Atacama
Large
Millimeter /
submillimeter
Array

Pipeline flagging report II

Prepared By: Sandra Burkutean with contributions from the Italian ARC

Date : 06.05.2017

Pipeline manual flagging report II

1. The Aim

We were asked to examine the effect of the new visibityOutliers flagging code on the 30 measurement sets whose manual flagging intervention we previously looked at in our Pipeline manual flagging report (30.09.2016).

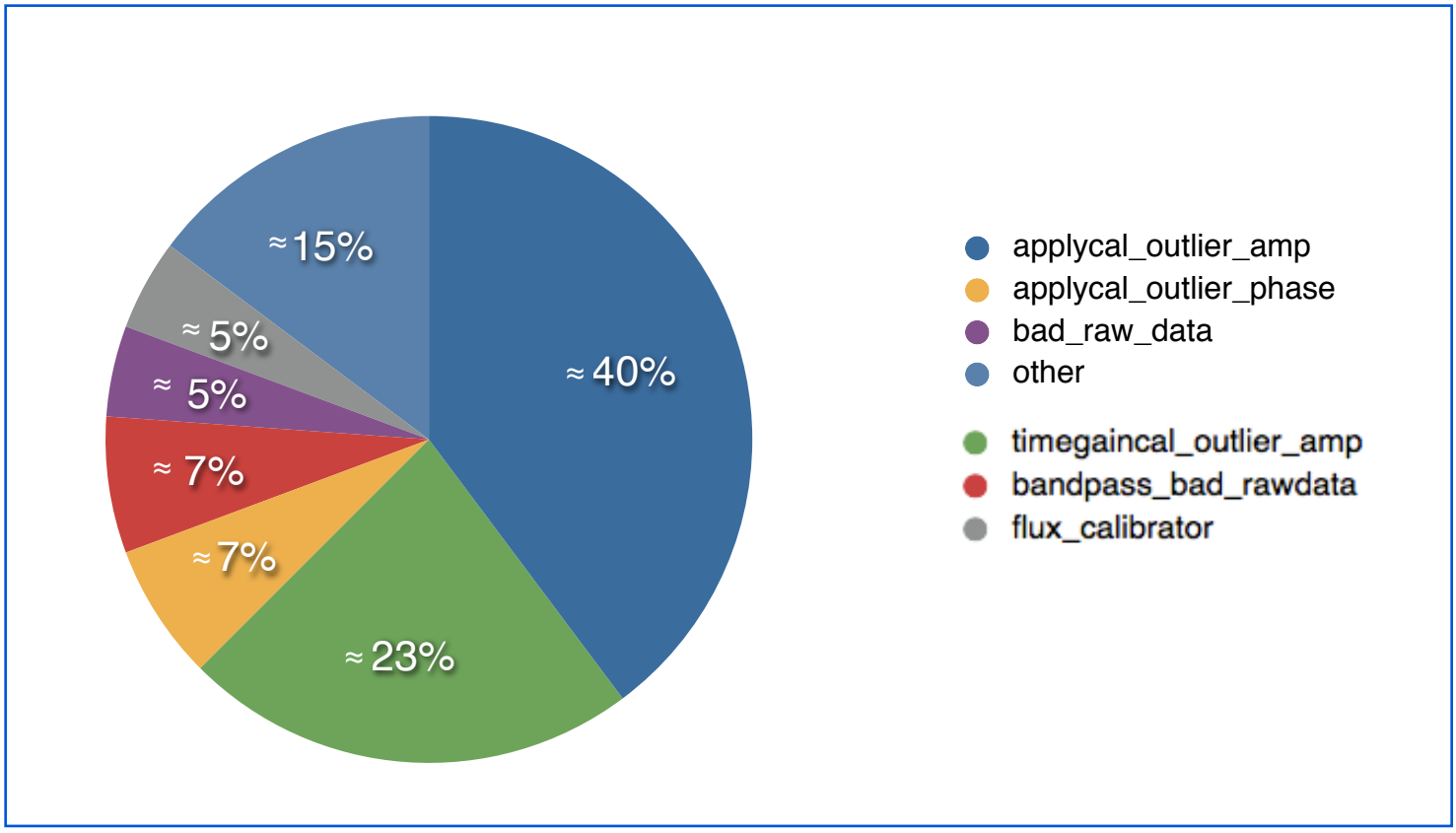
The project workflow focussed on the following steps:

- execute the pipeline reduction script for each measurement set using the pipeline version CASA 4.7.0 r38335(Pipeline-Cycle4-R2_b) (please note the change in pipeline version to the previous report - Pipeline-Cycle3-R4-B) as well as version 1.74 (2017/03/06) of the visibilityOutliers.py file
- note down the reasons for manual flagging as reported in the manual flagging templates provided
- note the flagging reasons in the *flagtemplate.txt file produced by visibilityOutliers.py
- examine the relevant diagnostic plots directly accessible via the pipeline weblog, thus mirroring the first PI experience with the data
- re-run the pipeline with the new flagtemplate file in cases where the different effects of the manual and pipeline (visibilityOutliers.py) flags were not obvious from the weblog alone and examine the data in plotms.

