NASA/TM–20210023307

PyHC Integration Strategy Workshop Report

Brian A. Thomas, Arnaud Masson, Julie I. Barnum, Aaron Roberts and Reinhard Hans Walter Friedel

October 2021

NASA STI Program ... in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NTRS Registered and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA Programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include organizing and publishing research results, distributing specialized research announcements and feeds, providing information desk and personal search support, and enabling data exchange services.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at http://www.sti.nasa.gov

- E-mail your question to help@sti.nasa.gov

- Phone the NASA STI Information Desk at 757-864-9658

- Write to:
  NASA STI Information Desk
  Mail Stop 148
  NASA Langley Research Center
  Hampton, VA 23681-2199

NASA/TM–20210023307

PyHC Integration Strategy Workshop Report

Brian A. Thomas
Goddard Space Flight Center, Greenbelt, MD

Arnaud Masson
European Space Astronomy Centre, Madrid, Spain

Julie I. Barnum
Colorado State University, Fort Collins, Colorado

Aaron Roberts
Goddard Space Flight Center, Greenbelt, MD

Reinhard Hans Walter Friedel
Los Alamos National Laboratory, Los Alamos, New Mexico

National Aeronautics and
Space Administration

Goddard Space Flight Center
Greenbelt, Maryland 20771

October 2021

Level of Review: This material has been technically reviewed by technical management.

Available from

NASA STI Program
Mail Stop 148
NASA's Langley Research
Center Hampton, VA
23681-2199

National Technical Information
Service 5285 Port Royal Road
Springfield, VA 22161
703-605-6000

PyHC Integration Strategy Workshop Report

ISWAT O2-06 workshop, organised by NASA, Aug 30 - 31st, 2021
Final Version

contact emails : brian.a.thomas@nasa.gov, Arnaud.Masson@esa.int

Table of Contents

1. Introduction/Background

The purpose of the workshop was to draw together key figures in PyHC (Python in Heliophysics Community) to discuss how the community can better advance our efforts to provide support in Python for the heliophysics research community. The objective of the workshop was to identify priorities for technical work in the coming years particularly with regard to a theme of "Integration". One goal was to produce a public report of findings and observations that will eventually enable a strengthening of the Space weather Information architecture, in relation with the ISWAT 02-06 action team. Additionally, this report will be used to develop a (non-public, internal) NASA report for recommended action. The key deliverable from the workshop is this report containing a list of 'key challenges' and associated suggestions for action (projects).

2. Workshop Organization

The workshop was a 2-day (~3 hrs/day) virtual meeting held August 30-31, 2021 with participation from PyHC members across the world (see Appendix A for a participant list). Portions of the deliverable were created on each day, as follows:

Day 1 (August 30): Identify challenges to developing heliophysics software
Day 2 (August 31): How to solve challenges, suggested actions

A full agenda can be found in Appendix B.

3. Results of the Workshop

The gathered key challenges and associated suggested solutions appear in this section. We have tried to capture the full range of opinion from the participants in describing these solutions. These results are presented in no particular order of priority.

1. Challenge: Sustainability of Projects/Software (people, dollars)

    1.1. Have PyHC projects leverage NumFocus

    NumFocus is a non-profit organization intended to support sustainable open source science software. A partnership between NumFocus and PyHC projects could provide increased visibility for a project which could be leveraged to expand the user base as well as open the range of potential funding options, potentially including those outside standard government channels. A package would likely require functionality that is applicable to fields outside space science to get additional outside funds. One way for packages to solve this would be for them to become part of a larger, successful project

(e.g., a mission or missions). Each package would need to apply and meet NumFocus standards.

1.2. Funded position(s) dedicated to Continuous Integration (CI) and Continuous Deployment (CD) issues of PyHC projects.

This would be a dedicated PyHC Developer who would work on installation methods (binary wheels, pip install, conda-forge), implementing integration tests, managing documentation infrastructure, performing code reviews, fixing bugs, and adding new requested features to packages. There would be a focus in the role on both project maintenance *and* improvements. The PyHC Developer would ideally be someone who's active in the field of heliophysics and gets a salary to do the proposed work.

This technically falls under the role of the PyHC Technical Lead, but has not been able to be adequately implemented in practice because the Technical Lead has experienced project leadership overhead such that digging into the code has been difficult. Namely, the current Technical Lead is not actively involved in the field of heliophysics research, and thus has run into issues therein. This speaks to the benefit of the above-mentioned dedicated PyHC developer who *is* active in the field of heliophysics and receives a salary to improve code, fix GitHub issues, etc.

