Python ARM Radar Toolkit (Py-ART) Roadmap 2

Scott Collis, Zachary Sherman, Robert Jackson, Jason Hemedinger, Kathryn Dosey, Stacy Toyooka and Stacy Larsen

1 Introduction and Aims

Radar software is key for producing scientific insight from radar data. 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. After its release in 2013, Py-ART and its collection of algorithms and radars grew it became clear that, while well meaning, third party contributions would subsume other ARM funded efforts by the Py-ART development team. Due to this, and to ensure the toolkit has maximal impact, a roadmap to chart development priorities for the next five years was needed. The roadmap was developed starting in 2015 and was followed until the end of 2020. Due to the original 5 year roadmap, we were able to prioritize development and had much success in development in needs of ARM and the community as a whole. Due to this success, a second roadmap is needed for the next 5 years of Py-ART's development.

1.1 The Python ARM Radar Toolkit

The motivation of Py-ART is to provide a package, in the popular and easy to use coding language Python, that allows the user a suite of tools to read, analyze and correct radar data. Py-ART grew out of a collection of radar algorithms generated to support the new radar capability in the ARM program (Mather and Voyles, 2012). The two original features were a Linear Programming (LP) technique for polarimetric phase processing (Giangrande el at, 2013) and mapping radar data onto a cartesian grid. Shortly after, development on Py-ART began in earnest. ARM provided support specifically to release Py-ART open source. In September of 2012, Py-ART was uploaded to the social coding platform GitHub at https://github.com/ARM-DOE/pyart. Over time, due to contributions from within ARM but also the open source community as a whole, Py-ART has evolved to have tools such as dealiasing, kdp processing, polar to Cartesian gridding and more. Py-ART also has evolved to read a variety of radar formats, such as cfradial, ODIM, RSL, sigmet, radar spectra and more. Release notes can be found here: https://github.com/ARM-DOE/pyart/releases.

Py-ART has benefited from code from 38 individual contributors. When the original roadmap was written, Py-ART had 22 contributors. The 38 contributors with their commits over time can be seen in **Appendix 2.** 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. The lead developer as well as the associate developers, review these pull requests and guide the user submitting the code, to make the code acceptable for Py-ART. 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. Currently on Anaconda, Py-ART has been downloaded 289K times. A note for clarification, the lead developer and associate developers are typically in development mode for Py-ART which does not increase downloads on Anaconda. This means that Py-ART has been downloaded 289K mostly outside the main developer group. ARM is referenced not only within Py-ART's code, but also in its documentation. With the surge of users, ARM is shown in the mentioned documentation 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 Climate and 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 2.0) VAP. CMAC 2.0 uses Py-ART for most of its working code, but has a configuration based on Py-ART's configuration, which allows for new radars to be added with ease. Having Py-ART in the community allows ARM to seamlessly incorporate new research from DoE and non-DoE sources into VAPs without the laborious task of transcribing from papers or other languages. 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. There is also integration now with the ARM funded Atmospheric Community Toolkit (ACT) and plans for other future integration as well.

1.3 Targeted Reviews

This document currently contains responses from the general community and possibly responses from stakeholders. 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.

2 The Py-ART Roadmap 2 Survey and Reviews

2.1 Success of the First Roadmap

The first roadmap was developed starting in 2015 and was followed until the end of 2020. The roadmap consists of many key inputs from users, non users, academia, ARM and more. From the roadmap we were able to learn that code for reading radar spectra was desired, which was added in late 2019, cartopy is a preferred plotting library for radar ppi, also added with basemap priority removed. We did change priorities in the roadmap based on survey results and feedback, such as leaving CFADs to the PI or in another package. The feedback from the surveys and community helped shape the roadmap, we then just had to follow course. With the success of the first roadmap over that 5 year period, we decided that a second roadmap that charted the next 5 years was necessary.

2.2 The Survey

Similar to the first roadmap, 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. The questions regarding the user's background are similar to the first roadmap. Overall the survey has been toned down, as the previous survey might have been too long as we did receive responses, but many questions were skipped. The survey was hosted by the communications team from PNNL and was included in the July newsletter as well as was posted on Py-ART’s user google groups page. Many of the plots were created by the communications team using the results from SurveyMonkey. The survey had 37 respondents, with most of the respondents from University or other academic settings **Figure 1**.

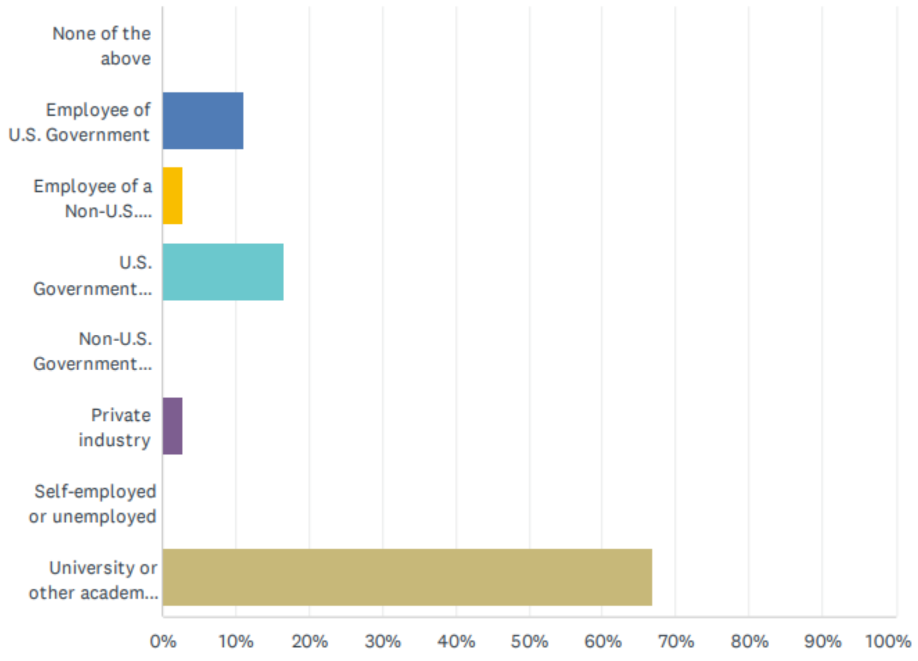
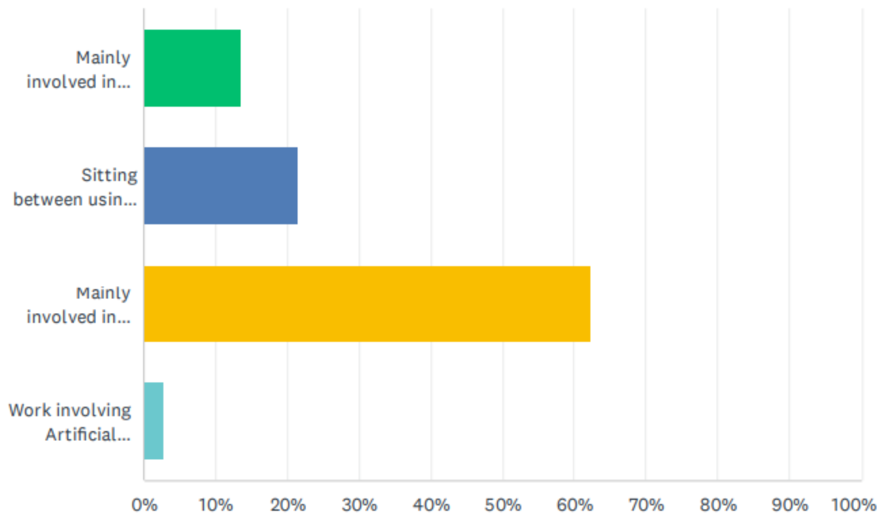


Figure 1: Organization of respondents.

Similar to the first roadmap, we did not receive enough responses to differentiate between user groups, however we can separate the responses from users and no-users, which were 26 and 11 respectively, similar to 24, 11 from the original roadmap. We also wanted to understand the field of work or study for users **Figure 2**, with most respondents being in observations followed by modeling. Operating system used was also asked, with results split between Linux and Mac. For this survey, we did add a question on the Atmospheric Community Toolkit (ACT) to gauge from users and stakeholders on how Py-ART and ACT should better integrate. A key note to mention, is that, we do receive quite a lot of traffic from the Py-ART google groups page as well as the GitHub issue tracker. Much of these questions are inputting specific radar formats, corrections and examples. As the survey does give us a picture on the community, we will also include the common theme of the Py-ART google groups and issue tracker as well as it is quite possible some individuals might have used those as an outlet and not the survey. Survey questions asking for user or non-user rankings are based on scores, with the higher the score, the higher that response and its request or impact.



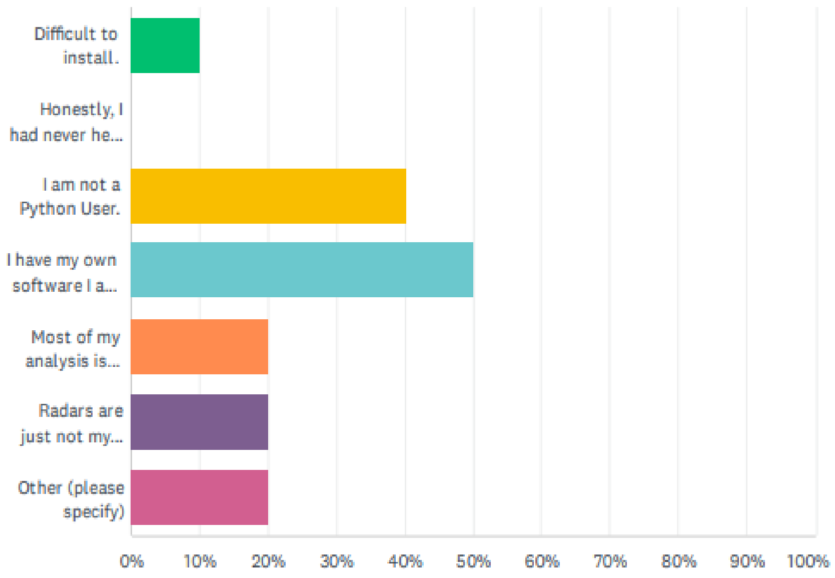
ANSWER CHOICES	RESPONSES	
Mainly involved in using models, limited use of observations	13.51%	5
Sitting between using models and observations	21.62%	8
Mainly involved in using observations, rarely use or interact with numerical models	62.16%	23
Work involving Artificial Intelligence	2.70%	1
TOTAL		37

