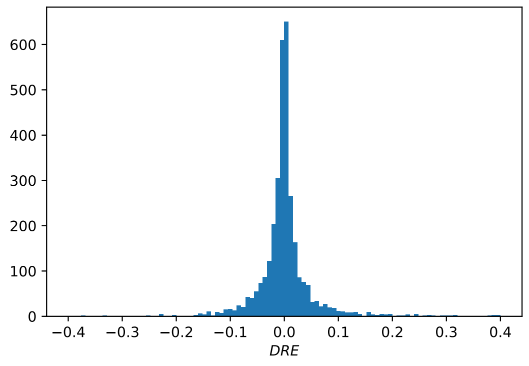
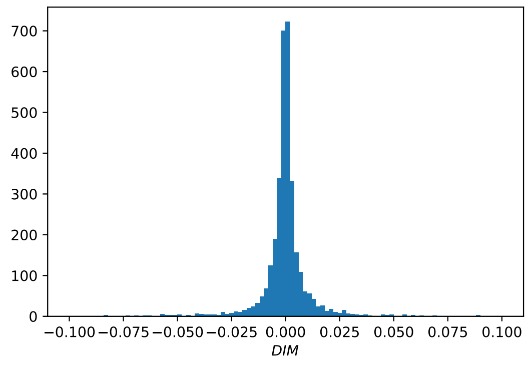


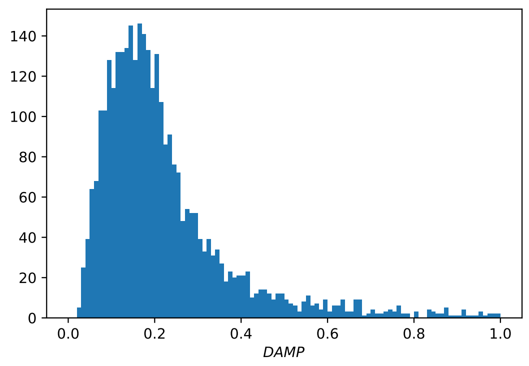
Figure 4. These histograms show the distributions of the 4 'subtracted' quality scoring components used to assess the reliability of the data (equation 21) we are looking at to construct the score (equation 22) for a calibrator. A good point source should be closer to 0 for all of these properties.



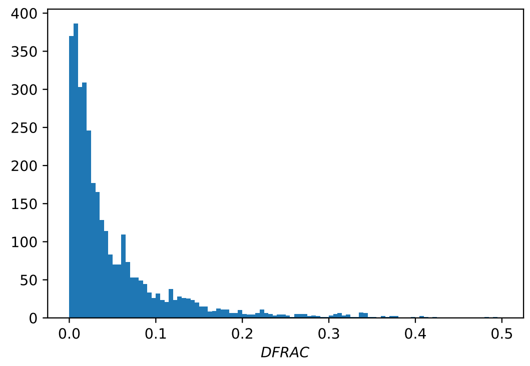
(a)



(b)

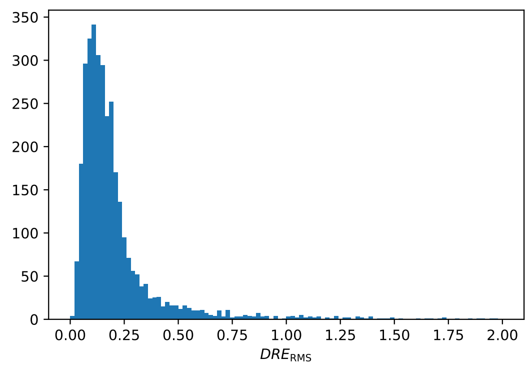


(c)

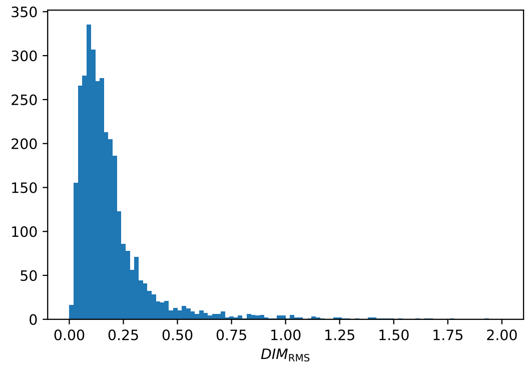


(d)

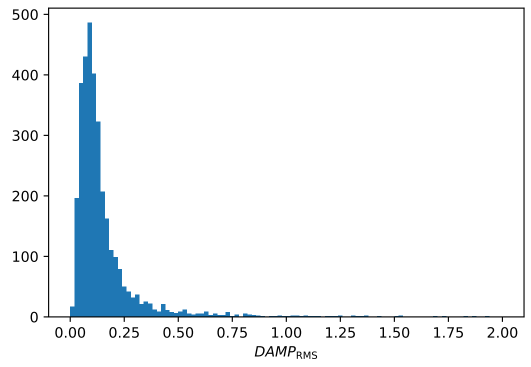
Figure 5. These histograms show the distributions of the 4 ’divided’ scoring components (equations 16–19) we are looking at to construct the score (equation 22) for a calibrator. A good point source should be closer to 0 for all of these properties.