1.3. Template repository for package onboarding.

The creation of a Git repo with boilerplate/filler content that resembles what a well-structured PyHC repo should look like. This could help existing projects conform to our standards and help newcomers make good repos. Work would include both technical layout and developing consensus within the PyHC community on what should be part of this template repo.

1.4. Establish different funding levels and mechanisms in PyHC for projects at different levels of maturity

Establish within PyHC several funding profiles for projects. This would be implemented in three tiers:

1. Yearly Announcement of Opportunity (AO): An initial, low-level funded grant for one year to prove a concept or get a package initially connected to PyHC or to do work for common good of all projects such as a common data model effort. This is a grant and is a small amount of funding to demonstrate that it will work, (on the order of $50K - $75K). Many projects could be funded by this.

2. Second-year effort proposals: A larger grant for two to three years to establish and solidify a project and make it stable and maintainable. These would be awarded to projects that have proven themselves via the first year efforts, or that

are somehow already more established in the community. Fewer projects are selected at this level, since the entry requirements are higher (e.g., demonstrated adherence to PyHC standards, a threshold number of users and/or developers using / working on the package, tutorials or at least a getting started guide).

3. Infrastructure projects: A lower level of baseline maintenance funding for 5 years as a contract. The benefit to PyHC from using contracts is that they have the potential to provide greater funding stability for core projects. Note that since NASA is making a commitment to a project, the project needs to also make a commitment on deliverables and responsibilities. These would include project maintenance (e.g., automated, rigorous test code, a regular release schedule happening at least once per year), user support (e.g., a site for users to submit bug reports, an user guide updated on at least a yearly basis, giving tutorials at least twice per year at a relevant science conference and also online), and collaboration with instrument teams (could also involve tutorials). A comparable number of these projects as for the second level will be funded.

1.5. Offer pathways and incentives for users of PyHC packages to transition to developers of those packages.

Incorporate a funding model for projects to source additional developers on PyHC packages, to the end of eliminating a reliance on a core group of people ad infinitum.

This could be done one of two ways (or both):

1. Target junior developers. This would be framed as a trade-off; a developer joins a project and contributes code, and project leadership would teach that developer how to run a software project. This could be implemented in 3-6 month stints of development time to allow the developer to determine if they enjoy the work. Hence, there should be a fellowship of sorts to offer funding in exchange for both learning to code via working on a project's code base. The work would be geared towards undergrad, or more likely, graduate students. This could also be an REU or internship program to let undergraduates try out scientific software development.

2. Create a "Visiting Scientist" (or "Visiting Scientific Developer") position for additions/changes to packages. This would get them a substantial amount of additional funding (~3-6 months) to devote to not just incorporating their code into the library, but to also help out with other ongoing projects within that library. This would require projects to have lists of current coding or design needs, so that applicants to the Visiting Scientist position for a project could know about, get excited about, and respond to the currently expressed needs of that project. For example, if SunPy had two of these at a time continuously over the next 5 years, it would have a transformative effect on the library, adding many vested

developers into the project. For people outside the US such as Europe or Japan, this could cover 3 months mission costs. This would enable smart but rather limited Python initiatives to be integrated in larger PyHC libraries like SunPy for instance. (Often, the undergraduates or grad students are the ones developing the code for a mission, and may also be attracted to this.)

2. Challenge: How to get new developers

2.1. Collect a master list of known issues and desired additions on PyHC's website to provide a list of possible contributions for new developers.

For example, a 'help wanted' label next to a package name in the main table in addition to the master list, or a 'help wanted' section on the PyHC website. Also need a way for new developers to add a proposed addition and let package leads 'claim' the project or provide recommendations on the next steps. Presumably scraped from a GitHub issues label.

2.2. Hold a week-long summer school that covers the skills necessary to contribute to an open source project.

The summer school would cover essential research software engineering skills such as git and GitHub; Python packaging; developing documentation using Sphinx, reStructuredText, and Read the Docs; continuous integration testing; and introductions to core PyHC packages. The content for the course could be adapted from past events, such as the URSSI Winter School in Research Software Engineering and Plasma Hack Week.

