



PyDSD, PARSQUANTS, and experience transitioning open source code into an ARM VAP

June 13, 2019

**Joseph C. Hardin¹, Scott Giangrande²,
Aifang Zhou², Die Wang²**

Pacific Northwest National Laboratory¹ Brookhaven National Laboratory²

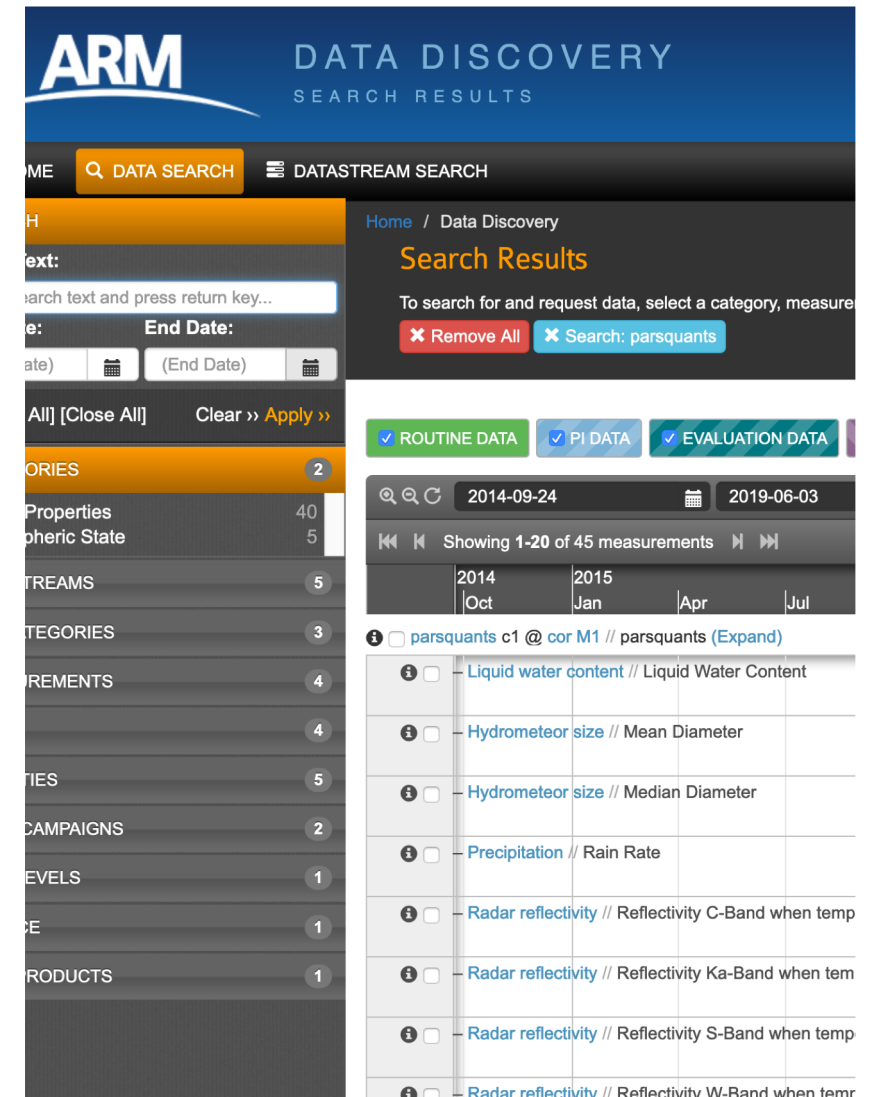
Acknowledgement: Mary Jane Bartholamew for serving as the disdrometer mentor and offering advice on ARM disdrometers.



PNNL is operated by Battelle for the U.S. Department of Energy

Introduction

- Over the last year, Scott Giangrande, Aifang Zhou, and I have taken an open source library (PyDSD) and transitioned it into an operational ARM VAP that runs at multiple ARM sites
- I want to discuss the process, how ARM benefits, and more importantly how I (and thus you as a scientist) benefit.
- I will present some key steps along the way
- Finally I will offer up some general advice for transitioning code into ARM, but more generally between organizations.



