Interoperability in Radio Interferometry Software

S. Bourke

21 May 2012

Contents

1 Introduction

Many software packages exist for the purpose of processing radio interferometric data. Prominent examples include AIPS, Casa, Miriad and Difmap. Each package has its own particular strengths and weaknesses, and in many cases it is desirable to use more than one package in the end to end processing of an astronomical dataset. A common such use case is the use of AIPS to perform the initial calibration of a dataset, followed by the use of Difmap for further editing, calibration and imaging. Similarly, the use of Casa's advanced imaging capabilities such as W-Projection (Cornwell et al., 2005), Wide-band Imaging (Rau and Cornwell, 2011), and A-Projection (Bhatnagar et al., 2008) may be desired following calibration of the data using AIPS' robust routines. Traditionally, the software packages have been used independently: data is exported from one software in a commonly understood format (usually FITS (Hanisch et al., 2001)) before being imported in the other software package. This report covers work done to facilitate interoperability of the software package AIPS and Casa. A single scripting environment, and a facility for the transfer of calibration meta-data were developed. The former allows a single pipeline or script to use both packages simultaneously. The latter allows calibrations to be transferred from AIPS to Casa without exporting and importing the often extremely large datasets.

2 Python Interfaces to Interferometry Packages

Python interfaces are available for both AIPS and Casa. The first goal of the interoperability project was to facilitate the a single environment from which a user would have access to AIPS and Casa. In this section an overview of each packages python integration is presented followed by a description of combined environment that was developed to allow usage of both packages simultaneously. Finally a binary ParselTongue/Obit distribution is reported on.

2.1 AIPS / ParselTongue

Natively, AIPS uses POPS as its shell. Python access is provided by ParselTongue (Kettenis et al., 2006), developed under the ALBUS project. ParselTongue accomplishes its interfacing with AIPS via complete external mechanisms. That is, no AIPS code is used or AIPS libraries linked against. ParselTongue directly [1] interacts with AIPS storage files for access to data and task parameter setting. AIPS tasks are run from ParselTongue by invoking the task executables via Python's OS module.

2.2 Casa / IPython

Casa is integrated with Python at a much lower level than AIPS. Casa's functionality is contained in libraries and IPython extended to provide access to these routines. Unlike AIPS, Casa tasks can not be run from the operating system level, they must be linked at the application level.

2.3 Combined AIPS/Casa Python Environment

Due to the tight integration of Casa with Python, it is simpler to bring ParselTongues features to Casa than vise versa. This was accomplished by building Obit and its dependencies with the version same tool-chain version as Casa and using Casa's version of Python to build ParselTongue.

2.3.1 Distribution

The use of ParselTongue from within Casa requires ParselTongue and it's dependency Obit which in turn brings many package dependencies to be build with binary compatibility with Casa. Casa is distributed in an isolated manner including all its library dependencies. It does not

[1] The Obit library is used as a layer on top of the binary file formats

use the local system libraries. This simplifies the distribution of a ParselTongue binary distribution by limiting the number of target platforms to those which Casa provides, namely Linux 32-bit, Linux 64-bit and Mac. Initially, a Linux 32-bit binary distribution of ParselTongue is provide which when extracted to the Casa directory, provides ParselTongue functionality to Casa.

3 Data Storage in Interferometry Packages

With a single execution environment established, the next hurdle is data interoperability. AIPS and Casa use different storage formats. The initial solution is to use FITS as a intermediate format, as both packages support the format. However for the large visibility dataset than are produced by the latest generation of interferometers, this method is not acceptable. An alternative solution is presented in the following section.

4 Calibration Transfer

As described in Section 1, it can be desirable to make use of the extensive calibration routines present in AIPS, followed by the advanced imaging routines of Casa. The closed loop calibration technique known as Self Calibration (Readhead and Wilkinson, 1978) which is frequently used in radio astronomy iteratively uses these two processes in a feedback loop, whereby an initial calibration is performed, the data are imaged, this initially poor image is as an input model to recalibrate the data, and the sequence repeated until convergence is achieved. For such a process is to be performed with AIPS' calibration routines and Casa's imaging routines the visibility data must be passed from AIPS to Casa, and Images back from Casa to AIPS. The latter case of passing images from Casa to AIPS is not a problem. Images can be exported from Casa in FITS format which can be imported in AIPS. The size of these images are typically in

the order of a few megabytes. The transfer of visibility data from AIPS to Casa can, however, represent a major I/O bottleneck, with data sizes from modern interferometers often ranging from hundreds of gigabytes to several terabytes.

To make the above scheme feasible we developed a calibration conversion tool which reads the AIPS internal calibration tables, interprets them, and generates corresponding Casa caltables. Calibration (AIPS Solution, and Calibration tables) and bandpass tables are supported. ParselTongue is used for reading AIPS tables and both pyrap, and casapy are supported for writing Casa caltables. A Casa task has been developed using this functionality, simplifying the process for the end user.