This item is heavily related to items 3.6 "Hold regular tutorials" and 3.7 "Software Carpentry Workshops" below.

2.3. Create a dedicated, permanent role for a PyHC Onboarding Scientist.

PyHC should create a permanent, "PyHC Onboarding Scientist" position. This person would sponsor events such as the following:

- Regular tutorials for PyHC projects at conferences.
- Could assist with the PyHC Summer School idea as a result of familiarity with tutorials.
- Software Carpentry workshops for members of the heliophysics community who are unfamiliar with Python (they don't have to run these, just orchestrate them).

- Assist with creating and curating effective templates for new projects and help newly selected PyHC developers get into an early pattern of good development (see item 1.3 above)

- Hack sessions to incorporate legacy code in other languages into a python package (likely quite useful for code from other missions).

2.4. Provide credit for software via citations.

Implement measures to expand PyHC's presence in publications (i.e., increase PyHC software tool citation). The importance to PyHC of this is four-fold: 1) those interested in a career in science need scientific currency to further that career (i.e., citable work that is also cited in peer reviewed publications, not just potentially citable), 2) this gives PyHC packages an opportunity to advertise their tools, 3) it advertises PyHC, driving traffic to our community (which would ideally then increase PyHC package usage), and 4) this would demonstrate to scientists, with data, that software has a role in the scientific record.

One of the most straightforward methods to crediting PyHC software is to publish papers on the software. For example, there are a few recent publications on PyHC tools (e.g., a pysat-specific paper, "PYSAT: Python Satellite Data Analysis Toolkit", as well as a "Snakes on a Spaceship" paper, "Snakes on a Spaceship—An Overview of Python in Heliophysics"). However, steps should be taken to increase citation instances. One way to achieve this goal is to collect several PyHC manuscripts and submit them for publication simultaneously. This is done to increase publication likelihood via a "strength in numbers" approach. If enough packages participated, this could become a special issue (e.g., in JOSS or JGR Space Physics). This approach would benefit from considering international collaboration (e.g., consider bringing in IHDEA). Conversely, as opposed to submitting several individual manuscripts, they could be combined into one larger, overall publication (such as the above-mentioned "Snakes on a Spaceship" paper). Once this is successfully done, pushing through publications on PyHC project tools would become easier since they would be associated with that first round of a large number of PyHC publications (this process has the added benefit of increased advertising for PyHC). Something to consider with respect to the above papers is incorporating the idea of an "executable research article". This kind of article contains all of the data and code which can be executed to provide plots and analysis of outputs.

In addition to the paper submission ideas above, the community should keep in mind that:

- DOIs should be attributed to each PyHC python library.
    - Infrastructure is already in place (Zenodo, ORCID, ...)

        - These DOIs should be mentioned in outreach articles and on PyHC's website (via the addition of a Publications web page).

    - With respect to the above suggested PyHC-tool specific publication ideas, PyHC also needs to get scientific papers published that cite software.

        - PyHC could start by making connections with scientists and offering support in their research use of the package. This idea dovetails well with the creation of a PyHC Liaison position (see section 4.4).

    - PyHC needs to communicate with editors in chief of major heliophysics journals about the need to cite DOI related to software libraries, instead of software names with a website.

        - This of course starts with the first bullet point above; PyHC packages need to utilize DOIs and PyHC needs to make these DOIs easily available on the PyHC website.

2.5. Maintain a packaging guide for PyHC developers.

When developing a new project, there are certain tasks related to packaging, testing, and documentation that are needed in order to turn it into an actual package. This effort could involve participating in an existing guide, such as the OpenAstronomy packaging guide. This is a sister project to "Template repository for Package Onboarding" (section 1.3).

3. Challenge: How to get new users involved

3.1. Enhance and Expand PyHC Website

Enhance and expand functionality and documentation on the PyHC website (heliopython.org) to showcase the vibrant PyHC community. Items to tackle include:

- Develop a reference presentation on PyHC to be shown on the front page of the PyHC website.
- Develop a slide deck covering PyHC, which makes it easy for community members to advertise PyHC via the ability to provide high quality slides.
- Develop "Quick Start Guides" for PyHC projects.
- Post video presentations/demos and tutorials on PyHC packages. This makes it easy for people to get started with all of the packages within PyHC. These should be easily available from the PyHC website (perhaps on the package page). Can start with the videos from 2021 recorded telecons.
- Host working examples on the PyHC Gallery page.