2. Comparison of old and new flagging approaches

In the project focusing on the manual flagging intervention, “Pipeline manual flagging report” (30.09.2016), 30 measurement sets were chosen at random from a wider selection of pipeline-reduced projects (no selection was made in terms of focusing on particular manual flagging applications so as to examine the whole range of possible flagging scenarios). The individual ms files that were examined are listed in Appendix A for further reference.
We noted that timegaincal and applycal were the most frequent stages for flagging related to outliers in amplitude and, less often, phase (Figure 1).

Re-running all 30 projects with the Cycle 4 pipeline as well as the newly developed visiblityOutliers python task in analysisUtils, we find that in 24% of the 30 projects all of the manually flagged data issues are indeed detected (Fig. 1, bottom). We note however that manual flagging focuses on whole-antenna flags whereas the visiblityOutliers.py approach is more baseline-flagging-based. In addition, we found that the new Cycle 4 pipeline manages to catch some outliers that needed previous manual intervention at the msnrmsdeviant-flag_commands.txt stage. Issues related to the gain solution step, bandpass artefacts or Tsys issues were not accounted for by the new task. The 76% of projects without complete manual flag overlap were mostly due to gainsol outliers,bandpass artifacts or Tsys issues not being accounted for or due to baseline as opposed to whole antenna flagging.