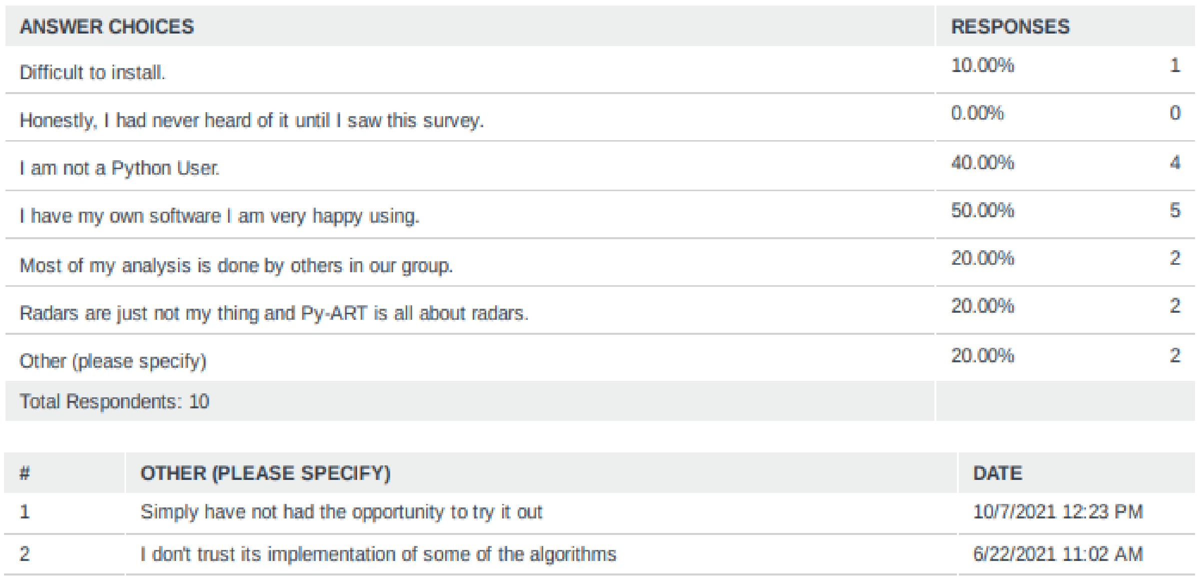
Figure 2: Field of work or study of respondents.

2.2.1 Non Py-ART Users Survey

First we will start with non users to gauge the barriers to using Py-ART as well as possible changes to attract more users. With the most common responses, we will provide some possible solutions to address these issues for these non-users.

The response chosen the most to being a barrier to using Py-ART was that the non-user is ‘happy with their own software’, followed by ‘not a python user’ **Figure 3**. ‘Not a python user’, we can continue providing python tutorials followed by Py-ART tutorials at conferences and workshops and teach the scientific stack for those who are interested in transitioning to python. For this survey, compared to the original, we added an input section as well to capture responses that we did not consider in **Figure 3**.





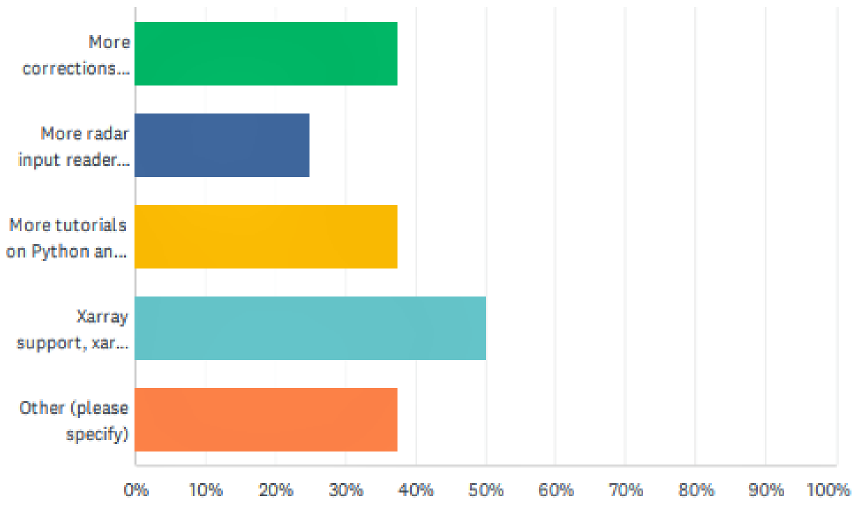
ANSWER CHOICES	RESPONSES	
Difficult to install.	10.00%	1
Honestly, I had never heard of it until I saw this survey.	0.00%	0
I am not a Python User.	40.00%	4
I have my own software I am very happy using.	50.00%	5
Most of my analysis is done by others in our group.	20.00%	2
Radars are just not my thing and Py-ART is all about radars.	20.00%	2
Other (please specify)	20.00%	2
Total Respondents: 10		

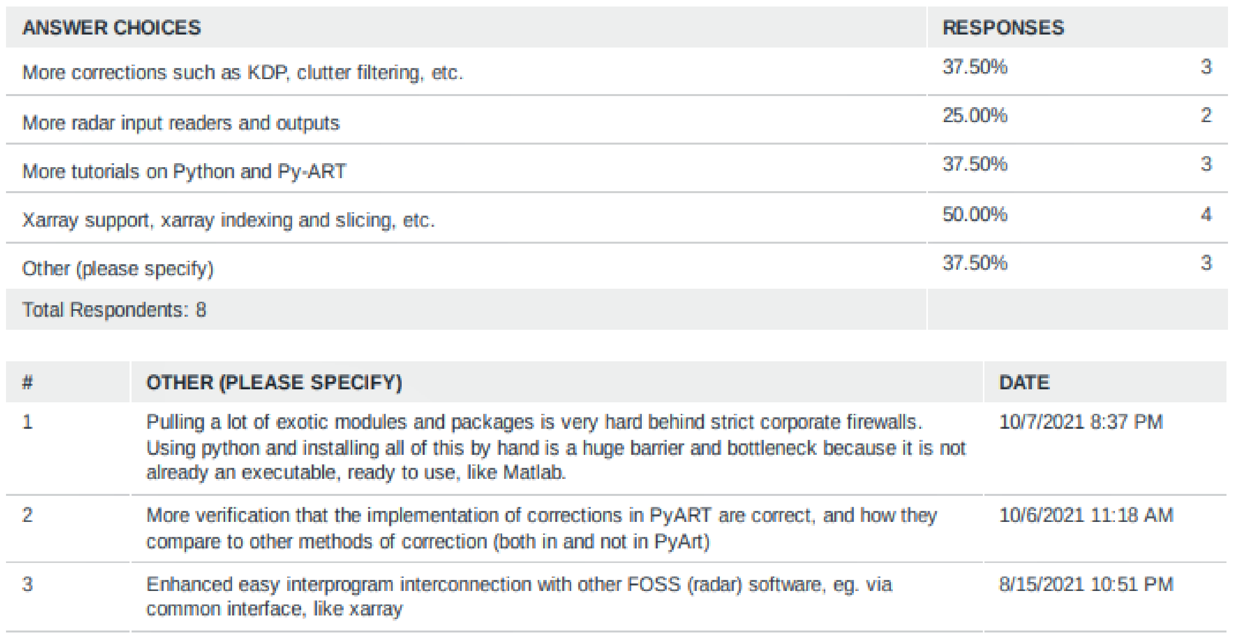
#	OTHER (PLEASE SPECIFY)	DATE
1	Simply have not had the opportunity to try it out	10/7/2021 12:23 PM
2	I don't trust its implementation of some of the algorithms	6/22/2021 11:02 AM

Figure 3: Barriers to using Py-ART as well as open reponses.

The response that was interesting and that we did not expect was that 'I don't trust its implementation of some of the algorithms'. This response was slightly concerning as we want the science to be right to help individuals further their work. This response is also very helpful, because it's a problem for which we need to tackle. A way to address such an issue is to provide real comparisons to real cases, possibly in a new documentation section within the Py-ART webpage. We do include references to each algorithm, but it seems we need to provide better examples of the algorithms in action in comparison to real case studies as well as better transparency.

The question was also asked, 'What would get you interested in using Py-ART?' **Figure 4**. Xarray support was chosen the most with 4 reponses, followed by more tutorials and corrections with a response of 3. Radar inputs and outputs were the least chosen.





ANSWER CHOICES	RESPONSES	
More corrections such as KDP, clutter filtering, etc.	37.50%	3
More radar input readers and outputs	25.00%	2
More tutorials on Python and Py-ART	37.50%	3
Xarray support, xarray indexing and slicing, etc.	50.00%	4
Other (please specify)	37.50%	3
Total Respondents: 8		

#	OTHER (PLEASE SPECIFY)	DATE
1	Pulling a lot of exotic modules and packages is very hard behind strict corporate firewalls. Using python and installing all of this by hand is a huge barrier and bottleneck because it is not already an executable, ready to use, like Matlab.	10/7/2021 8:37 PM
2	More verification that the implementation of corrections in PyART are correct, and how they compare to other methods of correction (both in and not in PyArt)	10/6/2021 11:18 AM
3	Enhanced easy interprogram interconnection with other FOSS (radar) software, eg. via common interface, like xarray	8/15/2021 10:51 PM

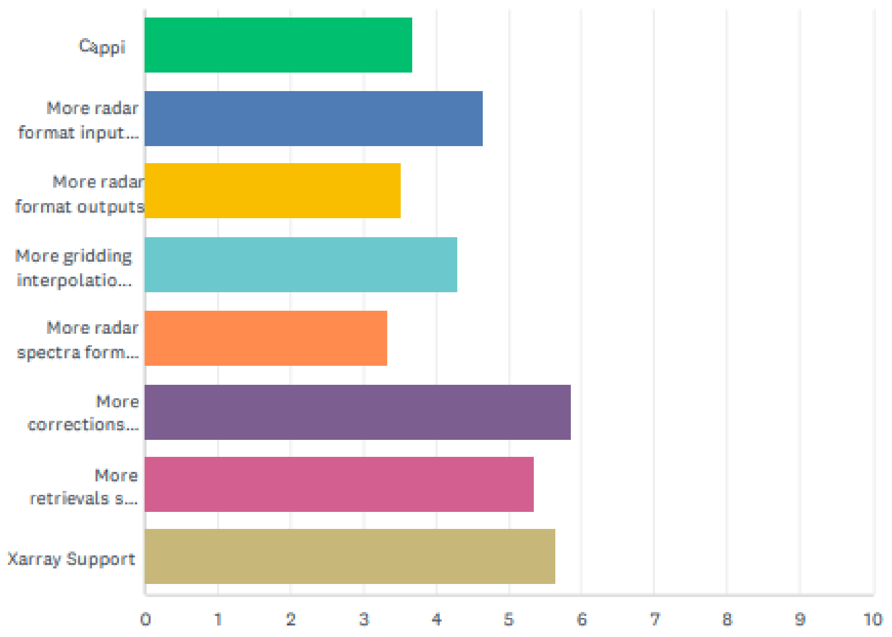
Figure 4: Non-user responses on what would get you interested in Py-ART

Much of the traffic on the Py-ART google groups page as well as the GitHub issue tracker seem to be of Py-ART not being able to read specific radar formats from across the globe; an example being the radar sites in India. An assumption is that those who can not use Py-ART due to inputs and outputs might have not participated in the survey but have used the google groups and GitHub issue tracker instead. This is an example of the mentioned above on using both the survey and traffic on the forums as a way to gauge the community. Xarray support is a common request that would enhance Py-ART greatly. We do plan on making this change, however it is a great undertaking as many of Py-ART's functions work on the original Radar Object. Xarray support is planned for this fiscal year. With the change, not only would it bring in non-users, it is a feature requested by users as well, as shown later with examples on its importance. We again received a custom response on better verification of corrections and algorithms within Py-ART. Another response was issues with installing Py-ART behind strict corporate firewalls.