- Implement functionality search feature for users to find packages. This should link to the package *and* to demos of that functionality. Can have pointers to videos at specific time for this in addition to or in place of a tutorial. Could also point to papers and sections of papers, if appropriate.

These would show useful use cases, as well as demonstrate functionality of our packages. Bonus if we can do something easily using our tools that would be very hard otherwise. Jupyter notebooks, which are backed by BinderHub, would be a desirable means of capturing examples.

3.2. Improve Documentation within all projects

Improve documentation within the projects to support new users, advanced users, and the transition to development. Developers would be responsible for this effort, however, the community would need to develop a standard for documentation and a few common documentation styles, which give similar results, should be employed (e.g. Sphinx). The documentation standards would be focused more so on the actual content than style.

3.3. Present software in science sections (of conferences), not just informatics/software sections

Presentation of technical work, particularly working applications, can be an effective means of outreach to the user community. By presenting in the science sections, one can reach non-developers who often are not aware of (easily) available tools they might be interested in and who won't necessarily attend software sections more aimed at developers.

3.4. Position for Community Engagement officer

Create a Community Engagement Officer position in PyHC (or split up amongst interested parties). This person would advertise events, new features, tutorials at conferences, etc. to the wider community. Duties could include:

1. Tweet about current events.
2. Post in community newsletters (link to PyHC blog on website), e.g. SPA newsletter and SolarNews.
3. Solicit blog posts from community members - i.e., get projects to write about new things that are happening!
4. "This Week in PyHC" on the blog, for example (could also be monthly)
5. Highlight new features, spotlight packages in rotation, summary in plain English of new/useful tutorials, etc (e.g. https://sunpy.org/project/roles.html#communication-and-education-lead).
6. Entice users with tutorials using multiple PyHC packages to achieve an analysis/science goal (posted on the PyHC Gallery page).

3.5. Hold regular tutorials for PyHC projects at conferences.

We should hold regular tutorials for PyHC at conferences to continuously remind others of our efforts and increase visibility of PyHC packages. The tutorials would likely last a few hours each, and could cover either a single PyHC package or how to use multiple PyHC packages in conjunction with each other. The initial setup of these tutorials would likely require some time devoted by project teams. Basing these tutorials on existing example notebooks would reduce the amount of preparation time needed for holding one of these events.

3.6. Hold Software Carpentry workshops

These workshops would be for members of the heliophysics community who are unfamiliar with Python.

These two-day workshops cover foundational skills like using the Unix shell, version control with git & GitHub, and programming with Python. The topics covered by these workshops are necessary for both using and contributing to PyHC packages.

3.7. Hire a dedicated PyHC training coordinator.

This staff person would be responsible for organizing training events such as summer schools, conference tutorials, and Software Carpentry workshops.

4. Challenge: Improve engagement with PyHC

4.1. PyHC Social Media and Outreach

Improve engagement within the PyHC through utilizing social media tools including

- Twitter
  - Post examples from packages
  - Provides outreach to wider groups.

- Twitch (e.g., Dan Welling). https://www.twitch.tv/spacecataz1663
  - Streams working on papers. Provides commentary and humor. Has many visual results to work with. Helps ensure Dan works on papers if he schedules a stream.
    - Found a way to perform outreach while getting “real” work done
  - Regularly link to Dan and others who do blogs, streams, etc.
  - Maintain a list on heliopython.org

Improve PyHC outreach as follows

- Creation of a PyHC Newsletter with announcements of new results, project highlights, etc.

- Improve the PyHC blog content (input from developers) including
  - Package examples (in gallery)
  - Package release or other milestone announcements
  - Package introductions/Intro text (if they give an intro presentation, could include link to that)
  - Post standard Developer Questions
    - Standard set of written questions to be answered by every developer.
    - Provides a base level of content to supplement blog as needed

- Help PyHC members get to learn about everyone in the community. Being a community requires a level of regular interaction. New users will have a chance to directly learn about the different people working in PyHC. The more people we cover the more likely it is that a new user (or PyHC 'developer') will find some commonality with another person in the community. PyHC Mixers? If we were all local we could have mixer events and just interact socially. Perhaps a need to do the same online? Monthly? Quarterly? Overall, form enough connections between people to make a community.

