Challenges in Releasing Academic Software Packages

Dogacan Su Ozturk

UAF Students: Hunter Barndt, Jasmine Kobayashi

JPL Colleagues: Xing Meng, Olga Verkhoglyadova

contact: dsozturk@alaska.edu

Software Packages Developed

- High-latitude Input for Meso-scale Electrodynamics (HIME) Framework at JPL
  https://github.com/dcsozturk/hime

- ML-SoWGap: Machine Learning Models for Solar Wind Data Gaps at UAF (lead by Jasmine Kobayashi)
  https://github.com/Rokkaan5/MLSoWGap-Private

- SAM-III Magnetometer Data Analysis at UAF (lead by Hunter Barndt)
  https://github.com/whbarndt/samIII-magnetometer-data-analysis

Software Packages Developed but not all are published 😭

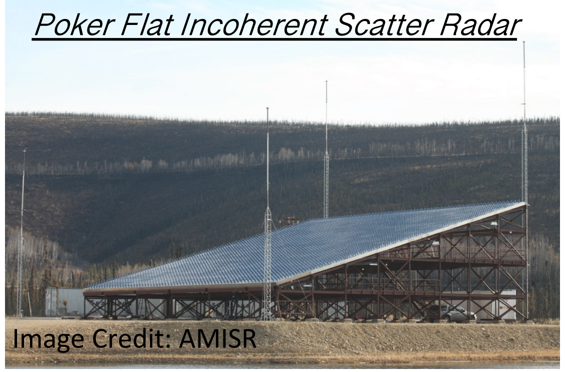
- High-latitude Input for Meso-scale Electrodynamics (HIME) Framework at JPL
  https://github.com/dcsozturk/hime

- ML-SoWGap: Machine Learning Models for Solar Wind Data Gaps at UAF (lead by Jasmine Kobayashi)
  https://github.com/Rokkaan5/MLSoWGap-Private

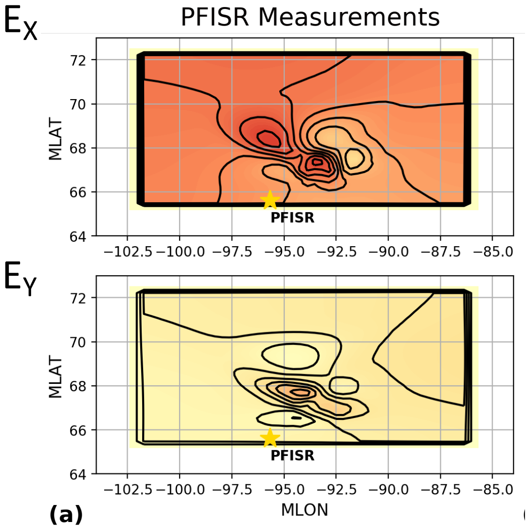
- SAM-III Magnetometer Data Analysis at UAF (lead by Hunter Barndt)
  https://github.com/whbarndt/samIII-magnetometer-data-analysis

High-latitude Input for Meso-scale Electrodynamics (HIME) Framework

HIME framework enables using local 2D electric field estimates to drive global circulation models.