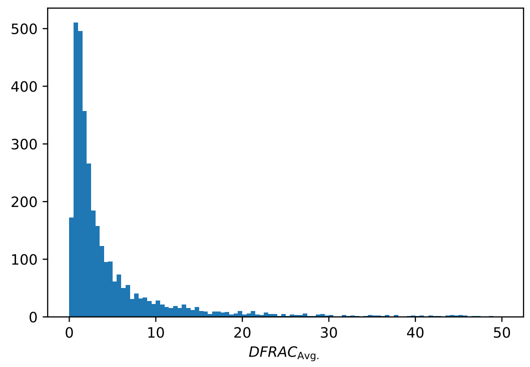
(a)



(b)

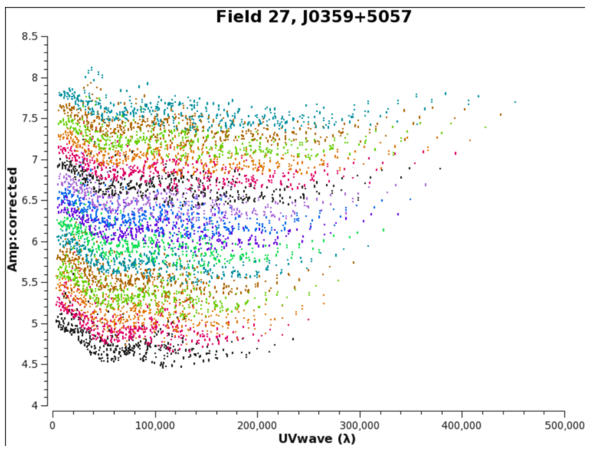


(c)



(d)

Figure 6. These histograms show the distributions of the 4 'divided' quality scoring components used to assess the reliability of the data (equation 21) we are looking at to construct the score (equation 22) for a calibrator. A good point source should be closer to 0 for all of these properties.



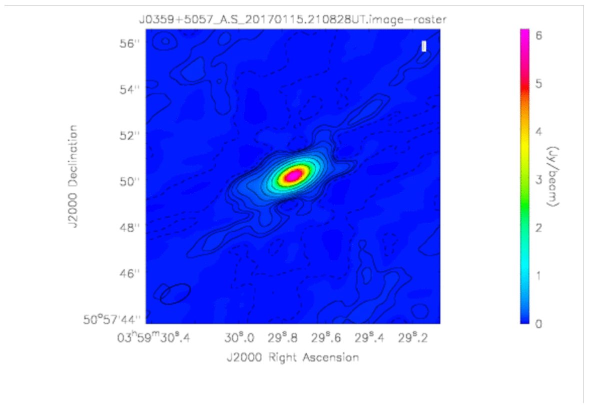


Figure 7. As of the iteration in use on 14 August 2020, source J0359+5057 is rank 1 of all scored sources. As can be seen from the Amplitude vs. UV diagram (left), it has a maximum flux on the order of 1 Jy/beam, sufficient for calibration, and has a constant amplitude with baseline length λ. In the sky (right), the source is compact, but is not what we would describe as a point source due to the clear surrounding structure. It serves as an example that the absolute ranking is not finalized; in this case, the flux of this source is inflating its score past conventionally better calibrators.

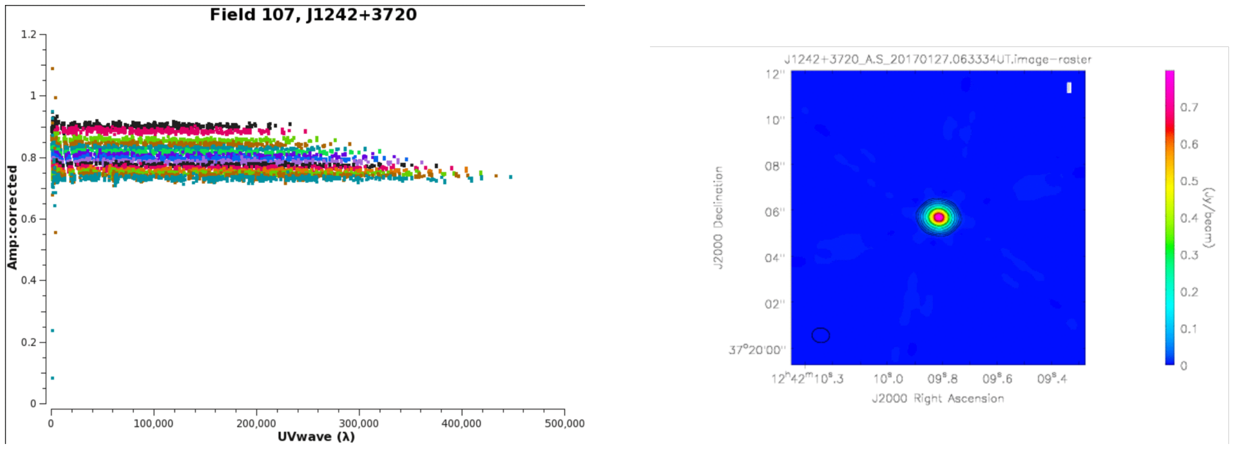


Figure 8. As of the iteration in use on 14 August 2020, source J1242+3720 is rank 63 (98th percentile) of all scored sources. As can be seen from the Amplitude vs. UV diagram (left), it has a maximum flux on the order of a 0.95 Jy, sufficient for calibration but weaker than the source in Fig. 7, and has a constant amplitude with baseline length λ. It can also be seen that the spectral index is small. In the sky (right), the source is very compact, and is what we would describe as a point source. It is what we would consider a high quality calibrator.