- Replace use of Element with Helionauts?
  - The Element group chat is not used well, and importantly is not indexed by Google. If we had discussions on a google-able discussion board, people could find us when googling their problems. Helionauts may not be openly searched/accessed, but they are working on that. (https://community.openastronomy.org/). Investigate if we could create a PyHC category on helionauts and switch to using that for questions from the community (bonus is that other non-PyHC members could see and provide ideas/suggestions as well).

4.2. Searchable PyHC website

Contents (e.g. tutorials, blogs, PyHC projects) need to be searchable (by Google) to get more 'hits'. Most people do not immediately go first to the PyHC page to look at tools, they go to Google and start searching for keywords. We need to make it such that the PyHC website is one of the first couple options that pops up when people do this. Although we have a lot of information on Github, it seems that the Google search engine doesn't index deep enough on GitHub. Some ideas that could be investigated with respect to this:

- Leverage PyHC docstrings: Make sure PyHC docstrings get indexed by Google so they're more easily found. E.g. we know what projects are, PyHC could go through docstrings on projects and pull info out --> exposes PyHC more in Google
    - Investigate this with intersphinx?
    - Post somewhere on our website the various package routines?
- Make a PyHC package requirement to have things be picked up (better Search Engine Optimization or SEO). This would need sustained effort to keep things up-to-date (monthly job that runs and checks?).
- Utilize SEO keywords in blogs, tutorials, etc.
- Consider using SPASE?
- Have packages link to PyHC website

4.3. Tutorials

To help developers/scientists understand the utility of PyHC packages, we need to populate the PyHC gallery with more tutorials. Many of the tutorial ideas discussed previously fall under this (e.g., onboarding new devs/users).

Interactive shared tutorials/package howtos

- Add more tutorials/examples to PyHC Gallery, emphasizing tutorials that use multiple PyHC packages. One FTE could work on this indefinitely; endless tutorials/examples can be imagined. We just have to start producing them.
- Have good demos and examples that show useful use cases.
    i. Shows how to use software and that it is useful at the same time.
    ii. Ensure tutorials are reproducible/copyable by containerizing (or something similar) OR have someone go check/update them all indefinitely. (The first is likely easier in the long run.)

This work would feed into the needs of the 2.2 section related to Summer School, the 2.3 section related to a PyHC Onboarding Scientist, and the 3.6 section on regular PyHC projects' tutorials at Conferences.

4.4. Have a PyHC liaison for instrument teams.

This is a funded position to go to new instrument teams and offer free Python development support to help ensure that new instrument team tools mesh well with or leverage or expand existing PyHC tools. The liaison would provide free (to the teams) software support for a specific time period during a mission and would work to make software tools developed by the team part of the PyHC ecosystem. These could be incorporated into the mission or instrument AOs, or at least timed to coincide with when new instrument teams are being formed after a proposal is selected.

Get scientists/users excited about this by pinging Instrument Teams via a PyHC liaison and suggesting PyHC tools!

4.5. Hold regularly scheduled virtual PyHC office hours.

Have those funded to do PyHC-specific tasks (e.g., PyHC PI, PyHC Technical Lead), hold regularly-scheduled office hours. Perhaps also have a couple of developers from the community participate (could be different each month)? These office hours would be an opportunity for members of the broader heliophysics community to stop by and talk with representatives from PyHC. These events should be advertised via newsletters so that the broader community is aware of them. On the occasions when nobody shows up, this time could be used as a co-working session among a few PyHC members. These office hours would require low effort (~2–3 person-hours per session) and would likely have a commensurate reward.

5. Challenge: Improve PyHC documentation/discoverability

5.1. Make Table of routines and functions with expanded descriptions

Table of routines and functions shall be made available with expanded descriptions.
Current table of routines could be supplemented with a written tour of the community
More information about each package would be helpful for new and current users
Current table requires keyword support by community packages

5.2. Software as Infrastructure

- Longer term financing to support inter-package connections, as well as a host of community maintenance and support activities, is essential for a healthy community.

- Freely available and open source development shouldn't be synonymous with free development. We are professionals and quality science software requires time and capable scientists and developers. All of that requires funding.

- Transition from a personal collection of science methods to a public, tested, documented, and well formed package requires significant time and energy. These activities may not reach a sufficient level for an isolated proposal to the government. However, if the government wants a solid foundation, and a working actualization, for open and reproducible science, then that requires funding for all kinds of 'little' things. Each important on their own, but even more so in aggregate. All the little things form the difference between a conglomeration of packages and a community.