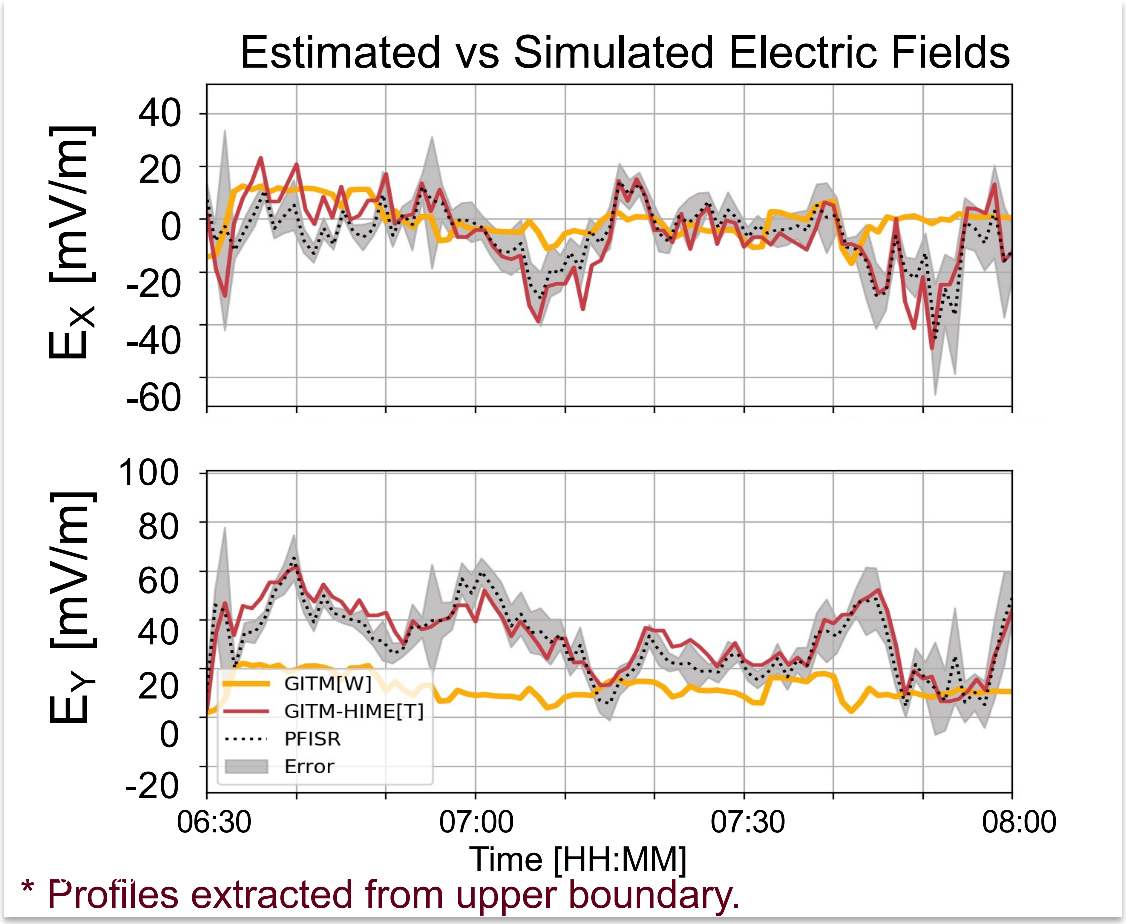


- Provides 3D “imaging” of electron density features.
- Can operate continuously.
- LOS velocities used in estimating 2D electric fields [Nicolls et al., 2014].



The 2D estimates captured variability that was not resolved by GCMs.

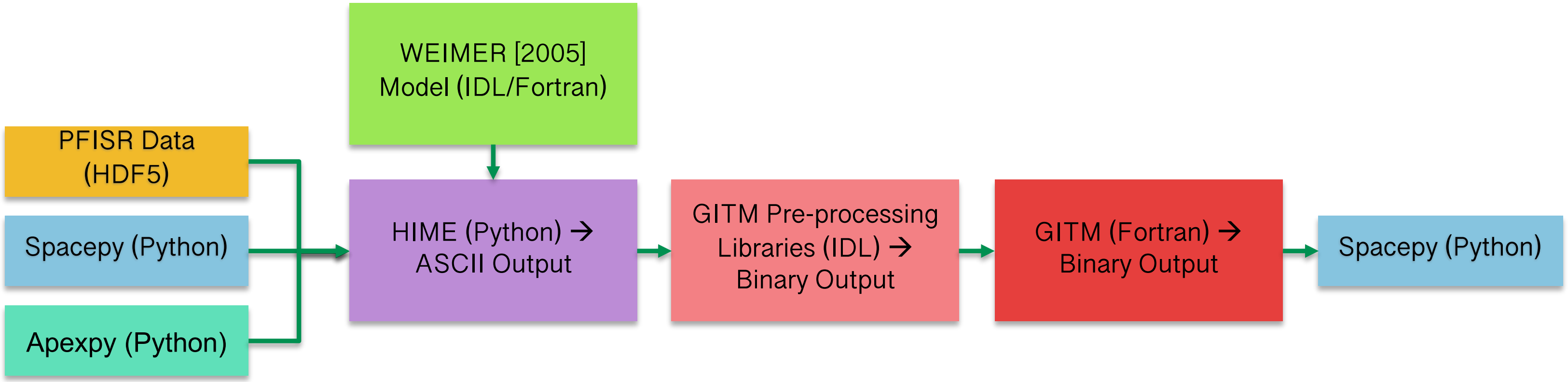
HIME Framework merges the potentials calculated from the PFISR E fields with other 2D global electric potential models to drive General Circulation Models.