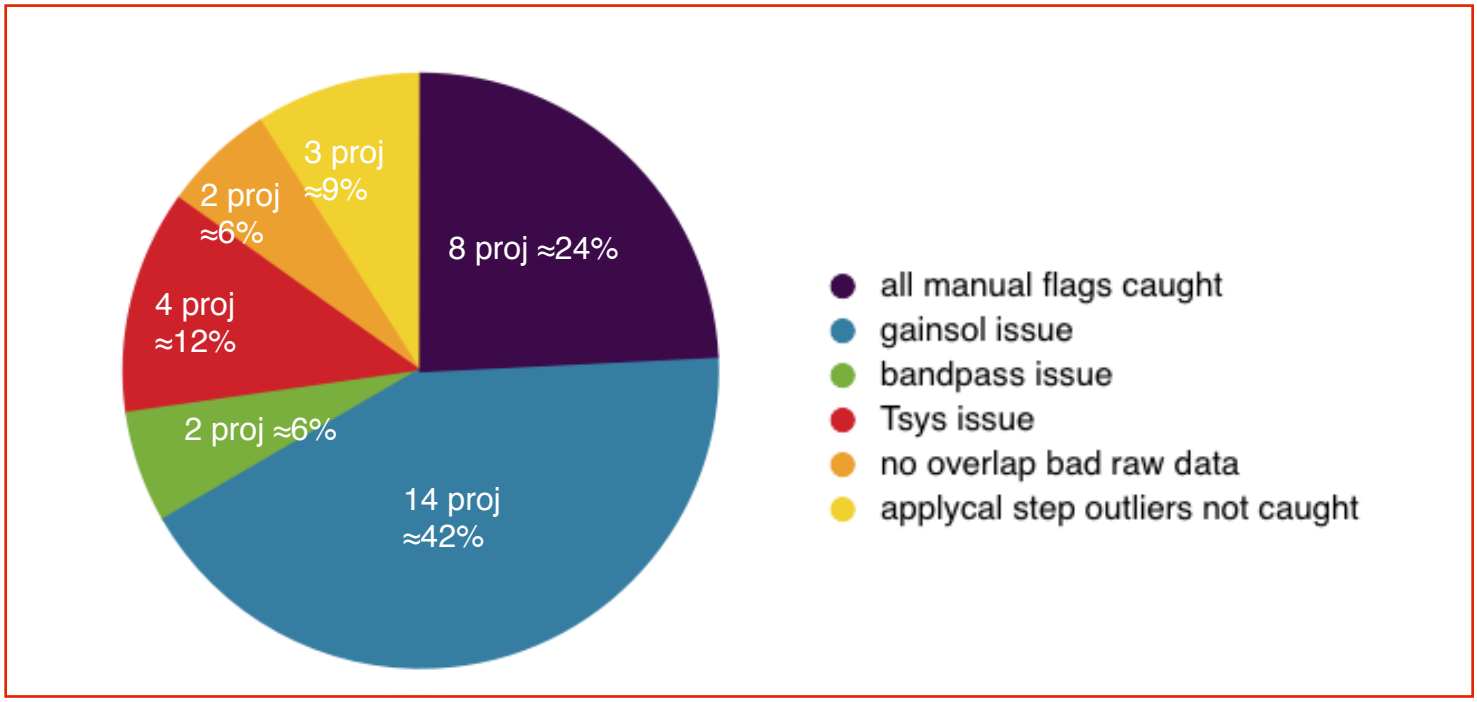


Fig. 1: Comparison of manual flagging and new VisibilityOutliers investigation. *top)* Distribution of manual flagging stages from the 30 measurement sets using the Cycle-3 pipeline (Pipeline report #1). *bottom)* A comparison study between the manual flagging commands and the joint effect of the flagtemplate file produced by visibilityOutliers.py and the Cycle 4 pipeline. 24% of cases have complete overlap with the manual flagging commands (partly also due to the fact that the new pipeline itself caught several outliers before the visibilityOutliers.py task).The remaining 76% had overlap with the manual flags in most cases but missed out on other flagging issues. Gain solution outliers were the most frequent (note this number has increased compared to the top graph as the former lists the number of flagging reasons in the manual flagging files whereas the bottom graph is calculated per flagging occurrence class in the projects). Please note that a few measurement sets entered into several categories. The reason for flagging in the "bad_raw_data" category was not obvious in several cases judging from the information in the pipeline html reports.

Flagging reason distribution of the new flagging task:

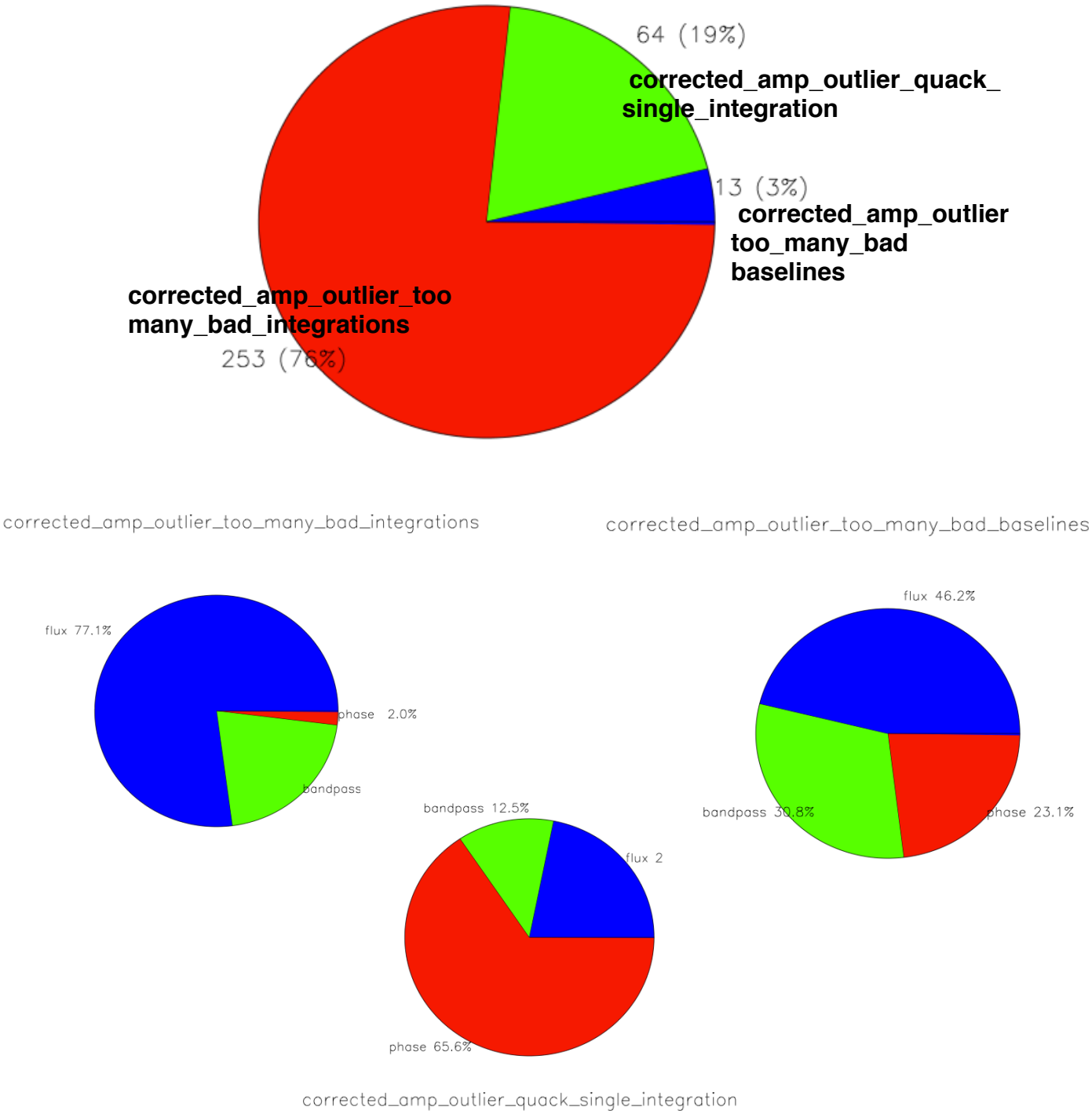


Fig. 2: Reasons for flagging in the visibilityOutliers.py code. *top)* Distribution of flagging reasons for the visibilityOutliers.py code applied to 30 projects. *bottom)* Intent distribution for all three flagging reasons in visibilityOutliers.py in the project runs.

3. A detailed look at residual outliers

Due to the presence of residual outliers not caught by the new visibilityOutliers.py code (as the latter was designed to detect outliers primarily after the applycal step), we give example plots below for further cases where additional flagging could be applied. We stress that the effect of additional flagging, besides the flags from visibilityOutliers.py, is not quantifiable at this stage and was not the purpose of this report.

Bandpass outliers:

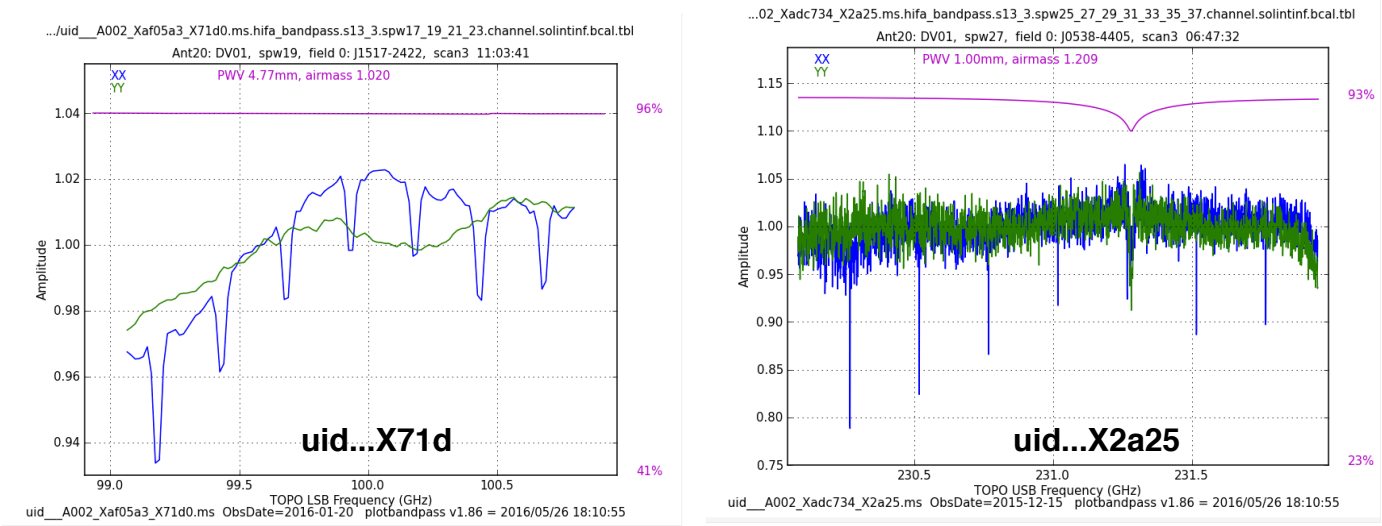


Fig. 3: Bandpass spikes as also reported in the Pipeline manual flagging report (30.09.2016).

Tsys outliers:

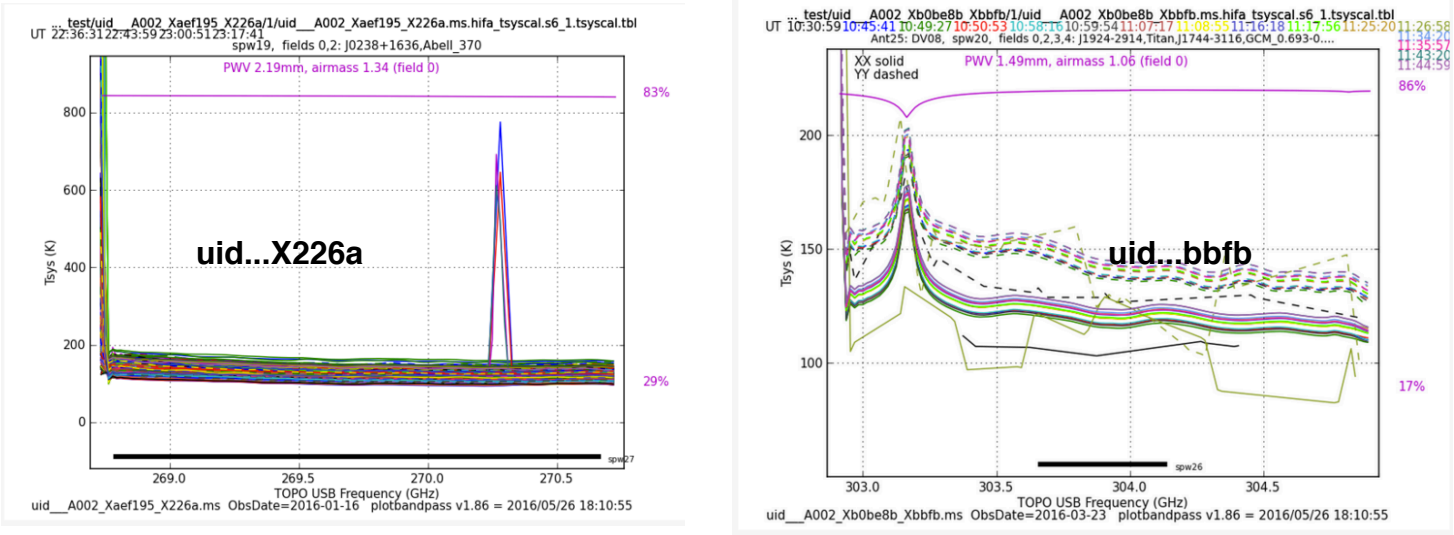


Fig. 4: While positive spikes in Tsys on individual antennas will result in this data to be down-weighted, low Tsys outliers would lead to an opposite effect - the impact of additional flagging heuristics cannot be predicted at this stage.

Outliers after the applycal step (9 % in Fig. 1):