5.3. Advertise PyHC and increase discoverability of PyHC website.

'

Advertise PyHC telecons, new releases, package improvements and capabilities, etc. in community newsletters (e.g. SolarNews, UK Solar, SPA, GEM, SHINE, CEDAR), see also section 4.1

Advertise PyHC at meetings (needs to be consistent and everywhere PyHC folks go).

Cite Software DOIs.

Increase discoverability of PyHC website

- ask package repositories/websites to link to PyHC website
- to increase 'Googling' status, paid advertising?
- see also Section 4.2

Improve ranking in search engines (get links from other sites, ping other places about underlying package support)

5.4 Community Reviews

The community could band together to help review each others' software packages in code review. The process might be one of:

- Developer(s) from package A could review package B.
- Developers could form connections, possibly resulting in collaboration.
- Developer A's Improves knowledge of package B

Documentation review, and code creation, can generally benefit from outside perspectives. Developers already have an established mental model of the software and it can be difficult to see things as a new user may.

The use of community review could be applied as a requirement to join PyHC, with reviewers selected by the community. To begin, developers from the most established community packages could review a select few PyHC packages. Once approved, developers of those packages would be added to a potential pool of review committee members. The number of reviewers, etc. should be established via a community generated and approved standard.

Continuing reviews of approved packages could be enforced at major releases.

The time and effort in reviewing a package depends upon the size of the package involved. Review of a larger package, including code, documentation, unit tests, and actual use, could take 3 FTEs a month or more.

5.5. Keyword Search of Packages

We need packages to associate with all possible taxonomy keywords; this should improve search functionality on the PyHC website and onGoogle. We have a taxonomy of keywords on the PyHC site already (although this should be vetted and improved by the community), but only some projects add keywords to their projects. SPASE provides a possible source of keywords. We have filtering capabilities on the Projects page based on these keywords already for maximum utility the keywords need to be widely used. Additionally, we aren't displaying the keywords anywhere on the site. Once keyword usage is consistent across projects we should display those keywords somewhere.

Create and integrate a common template for auto-document generation for packages which includes HTML-based META tags along with places where links to the PyHC website occur in standard places. Create a standardized 'look n feel' for documentation and cross referencing. This issue is also related to issue 4.2.

Create a keyword-based search interface on the PyHC website, perhaps using intersphinx.

6. Challenge: Reduce overlapping functionality

6.1. Put together a master table showing what functionalities are offered by each package and each routine within each package.

The first step in reducing overlap is to discover what overlap exists. Making such a master table would accomplish that. It could also address onboarding topics by showing users and new developers what packages have what functionalities (if we made the master table look pretty and shared it on our site). The table, and thus the taxonomy, should include methods: coordinate transformation, time, units, data search & retrieval, etc.. The basic table entries will be quick for each method, so cost depends on how many methods there are in the package. A few days of work could go a long way.

6.2. Data access duplication: align behind one or a few core access packages? (HAPI, VSO, and something else?).

Data access is a functionality that's duplicated across so many projects. It seems obvious to us that we should strive for a one-stop-shop kind of solution for data access, like HAPI. Then every PyHC package gets its data in the same way from the same place. HAPI is the obvious choice, the only issue is that it can't serve some percentage of datasets. There'd have to be a backup for such datasets. Some packages already have a HAPI implementation, others would need to start fresh. There is also the question of what type of data format is used (xarray? …) and therefore how it plays with other packages. There is not complete consensus on this. This is likely opening a can

of worms. The various packages all had specific reasons for choosing the structure they have. While some standardization across the packages can (and should) be done, it will cause a divide in other cases. It is better to choose a few options as a best practice to follow rather than attempting to enforce this kind of thing.

6.3. Create (?) an architectural review board for PyHC. (This is contentious.)

An architectural review board is a governance body that makes major decisions regarding the overall architecture of a software system. A board for PyHC would make major decisions not so much on the internal structure of different software projects but rather on interoperability between different PyHC packages. A goal of this board would be to decide how to consolidate overlapping functionality and made recommendations but does not make mandates.

Because PyHC software projects are currently largely autonomous, this board would need buy-in and active participation from the different major projects in order for it to be successful, including from projects with pre-existing governance bodies. There should be a mechanism for members of PyHC to propose architectural changes.