These responses have been quite useful and share a common theme on some of the questions shared on Py-ART google groups and the GitHub issue tracker. We believe many of these issues can be addressed to make Py-ART more accessible to non-users, whether adding better documentation and algorithm verification or continuing the plan to add Xarray support as soon as possible.

2.2.2 Py-ART Users Survey

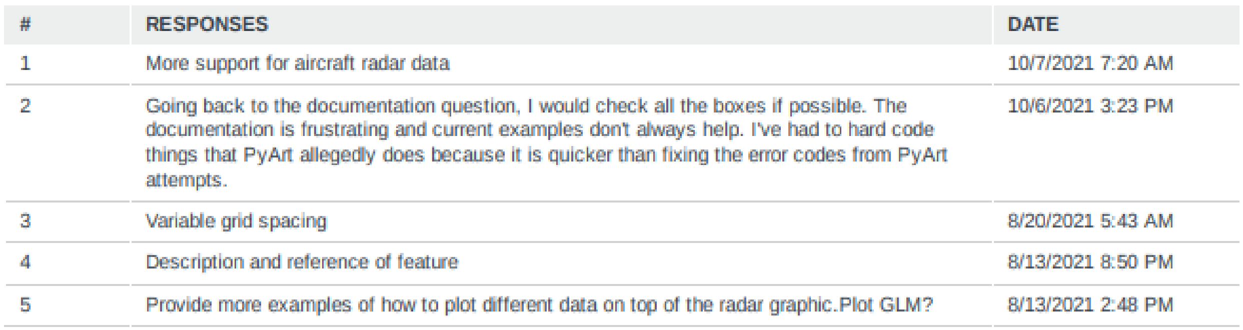
Many of the responses from the previous survey in the first roadmap on features requested now exist in Py-ART such as VADs and Cartopy support. Some of the responses such as cell tracking and multi-doppler winds were not added into Py-ART, but were decided to be better addressed by having Py-ART as a dependency and having them as separate packages (Tint Is Not Titan) (Dixon and Wiener, 1993) and (PyDDA). These responses were planned after the first survey and many were accomplished during the first 5 years. With this success, we then again ask questions again, but also with the added open response. As mentioned above, Xarray support was a popular response, but as well as more corrections and retrievals, such as dealiasing and quasi-vertical profiles. **Figure 5.** Following close behind is more radar input formats and gridding interpolation functions. More radar inputs is not surprising, as much of the traffic from google groups and GitHub is on trying to read formats not native to Py-ART. To address this, we would need to work with the community on these formats and add new readers to Py-ART's IOs. This has been the process for many of the current inputs and outputs for Py-ART, and this also allows us to focus on key stakeholders and not put too much of our effort on very unique radar formats. We can continue to review and direct for unit testing and where to add the new code. When addressing more corrections and retrievals, we should also keep in mind the response on verification that the science is correct.



	1	2	3	4	5	6	7	8	TOTAL	SCORE
Cappi	18.18% 4	4.55% 1	4.55% 1	9.09% 2	4.55% 1	13.64% 3	13.64% 3	31.82% 7	22	3.68
More radar format input readers	18.18% 4	9.09% 2	9.09% 2	4.55% 1	22.73% 5	18.18% 4	13.64% 3	4.55% 1	22	4.64
More radar format outputs	4.76% 1	14.29% 3	4.76% 1	4.76% 1	14.29% 3	9.52% 2	28.57% 6	19.05% 4	21	3.52
More gridding interpolation weight functions (Current: Barnes Cressman, Nearest)	0.00% 0	14.29% 3	9.52% 2	28.57% 6	9.52% 2	19.05% 4	14.29% 3	4.76% 1	21	4.29
More radar spectra format inputs	0.00% 0	0.00% 0	4.55% 1	18.18% 4	22.73% 5	27.27% 6	13.64% 3	13.64% 3	22	3.32
More corrections such as dealiasing, KDP processing and more	19.05% 4	28.57% 6	19.05% 4	14.29% 3	4.76% 1	4.76% 1	4.76% 1	4.76% 1	21	5.86
More retrievals such as rain rate, Quasi Vertical Profiles (QVP), etc.	8.70% 2	13.04% 3	39.13% 9	17.39% 4	4.35% 1	4.35% 1	8.70% 2	4.35% 1	23	5.35
Xarray Support	34.78% 8	17.39% 4	13.04% 3	0.00% 0	17.39% 4	0.00% 0	0.00% 0	17.39% 4	23	5.65

Figure 5: Most requested feature to be added to Py-ART. The higher the score, the higher it is requested.

So when adding new algorithms, we need to provide examples and real case studies for comparisons and be more transparent outside of references to scientific journals. In **Figure 6**, many of the open responses were geared towards better documentation and examples. An ideal solution to this would take common questions on the issue tracker and google groups and provide the code of the solution or plot etc. in Py-ART's documentation. When the current documentation was created, many of the examples reflected current content as expected. Overtime, many new features have been added. We need to have effort in showcasing these features as well as how to use them.

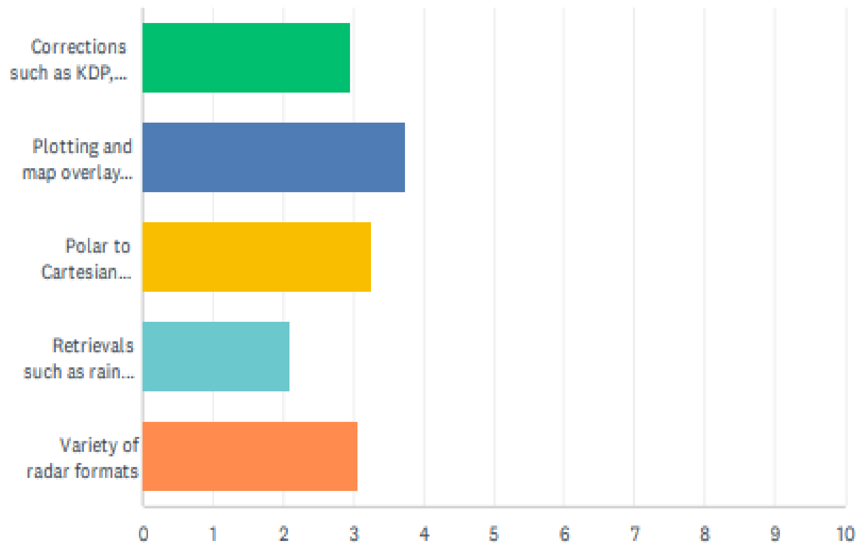


#	RESPONSES	DATE
1	More support for aircraft radar data	10/7/2021 7:20 AM
2	Going back to the documentation question, I would check all the boxes if possible. The documentation is frustrating and current examples don't always help. I've had to hard code things that PyArt allegedly does because it is quicker than fixing the error codes from PyArt attempts.	10/6/2021 3:23 PM
3	Variable grid spacing	8/20/2021 5:43 AM
4	Description and reference of feature	8/13/2021 8:50 PM
5	Provide more examples of how to plot different data on top of the radar graphic.Plot GLM?	8/13/2021 2:48 PM

Figure 6: Open responses to what feature users would like to see added into Py-ART.

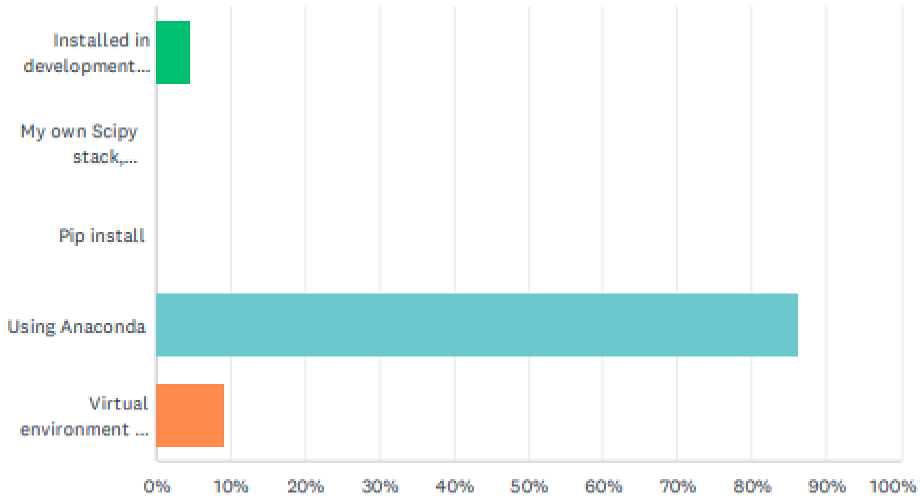
We also asked users their favorite features. Plotting with map overlays and polar to Cartesian gridding were the highest rankings, with corrections and a variety of radar formats following close behind in **Figure 7**. The reason we ask this question is more to understand what features might not necessarily 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.

Another question asked is how do users install Py-ART. Anaconda was the overwhelming response with 19 followed by virtual environments outside of Anaconda with 2 **Figure 8**. This response is also not surprising, as mentioned before, there are 289K downloads of Py-ART from Anaconda. Anaconda is an environment manager which allows for the download of many Python packages in an environment, but also cross checks dependencies and conflicts to ensure a working environment. Many users will not be in development mode if they aren't doing code development. Pip installation is useful, but can have conflicts at times, so many of these responses are not surprising at all.