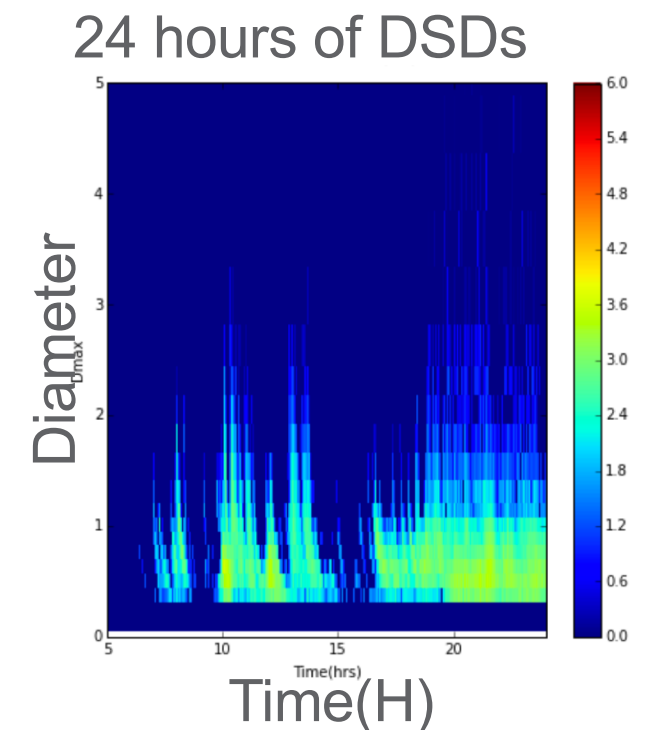
The screenshot shows the ARM Data Discovery Search Results interface. The top navigation bar includes the ARM logo and the text "DATA DISCOVERY SEARCH RESULTS". Below this, there are search filters for "DATA SEARCH" and "DATASTREAM SEARCH". The main content area displays search results for "parsquants". On the left, a sidebar lists categories such as "Properties", "Streams", "Categories", "Measurements", "Sites", "Campaigns", "Levels", "Phase", and "Products" with corresponding counts. The main results area shows a table of search results with columns for "Liquid water content", "Hydrometeor size", "Precipitation", and "Radar reflectivity". The interface also includes a date range selector (2014-09-24 to 2019-06-03) and a "Showing 1-20 of 45 measurements" indicator.

Disdrometers

- Disdrometers are instruments designed to measure the drop size distribution of hydrometeors as they pass through a collection area.
- Utilize a variety of measuring technologies (laser, video, impact, etc)
- Generally they measure a 2d matrix of speed vs size.
- If we collapse the speed dimension we get a binned time series of DSDs.
- ARM hosts a series of disdrometers at all of its sites.
- There are data quality issues however with all disdrometers.



Parsivel Disdrometer



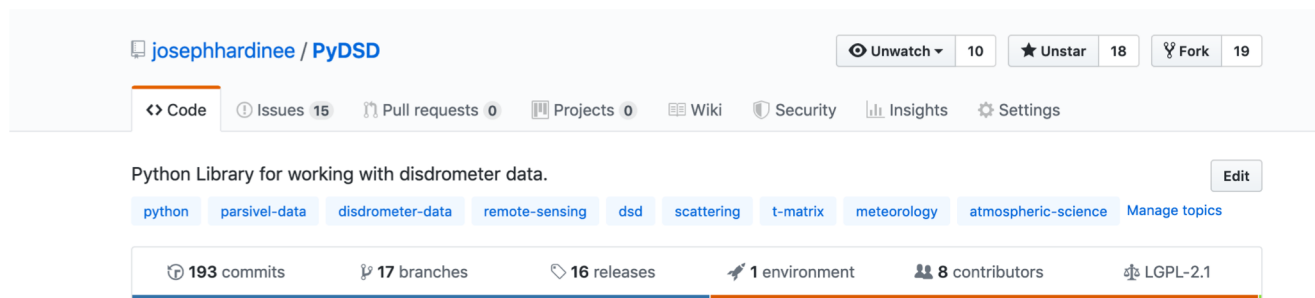
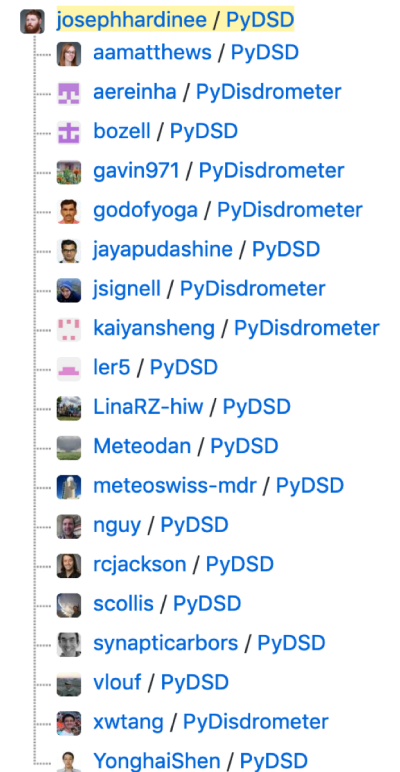
Disdrometer Comparison issues