APPENDIX

A. PYTHON CODE

The program to calculate the score was written in Python3 (Van Rossum & Drake 2009). The package NumPy (Oliphant 2006) was used extensively for calculating values, and MatPlotLib (Hunter 2007) was used for generating the figures found within this article. Running the program was done inside the Jupyter (Kluyver et al. 2016) notebook environment. The following section of code makes up the preamble of the program, importing all packages needed for use.

```
#!/usr/bin/env python
# coding: utf-8

import numpy as np
import os
import matplotlib
from datetime import datetime
import matplotlib.pyplot as plot
```

The following section of code takes a data file generated in AIPS and reads it into the program as an array. This is requires sorting around the descriptive text in the process and making exceptions for accidentally concatenated strings in AIPS. To see how this data as structured, see appendix B.

```
def data_lines_only (opened_text, index_array):
# This function will take the raw AIPS data and return only the rows with data as strings,
# discarding any with extraneous information (for our purposes)
    new_array = []
    for i in range (0,len(index_array)):
        data_rows = opened_text[index_array[i]:index_array[i]+5]
        new_array.append(data_rows)
    return new_array

# Necessary information in this file appears after a string "Task EVAUV   has finished",
# occupying five lines
Raw_AIPS_File = open('evauv/LOOPEVAU.PRT.org','r')
# Open a raw tect file of AIPS outputs from running evauv
Raw_AIPS_File_bylines=Raw_AIPS_File.readlines()
# Saves a line-by-line version of this output into memory
finished_text = "Task EVAUV   has finished"
# Text that tells us when data is about to be output
data_indices = np.asarray([i for i, s in enumerate(Raw_AIPS_File_bylines) if finished_text in s])
# Save the index of all lines where above text occurs
data_indices = data_indices+1
# Increment all elements by one, to the lines where data actually starts
AIPS_FILE_DATA = data_lines_only(Raw_AIPS_File_bylines,data_indices)
# cut out all the extra stuff and leave only the data using the function defined above
Raw_AIPS_File.close()
# Close the raw data file because we extracted everything of value
# The above is a list of lists, the main list holding all data and a sub-list
# holding an individual source
# Next, we need to read each source and get the data out of it as strings or floats,
# as appropriate


header = ['Source Name', 'Visibil', 'Flux (Jy/b)', 'SUB Re', 'Sub Re RMS', 'Sub Im',
'Sub Im RMS', 'Sub Amp', 'Sub Amp RMS', 'Sub Frac', 'Sub Frac Avg', 'DIV Re', 'DIV Re RMS',
'DIV Im', 'DIV Im RMS', 'DIV Amp', 'DIV Amp RMS', 'DIV Frac', 'DIV Frac Avg' ]

def extract_data_from_AIPS (DATA):
# Using the data only array of strings generated in the previous function, extract the data for
# each calibrator as floats or strings
    read_data = []
    read_data.append(header)
    for i in range(0,len(DATA)): # Loop over each string in the array
```

```
        source_data = []
        source_to_read = DATA[i]
        string_1 = source_to_read[0]
        sstring = string_1.split('AIPS',1)[1] # Cut off the output details
        sstring = sstring.split("'\n",1)[0] # Remove New Line '\n'
        data = sstring.split(' ',13) # Split string into substrings that have data we want
        source_name = data[6].split("'",1)[1] # Cut off the single quite at the start of the
        # name, and save to source_name variable
        source_vis = int(data[8]) # Turn the source visible points into an integer
        # and save as source_vis
        source_fluxbeam = float(data[13]) # Take the Jy/beam and turn into a float
        source_data.append(source_name) # Append name to array holding data from this source
        source_data.append(source_vis) # Append visibil to array holding data
        # from this source
        source_data.append(source_fluxbeam) # Append flux (in Jy/beam) to array holding data
        # from this source
        for j in range (1,len(source_to_read)):
        # This loop itterates over individual data now that the original string is split into
        # an array of strings.
            string_2 = source_to_read[j]
            sstring = string_2.split('AIPS',1)[1]
            sstring = sstring.split('\n',1)[0] # Remove New Line '\n'
            data = sstring.split()
            if len(data) <= 3:
            # Check if succcessfully split into 4 parts, if not look at each of the (likely) 3
            # strings
                if len(data[0])>10:
                    datum0 = float(data[0][0:11])
                    datum1 = float(data[0][11:])
                    datum2 = float(data[1])
                    datum3 = float(data[2])
                if len(data[1])>10:
                    datum0 = float(data[0])
                    datum1 = float(data[1][0:11])
                    datum2 = float(data[1][11:])
                    datum3 = float(data[2])
                if len(data[2])>8:
                # Most common error. The final datum in some sets begins adjacent to the
                # penultimate. The penultimate is 9 characters long (precision limit), so the
                # third string is split into one of length 9 and another of the remainder.
                    datum0 = float(data[0])
                    datum1 = float(data[1])
                    datum2 = float(data[2][0:9])
                    datum3 = float(data[2][9:])
                # Should change to check if a substring has more than one dot
                # Note: Need to change to exception if conversion to float fails.
            else:
                datum0 = float(data[0]) # Individual values from line
                datum1 = float(data[1]) # Individual values from line
                datum2 = float(data[2]) # Individual values from line
                datum3 = float(data[3]) # Individual values from line
            source_data.append(datum0)
            source_data.append(datum1)
            source_data.append(datum2)
            source_data.append(datum3)
        read_data.append(source_data)
    return read_data

Extracted_Data = extract_data_from_AIPS(AIPS_FILE_DATA)
```