	1	2	3	4	5	TOTAL	SCORE
Corrections such as KDP, dealiasing	15.79% 3	5.26% 1	47.37% 9	21.05% 4	10.53% 2	19	2.95
Plotting and map overlay modules	36.84% 7	42.11% 8	0.00% 0	0.00% 0	21.05% 4	19	3.74
Polar to Cartesian gridding	30.00% 6	25.00% 5	10.00% 2	10.00% 2	25.00% 5	20	3.25
Retrievals such as rain rate, QVP and VAD	0.00% 0	0.00% 0	25.00% 5	60.00% 12	15.00% 3	20	2.10
Variety of radar formats	19.05% 4	28.57% 6	19.05% 4	4.76% 1	28.57% 6	21	3.05

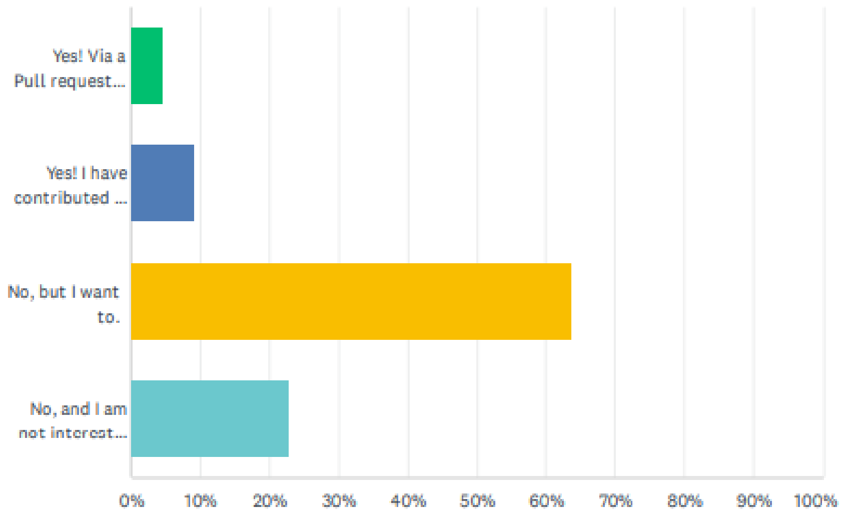
Figure 7: Favorite features currently within Py-ART. Higher the score, the higher that feature is a favorite.



ANSWER CHOICES	RESPONSES
Installed in development mode 'pip install -e .'	4.55%
My own Scipy stack, Installed from source	0.00%
Pip install	0.00%
Using Anaconda	86.36%
Virtual environment not Anaconda	9.09%
TOTAL	

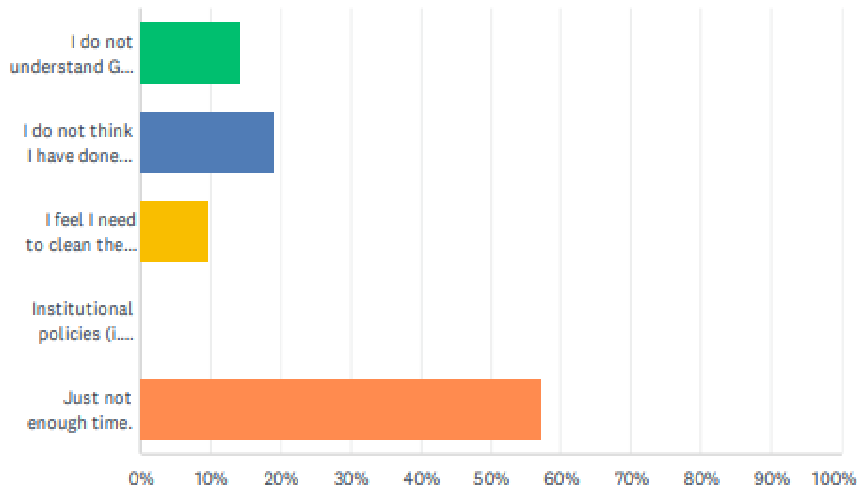
Figure 8: Installation method used for Py-ART.

The question was also asked, 'Have you ever contributed to Py-ART?'. Overwhelmingly, the response was 'No, but I want to' **Figure 9**. This question provided great insight that much of the community does want to contribute to Py-ART, but just has not been able to yet. This transitions into the next question on barriers to contributing, which was overwhelmingly not having enough time **Figure 10**. Responses for 'I don't understand Git or GitHub' as well as 'I do not think I have done anything worth contributing' were also chosen. Py-ART's documentation does provide information on the process of contributing to Py-ART and using GitHub. If time is a limiting factor due to complexity of the process, we might need to work on a way to streamline the contribution process without sacrificing quality. If time is a factor due to outside factors related to the user, maybe we can provide on the google groups or elsewhere a forum of what they wish to provide, but a way for them to be credited with the assistance of a current contributor or developer. GitHub does provide a way to contribute code, but sets the author as another user. We have done this before with code from the MeteoSwiss group who also does development of Py-ART on their own version for their organization.



ANSWER CHOICES	RESPONSES	
Yes! Via a Pull request on GitHub.	4.55%	1
Yes! I have contributed IP or code that was implemented by someone else.	9.09%	2
No, but I want to.	63.64%	14
No, and I am not interested in doing so.	22.73%	5
TOTAL		22

Figure 9: Has the user contributed to Py-ART and if yes how, and if no why not.

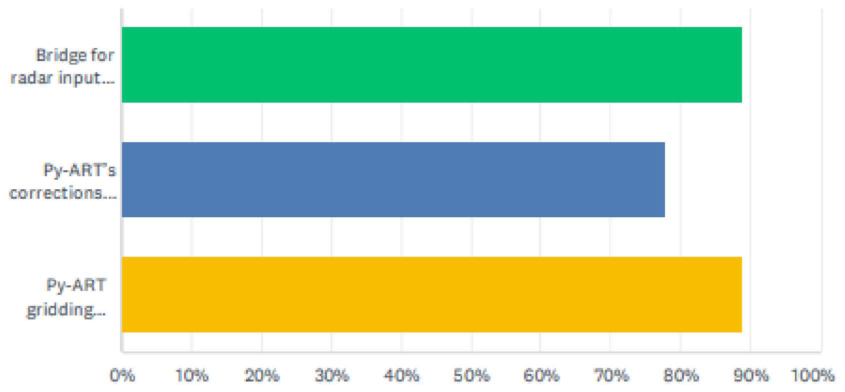


ANSWER CHOICES	RESPONSES	
I do not understand Git or GitHub.	14.29%	3
I do not think I have done anything worth contributing.	19.05%	4
I feel I need to clean the code and add unit tests.	9.52%	2
Institutional policies (i.e., IP issues).	0.00%	0
Just not enough time.	57.14%	12
TOTAL		21

Figure 10: Has the user contributed to Py-ART and if yes how, and if no why not.

2.2.3 ACT Integration Survey

With ACT growing within the ARM and atmospheric community, we wanted to gauge on how we can better integrate with ACT. We asked the Py-ART community if they have heard of ACT with 40.91% of respondents with yes and the remainder as no. Close to an even split, but with more responding that they have not heard of ACT. It is possibly that as ACT is newer and growing, that its reach outside of ARM is still growing. With Py-ART's popularity, with better integration, it might be possible to grow ACT alongside Py-ART. This leads into **Figure 11**, how would those who have heard of ACT, would like Py-ART to integrate with ACT. All three responses were chosen equally, for Py-ART to be used as a bridge for inputting radar formats, use of Py-ART's correction suite and use of Py-ART's polar to Cartesian gridding algorithm. With these responses, it seems using Py-ART as an overall backend to ACT might be the route to take with collaborating between the two packages.



ANSWER CHOICES	RESPONSES	
Bridge for radar input reading in ACT	88.89%	8
Py-ART's corrections integrated with ACT	77.78%	7
Py-ART gridding algorithm used within ACT	88.89%	8
Total Respondents: 9		

Figure 11: How Py-ART should integrate with ACT, multiple choices allowed.

2.2.4 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 Google groups forum is needed. Many of the questions on the Google groups forum are radar science questions such as reflectivity masking and more. 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 requests are radar formats and reading them with Py-ART. We try to point these users to the custom radar object notebook, but some formats require more work. Similar to the survey results, new radar inputs would be ideal, however we would need assistance from the community with these, affecting the priority of this feature in the section below. Many questions involved are similar to the survey results of new retrieval and correction features. Many users on the forums ask questions on new filtering methods or how to correct noise in reflectivity, as examples.

3 Modified Governance Structure

Similar to the first roadmap, 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 as well as the role of the associate developers. The lead developer maintains the overall package, but the associate developers assistance when these tasks require secondary input as well as assistance with the increase of traffic due to the growing popularity of the package. The roles will continue to be, with slight modification:

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 and google groups 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. The Xarray integration will also require much effort as it requires an overhaul of Py-ART. 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 lower barriers to doing science with radar data, in particular for Department of Energy stakeholders. Expected from the survey results, Xarray support garnered many responses from both users and non-users. This would increase not only usability with slicing radar data and working with arrays easier, but adds output and input support as well as a variety of integration with other packages such as ACT. This task will be the most demanding of effort to accomplish but would have the greatest impact.

Increased documentation and examples was also highly requested. This was a high priority from the previous roadmap. Though the documentation has been overhauled, with the addition of many notebook examples and plots, there is still a response for more to be added and improved on.

More algorithms for retrievals and corrections, such as QVP (Ryzhkov et al, 2016) , filtering and more were requested from the user survey. There were also responses on science credibility on these algorithms, so included with these additions to the corrections and retrievals suites, we will provide more real case studies as well as documentation to support current and future algorithms.

More inputs and outputs for radar formats as well as increased default formats to support changing radar formats such as cfradial. Focus will be on supporting current radar formats and adjusting for changes for future standard changes. New radar formats will be added based on increased demand, but with developers of Py-ART assisting with reviews of new code and standards.

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. **Xarray integration**: Overhaul the radar object in Py-ART to be an Xarray object. This functionality will allow Py-ART to become more powerful with slicing and indexing data. Integration will allow for better interaction with other packages such as ACT. Also allows for increased output formats, built into Xarray. This will take a lot of effort, but will evolve Py-ART to new heights.

2. **Integration with ACT**: With ACT’s growth within ARM, having Py-ART integrated within ACT would improve ACT’s functionality, especially once Py-ART uses Xarray.

3. **Improved references for algorithms**: For better confidence from the science community, 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 the code is matching the scientific algorithm.

4. **Improved examples and documentation**: Add more examples and documentation. Many users requested better documentation and examples for plotting as well as custom data objects for unique datasets. More examples for retrievals and corrections is also

desired. This would also help with the barrier for individuals, not only for the algorithm confidence mentioned above, but also for those learning Python.

5.2 Moderate Priority

1. **Increased radar inputs and outputs**: Many of the responses on google groups and the GitHub issue trackers is that of radar formats not readable by Py-ART. Some of these formats would be easier than others to implement. If demand for a format increases, we can add them to Py-ART. For more unique cases, we can help guide the community in Pull Requests.

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 QVP and VAD (Lee et al, 2014) methods within Py-ART. As new journals come out, we would like to keep up with new methods.

