Python ARM Radar Toolkit (Py-ART) Roadmap

Scott Collis, Cory Weber, Jonathan Helmus and Zachary Sherman

Introduction and Aims

Everyone one who uses weather radar to do science uses, in one form or another, radar software. Software is a science enabling piece of infrastructure and good software minimizes frustration and allows the domain expert to get to understanding the phenomena being studied without needing to be an expert on numerics, data informatics and software engineering.

There are several platforms for interacting with radar data, the open source variants are well documented in (Heistermann et al 2014). The Python ARM Radar Toolkit (Py-ART, Helmus and Collis, 2016) is one of these.

Py-ART grew out of a collection of radar algorithms generated in support of the new radar capability in the ARM program (Mather and Voyles, 2012). One of the first contributions was a Linear Programming (LP) technique for separating propagation polarimetric phase shift from other local impacts (Giangrande el at, 2013). As the collection of both algorithms and radars grew it became clear that the problem would become intractable unless a carefully designed architecture was designed that allowed application chains to be developer via a common data model approach.

Shortly after, in early 2012 development on Py-ART began in earnest with the support of the ARM program. In September of 2012 Py-ART was uploaded to the social coding platform GitHub at https://github.com/ARM-DOE/pyart. Py-ART was unofficially bumped to version 1.0.0-Dev in May of 2013 and publicly released. The first "stable" release was 1.2.0 in February of 2015 and the most recent release was 1.7.0 in September of 2016. Release notes can be found here: https://github.com/ARM-DOE/pyart/releases.

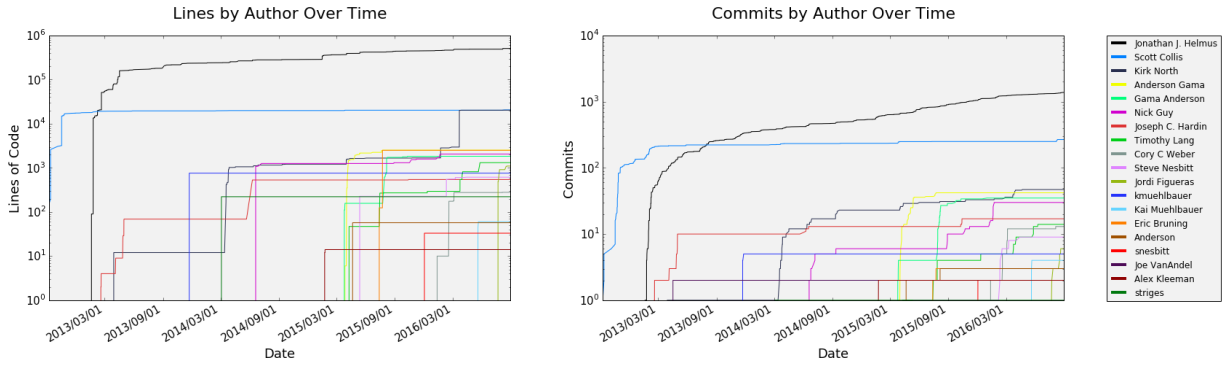

Figure 1: Lines of code by Contributor as a function of time. Note, some lines are automatically generated by Cython to C complilation.

Figure 1 shows the growth of the project as a function of time broken down by contributor. Note that some of the code is C which has been generated by Cython which inflates the number some what.

Py-ART has benefited from code from 15 individual contributors. This has been enabled by careful implementation of unit tests and continuous integration. Every time a pull request is submitted against the Py-ART codebase a set of tests run and a report is generated so the developers know if a contribution causes any unit tests to fail.