High-latitude Input for Meso-scale Electrodynamics (HIME) Framework

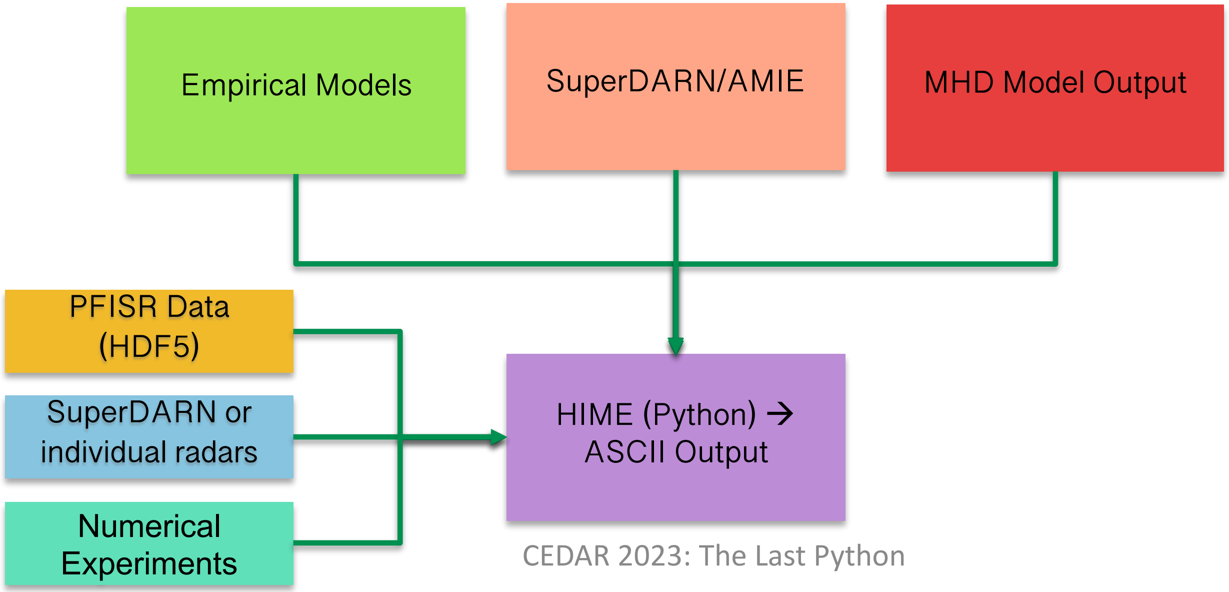
Interfacing Process:

Various data formats and programming languages are used in HIME Framework.



Versatility Process:

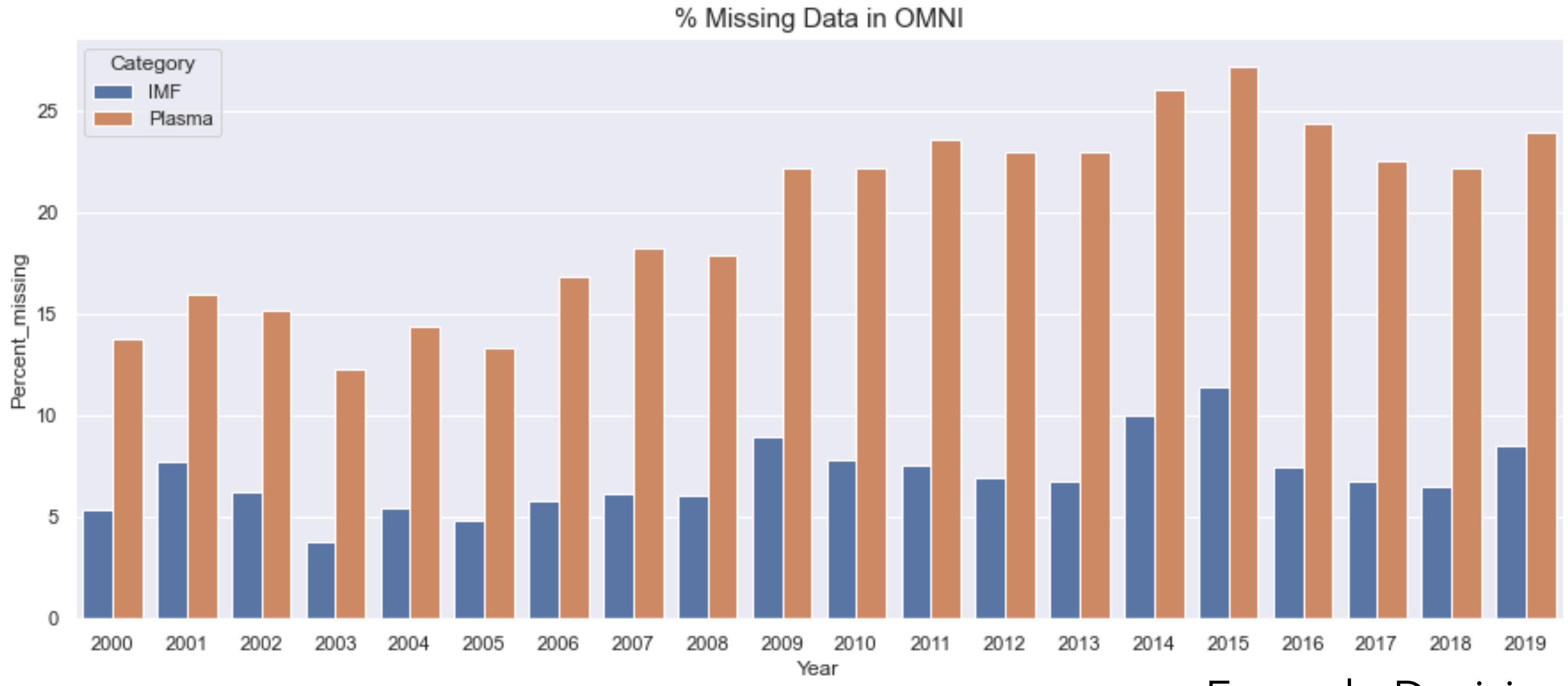
HIME's functionality can be separated into downsampling, conversion, and merging operations.



HIME Framework requires:

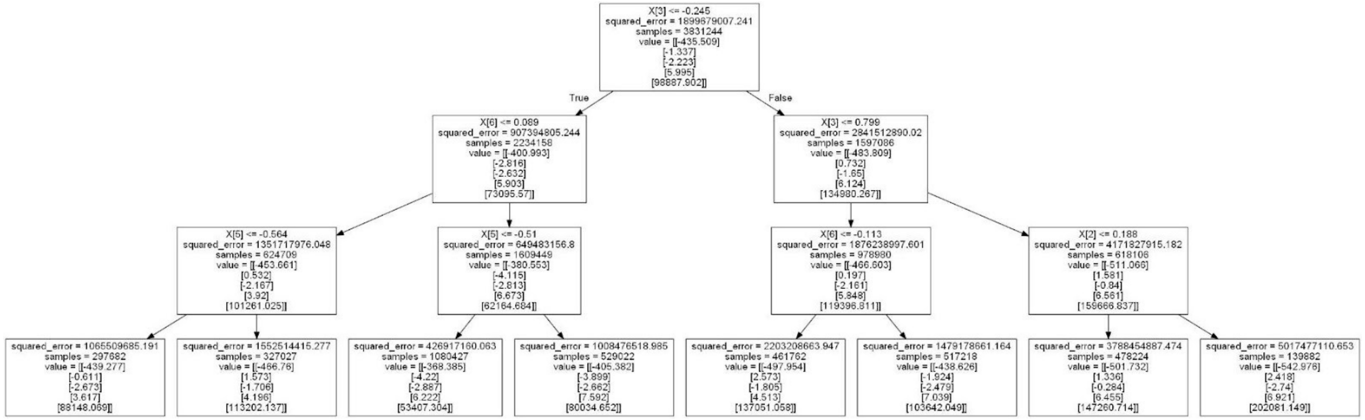
- New Technology Release
- A dependencies file.
- Tutorials for different functionality.