3. **Increase of gridding interpolation methods**: Currently Barnes, Cressman and Nearest neighbor weighting functions for interpolation exist in Py-ART. Custom weighting functions as well as variable grid spacing has been requested by users.

5.3 Low Priority

1. **CAPPI support**: Constant Altitude Plan Position Indicator (CAPPI) has been requested from a few users. However it does not seem to be urgent or requested in the ARM space.

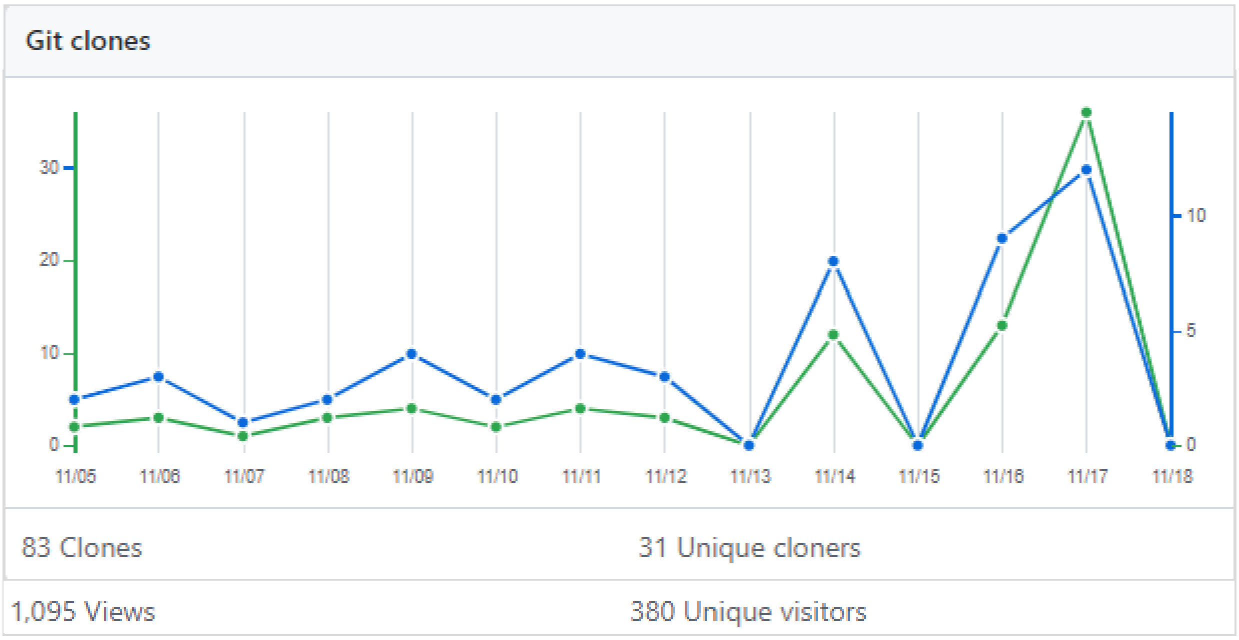
2. **Spectra I/Os**: Spectra input for KAZR was added in 2019 to Py-ART as part of the first roadmap. Current survey results show Spectra dataset readers in Py-ART as a low priority.

3. **More aircraft datasets support**: Increase of inputs and outputs for aircraft radar datasets as well as plotting support.

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 the first 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 12** shows that the main repository is viewed by over 300 unique visitors and in one week the GitHub repository was cloned 31 times. We can see an increase in traffic as compared to the first roadmap in **figure 13**. For the main download source of Py-ART, Anaconda, the downloads were under 100K. Py-ART was introduced to Anaconda later on in development. Py-ART is now a little under 300K downloads as seen in **Figure 14**. 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. There are currently 445 forums on the Py-ART google groups with users asking for help with radar science, Py-ART syntax, bug reports, tutorials and more. Also, the many emails as developers we receive that we target to the google groups or respond to.



83 Clones 31 Unique cloners

1,095 Views 380 Unique visitors

Figure 12: Visitor traffic on GitHub, unique downloads via GitHub as well as unique views.

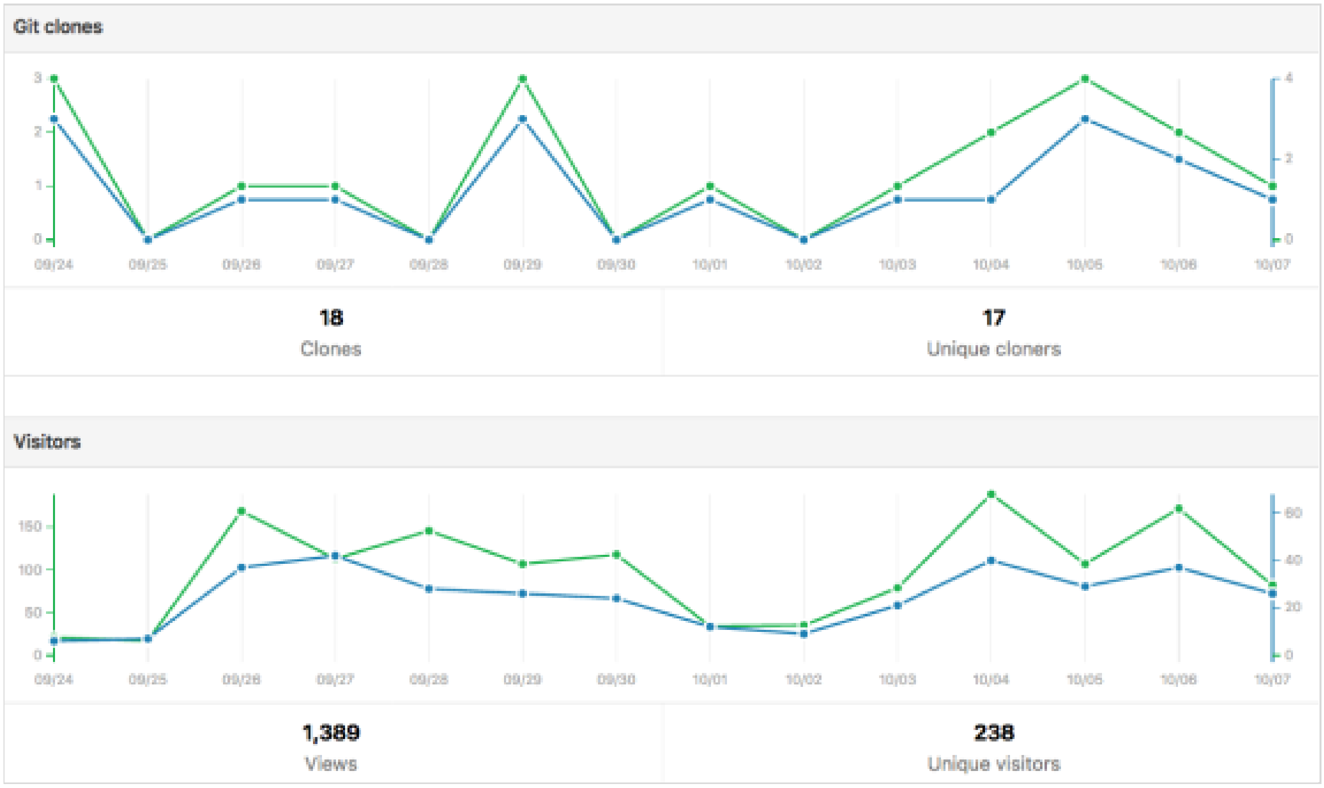
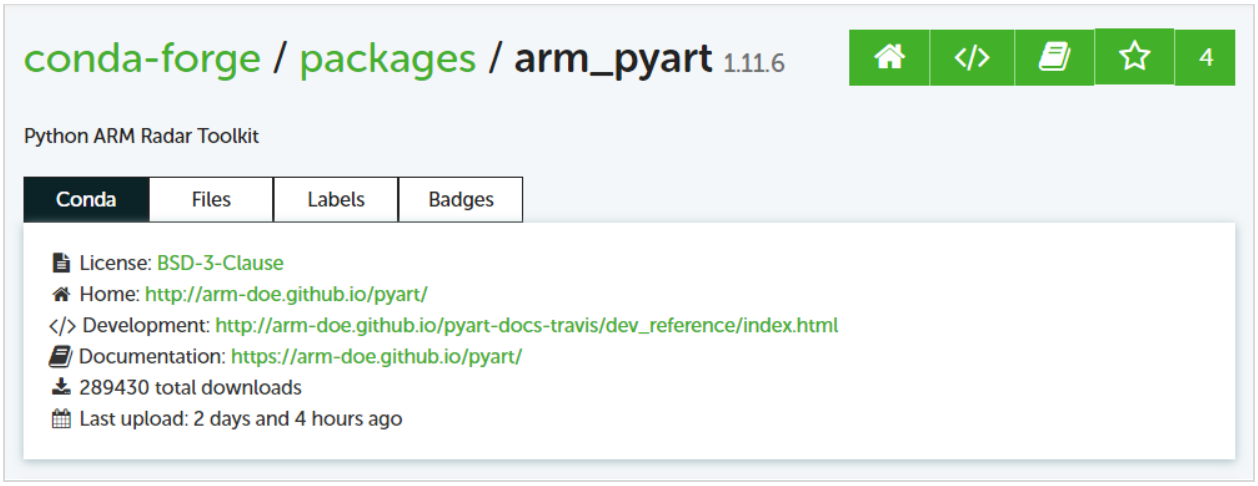


Figure 13: GitHub traffic from the first Py-ART roadmap, similar to figure 14, showing unique visitors and downloads.



conda-forge / packages / arm_pyart 1.11.6

Python ARM Radar Toolkit

Conda | Files | Labels | Badges

License: BSD-3-Clause
Home: http://arm-doe.github.io/pyart/
</> Development: http://arm-doe.github.io/pyart-docs-travis/dev_reference/index.html
Documentation: https://arm-doe.github.io/pyart/
289430 total downloads
Last upload: 2 days and 4 hours ago

Figure 14: Downloads of Py-ART from conda-forge on Anaconda.

2. **Number and success of dependent projects**: Appendix 1 shows projects that require Py-ART as a dependency. Py-ART needs to have a clearly defined scope and the needs of the community outside of this scope are better served with new packages. This was seen

in the first roadmap for both cell tracking and multi-doppler retrievals. 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 218 times. This is an increase from just 10 during the writing of the first 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 (via the Py-ART email list, Facebook page, and Twitter feed) users to self-report so we can build a Py-ART publications database. Py-ART was also recently used by The New York Times as a dependency for plotting forest fire plumes.

