

Python ARM Radar Toolkit (Py-ART) Roadmap 2025-2030

**Zachary Sherman, Max Grover, Scott Collis, Robert Jackson,
Joseph O'Brien and Bhupendra Raut**

Executive Summary

The five-year development plan for the Python ARM Radar Toolkit (Py-ART) is designed to maximize its scientific value for ARM and the wider research community. This roadmap identifies five priority tasks. 1. to improve documentation and examples, 2. to improve citations for algorithms from the literature, 3. a thorough assessment of existing retrieval code to ensure it authentically represents the intent of the algorithm (in a paper or elsewhere), 4. to establish a formal, documented procedure to detect and mitigate risks associated with AI-generated or intentionally harmful code and files and 5. to continue coordinated integration with the multi-agency xradar project to maintain interoperability, shared standards, and a consistent software ecosystem.

1 Introduction and Aims

Radar software is key for interacting with weather radar data and for analyzing, visualizing, and interpreting it. Several open-source variants are documented in Heistermann et al. (2014). After its release in 2013, the Python Atmospheric Radiation Measurement Radar Toolkit (Py-ART) steadily expanded its collection of algorithms and radar interfaces. However, while well-intentioned, third party contributions would distract from other higher priority ARM funded efforts by the Py-ART development team. Due to this, and to ensure the toolkit has maximal impact, roadmaps have been developed. These 5 year roadmaps, the first being 2015-2020 and the second 2020-2025 were able to prioritize development and needs of ARM and the community as a whole. In this report we outline the next 5 years of development for Py-ART.

1.1 The Python ARM Radar Toolkit

In September of 2012, Py-ART was released to the social coding platform GitHub at <https://github.com/ARM-DOE/pyart>. It has a wide collection of radar algorithms generated to support the new radar capability in the ARM program (Mather and Voyles, 2012) and it has evolved to have tools such as dealiasing, kdp processing, polar to Cartesian gridding, Xarray

support and more, due to contributions from within ARM and the open source community. It also is capable of reading cfradial, ODIM, RSL, sigmet, radar spectra and a variety of other radar formats. Recently Py-ART had its v2.0 release with xradar support for processing radar data as xarray datasets. (see release notes <https://github.com/ARM-DOE/pyart/releases>).

When the original roadmap was written, Py-ART had 22 contributors which have grown to 58 in 2025. Appendix 2 shows the top 10 contributors with their commits as of December 2025. We encourage the contributions to Py-ART by implementing unit tests and continuous integration. A set of tests are run for every pull request (PR) and a report is generated to suggest if a contribution causes any unit tests to fail. After review, the lead developer and the associate developers guide the code contributor, to make the PR acceptable for Py-ART standards. When the code is acceptable, it is merged into Py-ART.

1.2 Value of Py-ART to ARM

Py-ART has grown not only in code, but also in use from users around the globe. Recently Py-ART surpassed 1 million downloads on conda-forge and the [Py-ART metapaper](#) has over 500 citations, many of them using ARM data. ARM is also referenced not only within Py-ART's code, but also in its documentation and with many downloads, ARM is shown in the mentioned documentation and blog posts to new users most likely on a daily basis, which showcases ARM. With more users, we tend to get an increase in pull requests from users for new additions as well as questions or bug catches. Because of this Py-ART requires financial support for funding developers to review code and implement automated systems like continuous integration and document generation. Py-ART receives vital support for accepting pull requests, bug fixing, documentation, outreach and education through the ARM program which is part of the Environmental Sciences Division of the Office of Science in the Department of Energy. The demonstrable value of Py-ART to ARM is that any code that is contributed to Py-ART can be easily be used by ARM in engineering and operations. Since Py-ART carefully conserves radar metadata including data pertinent to ARM new code can be inserted into Value Added Product (VAP) implementations preserving data pertinent to discovery and dissemination. Py-ART is used in all VAPs applied to the Precipitation Scanning Radars and could easily be applied to all ARM radars, which can be seen in the Corrected Moments in Antenna Coordinates (CMAC) VAP. In addition Py-ART gives PIs of DoE funded programs and activities a convenient way of answering questions pertaining to the openness of their science and their contributions to ARM and DoE.