ML-SoWGap: Machine Learning Models for Solar Wind Data Gaps



OMNI data is widely used in statistical analysis and numerical modelling for describing the external driving conditions. However, significant data gaps occur during large geomagnetic storms, creating "an unbalanced" data set.

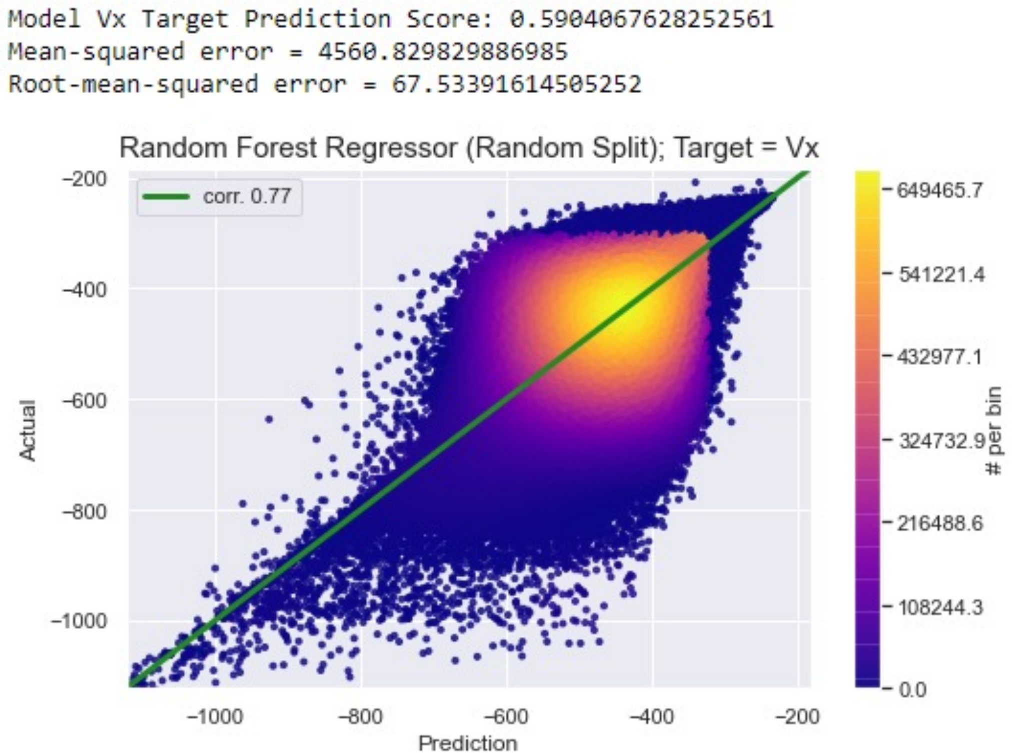
Example Decision Tree Architecture

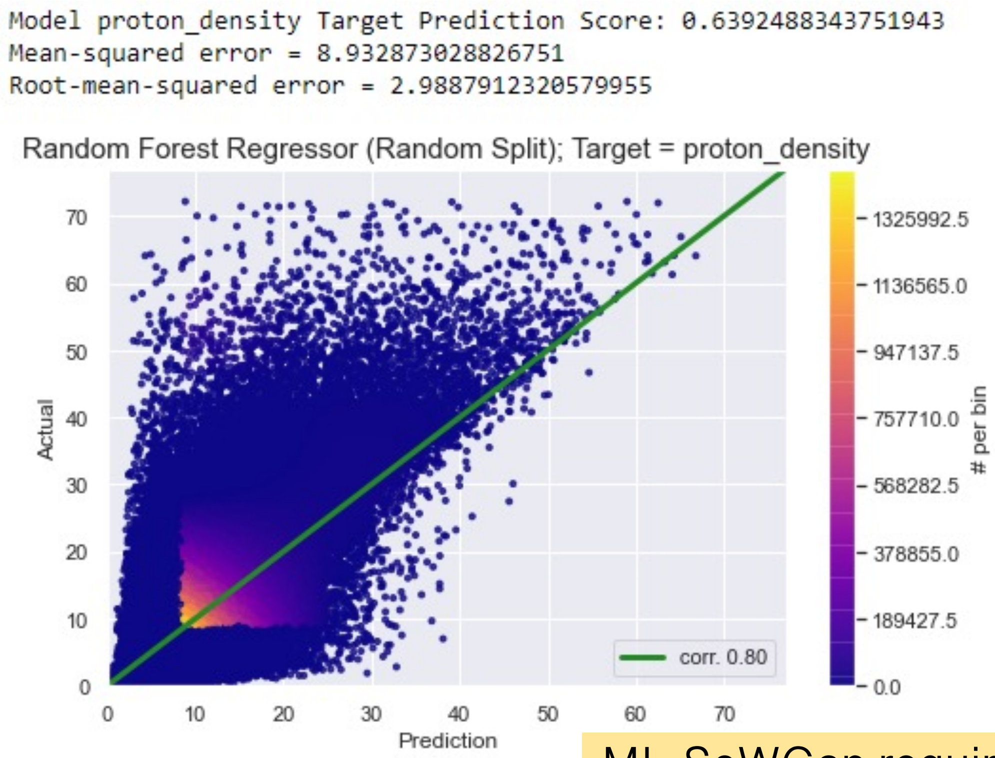