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: Packages that use Py-ART as a dependency

ACT https://github.com/ARM-DOE/ACT The Atmospheric data Community Toolkit (ACT) is an open source Python toolkit for working with atmospheric time-series datasets of varying dimensions. The toolkit has functions for every part of the scientific process; discovery, IO, quality control, corrections, retrievals, visualization, and analysis. It is a community platform for sharing code with the goal of reducing duplication of effort and better connecting the science community with programs such as the Atmospheric Radiation Measurement (ARM) User Facility.

ARTView https://github.com/nguy/artview ARTview is an interactive GUI viewer that is built on top of the Py-ART toolkit. It allows one to easily scroll through a directory of weather radar data files and visualize the data. All file types available in Py-ART can be opened with the ARTview browser. You can interact with data files through "Plugins". Many functions from the Py-ART package can be selected. In addition, ARTview plugins allow querying data by selecting regions or points visually.

SingleDop https://github.com/nasa/SingleDop SingleDop is a software module, written in the Python programming language, that will retrieve two-dimensional low-level winds from either real or simulated Doppler radar data. It mimics the functionality of the algorithm described in the following reference: - Xu et al., 2006: Background error covariance functions for vector wind analyses using Doppler-radar radial-velocity observations. Q. J. R. Meteorol. Soc., 132, 2887-2904.

The interface is simplified to a single line of code in the end user's Python scripts, making implementation of the algorithm in their research analyses very easy. The software package also interfaces well with other open source radar packages, such as the [Python ARM Radar Toolkit (Py-ART)](https://github.com/ARM-DOE/pyart). Simple visualization (including vector and contour plots) and save/load routines (to preserve analysis results) are also provided.

PyTDA https://github.com/nasa/PyTDA software providing Python functions that will estimate turbulence from Doppler radar data. It is tested and working under Python 2.7 and 3.4. DualPol https://github.com/nasa/DualPol This is an object-oriented Python module that facilitates precipitation retrievals (e.g., hydrometeor type, precipitation rate, precipitation mass, particle size distribution information) from polarimetric radar data. It leverages existing open source radar software packages to perform all-in-one QC and retrievals that are then easily visualized or saved using existing software.

CSU Radar Tools https://github.com/CSU-Radarmet/CSU_RadarTools Python tools for polarimetric radar retrievals.
PyDDA https://github.com/openradar/PyDDA This software is designed to retrieve wind kinematics (u,v,w) in precipitation storm systems from one or more Doppler weather radars using three dimensional data assimilation. Other constraints, including background fields (eg reanalysis) can be added.

PyRAD https://github.com/MeteoSwiss/pyrad Pyrad is a real-time data processing framework developed by MeteoSwiss and MeteoFrance. The framework is aimed at processing and visualizing polar data from individual weather radars as well as composite Cartesian products both off-line and in real time. It is written in the Python language.

TINT (TINT Is Not TITAN) https://github.com/openradar/TINT TINT (TINT is not TITAN) is an easy-to-use storm cell tracking package based on the TITAN methodology by Dixon and Wiener.

CMAC 2.0 (Corrected Moments in Antenna Coordinates)
https://github.com/zssherman/cmac2.0 CMAC 2.0 (Corrected Moments in Antenna Coordinates version 2) is a set of algorithms and code that does corrections to Radar data, but also adds fields to the original data. Using fuzzy logic CMAC also calculates gate IDs such as rain, snow and second-trip.

PyHail https://github.com/joshua-wx/PyHail This toolkit provides a collection of hail retrieval techniques for weather radar data using the Py-ART toolkit.

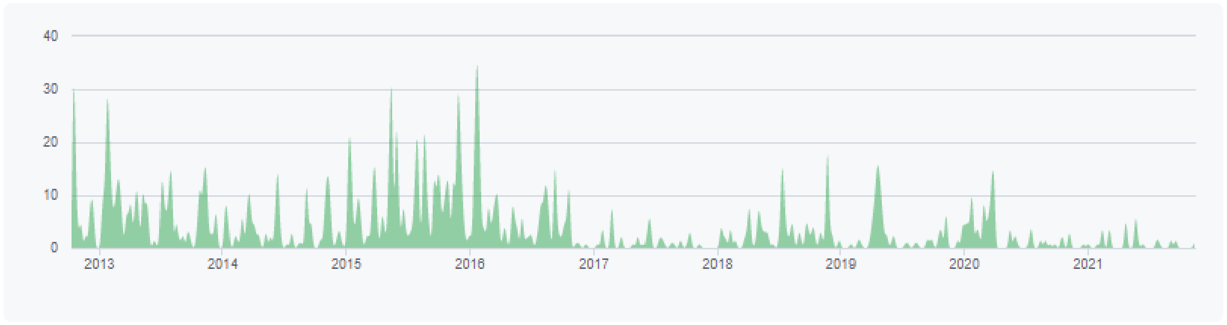
PyBlock https://github.com/nasa/PyBlock PyBlock is a Python 2 or 3 module that enables the end user to estimate partial beam blockage using polarimetric radar data.

Appendix 2: Contributors to Py-ART

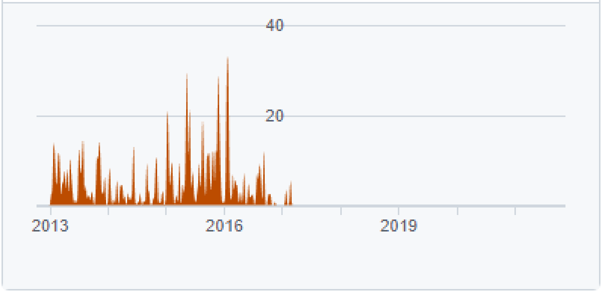
Oct 14, 2012 – Nov 18, 2021

Contributions: **Commits**

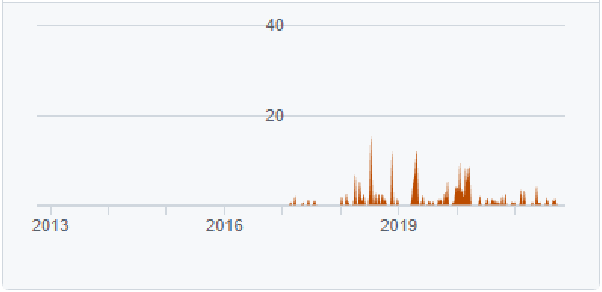
Contributions to main, excluding merge commits and bot accounts



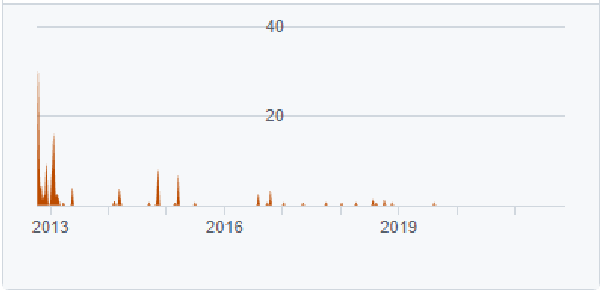
jjhelmus #1
1,061 commits 510,212 ++ 245,479 --



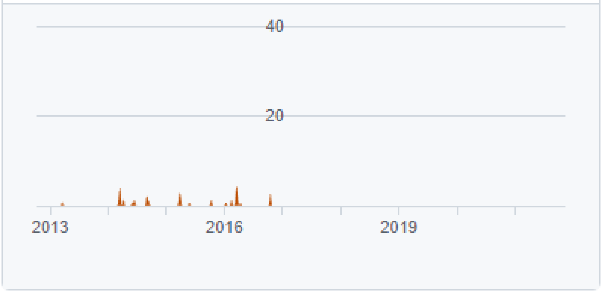
zssherman #2
298 commits 49,828 ++ 46,785 --