1.3 Targeted Reviews

This document currently contains responses from surveys provided to the open radar community and stakeholders. Similar to previous roadmaps, after the first draft of this roadmap,

the roadmap will be circulated through key stakeholders and science users, as well as the chair of the ASR Radar Science group. And the document will be edited to represent the reviews.

1.4 Success of Past Roadmaps

Previous roadmaps consist of many key inputs from users, non users, academia, ARM and more. From these roadmaps we were able to learn that support for working with radar data in an Xarray dataset was desired, which was added in 2025, that radar colormaps (Sherman et al, 2024) should be in a unified space for the community to use, and more. We do change priorities based on survey results. We as developers might have ideas for new features, but need to confirm with the community on what is currently a necessity. The feedback from the surveys and community is valuable in shaping the roadmap. It is now time for the next 5 year roadmap.

2 The Py-ART Roadmap Survey and Reviews

2.1 The Survey

Similar to the first two roadmaps, a survey was needed to get the views of the users and stakeholders for what direction the toolkit should take to meet the needs of the users and stakeholders. Overall the survey has been toned down after feedback from ARM's communications team, as the previous surveys might have been too long. The survey was hosted by the communications team from PNNL, was shared with the BNF ARM summer school participants and was shared with the open radar community.

The survey had 17 respondents, with most of the respondents from University or other academic settings, which was similar to the last roadmap (**Figure 1**). Py-ART and its use is taught at many tutorials such as the open radar short course as well as the ARM summer school. University is not surprising as a top response as many university students attend these events. This question also gives the developers an idea of where Py-ART is being used and where we can also showcase Py-ART more to help individuals with their workflows.

What type of organization do you belong to?

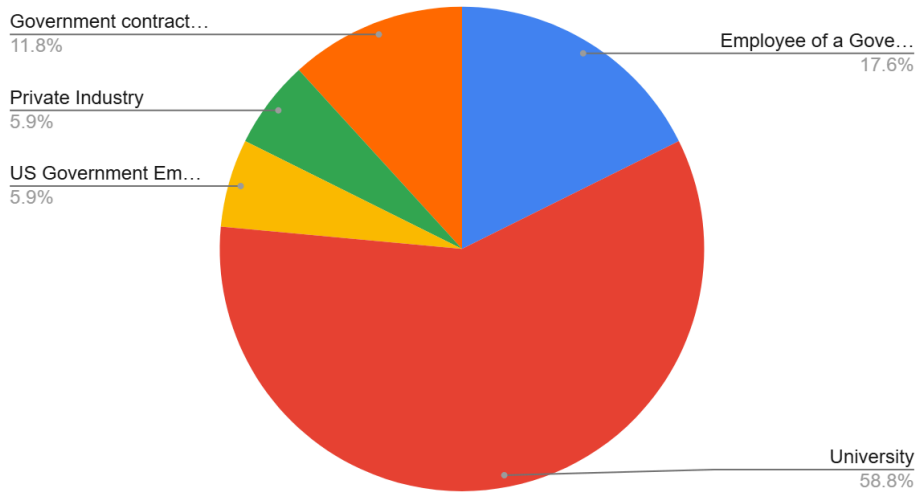


Figure 1: Organization of respondents.

Another question that was asked was what was the role of the respondent within their organization (**Figure 2**). The highest response was scientist/professional, followed by professors and then graduate students. To then understand how Py-ART is being used by these individuals in these organizations, we asked respondents on what they were using Py-ART for (**Table 1**). A summary or categories of common responses on what Py-ART is used for from **Table 1**. is as follows:

1. Radar data processing and analysis
2. Academic/scientific research
3. Visualization and plotting
4. Data ingestion/conversion and handling multiple radar sources
5. Quality control, correction, and gridding/product generation

What is your role within your organization?