Py-ART receives vital support for accepting pull requests, bug fixing, documentation, outreach and education through the ARM program which is part of the Climate and Environmental Sciences Division of the Office of Science in the Department of Energy. Due to this, and to ensure the toolkit has maximal impact a roadmap to chart development priorities for the next five years is needed and is the subject of this document. The roadmap document is broken down into:

1. This introduction

2. The results of the Py-ART roadmap survey
3. Proposed governance for accepting pull requests
4. Overarching goals for the next five years
5. Specific features that will be a priority for development

We also include a list of papers that have been accepted or are in process that have made use of Py-ART as a reference at the end of this document.

The Py-ART Roadmap Survey

In order to produce a development roadmap we first needed to get the views of users and stakeholders as to what should be in the toolkit. To this end we designed the Py-ART Roadmap survey. The survey was hosted on SurveyMonkey and we got some much appreciated assistance from the ARM outreach office in editing some questions for clarity. The Survey asked users to self identify as either a Py-ART user or not and then asked if they would identify as:

1. A person who mainly works with observational data
2. A person who uses a mix of modelling and observational data
3. A person who mainly works with model data

Unfortunately we did not get a statistically significant enough sample to discriminate between this groups so for this document *all user groups will be combined* effectively giving two groups: those that do and do not use Py-ART.

The survey had 35 respondents which were solicited by the ARM and Py-ART mailing lists, Facebook and Twitter. Of those 11 had never used Py-ART and 24 had.

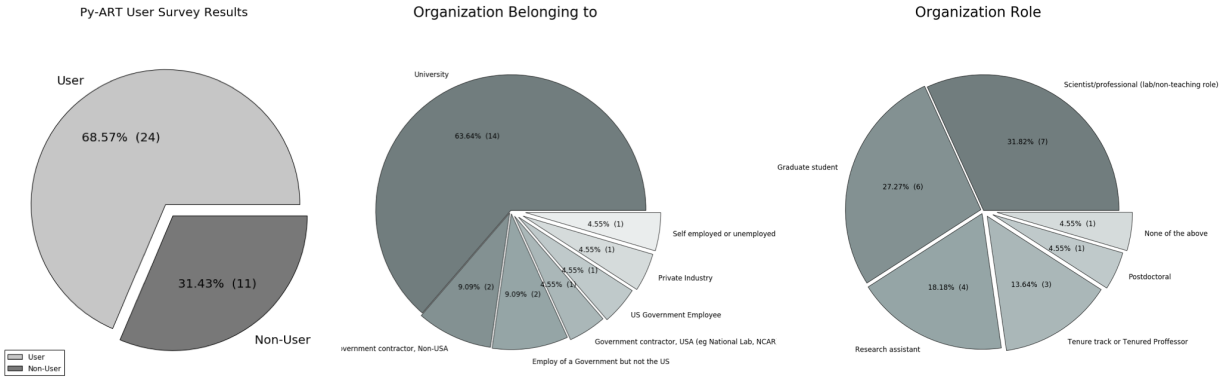

Figure N: Pie charts showing the split between Py-ART user and non-user respondants and aggregated (user/non-user) self identification of organization and role within that organization.

Respondents were presented with a list of possible additions to Py-ART's suite of algorithms consisting of:

1) **Ingest of WRF data to the Py-ART grid model.** The ability to ingest WRF out NetCDF files into the Py-ART Grid data model.

2) **Cell/Object Tracking.** The implimentation of TITAN's cell tracking (Dixon and Wiener, 1993) or similar to create cell tracks.

3) **Multi-Doppler wind retrievals.** Variational or other retrievals of meridional, zonal and vertical wind velocities from Doppler velocity measurements.

4) **More bulk statistics of grid or radar contents (CFAD, echo top heights etc..).** Functions that reduce radar volumes and grids down to descriptive parameters that could be visualized as a time series.

5) **Easier "one step" rainfall retrievals.** Making it easier to simply go from a radar volume to a rainfall map.

6) **More output formats.** More formats to write to.

7) **More input formats.** More ingests.

8) **Quasi-Vertical Profile reconstruction from a list of radars.** A specific case of item 4 along the lines of (Ryzhkov et al, 2016).