The next section of code implements the scoring system.

```
# For easier changing later on, if necessary, let's assign a variable for each column index
SrcNm_Indx = 0 # Tells us the name of the source. Good for identifying only (clearly)
vsbl_Indx = 1 # Tells us how many individual detections make up the source. More is better.
flx_Jyb_Indx = 2 # Tells us how bright the source is. Higher is better.
SRE_Indx = 3
SRE_RMS_Indx = 4
SIM_Indx = 5
```

```
SIM_RMS_Indx = 6
SAMP_Indx = 7
SAMP_RMS_Indx = 8
SFRAC_Indx = 9
SFRAC_AVG_Indx = 10 # Tells us if points are distributed in lines in FLX-UV space at given lambda.
# Lower is better.
# NOTE: IN THE DATA SET, WE HAVE DIV-1, NOT DIV!
DRE_Indx = 11
DRE_RMS_Indx = 12
DIM_Indx = 13
DIM_RMS_Indx = 14
DAMP_Indx = 15
DAMP_RMS_Indx = 16
DFRAC_Indx = 17
DFRAC_AVG_Indx = 18
Score_Indx = 19

# Using the above columns, we can create a scoring system by weighting values. This score should
# quantify how good of a calibrator the source is.

# The scoring system is a combination of the various values above.

def calculate_score_vector_space (data, vsbl_wt=1, flx_Jyb_wt=1, SRE_wt=1, SRE_RMS_wt=1, SIM_wt=1, SIM_RMS_wt=1, SAMP_wt=1,
                                SAMP_RMS_wt=1, SFRAC_wt=1, SFRAC_AVG_wt=1, DRE_wt=1, DRE_RMS_wt=1, DIM_wt=1, DIM_RMS_wt=1,
                                DAMP_wt=1, DAMP_RMS_wt=1, DFRAC_wt=1, DFRAC_AVG_wt=1, vsbl_cut=0, flx_Jyb_cut=0,
                                SRE_cut=np.inf, SRE_RMS_cut=np.inf, SIM_cut=np.inf, SIM_RMS_cut=np.inf, SAMP_cut=np.inf,
                                SAMP_RMS_cut=np.inf, SFRAC_cut=np.inf, SFRAC_AVG_cut=np.inf, DRE_cut=np.inf,
                                DRE_RMS_cut=np.inf, DIM_cut=np.inf, DIM_RMS_cut=np.inf, DAMP_cut=np.inf, DAMP_RMS_cut=np.inf,
                                DFRAC_cut=np.inf, DFRAC_AVG_cut=np.inf):
     # This scoring system uses what we know about an ideal point source to create a vector space around which we can make score base
     # on the distance in this space from the ideal. This funtion also takes optional arguments to weight values and establish cut-of
     # above which we shouldn't consider the source anymore; visibilities and flux are exceptions and are minima instead.
     scores_column = []
     cut_sources = 0
     for lpng_indx in range(1,len(data)):
          if data[lpng_indx][vsbl_Indx] < vsbl_cut:
               raw_score = 0
               cut_sources += 1
               print ('Source', data[lpng_indx][0], 'assigned score=0 due to number of visibilities being below minimum.')
          elif data[lpng_indx][flx_Jyb_Indx] < flx_Jyb_cut:
               raw_score = 0
               cut_sources += 1
               print ('Source', data[lpng_indx][0], 'assigned score=0 due to flux/beam being below minimum (assuming Jy/beam).')
          elif np.abs(data[lpng_indx][SRE_Indx])/data[lpng_indx][flx_Jyb_Indx] > SRE_cut:
               raw_score = 0
               cut_sources += 1
               print ('Source', data[lpng_indx][0], 'assigned score=0 due to magnitude of observed minus model Re(V)/<S> being above
          elif np.abs(data[lpng_indx][SRE_RMS_Indx])/data[lpng_indx][flx_Jyb_Indx] > SRE_RMS_cut:
               raw_score = 0
               cut_sources += 1
               print ('Source', data[lpng_indx][0], 'assigned score=0 due to observed minus model Re(V)/<S> RMS being above maximum.'
          elif np.abs(data[lpng_indx][SIM_Indx])/data[lpng_indx][flx_Jyb_Indx] > SIM_cut:
               raw_score = 0
               cut_sources += 1
               print ('Source', data[lpng_indx][0], 'assigned score=0 due to magnitude of observed minus model Im(V)/<S> being above
          elif np.abs(data[lpng_indx][SIM_RMS_Indx])/data[lpng_indx][flx_Jyb_Indx] > SIM_RMS_cut:
               raw_score = 0
               cut_sources += 1
               print ('Source', data[lpng_indx][0], 'assigned score=0 due to observed minus model Im(V)/<S> RMS being above maximum.'
          elif np.abs(data[lpng_indx][SAMP_Indx])/data[lpng_indx][flx_Jyb_Indx] > SAMP_cut:
               raw_score = 0
               cut_sources += 1
               print ('Source', data[lpng_indx][0], 'assigned score=0 due to magnitude of observed minus model |V|/<S> being above ma
          elif np.abs(data[lpng_indx][SAMP_RMS_Indx])/data[lpng_indx][flx_Jyb_Indx] > SAMP_RMS_cut:
               raw_score = 0
               cut_sources += 1
               print ('Source', data[lpng_indx][0], 'assigned score=0 due to observed minus model |V|/<S> RMS being above maximum.')
          elif np.abs(data[lpng_indx][DRE_Indx]) > DRE_cut:
               raw_score = 0
               cut_sources += 1
               print ('Source', data[lpng_indx][0], 'assigned score=0 due to magnitude of observed divided by model Re(V) being above
          elif data[lpng_indx][DRE_RMS_Indx] > DRE_RMS_cut:
               raw_score = 0
               cut_sources += 1
               print ('Source', data[lpng_indx][0], 'assigned score=0 due to observed divided by model Re(V) RMS being above maximum.
          elif np.abs(data[lpng_indx][DIM_Indx]) > DIM_cut:
               raw_score = 0
               cut_sources += 1