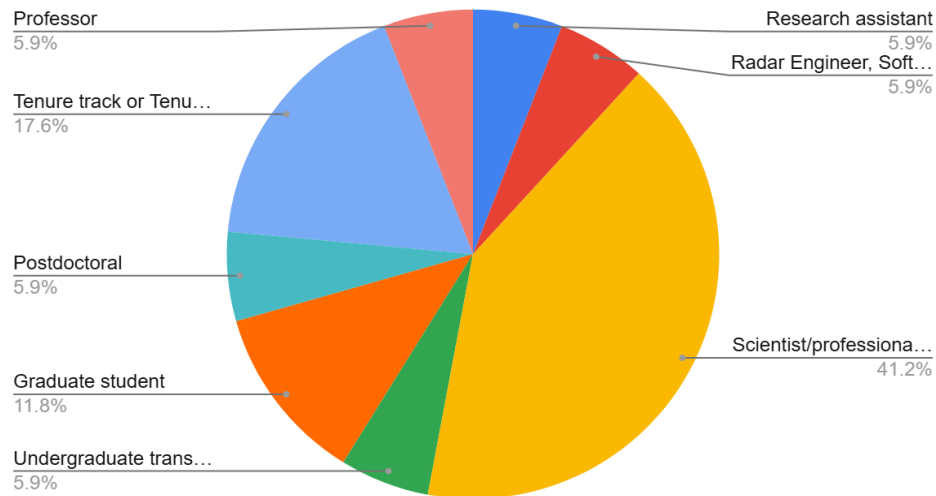


Figure 2: Role of respondents within their organization.

What do you use Py-ART for?

Research in weather radar at university

processing polarimetric radar data

Processing and analysis of radar data from a range of sources, for academic research.

I use Py-ART for the convenience of how easy it can analyze atmospheric datasets of various instrumentations. I also used it in previous ARM summer school where I got to see how it was applied on radars and model simulations.

Processing the ARM CSAPR2 datasets

scientific research

Quantitative precipitation estimation from 5-minute radar scans, including cleaning, estimation, interpolation, and mapping.

Radar data analysis
Retrospective analysis of both operational NEXRAD and research C-band radar for urban precipitation
Radar processing and analysis
radar VAD analysis
Loading and visualising radar data, regridding
Read data, converter to xarray and plots.
Use it in our radar processing software to QC , plot, grid, create products.
Reading, writing, and analysing the multiple radar data.
Reading and visualization and correction of data.
Radar data analysis tracking

Table 1: What do the respondents use Py-ART for.

2.1.1 Non Py-ART Users Survey Feedback

For this roadmap, we did not have any non-user survey respondents at this time. This could have been due to less responses for this survey compared to the last roadmap or increased usage of Py-ART within the community.

2.1.2 Py-ART Users Survey Feedback

Similar to previous roadmaps, many features requested by the user community were added to Py-ART such as Xarray support, CAPPIs and more. Some features were not added, either due to low priority or a lack of time to implement these features (ex: gridding interpolation). The questions for this roadmap and survey were fully open-ended this time, and by doing so, there seemed to be a variety of different feature requests and there were not many overlapping feature requests as well (**Table 2**). One of the requests was updating the VAD code. The VAD code has had a few issues opened on GitHub with concerns to overestimating wind values. This feature seems to be one that is not only used quite often, but does require more attention. With xradar growing in popularity and slowly being a replacement in Py-ART for reading and working with different radar formats, it is not surprising that users would want more integration for the xarray/xradar ecosystem. Similar to previous roadmaps, we are seeing more

reader and plotting functionality as well being requested. Some options are worth exploring such as alternative geographic maps, but at the moment, available packages for accomplishing such, are limited. Similar to previous roadmaps as well, responses such as digital terrain maps were present. We would suggest using other packages in the open radar ecosystem that do accomplish this feature to avoid duplicating effort.