Geomagnetic activity indices and IMF values can be used to estimate the solar wind velocity and density values with a lower error compared to widely used interpolation techniques.

From Jasmine Kobayashi's Capstone Project

ML-SoWGap: Machine Learning Models for Solar Wind Data Gaps





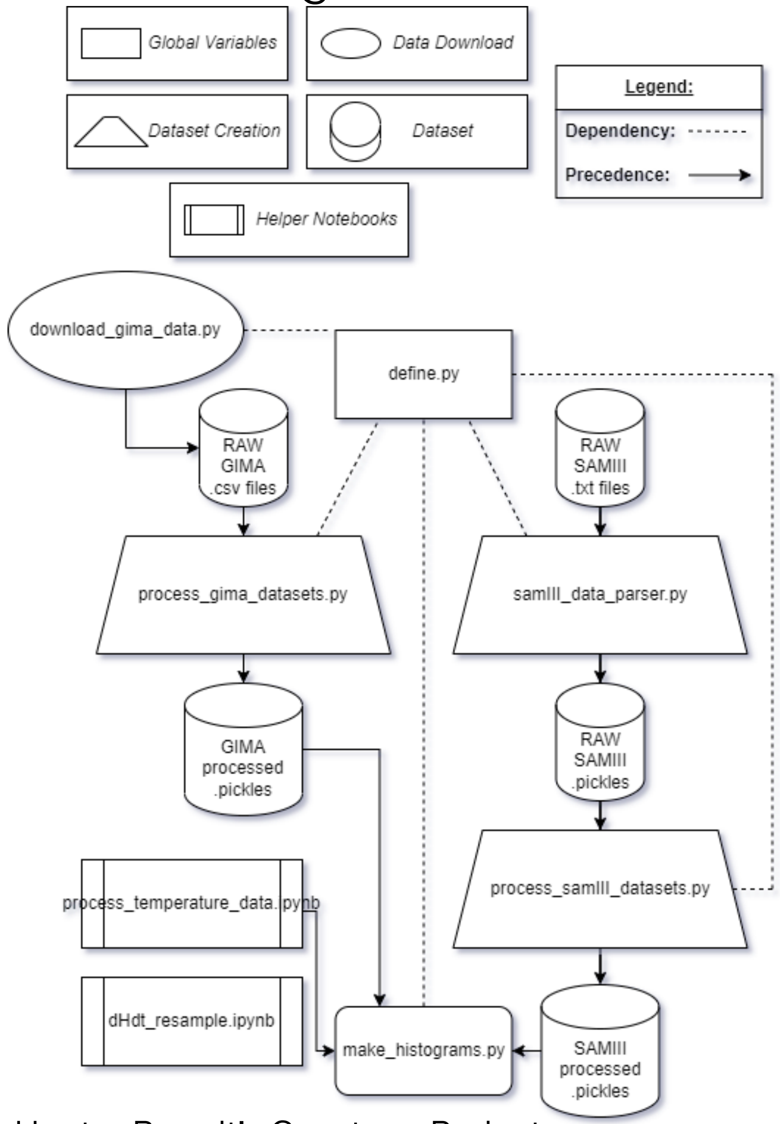
The Random Forest ensemble method achieves superior results compared to commonly used linear interpolation.