               print ('Source', data[lpng_indx][0], 'assigned score=0 due to magnitude of observed divided by model Im(V) being above
          elif data[lpng_indx][DIM_RMS_Indx] > DIM_RMS_cut:
               raw_score = 0
               cut_sources += 1
               print ('Source', data[lpng_indx][0], 'assigned score=0 due to observed divided by model Im(V) RMS being above maximum.
          elif np.abs(data[lpng_indx][DAMP_Indx]) > DAMP_cut:
               raw_score = 0
               cut_sources += 1
               print ('Source', data[lpng_indx][0], 'assigned score=0 due to magnitude of observed divided by model |V| being above m
          elif data[lpng_indx][DAMP_RMS_Indx] > DAMP_RMS_cut:
               raw_score = 0
               cut_sources += 1
               print ('Source', data[lpng_indx][0], 'assigned score=0 due to observed divided by model |V| RMS being above maximum.')
          elif np.abs(data[lpng_indx][SFRAC_Indx]) > SFRAC_cut:
               raw_score = 0
               cut_sources += 1
               print ('Source', data[lpng_indx][0], 'assigned score=0 due to subtracted fraction of bad gains above maximum.')
          elif np.abs(data[lpng_indx][SFRAC_AVG_Indx]) > SFRAC_AVG_cut:
               raw_score = 0
               cut_sources += 1
               print ('Source', data[lpng_indx][0], 'assigned score=0 due to subtracted average of bad gains above maximum.')
          elif np.abs(data[lpng_indx][DFRAC_Indx]) > DFRAC_cut:
               raw_score = 0
               cut_sources += 1
               print ('Source', data[lpng_indx][0], 'assigned score=0 due to divided fraction of bad gains above maximum.')
          elif np.abs(data[lpng_indx][DFRAC_AVG_Indx]) > DFRAC_AVG_cut:
               raw_score = 0
               cut_sources += 1
               print ('Source', data[lpng_indx][0], 'assigned score=0 due to divided average of bad gains above maximum.')
          else:
               # RMS values should ideally be zero (perfectly reliable data), so we can create a reliability measure with that data.
               real_reliability = np.sqrt((((SRE_RMS_wt)*data[lpng_indx][SRE_RMS_Indx]/data[lpng_indx][flx_Jyb_Indx])**2 +
               ((DRE_RMS_wt)*data[lpng_indx][DRE_RMS_Indx])**2))
               imag_reliability = np.sqrt((((SIM_RMS_wt)*data[lpng_indx][SIM_RMS_Indx]/data[lpng_indx][flx_Jyb_Indx])**2 +
               ((DIM_RMS_wt)*data[lpng_indx][DIM_RMS_Indx])**2))
               amp_reliability = np.sqrt(((SAMP_RMS_wt*data[lpng_indx][SAMP_RMS_Indx]/data[lpng_indx][flx_Jyb_Indx])**2 +
               (DAMP_RMS_wt*data[lpng_indx][DAMP_RMS_Indx])**2))
               aggregate_reliability = np.sqrt(real_reliability**2 + imag_reliability**2 + amp_reliability**2)
               flatness_score = np.sqrt(SFRAC_AVG_wt*(data[lpng_indx][SFRAC_AVG_Indx])**2 + DFRAC_AVG_wt*(data[lpng_indx][DFRAC_AVG_I
               # Using the RMS values, calculate reliability as the quadrature sum of individual RMS values
               # We can also multiply by the raw score to weight based on reliability.
               SFRAC_score = np.abs(data[lpng_indx][SFRAC_Indx])
               DFRAC_score = np.abs(data[lpng_indx][DFRAC_Indx])
               SIM_score = np.abs(data[lpng_indx][SIM_Indx])/data[lpng_indx][flx_Jyb_Indx]
               DIM_score = np.abs(data[lpng_indx][DIM_Indx])
               SRE_score = np.abs(data[lpng_indx][SRE_Indx])/data[lpng_indx][flx_Jyb_Indx]
               DRE_score = np.abs(data[lpng_indx][DRE_Indx])
               SAMP_score = np.abs(data[lpng_indx][SAMP_Indx])/data[lpng_indx][flx_Jyb_Indx]
               DAMP_score = np.abs(data[lpng_indx][DAMP_Indx])

               imag_dist = np.sqrt(SIM_wt*(SIM_score)**2+DIM_wt*(DIM_score)**2) # Ideally SIM = DIM = 0. This calcultates the distanc
               real_dist = np.sqrt(SRE_wt*(SRE_score)**2+DRE_wt*(DRE_score)**2) # The same holds for Real space
               amp_dist = np.sqrt(SAMP_wt*(SAMP_score)**2+DAMP_wt*(DAMP_score)**2) # ibid
               frac_dist = np.sqrt(SFRAC_wt*(SFRAC_score)**2+DFRAC_wt*(DFRAC_score)**2) # ibid


               flux_score = np.abs(data[lpng_indx][flx_Jyb_Indx]) # Weight the score as avg. flux/flux. This way, stronger sources wi
               CSS = (1/(SRE_wt + DRE_wt + SIM_wt + DIM_wt + SAMP_wt + DAMP_wt + SFRAC_wt + DFRAC_wt))*np.sqrt((real_dist**2 + imag_d
               RSS = (1/(SRE_RMS_wt + DRE_RMS_wt + SIM_RMS_wt + DIM_RMS_wt + SAMP_RMS_wt + DAMP_RMS_wt + SFRAC_AVG_wt + DFRAC_AVG_wt)
               raw_score = 1/(1+score_wt*(1/((CSS_wt+RSS_wt+flx_Jyb_wt)))*((CSS_wt)*CSS + (RSS_wt)*RSS + flx_Jyb_wt/flux_score))
               if raw_score == np.inf: # No real data should be perfect
                    raw_score = 0
                    cut_sources += 1
               print('Source', data[lpng_indx][0], 'assigned score=0 due to calculated score being infinity.')

               scores_column.append(raw_score)
          print ('Total of', cut_sources, 'sources cut due to declared cut-offs.', len(data)-cut_sources,'ranked.')
          return scores_column


score_list = calculate_score_vector_space(Extracted_Data, vsbl_wt, flx_Jyb_wt, SRE_wt, SIM_wt,
SAMP_wt, SFRAC_wt, DRE_wt, DIM_wt, DAMP_wt, DFRAC_wt)
score_list.insert(0,'Scores') # Appends a header to this array.
```