What additional features would you like to see in Py-ART, whether it's new readers or plotting functions or new retrievals or corrections? (Optional)

Surprise me!

New readers and plotting functions.

I WOULD LOVE TO SEE, IF THERE IS A FEATURE, BASED ON PPI SCANS AND GATE RESOLUTION PROVIDES THE CELL AREA, CELL HEIGHT AND WIDTH, SPECIFICALLY FOR ISOLATED CELLS

Integration with other tools is a big pain point for me. Customizing plots outside the matplotlib ecosystem is especially tricky and I would really benefit from additional work on that side of things along with some more documentation of complete dependencies and pipeline examples. This was a large time sink for me coming into this library as the matplotlib outputs didn't work for my org's needs (I'm using hvplot+panel or pyvista/VTK depending on the project).

Better integration with xarray and more generally outputting radar objects to be used in other packages (such as pandas). One example, currently facing difficulties with saving just a few days of Pyart gridded NEXRAD data because of large (~50 GB) files associated with high float precision.

Update the VAD code may be necessary to match the quality of LROSE package.

The blocking map using digital terrain.

An alternative to geographical maps, cartopy is very slow.

Reading GRIB2 formatted radar data. If needed, I could contribute a little bit.

More algorithms for analysis for scientific use.

Table 2: Features requested by the community to be added to Py-ART.

We also asked users their favorite features both as a ranking (**Figure 3**) and also provided an open ended response if there are other features that we may have overlooked (**Figure 4**). The reason we ask this question is more to understand what features might not necessarily need changes or additions and are already popular in the user space, but also if a bug does occur, where to prioritize effort as these features most likely will have an impact on operations for these individuals. Variety of readers was the most voted favorite feature which is similar to previous roadmaps. Surprisingly xradar was the least voted, this is most likely due to it being a newer feature within Py-ART and has yet to be utilized by a majority of the Py-ART user community.

What is your favorite/most beneficial feature within Py-ART, please rank your choices 1-5, 1 being most beneficial (Optional).

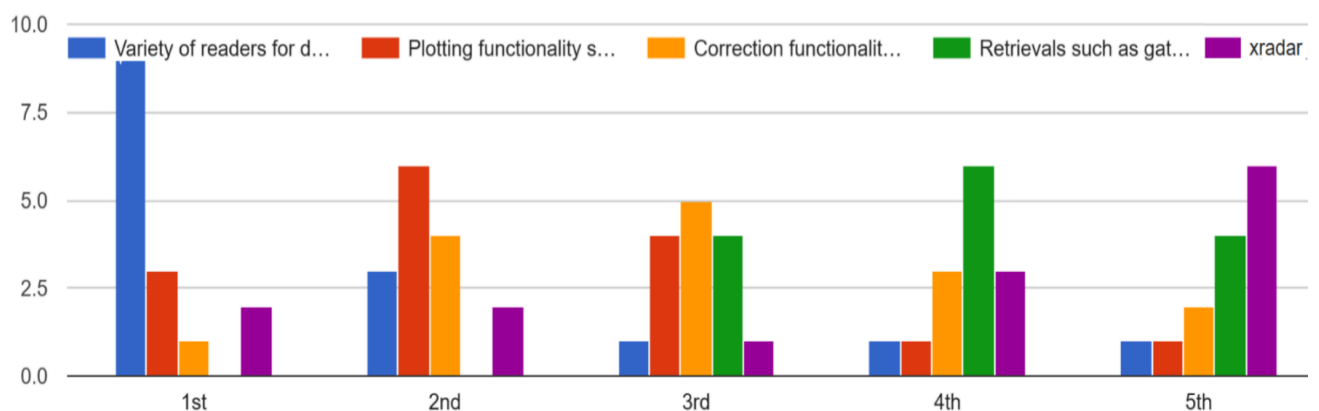


Figure 3: Favorite features currently within Py-ART. These features are the modules within Py-ART: Input and Output, Plotting, Corrections, Retrievals and Xradar integration. Users rank each of these features as their favorite choice between 1-5, 1 being the highest.