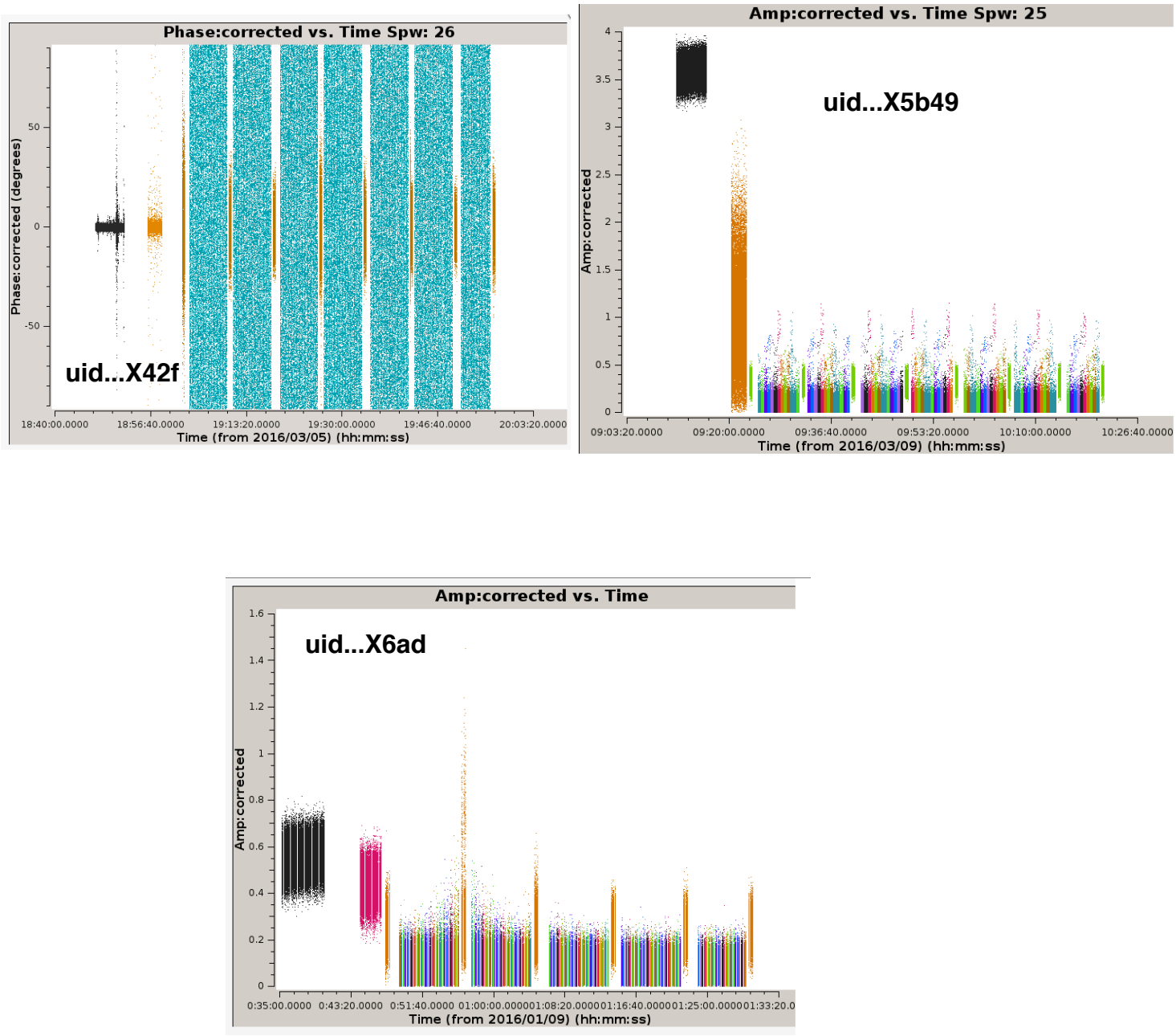


Fig. 5: These outliers are partly due to gainsol outliers propagating to the applycal stage. The effect of more stringent flagging on the imaging stage would need to