The following section of code combines the new score array to the data array generated earlier. The new combined array is then sorted based on score, and an HTML file is created that shows the calibrators in ranked order with associated images.

```
transposed_Data = [list(x) for x in zip(*Extracted_Data)]
transposed_Data.append(score_list)
scored_Data = [list(x) for x in zip(*transposed_Data)]
data_header = scored_Data[0]
del scored_Data[0] # Remove the header in preparation for sorting

all_cal_images = os.listdir('all_cal')  # Obtain the location of images to be included in webpage.

def examine_final(elem):
    # Sorting Key. Will sort a list based on the final element in the list. For a list of lists,
    # like what we have, this sorts based on the final element of each sub-list (the score).
    return elem[-1]

def write_html_table_row (data_row_1, data_row_2, iterating_index):
    source_1_name = data_row_1[0]
    source_2_name = data_row_2[0]
    source_1_images = [iterator for iterator in all_cal_images if source_1_name in iterator]
    source_2_images = [iterator for iterator in all_cal_images if source_2_name in iterator]

    if len([iterator for iterator in source_1_images if 'UT' in iterator])<1:
        image_1_map = 'all_cal/'+'image_not_found.gif'
    else:
        image_1_map = 'all_cal/'+[iterator for iterator in source_1_images if 'UT' in iterator][0]

    if len([iterator for iterator in source_1_images if 'uvd' in iterator])<1:
        image_1_uv = 'all_cal/'+'image_not_found.gif'
    else:
        image_1_uv = 'all_cal/'+[iterator for iterator in source_1_images if 'uvd' in iterator][0]

    if len([iterator for iterator in source_2_images if 'UT' in iterator])<1:
        image_2_map = 'all_cal/'+'image_not_found.gif'
    else:
        image_2_map = 'all_cal/'+[iterator for iterator in source_2_images if 'UT' in iterator][0]

    if len([iterator for iterator in source_2_images if 'uvd' in iterator])<1:
        image_2_uv = 'all_cal/'+'image_not_found.gif'
    else:
        image_2_uv = 'all_cal/'+[iterator for iterator in source_2_images if 'uvd' in iterator][0]

    row_str_1 = '<tr><td align=center>Rank ' + str(iterating_index+1) + '<br />' + data_row_1[0]
    + '<br />value: ' + str(data_row_1[Score_Indx]) + '</td><td align=center><a href=' + image_1_uv
    + '><img width=200 src=' + image_1_uv + '></a></td><td align=center><a href=' + image_1_map
    + '><img width=200 src=' + image_1_map + '></a></td>'

    row_str_2 = '<td align=center>Rank ' + str(iterating_index+2) + '<br />'+data_row_2[0]
    + '<br />value: ' + str(data_row_2[Score_Indx]) + '</td><td align=center><a href=' + image_2_uv
    + '><img width=200 src=' + image_2_uv + '></a></td><td align=center><a href=' + image_2_map
    + '><img width=200 src=' + image_2_map + '></a></td></tr>\n'

    return row_str_1+row_str_2

scored_Data.sort(key = examine_final, reverse=True)

def write_sorted_html (data_with_score,score_method):
    now = datetime.utcnow()
    current_time = now.strftime("%d %h %Y, [%H:%M:%S UTC]")
    current_time_file_name = now.strftime("%d%h%Y_%H_%M_%S")
    Header_text= """<html>
    <head><title>/users/hprager/public_html/score_sorted</title></head>
    <body alink="#ffff00" bgcolor="#eeeeee" link="#ffff00" text="#000000" vlink="#ffff00">
    <p><center><h1>Column 19: sorted on score</h1></center></p>
    <p><hr></p><p><br>
    """
    text_date = '<font color="#000000" size="-1">Latest update: ' + current_time + '</font>\n'
    text_score_method = ', <font color="#000000" size="-1">Scoring Method: '+ score_method
    + '</font>\n'
    text_table_def = '</p><p><div align="center"><table border="1" cellpadding="4"
    cellspacing="1%" width="95%"><tbody>'
    file_text = []
    file_text.append(Header_text)
    file_text.append(text_date)
    file_text.append(text_score_method)
    file_text.append(text_table_def)
    indx = 0
    print('Sources: ',len(data_with_score)) # Tell user how many sources are going to be displayed
    # on the generated webpage
    while indx< len(data_with_score)-1:
        row = write_html_table_row (data_with_score[indx], data_with_score[indx+1], indx)
        file_text.append(row)
        if len(data_with_score)-indx == -1:
        # Check if we have an odd number of values at the end, so we can do something
        # special with the last row
            row = write_html_table_row (data_with_score[indx+2], data_with_score[indx+2], indx+2)
            # for now, duplicate final entry; clean-up later
            file_text.append(row)
            break
        else:
            indx = indx + 2 # Iterate by two because we have two sources per row
    closing_text =''''</tbody></table></div></p>
    <p><font color="#000000"size=-1>Auto-created by Henry Prager</font></p>
    <p><hr></p></body>
    </html>
    '''
    file_text.append(closing_text)
    joined_text = ''
    joined_text=joined_text.join(file_text)

    html_file = open("score_sorted"+current_time_file_name+'.html',"w")
    html_file.write(joined_text)
    html_file.close()
    print('HTML file "score_sorted'+current_time_file_name+'.html" written to current directory.')
    return 1
```