If a favorite/most beneficial feature wasn't mentioned in the question above, what is your favorite feature? (Optional)

KDP derivation

The common data model

Personally, for me the best thing was function called "get_column" which i can use to get the vertical profile or to get the RHI plots for any cross section using the NEXRAD

PyDDA (although I don't use it personally)

Qpe retrieval.

griding

Figure 4: Open-ended favorite feature if not provided in the ranking of Figure 3.

For this roadmap we also combined the question of contributing to Py-ART into just one question if there are barriers preventing the respondent from contributing to Py-ART (**Figure 5**). Tied for the most responses are “Just not enough time” and “I do not think I have done anything worth contributing” followed by “I need to clean up my code and add unit tests”. What can be learned from this is that users might face some road blocks in contributing code if they have code

to contribute. One way the developer can assist, is by providing more documentation and guides on contributing to Py-ART. Or if time is available, provide guidance during pull requests on the process of writing unit tests and cleaning code.

Are there any barriers to you contributing to Py-ART? (Multiple Responses) (Optional)

15 responses

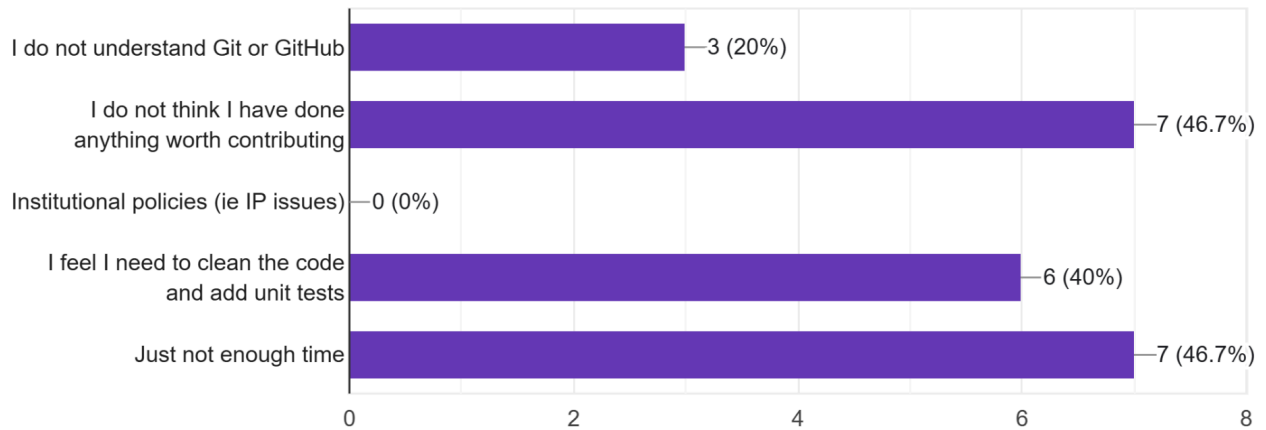


Figure 5: Barriers to contributing to Py-ART.

Every roadmap has had many responses on improved documentation. Because of this, we also inquired on where we can improve our documentation (**Figure 6**). Plotting, corrections and retrievals were the top choices and will be noted when improving our documentation. This will be the highest priority feature as it has been one of the most highly requested features in each survey while also being a key component of improving the user experience.