be examined to reach a quantitative conclusion on the necessity for such additional flagging.

New Flagging task in action:

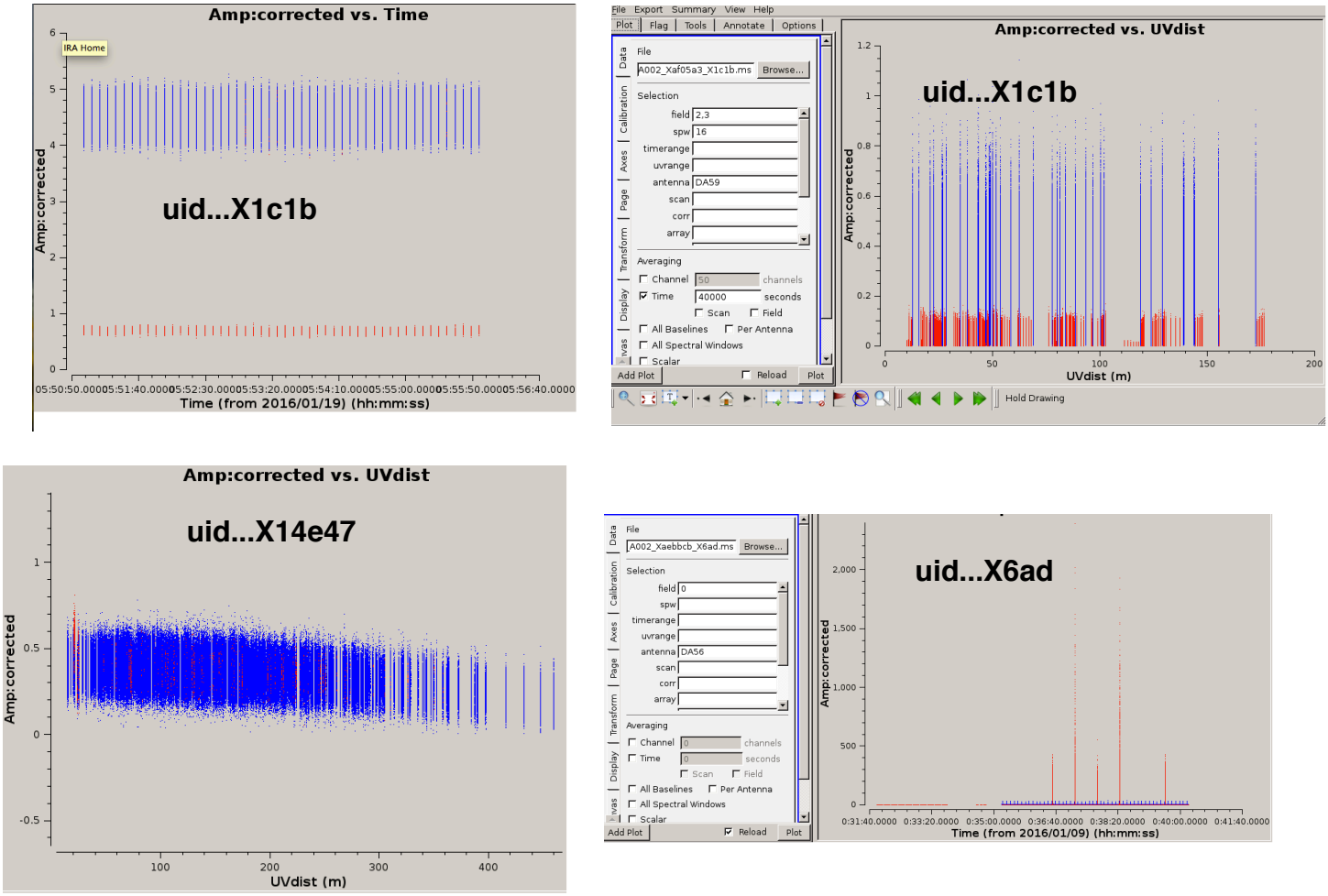
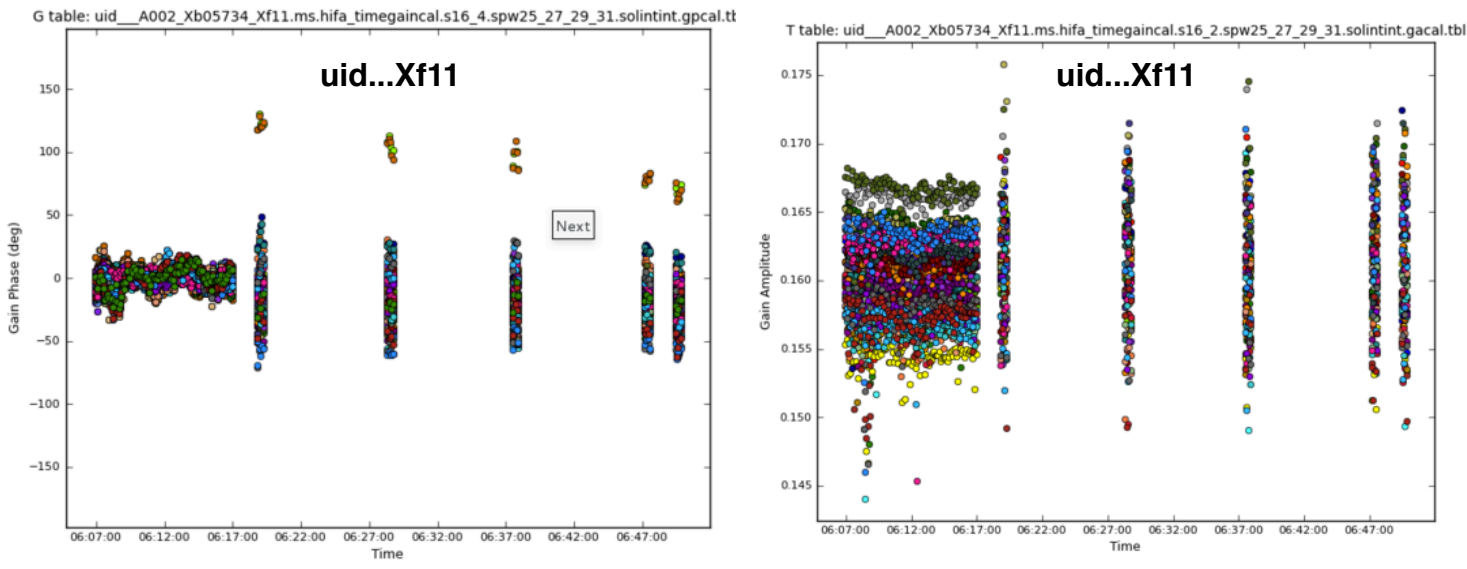