B. AIPS DATA

Below is a sample of the data being analyzed, as being output by AIPS. As can be seen, a new source is introduced with the statement ‘SOURCE’ ’J0302+5331’. The evaluation of this source is then declared finished with the phrase ‘Task EVAUV has finished’, which is the key phrase searched for by the extracting function (see appendix A). Data from the computed model then appears on the next 5 lines, with the first line being formatted uniquely. The extracting function takes the data from these lines, and sorts them into the appropriate column, simply based on the order they appear.

```
Pops  Prior    Date        Time       Task          Messages for user15933
1    2   14-FEB-2017  11:16:53     AIPS      Disk 12 No files needed renumbering
1    2   14-FEB-2017  11:16:53     AIPS      Got(1)   disk=12  user=****   type=UV   CAL_01.SPLAT0.1
1    5   14-FEB-2017  11:16:53     AIPS      'SKIPPING SOURCE'     '3C48'
1    2   14-FEB-2017  11:16:53     AIPS      Got(1)   disk=12  user=****   type=UV   CAL_01.SPLAT0.1
1    5   14-FEB-2017  11:16:53     AIPS      'SOURCE'     'J0302+5331'
1    2   14-FEB-2017  11:16:53     AIPS      Waiting for returned adverbs
1    5   14-FEB-2017  11:16:53     EVAUV     Task EVAUV  (release of 31DEC17) begins
1    3   14-FEB-2017  11:16:53     EVAUV     UVGET: doing no flagging this time
1    5   14-FEB-2017  11:16:53     EVAUV     Copied        8973 visibilities to the work file
1    3   14-FEB-2017  11:16:53     EVAUV     Copied XX file from vol/cno/vers 11    2   1 to 12   24   1
1    4   14-FEB-2017  11:16:53     EVAUV     Updating tables for IF/FREQID/channel selection
1    3   14-FEB-2017  11:16:53     EVAUV     Copied AN file from vol/cno/vers 11    2   1 to 12   24   1
1    3   14-FEB-2017  11:16:53     EVAUV     Copied CD file from vol/cno/vers 11    2   1 to 12   24   1
1    4   14-FEB-2017  11:16:53     EVAUV     Using SMODEL =   0.30528     0.00000     0.00000
1    7   14-FEB-2017  11:16:53     EVAUV     IGNORING MODEL IMAGE J0302+5331
1    2   14-FEB-2017  11:16:53     EVAUV     QINIT: did a GET  of      5120 Kwords, OFF   17559181948887
1    2   14-FEB-2017  11:16:53     EVAUV     EVADFT: Begin DFT component subtraction & division
1    2   14-FEB-2017  11:16:53     EVAUV     EVADFT: Model components of type Point
1    2   14-FEB-2017  11:16:53     EVAUV     Model computation is    30 percent complete
1    2   14-FEB-2017  11:16:53     EVAUV     Model computation is    60 percent complete
1    2   14-FEB-2017  11:16:53     EVAUV     Model computation is   100 percent complete
1    5   14-FEB-2017  11:16:54     EVAUV     method    real part          imaginary part     amplitude
1    5   14-FEB-2017  11:16:54     EVAUV     subtract -0.0003 +-  0.022   0.0004 +-  0.022   0.0274 +-  0.014
1    5   14-FEB-2017  11:16:54     EVAUV     divide-1 -0.0010 +-  0.072   0.0012 +-  0.071   0.0899 +-  0.047
1    5   14-FEB-2017  11:16:54     EVAUV     method   # bad samples  total samples  avg bad amp
1    5   14-FEB-2017  11:16:54     EVAUV     subtract            47         210446       0.1258
1    5   14-FEB-2017  11:16:54     EVAUV     divide-1            47         210446       0.4121
1    2   14-FEB-2017  11:16:54     EVAUV     returns adverbs to AIPS
1    2   14-FEB-2017  11:16:54     EVAUV     GFINIS: number records used      94
1    5   14-FEB-2017  11:16:54     EVAUV     Successful histogram plot file version      1 created
1    2   14-FEB-2017  11:16:54     EVAUV     GFINIS: number records used      94
1    5   14-FEB-2017  11:16:54     EVAUV     Successful histogram plot file version      2 created
1    2   14-FEB-2017  11:16:54     AIPS      Resumes
1    3   14-FEB-2017  11:16:54     EVAUV     Gridded image max=  2.7300E+02 counts; peak contour 2.0 in log10
1    4   14-FEB-2017  11:16:54     EVAUV     Plotted    210421. omitted       25. vis of RE/IM plot 1
1    4   14-FEB-2017  11:16:54     EVAUV     Used image of  121 pixels on a side, smoothed by  1 pixels
1    2   14-FEB-2017  11:16:54     EVAUV     GFINIS: number records used      74
1    5   14-FEB-2017  11:16:54     EVAUV     Successful histogram plot file version      3 created
1    3   14-FEB-2017  11:16:54     EVAUV     Gridded image max=  7.0900E+02 counts; peak contour 2.5 in log10
1    4   14-FEB-2017  11:16:54     EVAUV     Plotted    210443. omitted        3. vis of RE/IM plot 2
1    4   14-FEB-2017  11:16:54     EVAUV     Used image of  121 pixels on a side, smoothed by  1 pixels
1    2   14-FEB-2017  11:16:54     EVAUV     GFINIS: number records used      38
1    5   14-FEB-2017  11:16:54     EVAUV     Successful histogram plot file version      4 created
1    3   14-FEB-2017  11:16:54     EVAUV     Appears to have ended successfully
1    5   14-FEB-2017  11:16:54     EVAUV     euro 31DEC17 TST: Cpu=      0.6  Real=      1  IO=       155
1    3   14-FEB-2017  11:16:54     AIPS      Task EVAUV   has finished