With Py-ART's documentation, what examples would you like to see more of? (Optional)

16 responses

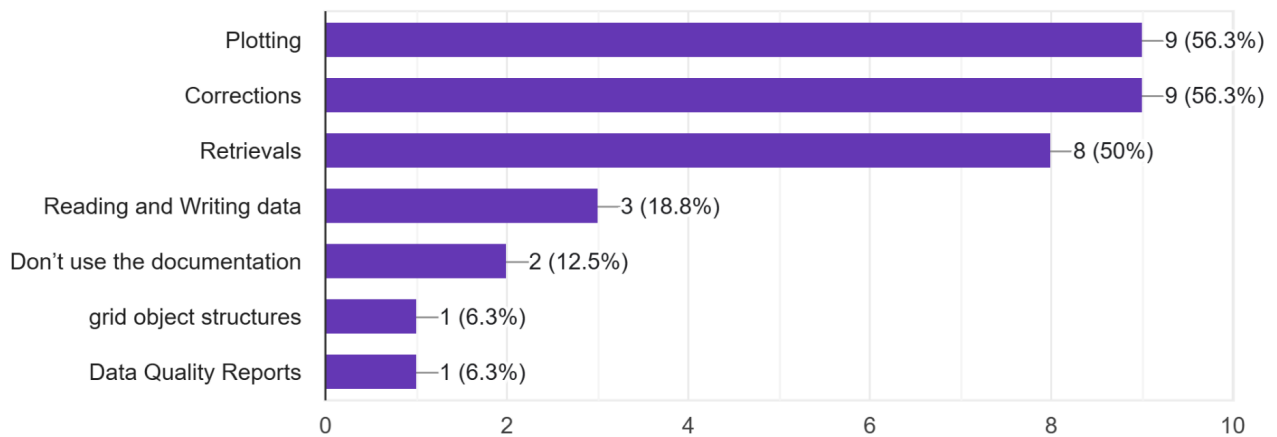


Figure 6: Question of which modules would users want more examples.

2.1.3 User Forums

The survey has given us insight on the user community, but to have a complete picture of the user community, mention of common traffic themes of the GitHub issue tracker and the GitHub discussion forum is needed. The lead developer and science lead answer these questions to the best of our ability with assistance from the associate developers. We also receive many feature requests on the forums which are similar to the survey results. Many issues that users encountered involved Py-ART's gridding module, velocity region dealiaser and Py-ART's auxiliary input module. We can learn from these issues and discussions to also shape this roadmap. For example, many of the issues involving Py-ART's gridding module and velocity dealiasing could be solved by creating more examples showing the use of these modules with different datasets. A common theme that we see is different sweep modes, scan strategies, gate spacing and more, that can impact the results of these modules if they are not adjusted to work with these different variables. There should be time invested in creating new examples for different workflows. We also will mention that due to Py-ART being so successful, users were using the issue tracker to ask basic radar science questions. We do attempt to answer these questions when they arise, however we have begun to push users to use the open radar science discourse to ask general radar science questions. This frees up time for the developers as well as allowing the broader community to be a part of the discussions.

3 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. The recommended governance structure will be identical to previous roadmaps. The governance structure is as follows:

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. The Science Lead will also coordinate reviews of the science behind a pull request where some claim has been made. There have been several pull requests accepted in the past which did not accurately implement the methodology from the literature. While it is difficult to catch all inconsistencies the Science Lead will make a judgement on if a pull request requires more review or (in the case of simple fixes) can be accepted as is.

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). Responds to users on the GitHub issue tracker, GitHub discussion page and open radar science discourse page (Py-ART related questions) with the assistance of the associate developers.

Associate Developers: Responsible, as time allows, for doing an initial check of pull requests for suitability and adherence to the Contributors Guide. Py-ART currently has about 2 associate developers, but this will most likely have to increase as the package continues to grow with popularity. It is expected that the associate developers will be recognized as key members of the project and are acknowledged accordingly in future publications and presentations.

4 Overarching Goals for the Next Five Years

The aim of Py-ART is to continue lowering barriers to doing science with radar data, in particular for Department of Energy stakeholders. One of the most requested features as well as barriers to users when implementing their workflows would be to increase documentation. Increasing examples, notebooks and blogs would also most likely reduce issues opened as many issues that arise are related to use of modules with documentation missing for different types of datasets throughout the community. This should be a high priority and most likely the main priority to improve the user experience.