There was considerable pushback on this suggestion. One consideration is that most of the PyHC projects in existence started prior to PyHC, and were developed without significant coordination. Many of the packages (including SunPy and PlasmaPy) have their own governance bodies which currently make these sorts of decisions. In the best case, there would be buy-in from the different projects and we'd get much closer to a cohesive Python ecosystem for all of heliophysics. In the worst case, individual projects would ignore the decisions of the board and we'd have wasted time and effort and made many of us grumpy. An alternative worth considering would be to hire a software architect, though the advantage of a board is that it would include representatives from the different software packages.

An alternative suggested approach instead of a board or software architect is to advertise to the packages what options are already in use (and why) for various scenarios and let them choose. There can be specific notes indicating possible incompatibilities (e.g. installing cdflib on a windows machine), but the packages need to be able to choose what is best for their development.

And finally, another point here is that duplication isn't something that is always undesirable. For example, users want to be able to generate the plots they are used to for their given conventions, so offering plotting in any particular package is not necessarily a duplication of effort. Now, using another package to do the plotting for you is good (e.g. plasmapy using another package's plotting features instead of developing its own). This is where the interoperability question becomes important: how to use data retrieved in one package in another package's function. If we can figure this out, THIS is where the user's options will suddenly expand to take advantage of the various versions

of functionality in the different packages. Such examples should be one focus of cross-package tutorials.

7. Challenge: Standardize and Share Data Models

7.1. Document current Data Models

Document use of existing data models, whether they are created in PyHC projects or are inherited by a supporting package such as Astropy.

7.2. PyHC community workshop to develop plan to coalesce Data Models

The community should offer a proposal to focus on coalescence on data models through an incremental approach. This probably relies on activity 7.1. (above).

7.3. PyHC might sponsor an AO dedicated to creating a common data model and modifying existing core PyHC libraries to use the new model. There needs to be a community consensus on the utility of such a model. (This is contentious and may lead to more standards.)

This probably relies on success of 7.2. (above). The AO could support one project per core library to come up with a joint set of Python data structures or interfaces that would underpin all current and future core PyHC modules. These data structures would need to offer ways of representing the common Heliophysics data types, including solar images and multivariate time series data (often in-situ measurements).

8. Challenge: Shared Services / APIs

8.1. Develop & document recommended methods of accessing data

PyHC should curate/support/recommend the ways of accessing data (uniform approach that outlasts a project, a promulgation support). It's important to get instrument teams/others to adopt these services for their own internal mechanisms. Possibly under jurisdiction of the above-proposed PyHC liaison.

9. Challenge : Mission Engagement

9.1. Develop and deploy a software/tool development plan for Heliophysics missions and instruments

This is becoming a NASA standard, namely, that missions use currently existing, open software when possible rather than reinventing. PyHC should be aware of this, and work with missions and international agencies as appropriate to understand new requirements for PyHC driven by missions.

9.2. Popularize PyHC APIs with Missions

PyHC should fund efforts to reach out to instrument teams to get them to adopt existing data APIs and other existing Python mechanisms/curate/support/recommend this kind of way of accessing data (uniform approach that outlasts a project, a promulgation support). Get instrument teams/others to adopt these services for their own internal mechanisms. Possibly under jurisdiction of the above-proposed PyHC liaison.

4. Summary

The results of the workshop show that there are many potential concrete actions which could be taken to improve the PyHC software and their associated projects. Key challenges may be grouped into items which involve sustaining/improving/expanding the community (key challenges 1, 2, 3, 4, and 9), improving information dissemination (challenge 5) and technical goals to better integrate and coalesce projects (challenges 6, 7, and 8). Solutions to challenges ranged from smaller and/or straight forward efforts (e.g., publishing results in science sections at conferences - section 3.3, holding regular, virtual office hours - section 4.5, or documenting current data models - section 7.1) to more detailed, long-term solutions (e.g., establishing new funding levels/mechanisms based on project maturity - section 1.4, creating and holding a yearly week-long “PyHC Summer School” - section 2.2, or creating new positions for PyHC - sections 1.1, 1.2, 2.3, 3.4, 3.7, and 4.4).