4.1 Internal Data Formats

AIPS and Casa have quite different internal calibration philosophies, however, for the purpose of complex gain, bandpass, and delays, their table storage formats are compatible. In both packages a solution is essentially represented as a complex number for complex gain, and bandpass, and a real number for delay. Rates are not handled by Casa but a workaround will be described in the following section.

4.2 Implementation

The conversion tool is implemented in Python. The stand alone (non-casapy) version is invoked from the command line as follows:

```
Usage: trans_cal <userno> <indisk> <cno> <inver> \
                  <ms> <outcal> [-b|-s] [antfile]
```

where the first four arguments specify the AIPS calibration table, argument five and six the equivalent Measurement Set and CalTable. A `-b` specifies a bandpass table while `-s` specifies a solution table, otherwise

a calibration table is assumed. `antfile` specifies a file containing the mapping of the antennas from AIPS to Casa which isn't necessarily a one to one relationship.

ParselTongue's Wizardry module is used for performance reasons when accessing AIPS tables. Pyrap, the python interface to casacore is used to write the caltable. Unlike AIPS, Casa requires caltables to be regular in time and include entries for all antennas. For this reason the tool reads the entire table before writing the Casa caltable. Dummy entries are entered in the caltable when a corresponding AIPS entry does not exist. These dummy entries are flagged as invalid. Internally the data is stored in a python dictionary containing numpy arrays. Blank serialized tables are stored in the tools module from which to generate the initial tables.

A Casa task was also developed based on this stand alone tool, with the pyrap code been ported to use casapy. The same functionality is then presented to the user via the familiar Casa task mechanism.

```
CASA <4>: default importaipscal
--------> default(importaipscal)

CASA <5>: inp
--------> inp()
#  importaipscal :: Convert AIPS calibration tables to MS cal tables.
vis                 =         ''        #  name of input visibility file
caltable            =         ''        #  Basename of output table(s)
userno              =          0        #  AIPS userno
indisk              =          0        #  AIPS disk number
cno                 =          0        #  AIPS catalog number
inver               =          0        #  AIPS table version
mode                =       'CL'        #  Calibration mode
antfile             =         ''        #  Text file providing AIPS to Casa
                                        #   antenna mapping
```

```
async               =       False       #  If true the taskname must be started
                                        #   using importaipscal(...)
```

The output of the conversion tool is graphically represented in Figure 4.2. A normalized bandpass table is generated and is seen to be in complete agreement with it's AIPS counterpart.

Phase rates are not currently supported in Casa. This can be addressed by generating high resolution calibration (CL) tables in AIPS and interpolating the fringe fitting solutions. The resulting table can then be converted to a Casa gain and delay table.

4.3 End to End Processing of VLBI Data in Casa

Using this tool in conjunction with the EVN archive, end users can process EVN data entirely in Casa, without the need to have AIPS installed on their workstation. This is facilitated by ParselTongue 2.0's AIPSLite features which allow ParselTongue to run without AIPS being present. To accomplish this, ParselTongue downloads the FITLD executable from NRAO's server, loads the EVN pipeline's calibration tables into AIPS format and then converts them to Casa caltables. With amplitude calibration and fringe fitting performed in JIVE by the EVN pipeline, the user can proceed to self calibrate and image their data.

5 Conclusion

A combined Python based environment was implemented, allowing for a single Python script to be written which can access both AIPS and Casa routines. Data interoperability is accomplished by each package having its own copy of the visibility data and the calibration meta-data being transferred by means of a conversion utility. A Casa interface was put on this tool to allow it to be seamlessly used from within the combined environment. The major drawback to this approach is that the visibility