9) **More data quality code (eg clutter rejection, biological masks..).** Code to create gatefilters to remove non-meteorological echoes.

10) **Add the option of Cartopy map backend to the existing basemap in RadarMapDisplay.** The ability to use the UK Metoffice developed Cartopy backend for map based displays. Cartopy is newer than the existing basemap backend but is likely to have a longer shelf life due to basemap not being supported beyond 2020.

11) **Ability to handle Radar Spectra and perform retrievals on.** Extension of the Py-ART data model to handle each gate having a spectra consisting of power as a function of velocity or phase. This will allow for an extension into spectra based retrievals such as clutter removal by interpolating over the "zero peak".

12) **More high level retrievals from the literature (Eg DSD, Particle ID..).** Systematic inclusion of various retrievals dealing with particle size retrieval and rain/snow/hail/ice retrievals.

13) **Velocity Azimuth Display wind retrievals.** Ability to retrieve flow vectors as a function of height. Could include advanced techniques such as DVAD (Lee et al, 2014).

Respondents were only presented with the bolded text, the extra information would have been excessive but it could be assumed that familiarity may have played some role in voting.

METHODOLOGY OF THE RANKING:

A selector drop-down ranks each feature between one and number of options. A count of the responses are multiplied against their ranked rank and summed. An example score would be three responses for rank 1 would be three points 4 responses for rank 3 means 16 points added together 19 points. That sum is divided by the total number of responses to that feature giving the feature a weighted ranked score, having the lowest score means that feature is the most important to the users. The example would yield a rank score of 2.7

Non Py-ART Users

Those who identified as non-users of Py-ART were asked "What feature would make you more likely to use Py-ART". In advertising the survey we made a particular effort to get respondents who do not use Py-ART so we did not suffer from an "echo-chamber" effect. Figure N shows the results of this question. The most popular weighted rank for new feature from non-users was "More high level retrievals". Unsurprisingly the item relating to the mapping back-end Cartopy was the least popular since a fair assumption is many of the non-users are also non-Python users and would not even know what Cartopy is. There is no real sudden decrease anywhere along the rankings.

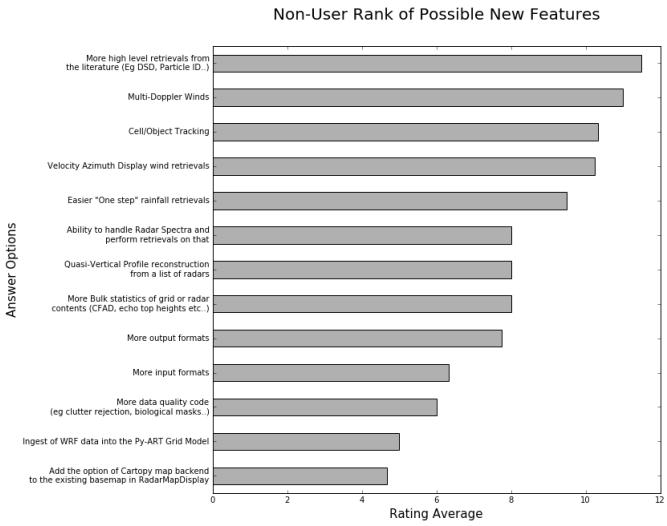

Figure N: Average ratings for the question "How likely would these added features be to get you to use Py-ART?" aimed at non-users

The survey also asked about barriers that non-users faced to using Py-ART. Surprisingly the number one barrier was "Difficulty to install" followed by "Most of my analysis is done by others in our group." The least popular barrier was "I am not a python user" which is pleasing as it is indicative of a large uptake of Python in the community.