The community displayed particular interest in solving challenges associated with sustaining, improving, and expanding the community (the first group of challenges indicated in preceding paragraph). As a starting point, the documentation of each major Python library shall be improved describing in more details its content. Tutorials deserve particular mention in this area. The creation and dissemination of project tutorials, especially for newcomers, was brought up several times and participants posited that tutorials leveraging more than one PyHC project would be of special benefit for the community. The consensus seemed to be that tutorials allow projects to show scientists what kind of functionalities are available, what science questions could be answered with a package (which helps keep a project relevant, and will help the project continue to secure funding), demonstrate ways to better interoperate with other PyHC projects (which helps improve projects overall), and improve outreach to others in the heliophysics and space weather realms with tutorial presentation at conferences, meetings, etc. Some ideas in this grouping had more overall consensus than others, but disagreements were for the most part related to differing ideas in implementing suggested solutions, as opposed to fundamental disagreement on the importance of solving the challenges themselves.

Interestingly, while the initial goal of the workshop was to focus particularly on project integration this was the area showing the most contention and lack of consensus (challenges in the 3rd grouping, i.e. technical goals to better integrate and coalesce projects). Many meeting participants encouraged PyHC to document current project data models, services/APIs, project functionalities, etc. and make *recommendations* based on the science question at hand.

Participants strongly discouraged a coalescence on one project (be that in totality or in specific functionality therein - e.g., data access) or data model to the end of reducing redundancy. Some argued that, in fact, not all redundancy is a bad thing; there's utility to having some functionality replicated when projects attack a goal in differing fashion because of differing importance of requirements. This approach, in fact, was felt to allow projects to tailor methods to work with their project's unique needs.

In closing, even though some of the results were not anticipated at the outset of the meeting (by the organizers at least) they are uniformly interesting and insightful and much of the captured content is actionable. The workshop may therefore be considered successful at achieving its goal to uncover areas for improvement, increased attention and/or resourcing in the PyHC community. The organizers wish to thank all of the participants for their time and effort to contribute to this report and attend the workshop itself.

Appendices

A. Participants

Name	Role	Institution
Aaron Roberts	Workshop Organizer	NASA/GSFC
Arnaud Masson	Workshop Organizer	
Baptiste Cecconi	VESPA/maserpy	
Bill Rideout	MADRIGAL	
Bob Weigel	HAPI Client	
Brian Thomas	Workshop Organizer	NASA/GSFC
David Stansby	HelioPy, pyPFSS	
Eric Grimes	PySPEDAS	
Jack Ireland	SunPy	
Jan Gieseler	SolarMACH	University of Turku
Jeremy Faden	AutoPlot	
Jonathan Niehof	SpacePy; db processing	UNH
Jon Vandergriff	HAPI Client	JHUAPL
Julie Barnum	Workshop Organizer	LASP
Michael Hirsh	General	
Nick Murphy	PlasmaPy	
Paul O'Brien	IRBEM, AE9/AP9-IRENE	Aerospace
Rebecca Ringuette	Kamodo (CCMC)	GSFC/NASA
Reinhard Friedel	Workshop Organizer	NASA/HQ
Russell Stoneback	pysat	Stoneris
Shawn Polson	LASP/PyHC (general)	LASP
Will Barnes	AIAPy/SunPy	NRL

B. Agenda

Day 1: [3 hrs]

- Introduction [30 min] - Aaron Roberts presents
    - Purpose of the workshop, collect findings and observations on what we can do -- feeds a private nasa report on actions to take/funding.
    - Some Provocations
        - ML integration?
        - Cloud integration? (ex. write CDF to S3)
        - Interoperability of Python Packages (ex. read in SpacePy and now want to calculate number of plasma parameters and call routines from PlasmaPy)
        - Interoperability with non-Python (APIs, IDL, MatLab, Java, etc)

- Breakout Sessions: Collect examples of challenges we have encountered -- challenges in specific research workflows [1:30 hrs] ~5 ppl per group.

- Open Discussion of Challenges, which are the most important [1 hr],
    - Each group has 5 min to present their thoughts,
    - Open discussion to try to rank priorities. [45 min]

Day 2: [3 hrs]

- Review of challenges from first day [30 min] (Brian)

- Breakout Sessions : Discussion of how to solve some key challenges [1:30 hr]

- Synthesis Discussion on Findings and Observations [1 hr]
    - Each group presents their way to solve/approach to the challenges [15 min]
    - Open discussion of approaches, feedback on suggested solutions [45 min]