1    5   14-FEB-2017  11:16:54     AIPS      'J0302+5331 HAS 8973 VIS, FLUX (JY/BM) = 0.3053'
1    5   14-FEB-2017  11:16:54     AIPS         -0.0002973     0.0219181     0.0003746     0.0217571
1    5   14-FEB-2017  11:16:54     AIPS          0.0274305     0.0141977     0.0002233     0.1257948
1    5   14-FEB-2017  11:16:54     AIPS         -0.0009738     0.0717969     0.001227      0.0712696
1    5   14-FEB-2017  11:16:54     AIPS          0.0898538     0.0465071     0.0002233     0.4120648
1    2   14-FEB-2017  11:16:54     AIPS      Got(1)   disk=12  user=****   type=UV   CAL_01.SPLAT0.1
1    5   14-FEB-2017  11:16:54     AIPS      'SOURCE'     'J0303+4716'
1    2   14-FEB-2017  11:16:54     AIPS      Waiting for returned adverbs
1    5   14-FEB-2017  11:16:54     EVAUV     Task EVAUV  (release of 31DEC17) begins
1    3   14-FEB-2017  11:16:54     EVAUV     UVGET: doing no flagging this time
1    5   14-FEB-2017  11:16:54     EVAUV     Copied        7020 visibilities to the work file
1    3   14-FEB-2017  11:16:54     EVAUV     Copied XX file from vol/cno/vers 11    3   1 to 12   24   1
```

```
1   4   14-FEB-2017  11:16:54    EVAUV    Updating tables for IF/FREQID/channel selection
1   3   14-FEB-2017  11:16:54    EVAUV    Copied AN file from vol/cno/vers 11    3   1 to 12   24   1
1   3   14-FEB-2017  11:16:54    EVAUV    Copied CD file from vol/cno/vers 11    3   1 to 12   24   1
1   4   14-FEB-2017  11:16:54    EVAUV    Using SMODEL =   2.45805     0.00000     0.00000
1   7   14-FEB-2017  11:16:54    EVAUV    IGNORING MODEL IMAGE J0303+4716
1   2   14-FEB-2017  11:16:54    EVAUV    QINIT: did a GET  of      5120 Kwords, OFF   17532279675351
1   2   14-FEB-2017  11:16:54    EVAUV    EVADFT: Begin DFT component subtraction & division
1   2   14-FEB-2017  11:16:54    EVAUV    EVADFT: Model components of type Point
1   2   14-FEB-2017  11:16:54    EVAUV    Model computation is    40 percent complete
1   2   14-FEB-2017  11:16:54    EVAUV    Model computation is    80 percent complete
1   5   14-FEB-2017  11:16:54    EVAUV    method    real part          imaginary part     amplitude
1   5   14-FEB-2017  11:16:54    EVAUV    subtract -0.0062 +-  0.250  -0.0046 +-  0.053   0.2250 +-  0.121
1   5   14-FEB-2017  11:16:54    EVAUV    divide-1 -0.0025 +-  0.102  -0.0019 +-  0.022   0.0915 +-  0.049
1   5   14-FEB-2017  11:16:54    EVAUV    method   # bad samples  total samples  avg bad amp
1   5   14-FEB-2017  11:16:54    EVAUV    subtract            97         161948       3.3920
1   5   14-FEB-2017  11:16:54    EVAUV    divide-1            97         161948       1.3799
1   2   14-FEB-2017  11:16:54    EVAUV    returns adverbs to AIPS
1   2   14-FEB-2017  11:16:54    EVAUV    GFINIS: number records used      95
1   5   14-FEB-2017  11:16:54    EVAUV    Successful histogram plot file version      1 created
1   2   14-FEB-2017  11:16:54    EVAUV    GFINIS: number records used      94
1   5   14-FEB-2017  11:16:54    EVAUV    Successful histogram plot file version      2 created
1   2   14-FEB-2017  11:16:54    AIPS     Resumes
1   3   14-FEB-2017  11:16:54    EVAUV    Gridded image max=  5.5500E+02 counts; peak contour 2.5 in log10
1   4   14-FEB-2017  11:16:54    EVAUV    Plotted    161853. omitted       95. vis of RE/IM plot 1
1   4   14-FEB-2017  11:16:54    EVAUV    Used image of  107 pixels on a side, smoothed by  1 pixels
1   2   14-FEB-2017  11:16:54    EVAUV    GFINIS: number records used      42
1   5   14-FEB-2017  11:16:54    EVAUV    Successful histogram plot file version      3 created
1   3   14-FEB-2017  11:16:55    EVAUV    Gridded image max=  1.6590E+03 counts; peak contour 3.0 in log10
1   4   14-FEB-2017  11:16:55    EVAUV    Plotted    161864. omitted       84. vis of RE/IM plot 2
1   4   14-FEB-2017  11:16:55    EVAUV    Used image of  107 pixels on a side, smoothed by  1 pixels
1   2   14-FEB-2017  11:16:55    EVAUV    GFINIS: number records used      26
1   5   14-FEB-2017  11:16:55    EVAUV    Successful histogram plot file version      4 created
1   3   14-FEB-2017  11:16:55    EVAUV    Appears to have ended successfully
1   5   14-FEB-2017  11:16:55    EVAUV    euro 31DEC17 TST: Cpu=      0.6  Real=      1  IO=       124
1   3   14-FEB-2017  11:16:55    AIPS     Task EVAUV   has finished
1   5   14-FEB-2017  11:16:55    AIPS     'J0303+4716 HAS 7020 VIS, FLUX (JY/BM) = 2.458'
1   5   14-FEB-2017  11:16:55    AIPS         -0.0062492     0.2495777    -0.0046047     0.0533107
1   5   14-FEB-2017  11:16:55    AIPS          0.2249807     0.1207269     0.000599      3.3919735
1   5   14-FEB-2017  11:16:55    AIPS         -0.0025424     0.1015349    -0.0018733     0.0216882
1   5   14-FEB-2017  11:16:55    AIPS          0.0915282     0.049115      0.000599      1.3799465
1   2   14-FEB-2017  11:16:55    AIPS     Got(1)   disk=12  user=****   type=UV   CAL_01.SPLAT0.1
```