More inputs and outputs for radar formats is also a highly requested feature, however this focus should be towards xradar and the Py-ART developers should assist with implementing those readers and moving current readers to that package, while directing users there as well. This would be a moderate priority.

5 Priority Features Summary

The Development team will prioritize the acceptance of Pull requests and perform targeted strategic development that adds the features outlined in the following subsections. As alluded to in the descriptions from previous sections ‘Highest priority’ means that ARM will accept pull requests that need significant (more than a few days) work or even perform some ARM funded work ourselves. "Moderate priority" means we will accept pull requests that may require some clean up and minor development. "Lower priority" are items where we will only advise the requester on changes required. This will change based on reviews from ARM stakeholders. Continued development, such as maintaining continuous integration, bug fix, general maintenance is expected and will be excluded from priority of features below.

5.1 Highest Priority

- 1. Improved documentation and examples:** Improve Py-ART’s documentation to include more user cases. This could help alleviate many issues that are opened on Py-ART’s GitHub issue tracker, as many questions relate to a user's specific use case that we might not have an example or blog post for. Also for better confidence from the science community, and to better understand we should continue to improve on references and examples of algorithms that are scientifically referenced in Py-ART. Currently code and documentation has references, but the best route to address some concerns would be to have a new examples section comparing the code to real case studies to show these algorithms in action. An example of this would be to show why the radius of influence or other parameters in gridding from antenna coordinates to a Cartesian grid can make a difference depending on type of radar, gate spacing and more. Finally we would want to add documentation referencing other open source packages in ARM such as ACT so that users can see other available resources for their research.
- 2. Investigate and validate the Velocity Azimuth Display (VAD) module:** The VAD code within Py-ART is used by many users, however there have been some issues raised on the estimates from the VAD code. Developers should investigate if there are issues present in the VAD code, but also determine if there are other VAD methods that can be added to Py-ART.
- 3. Continue xarray and xradar integration:** xradar enables cross package compatibility in the most natural way. This item directly addresses user feedback on desired compatibility. Basic functionality of xradar has been integrated into Py-ART. This allows users to work with their radar data in xarray datasets and data trees. There should be more effort to integrate new xradar functions into Py-ART as well as confirming that current functions and modules within Py-ART continue to operate correctly (ex: `extract_sweeps`, `radar.info` and more).

4. **New documented procedures and automated countermeasures against the rising issue of AI generated content:** Over recent years, the package had issues and pull requests opened, some possibly AI generated. To ensure that these issues are legitimate and not malicious, we need procedures and documentation alongside unit testing, CI, etc. to ensure that these problematic issues or pull requests are caught.

5.2 Moderate Priority

1. **Increased radar inputs and outputs in collaboration with xradar:** Many of the responses on the GitHub issue tracker and discussion forum are that of radar formats not readable by Py-ART. Some of these formats would be easier than others to implement. Moving these formats to xradar is ideal and developers should coordinate with implementing these readers into xradar while depreciating older readers in Py-ART.
2. **Growth of retrievals and correction suites:** Py-ART's suite of corrections and retrievals has quite few algorithms to improve on datasets, but there has been a growing demand for more of these features. An example of a popular retrieval recently added was the column extract function added to Py-ART. Developers should continue to add new algorithms and methods to Py-ART's correction and retrieval suites. An example of a requested retrieval or correction to be added to Py-ART that have not been added yet is, aircraft navigation corrections. We will want to continue to see which retrievals or corrections are in high demand, and see if they are a good fit for ARM and Py-ART.
3. **Integration with other radar packages:** Similar to Py-ART's integration with ACT and xradar. Determine if there are other plotting modules, retrieval packages and more that Py-ART can be integrated with. This includes updating the documentation to point the community to ACT as well for atmospheric research outside the scope of radars.
4. **Replace or update differential phase module:** The differential phase modules have been known to be slow or use dependencies that are not well supported. Ideally we would want to speed up these modules, replace them with new methods, or find dependencies that are better maintained or work.