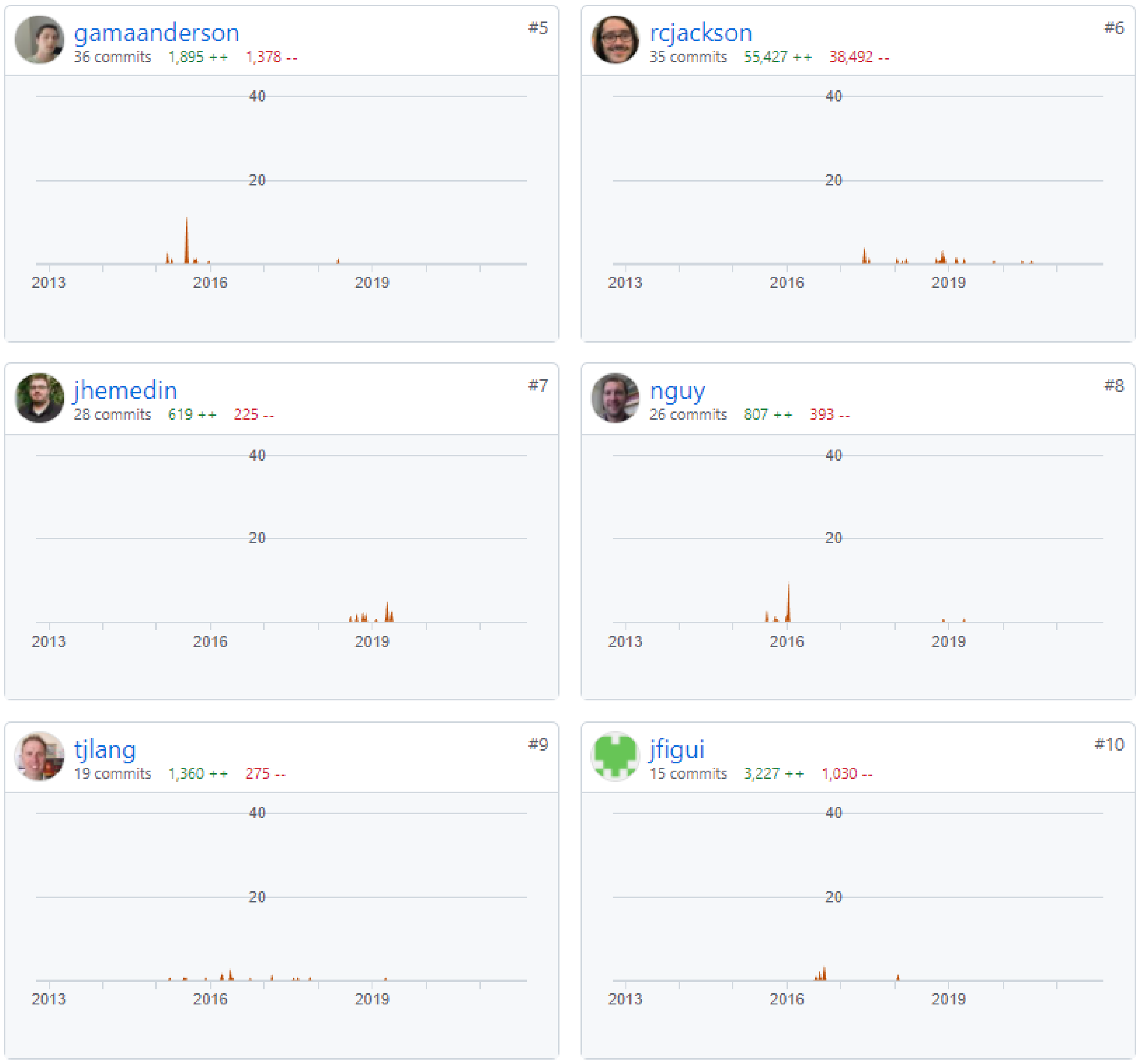


scollis #3
203 commits 14,951 ++ 5,823 --



kirknorth #4
45 commits 20,351 ++ 19,261 --





gamaanderson #5
36 commits 1,895 ++ 1,378 --

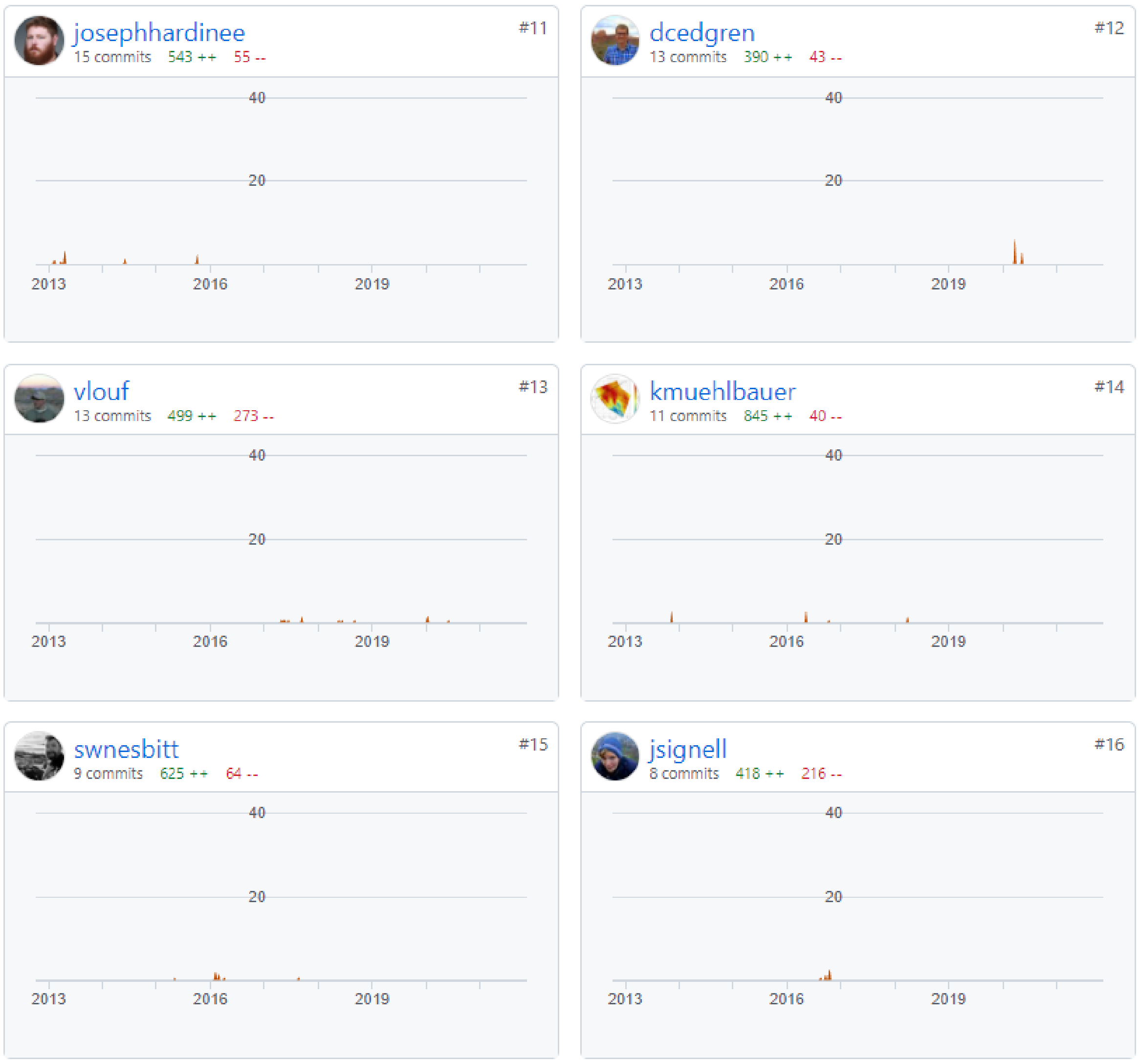
rcjackson #6
35 commits 55,427 ++ 38,492 --

jhemedin #7
28 commits 619 ++ 225 --

nguy #8
26 commits 807 ++ 393 --

tjlang #9
19 commits 1,360 ++ 275 --

jfigui #10
15 commits 3,227 ++ 1,030 --



josephhardinee #11
15 commits 543 ++ 55 --

dcedgren #12
13 commits 390 ++ 43 --

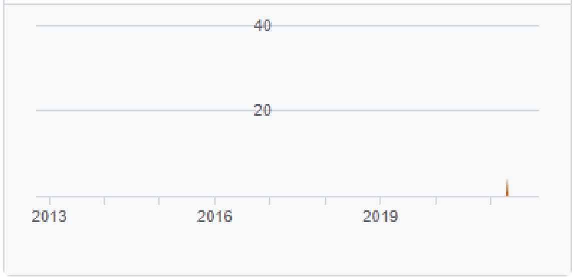
vlouf #13
13 commits 499 ++ 273 --

kmuehlbauer #14
11 commits 845 ++ 40 --

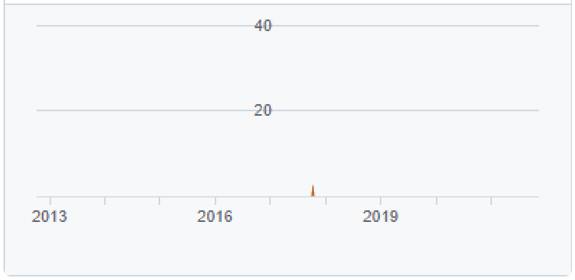
swnesbitt #15
9 commits 625 ++ 64 --

jsignell #16
8 commits 418 ++ 216 --

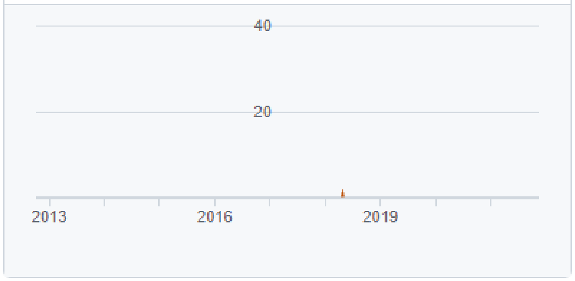
Quba1 #17
6 commits 13 ++ 4 --



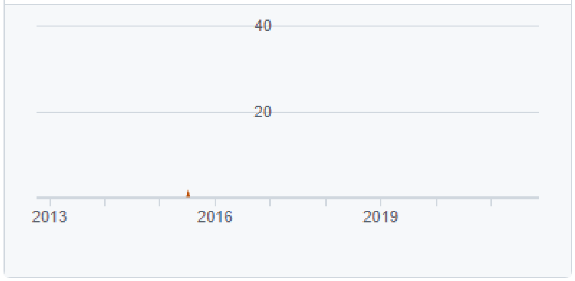
freemansw1 #18
4 commits 51 ++ 79 --



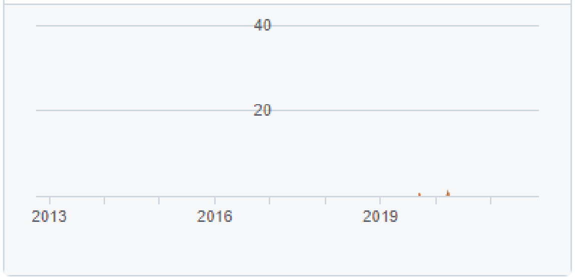
wave100 #19
4 commits 6 ++ 6 --



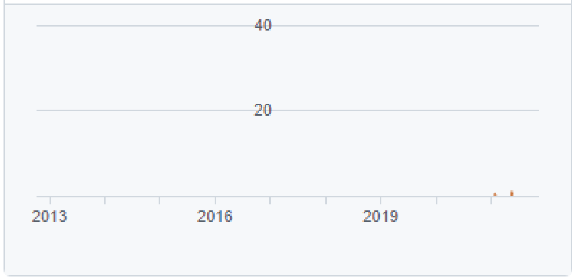
deeplycloudy #20
3 commits 2,500 ++ 2,444 --