- The first issue is unrealistic particles.
 - Bugs
 - Dripping from the hoods
 - Double bounce raindrops
- These can be filtered out by using their fall speed (50% terminal fall speed filter)
- The second issue is the lack of comparability.
 - Remote sensing does not retrieve a binned DSD
 - Models (usually) don't use a binned DSD as well.
- Most representations of DSDs are parameterized
 - Primarily as either exponential, or normalized gamma distributions.
- Comparisons with radar would often prefer to be in radar measurement space.

Phase 0: PyDSD

Step 0: Have existing (preferably coded) algorithms. Ideally vetted in a journal.

- PyDSD is a open source Python library for processing disdrometer and particle probe data.
 - Supports disdrometers of several varieties as well as several different particle probes.
 - Data quality routines
 - DSD estimation routines
 - Radar scattering support (via PyTMatrix) for a variety of frequencies/temperatures and microphysical parameters.
 - Plotting support
- Cited in several papers using ARM data.
- 40000+ Downloads(5K conda, 35K PyPI) *



*: Take this number with a huge grain of salt, real numbers are probably about 1% of this due to mirrors.

Phase 1: Conceptualization

Step 1: Decide on form of VAP.
Decide on role of science sponsor.
Decide on whether it is just ARM
integration, or whether new
algorithms are needed.

- Scientists were using PyDSD to do analysis, often doing the same steps
 - Filter data -> (Scatter at radar frequency, estimate DSD parameters, plot)
- Scott Giangrande (translator) and I brainstormed that we should do the initial leg work for scientists and release those products calculated for each of the ARM radars.
- This helps
 - ARM (more high-quality VAPS = quicker better science)
 - the library (More coverage and users)
 - **me** (Science sponsor) with more citations.
- We laid out the plans on what would be included, whether the existing library had all the features needed, and generally what distribution of work looked like.
 - A few new features were needed to make it easier, Scott's team would develop ARM side, and I would support as science sponsor.

PARSQUANTS

- Behind the scenes the translator does a few things
- Submits an ENG (Work order) to track the work and get approval
- Assigns a developer
- Allocates budget for the development (Through a yearly process)
- Starts an VAP (workflow ticket) that tracks the individual infrastructure components of a VAP development and release workflow.