5.3 Low Priority

1. **Read NEXRAD tabular data:** Many of our users are at the National Weather Service. A feature requested during the open radar short course was functionality to read NEXRAD level 3 tabular data.

2. **Determine if other geographic maps exist:** Users have raised issues on the slow performance on cartopy maps. Investigate if there are other packages that exist similar to cartopy and confirm concerns of performance issues.
3. **Increase of gridding interpolation methods:** This priority existed in the previous roadmap, however time was not allocated to work on this feature and should still be a priority during this roadmap. Examples of effort would be custom weighting functions as well as variable grid spacing, which has been requested by users.

6. Measuring Impact

As a Department of Energy supported project it is important, but not sufficient, to have a roadmap. It is important to monitor impact in order to justify investment and measure the success of the roadmap. Similar to previous roadmap, the impact of Py-ART can be measured three ways:

1. **Growing the number of users and installs:** While it is difficult to get exact statistics, several Py-ART distribution channels provide information of how widespread the usage of the toolkit is. For example, **Figure 7** shows that the main repository is viewed by over 238 unique visitors and in one week the GitHub repository was cloned 118 times. We can see an increase in traffic of clones of the Py-ART (essentially downloads from GitHub) as compared to the last roadmap in **Figure 8**, and a decrease in views on GitHub, possibly due to the overhauled documentation page that is the main landing page for users. Py-ART during the last roadmap had little under 300K downloads. Py-ART now has 1.25 million downloads on Anaconda as seen in **Figure 9**. The growth on Anaconda has been substantial. The increase of users and traffic can be seen on the issue tracker and google groups as well.

conda-forge / packages / arm_pyart 2.1.0

Python ARM Radar Toolkit

copied from cf-post-staging / arm_pyart

Conda	Files	Labels	Badges
<p>License: BSD-3-Clause</p> <p>Home: http://arm-doe.github.io/pyart/</p> <p>Development: https://arm-doe.github.io/pyart/API/index.html</p> <p>Documentation: https://arm-doe.github.io/pyart/</p> <p>1249143 total downloads</p> <p>Last upload: 11 days and 19 hours ago</p>			

Figure 9: Downloads of Py-ART from conda-forge on Anaconda.

2. **Number and success of dependent projects:** Appendix 1 shows a sample of projects that require Py-ART as a dependency. The increasing number of packages that use Py-ART as a dependency is a measure of success.
3. **Papers and presentations using Py-ART:** Publications are treated by many user facilities as a metric of scientific impact. To date, Py-ART has been cited 512 times according to google scholar. This is nearly double the amount of citations from the last roadmap.. Py-ART includes a message on start up encouraging users to acknowledge the ARM program and cite (Helmus and Collis, 2016). We will track and record instances of this appearing in major journals and encourage users to continue to self-report so we can continue recording these metrics.

References

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

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

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, 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, 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

Sherman, Z., and Coauthors, 2024: Effective Visualization of Radar Data for Users Impacted by Color Vision Deficiency. Bull. Amer. Meteor. Soc., 105, E1479–E1489, <https://doi.org/10.1175/BAMS-D-23-0056.1>.

Appendix 1: Packages that use Py-ART as a dependency

95 Repositories 9 Packages ⓘ		Owner ▾
 w-k-jones / tobac-flow	☆ 7 🍷 2	
 MeteoSwiss / pyrad	☆ 46 🍷 10	
 Kwonil-Kim / kkpy	☆ 2 🍷 2	
 rijaf-iri / mtorwaradar	☆ 0 🍷 0	
 jthielen / OpenMosaic	☆ 16 🍷 0	
 vlouf / radar_grids	☆ 5 🍷 2	
 vlouf / radar_quicklooks	☆ 0 🍷 0	
 vlouf / cluttercal	☆ 7 🍷 2	
 mcgillradar / bugtracker	☆ 3 🍷 0	
 kurtispinkney / WxProject	☆ 0 🍷 0	

Appendix 2: Contributors to Py-ART