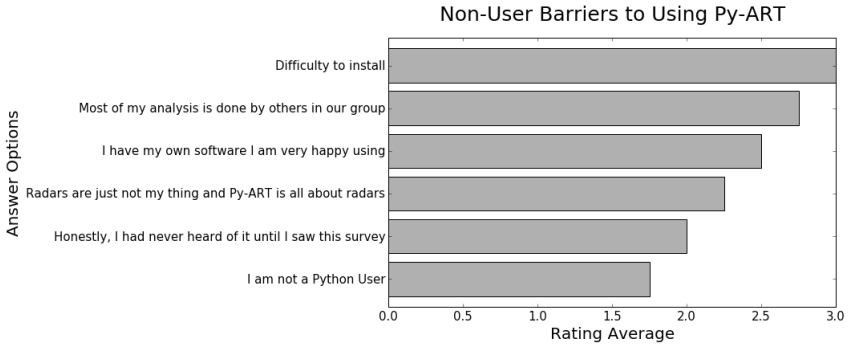

Figure N: Average ratings for the question "What is preventing you using Py-ART?"

Py-ART Users

With Py-ART users, in addition to asking what feature they desired the survey also asked them to identify their favourite feature. Figure N shows the weighted rankings for the responses. Visualization and plotting was the most popular feature closely followed by an appreciation for the wide variety of formats that Py-ART can read. While unsurprising this is in-line with the development priorities of Py-ART to-date as the team sees the two biggest barriers to new users of radar data being the reading of exotically formatted files and working out what those files contain. Rankings decreased gradually with a notable break when it came to "Knowing VAPS will work with ADI/ARM systems". Even though this is one of Py-ART's primary aims (to enable PI developed data to integrate easily with ARM systems) this is not surprising. If anything the development team is a victim of their own success in marketing Py-ART to the wider community. It does show, however, we have some work to do in helping DoE funded PIs in using the toolkit and advocating that funded retrievals be implemented in Py-ART.

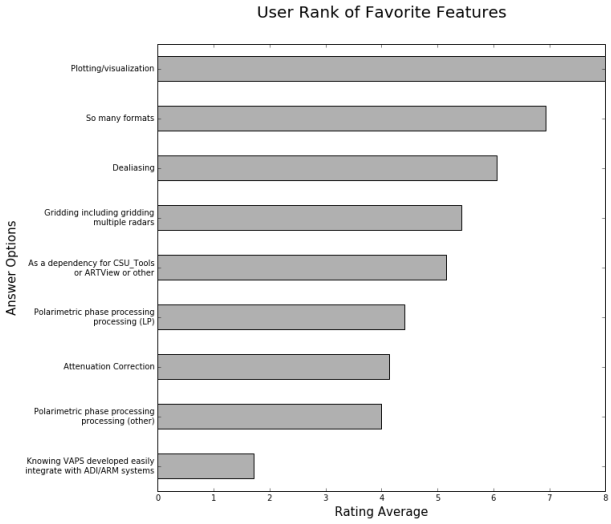

Figure N:

Figure N shows the weighted ranks for desired new features for existing Py-ART users. Figure N shows several key differences to figure N. Multi-Doppler retrievals is now the most popular feature very closely followed by Cell Tracking. And, notably, more literature based techniques is the lowest desire by existing users. Perhaps because many of them, using Py-ART's easy to use data model, have implemented many of these techniques at their home institutions.

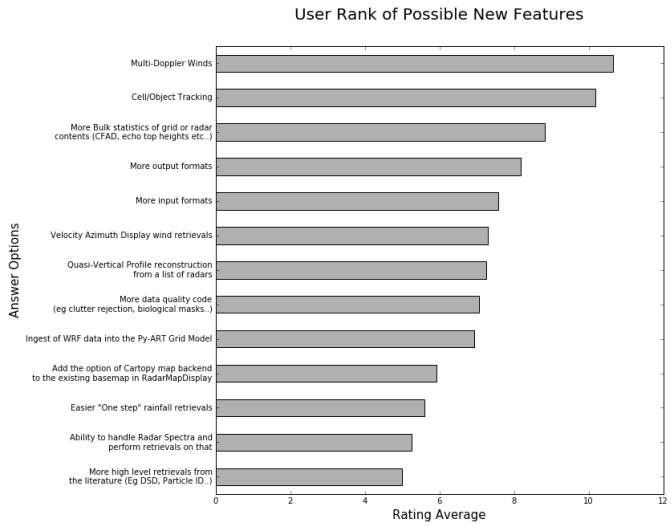

Figure N:

As well as having users pick from feature lists designed by the survey developers free-form answers were solicited with the questions "what would be Py-ART's Killer app". Users were allowed to enter three items each so they will not all be documented here and are available in Appendix 1. The key take aways are:

- Multi-Doppler retrievals are in high demand from the community.
- A functionality that allows cross-sections through a radar volume between two points is desired.
- Further desires for better dealiasing.

We took the opportunity to ask users about contributing. There were 18 responses to the question "Have you ever contributed to Py-ART?" Of the 18, 22.2%(4) said Yes via pull request through Github, 5.6%(1) said yes, by intellectual property implemented by someone else, 44.4%(8) said no, but they wanted to and 27%(5) said no and they were not interested in doing so.

Finally we asked those who have not contributed what the barrier was to contributing.

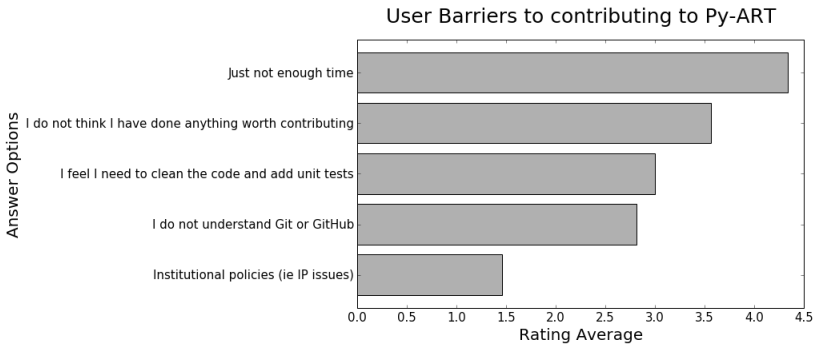

Figure N: Average ratings for the question "What is preventing you contributing to Py-ART?"

Figure N shows the average ranking with the most common barrier being "Just not enough time". This is not surprising as many researchers would not be judged by open source contributions and would not value such in advancing their careers. The second most common is "I do not think I have done anything worth contributing". This is more than likely a misunderstanding as even things as simple as correcting typographical errors in the documentation is a welcome contribution and small changes are much easier to accept than more substantial changes. The lowest ranking issue is to do with intellectual property issues. This is pleasing as it means there are few institutional roadblocks to our users contributing.

Proposed Governance Structure

The motivation of this roadmap is to ensure that the effort funded by the ARM program is responsive to the needs of the stakeholders of the program. A large task of the lead developer has been in assisting contributors in modifying pull requests (contributions) so that they can be accepted into Py-ART. While it is important to have a consistent standard across the codebase many of the tasks associated with accepting pull requests can be delegated to others in the community. Currently there are two ad-hoc defined roles in the Py-ART project: Science Lead and Lead Developer. This roadmap proposes the introduction of a third role: Associate Developer. The roles will be:

Science Lead: Provides high level leadership for the project, organizes outreach and education and coordinates contributor and stakeholder input to form a long term vision for the project.

Lead Developer: Responsible for overall architecture of the project. Final arbiter in what pull requests to accept. Develops the required style guidelines and coordinates the associate developers. Coordinates contributions from associated developers to a Contributors Guide (and contributes as well).

Associate Developers: Responsible, as time allows, for doing an initial check of pull requests for suitability and adherence to the Contributors Guide. Contributes to the Contributors Guide.