Fig. 6: Example cases for successful flagging by the visibilityOutliers flagtemplate - flagged points are in red.

Gainsol outliers (example for 42% item):



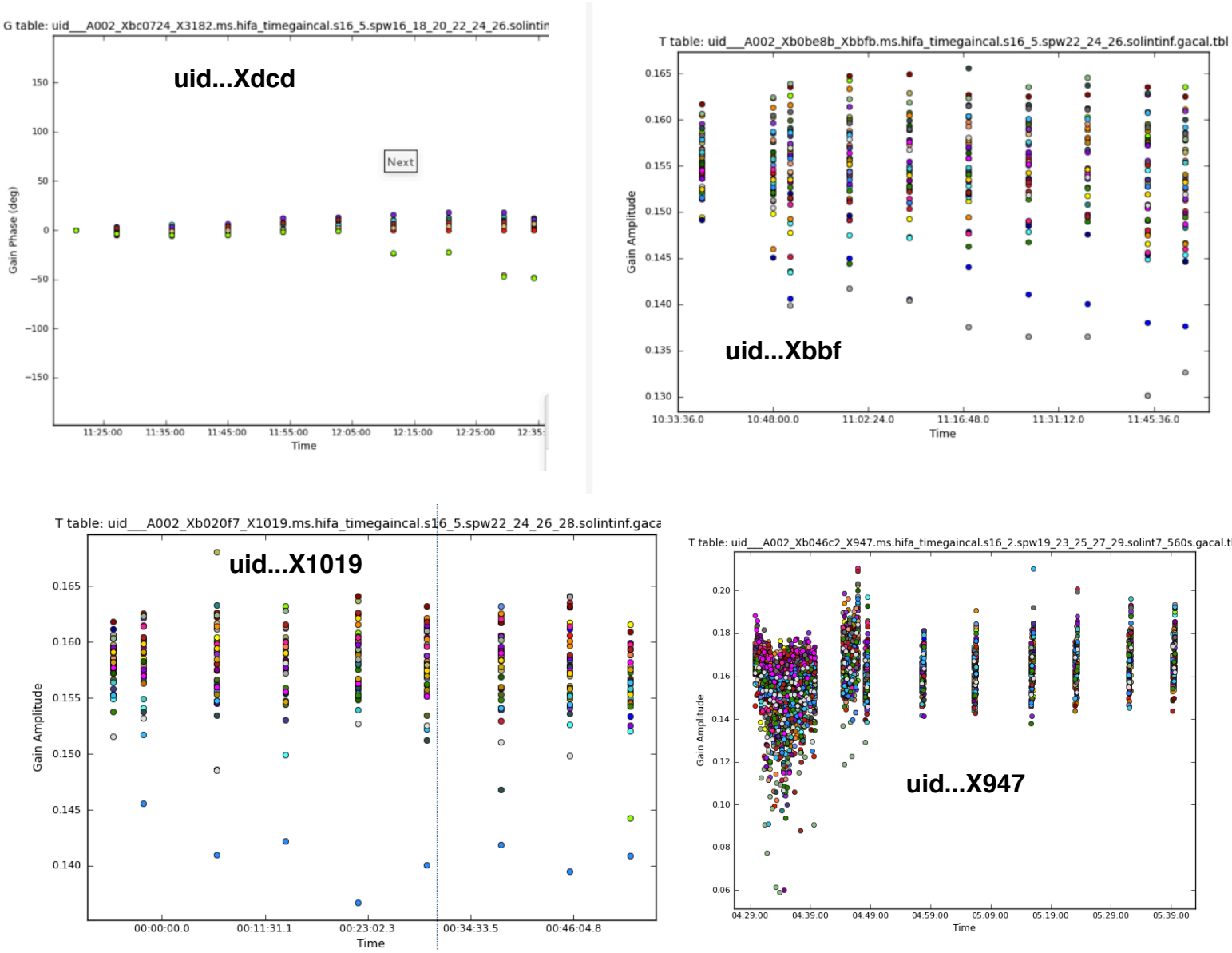


Fig. 7: Outliers in the gain solutions. The effect of more stringent flagging on the imaging stage would need to be examined to reach a quantitative conclusion on the necessity for such flagging.

4. Conclusion

We conclude that for 24 % of the projects, the combined effect of the Cycle-4 pipeline and the new flagging task commands is sufficient. The rest of the projects have partial flagging at the applycal step and/or additional gainsol, bandpass or Tsys outliers. The effect of adding additional flagging algorithms for these cases cannot be predicted from this study and would need further investigation through final image product comparisons. The diagnostic plots from the visibilityOutliers task would benefit from amp vs time graphs as well as a README file specifying and explaining the flagging thresholds.

Appendix A

Here is a list of measurement sets for which the pipeline weblogs and manual flagging reasons were examined.

uid_A002_Xadc734_X850	uid_A002_Xae5b1d_X482a
uid_A002_Xadc734_X2a25	uid_A002_Xaea19c_Xdcd
uid_A002_Xadabcb_X1dcc	uid_A002_Xb09eed_X2f8
uid_A002_Xadabcb_X1565	uid_A002_Xaf05a3_X45ea
uid_A002_Xaebbcb_X40b	uid_A002_Xaef195_X62bf
uid_A002_Xaecf7b_X1bd	uid_A002_Xaf05a3_X1c1b
uid_A002_Xaecf7b_X30da	uid_A002_Xaecf7b_X13f7
uid_A002_Xaebbcb_X6ad	uid_A002_Xaef195_X603c
uid_A002_Xaea19c_X8c9	uid_A002_Xaf05a3_X4171
uid_A002_Xaee04e_X468a	uid_A002_Xb05734_Xf11
uid_A002_Xaef195_X226a	uid_A002_Xb00ce7_X4bba
uid_A002_Xb0be8b_Xbbfb	uid_A002_Xb00ce7_X4e47
uid_A002_Xaf05a3_X71d0	uid_A002_Xb046c2_X5b49
uid_A002_Xb02e35_X42f	uid_A002_Xb046c2_X606
uid_A002_Xb020f7_X1019	uid_A002_Xb046c2_X947
uid_A002_Xadc734_X391c	uid_A002_Xb0ebd1_X978d