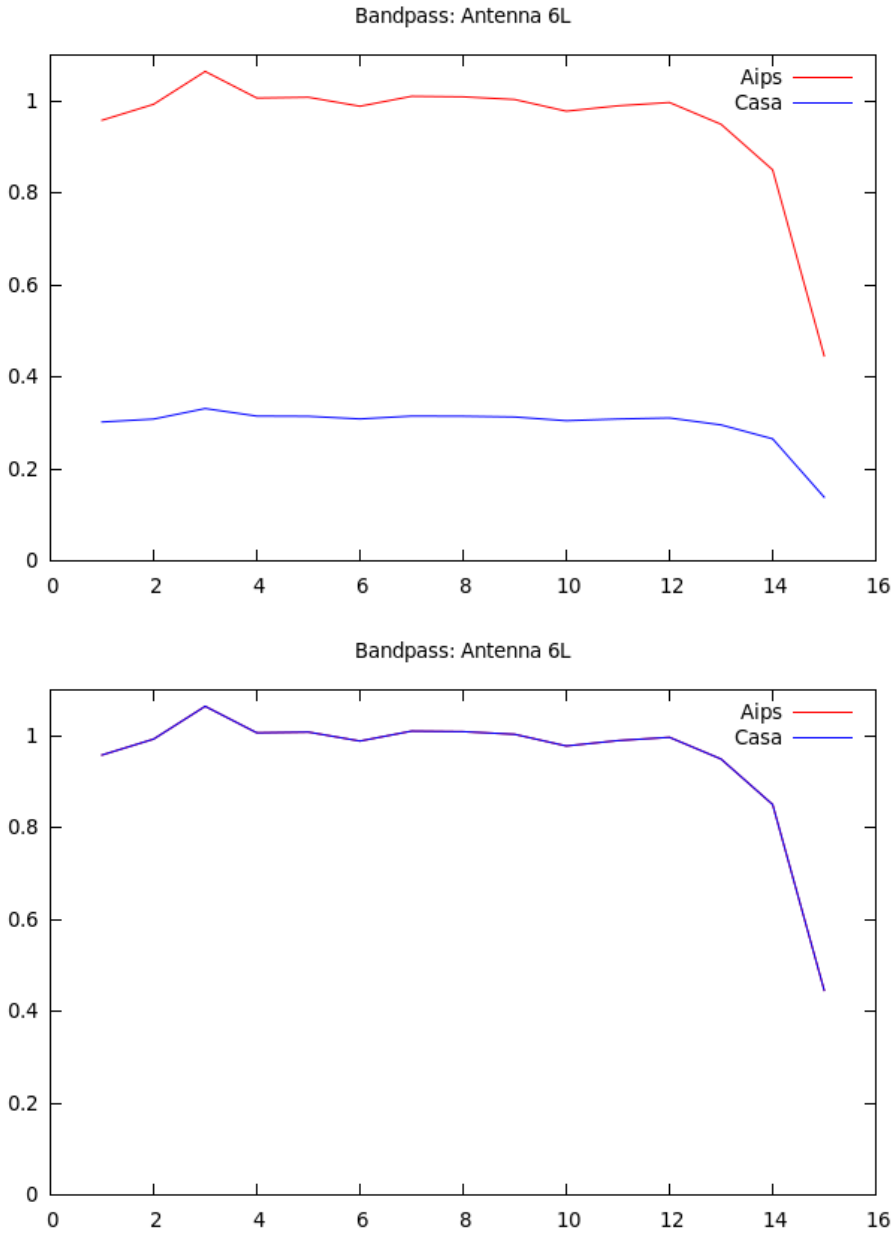

Figure 1: Graphical representation of a bandpass table converted from AIPS to Casa format. In the upper plot a normalized AIPS table is compared with an unnormalized Casa version. In the lower plot, the output of the conversion tool is show, generating an amplitude-normalized Casa table.

data is stored twice. This however is not uncommon in interferometry software and may be seen as acceptable in many cases.

5.1 Perspective on Future Use of Interoperability

Casa is targeted as the future interferometry package. AIPS, however is still in wide use and has kept pace with much of the recent advances in interferometers. It is currently often desirable to process data from the instruments being presently deployed (EVLA, LOFAR) with AIPS for performance reasons, AIPS being significantly faster than Casa for much of the data processing work-flow. Obit, a tool that provides a C interface to AIPS data structures is also developing rapidly and is being used to process EVLA data. It seems possible that interoperability between the packages is to become a fact of life. Currently Measurement Set access in Obit is one of the missing parts of the ecosystem. In future, it might be desirable further insulate the interferometry algorithms from the storage format as Obit has done. This however, often becomes difficult as the data formats are often driven by a particular packages algorithmic requirements. In any case it is clear that currently, the optimal routines for the various stages of the data processing work-flow are dispersed amongst the various software packages. For this reason interoperability is desired by the user if not by the package developers.

References

S. Bhatnagar, T. J. Cornwell, K. Golap, and J. M. Uson. Correcting direction-dependent gains in the deconvolution of radio interferometric images. A&A, 487:419–429, Aug. 2008.

T. J. Cornwell, K. Golap, and S. Bhatnagar. W Projection: A New Algorithm for Wide Field Imaging with Radio Synthesis Arrays. In P. Shopbell, M. Britton, and R. Ebert, editors, *Astronomical Data Analysis Soft-*

ware and Systems XIV, volume 347 of *Astronomical Society of the Pacific Conference Series*, page 86, Dec. 2005.

R. J. Hanisch, A. Farris, E. W. Greisen, W. D. Pence, B. M. Schlesinger, P. J. Teuben, R. W. Thompson, and A. Warnock, III. Definition of the Flexible Image Transport System (FITS). A&A, 376:359–380, Sept. 2001.

M. Kettenis, H. J. van Langevelde, C. Reynolds, and B. Cotton. ParselTongue: AIPS Talking Python. In C. Gabriel, C. Arviset, D. Ponz, and S. Enrique, editors, *Astronomical Data Analysis Software and Systems XV*, volume 351 of *Astronomical Society of the Pacific Conference Series*, page 497, July 2006.

U. Rau and T. J. Cornwell. A multi-scale multi-frequency deconvolution algorithm for synthesis imaging in radio interferometry. A&A, 532: A71, Aug. 2011.

A. C. S. Readhead and P. N. Wilkinson. The mapping of compact radio sources from VLBI data. ApJ, 223:25–36, July 1978.