It will be expected that the associate developers will be recognized as key members of the project and acknowledged accordingly in future publications and presentations.

Overarching Goals for Next Five Years

Freeform discussion of where we want to be

Priority Features

In priority order the features we want added either by ARM or features that if they are in a PR we will be very happy to help with this PR

References

(Heistermann et al, 2104) Heistermann, M., Collis, S., Dixon, M.J., Giangrande, S., Helmus, J.J., Kelley, B., Koistinen, J., Michelson, D.B., Peura, M., Pfaff, T., Wolff, D.B., 2014. The Emergence of Open Source Software for the Weather Radar Community. Bull. Amer. Meteor. Soc. doi:10.1175/BAMS-D-13-00240.1

(Helmus and Collis, 2016) Helmus, J.J. & Collis, S.M., (2016). The Python ARM Radar Toolkit (Py-ART), a Library for Working with Weather Radar Data in the Python Programming Language. Journal of Open Research Software. 4(1), p.e25. DOI: http://doi.org/10.5334/jors.119

(Mather and Voyles, 2012) Mather, J.H., Voyles, J.W., 2012. The Arm Climate Research Facility: A Review of Structure and Capabilities. Bull. Amer. Meteor. Soc. 94, 377–392. doi:10.1175/BAMS-D-11-00218.1

(Giangrande et al, 2013) Giangrande, S.E., McGraw, R., Lei, L., 2013. An Application of Linear Programming to Polarimetric Radar Differential Phase Processing. Journal of Atmospheric and Oceanic Technology 30, 1716–1729. doi:10.1175/JTECH-D-12-00147.1

(Dixon and Wiener, 1993) Dixon, M., Wiener, G., 1993. TITAN: Thunderstorm Identification, Tracking, Analysis, and Nowcasting—A Radar-based Methodology. Journal of Atmospheric and Oceanic Technology 10, 785–797. doi:10.1175/1520-0426(1993)010<0785:TTITAA>2.0.CO;2

(Ryzhkov et al, 2016) Alexander Ryzhkov, Pengfei Zhang, Heather Reeves, Matthew Kumjian, Timo Tschallener, Silke Trömel, and Clemens Simmer, 2016: Quasi-Vertical Profiles—A New Way to Look at Polarimetric Radar Data. J. Atmos. Oceanic Technol., 33, 551–562, doi: 10.1175/JTECH-D-15-0020.1.

(Lee et al, 2014) Wen-Chau Lee, Xiaowen Tang, and Ben J.-D. Jou, 2014: Distance Velocity–Azimuth Display (DVAD)—New Interpretation and Analysis of Doppler Velocity. Mon. Wea. Rev., 142, 573–589, doi: 10.1175/MWR-D-13-00196.1.

Appendix 1: Free form responses to "Killer App"

These comments have no order to them so they are listed below for reference:

Feature 1 (11 responses):

- Easier installation
- Dual-Doppler Wind Calculations
- More advanced feature with Cross-section cut, based on any two single points, similar to iris
- Dual-Doppler Winds
- Treat variable like this variable
- cross sections between any two points
- RadarCollection
- Advection correction
- More precise data model - e.g in Nexrad Level 3 the width of azimuth gates are not always uniform and in the data format the rays are described with "azimuth of the beginning of the ray" and width of the ray. See relevant ICDs on Level 3.
- Multi-Doppler wind retrievals
- Additional weighting function options when gridding radar data, besides the Barnes and Cressman schemes

Feature 2: (6 responses)

- Dealiasing X-Band Vertical Profiling Radar
- More advanced algorithm, like ZDR column detection or NCAR PID algorithms
- Easier Geotiff compatibility
- Carry along a map image/background to help speed up multiple plotting instances of same radar
- Improved dealiasing algorithms
- Hydro ID

Feature 3: (3 responses)

- Collaboration with SingleDop
- Improved dealiasing
- Improvements to ARTview to make it replace solo3