ARM Release Process: Test Files

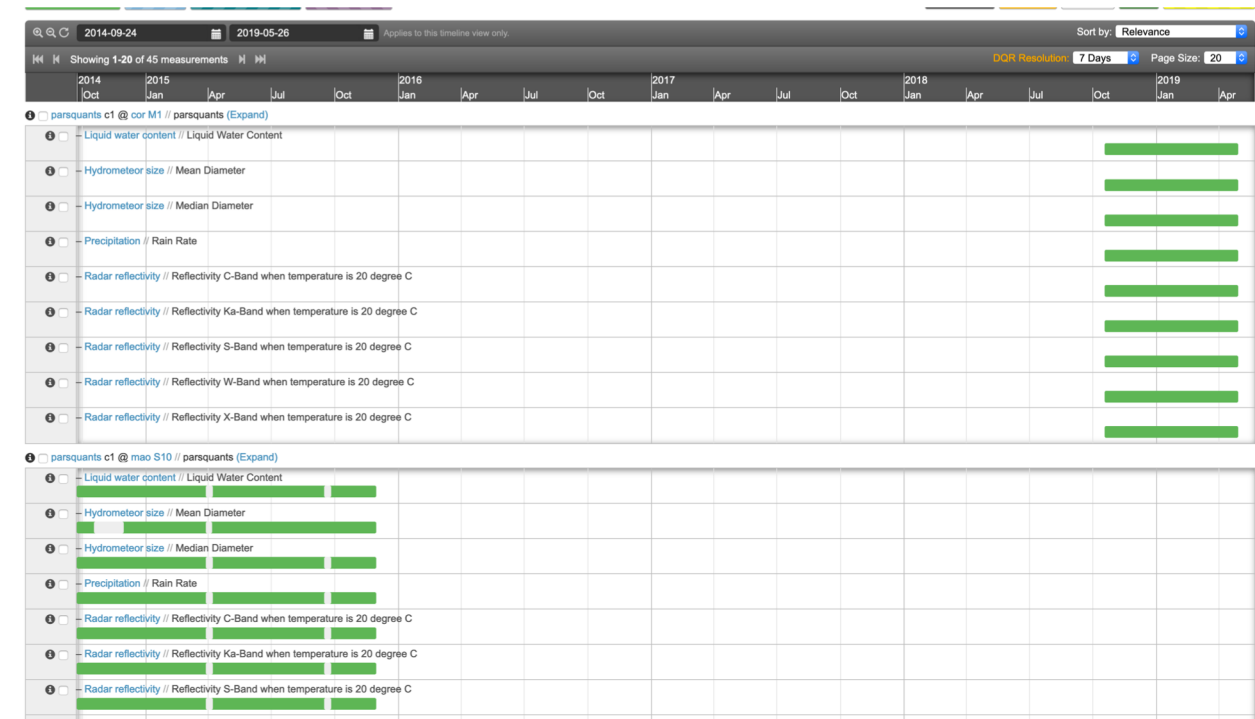
- A good starting point is for the science sponsor to generate some test files. These form a benchmark for discussion with the developer.
- If developers are re-implementing the algorithm, allows for verification of accuracy.
- Test files should span the planned implementation regimes (Don't just run one region).
- Assuming that because the code works in one place, the developer can just handle the “edge cases” in other places will not work. They are not the domain expert.

ARM Release Process: Development

- The developer will port the algorithm into ARM's infrastructure, and if needed, recode it into a usable (by ARM) language.
- This is where most of the back and forth is (In our case, about 200 e-mails and a site visit).
- As they hit edge cases, the science sponsor should help address deficiencies in the algorithm or implementation.
- Even if your algorithms are in a library, there can be back and forth improvements needed, or wrapper scripts to make internals more accessible. The developer can sometimes do this, but much more efficient to help them out.
- Use this as an opportunity to refactor your code and improve it.