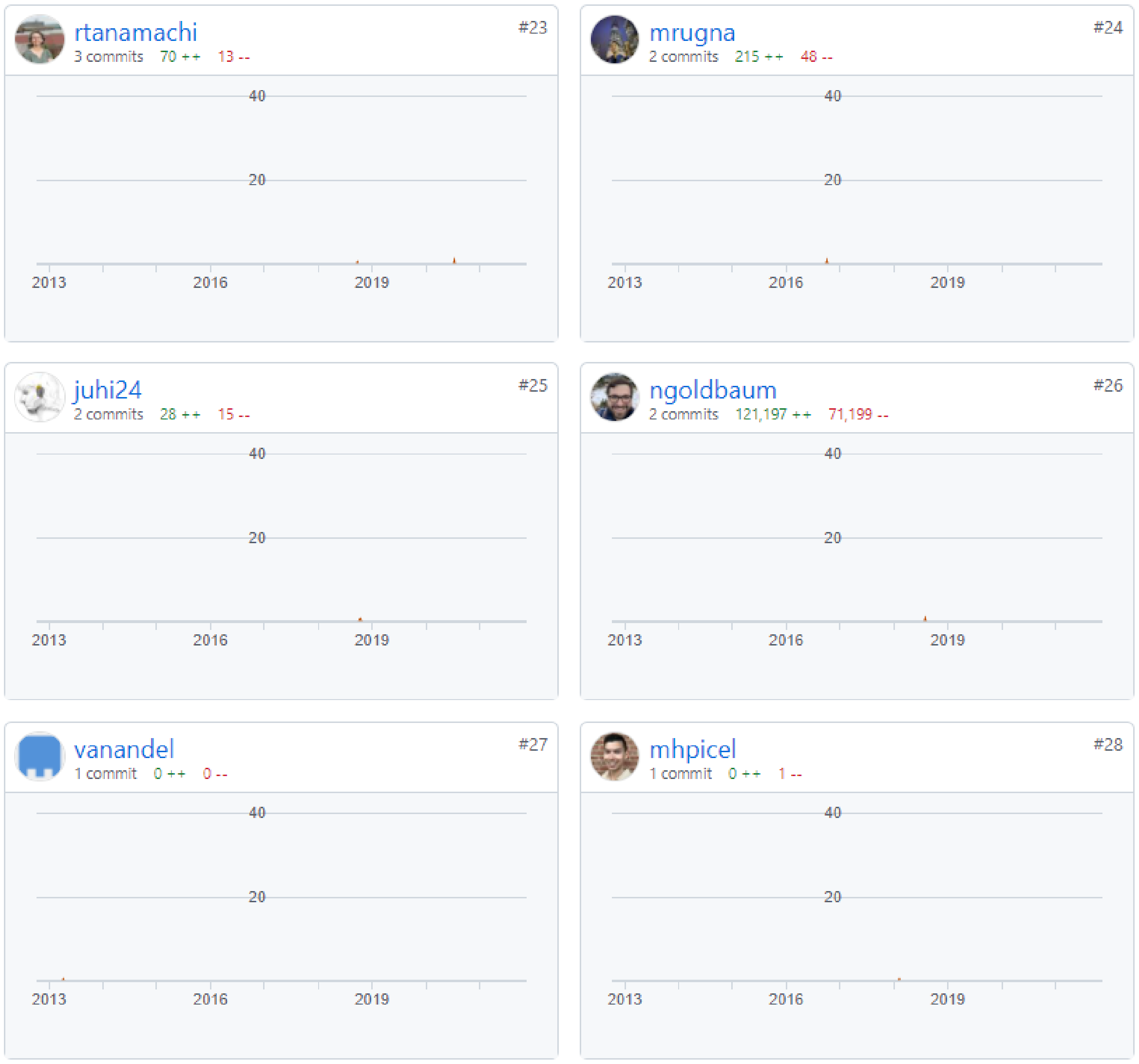


AdamTheisen #21
3 commits 164 ++ 5 --



HumphreysCarter #22
3 commits 36 ++ 34 --





rtanamachi #23
3 commits 70 ++ 13 --

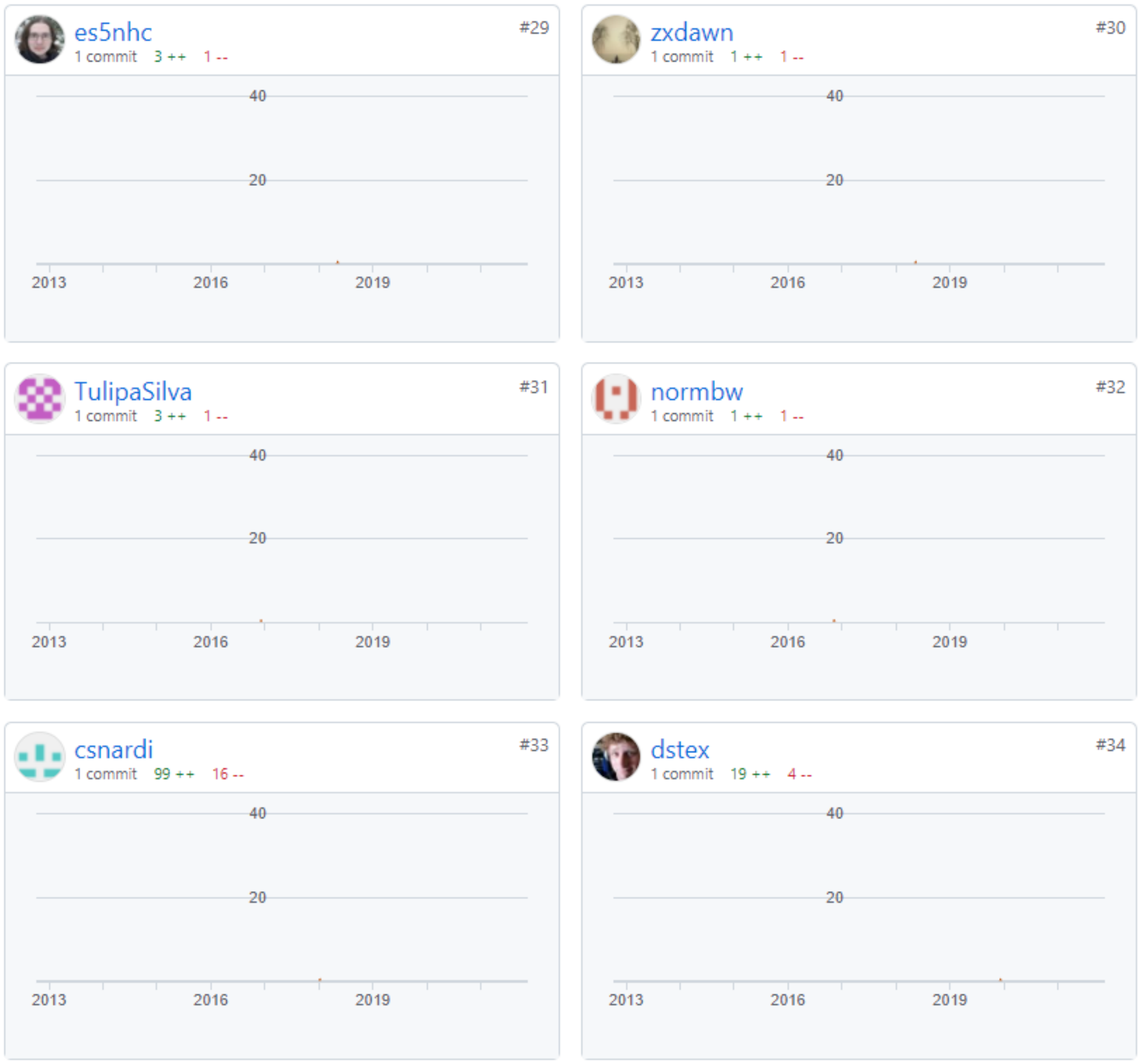
mrugna #24
2 commits 215 ++ 48 --

juhi24 #25
2 commits 28 ++ 15 --

ngoldbaum #26
2 commits 121,197 ++ 71,199 --

vanandel #27
1 commit 0 ++ 0 --

mhpicel #28
1 commit 0 ++ 1 --



es5nhc #29
1 commit 3 ++ 1 --

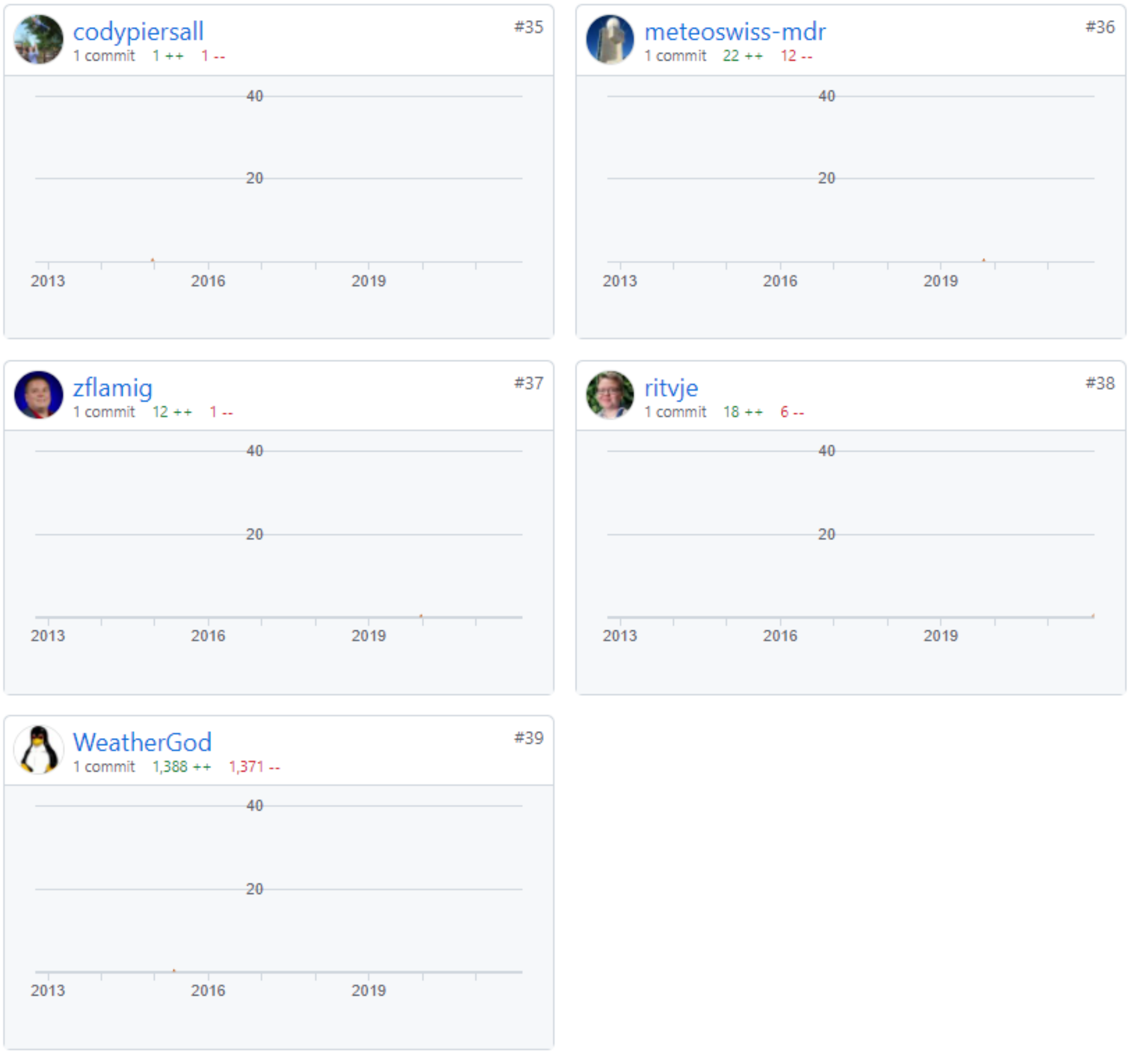
zxdawn #30
1 commit 1 ++ 1 --

TulipaSilva #31
1 commit 3 ++ 1 --

normbw #32
1 commit 1 ++ 1 --

csnardi #33
1 commit 99 ++ 16 --

dstex #34
1 commit 19 ++ 4 --



codypiersall #35
1 commit 1 ++ 1 --

meteoswiss-mdr #36
1 commit 22 ++ 12 --

zflamig #37
1 commit 12 ++ 1 --

ritvje #38
1 commit 18 ++ 6 --

WeatherGod #39
1 commit 1,388 ++ 1,371 --