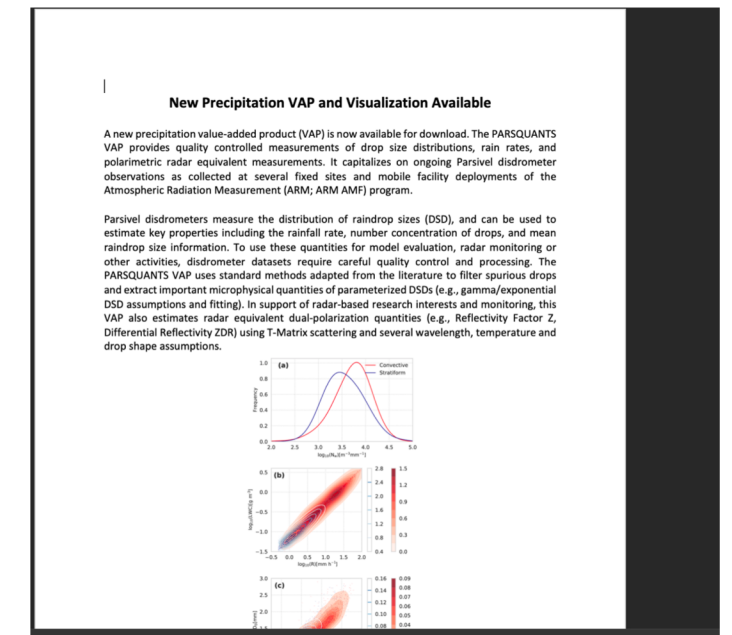
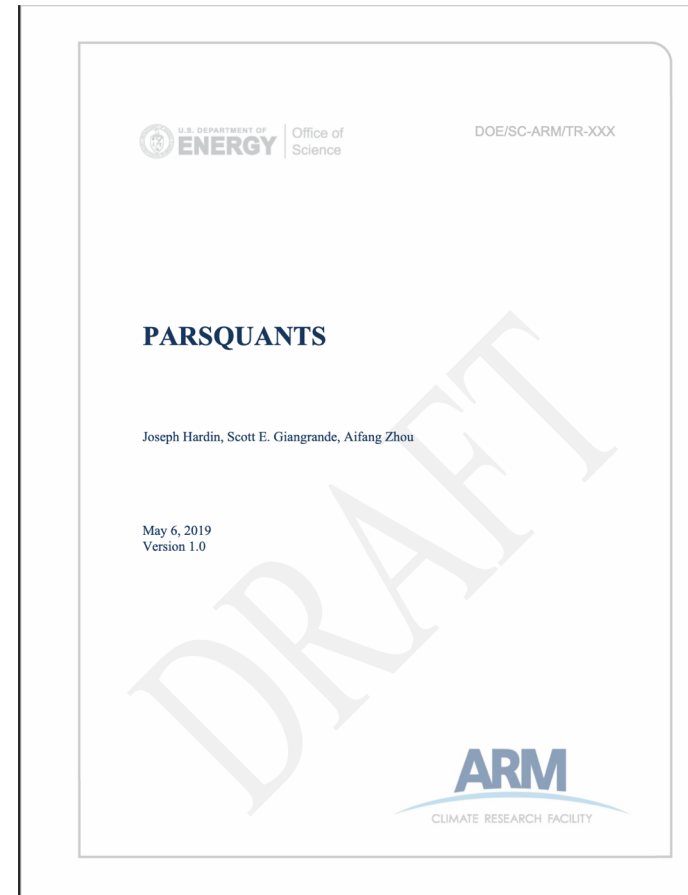
ARM Release Process: Reference files

- Once development is completed, you'll be asked to approve reference files.
- These are the results of running the operational VAP on a test data set.
- Agreement between you and translator that this is the output expected.
- Also serves to allow updates to code in future, while ensuring output does not (silently) change.
- Verify that the output is correct (Both data values, and metadata)
- Once this is done, your work is mostly completed.



ARM Release process: Approvals

- At this point there are many behind the scenes activities that ARM requires.
- Approvals of:
 - Metadata
 - QC flags
 - Test runs
- Technical Report for the VAP that you will be asked to review
- Data announcement with comms staff.
- Data uploaded to archive
- Hopefully automated processing turned on.



Advice for the transition

- Common Formats
 - Use common data formats (preferably netCDF4)
- Choice of Language
 - Languages that ARM ADI can use directly are Python, C, IDL, Matlab (Roughly in that order of preference)
 - If Fortran, it can be wrapped in Python
- "Seams"
 - Provide places to tap into code, not one factory method.
- Multiple ways in and out
 - The VAP code will not directly open a file.
 - Have alternate methods to construct an internal object to your code.
- If a library, have an easy way of updating the version and installation (conda package, PyPI, rpm, etc)

Advice for the transition

- Handle failures gracefully
 - Emit failure codes, sentinel values, etc.
 - This makes it easier for the developer to capture these and issue QC flags.
- Self documenting metadata is great
 - Self documenting formats and data structures are very helpful.
- Minimize external dependencies when possible
 - Have an easy way to set up a development environment as it must be recreated on ARM servers.
- Avoid pinning dependencies if possible.
 - This is somewhat an anti-pattern, but if conflicting pins happen, there is not currently a way to support this in ARM's infrastructure (Though this is being worked on).
- **Have a clear code license. (BSD, GPL, MIT, etc)**

Practical advice for dealing with ARM developers

- Don't assume the developer is an expert in
 - The instrument
 - The algorithm
 - The data
- Developer is there to transition your algorithms, into the ARM infrastructure.
- Minimize the work done by developer
 - Test with a wide variety of cases
 - Automate where possible
 - 80/20 does not apply to operational VAPS.
- Be explicit. If assumptions in code are made, document those. Don't leave a developer guessing
- If possible, move regime/site specific portions to a config file. This makes it easier to update future campaigns rather than codebases.