ML-SoWGap requires:
- Model features, input parameters, algorithm specifications for reproducibility.
- Extensive error/validation.

From Jasmine Kobayashi's Capstone Project

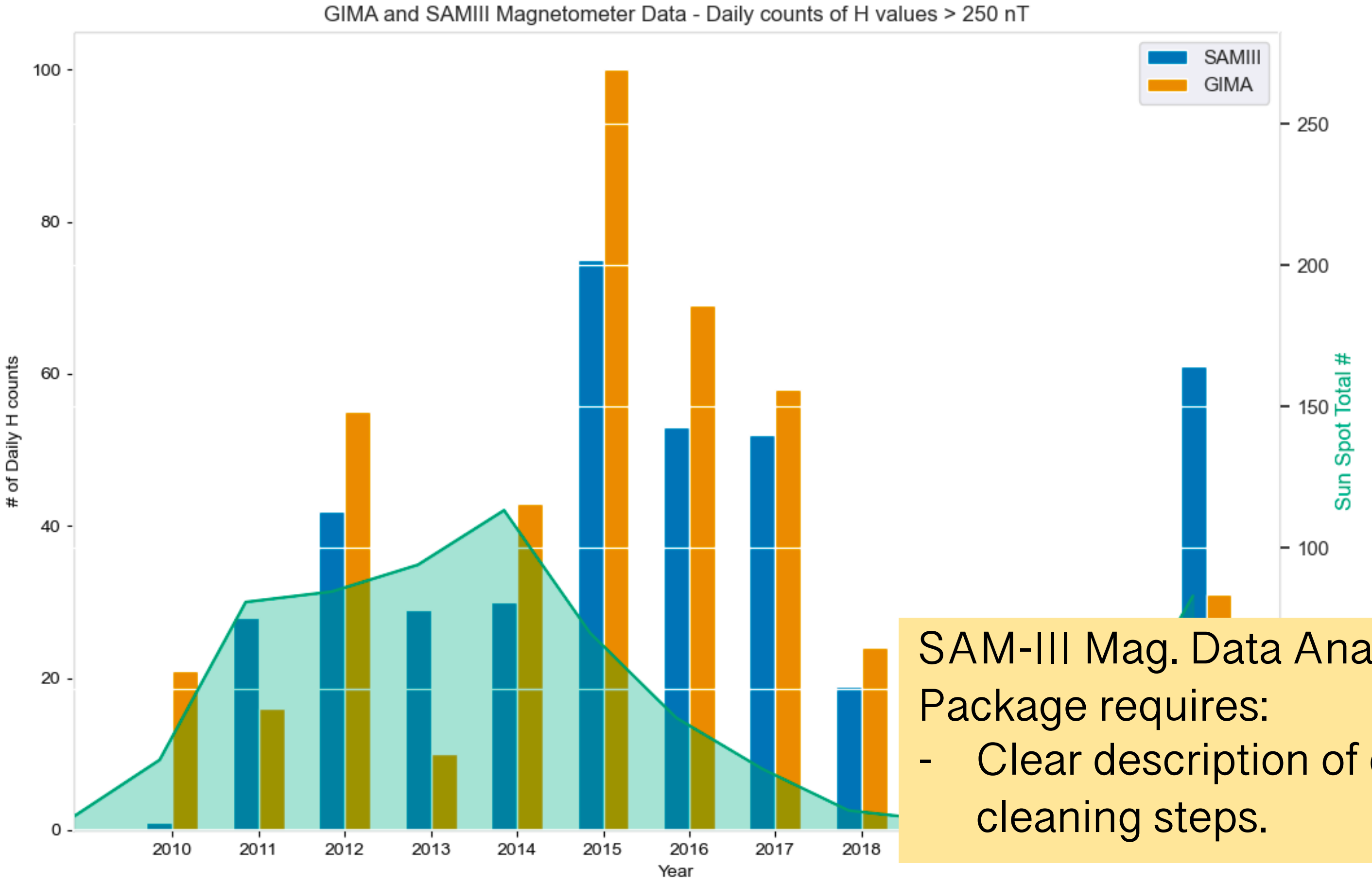
SAM-III Magnetometer Data Analysis

Package Architecture



From Hunter Barndt's Capstone Project

The software package downloads UAF GIMA and SWUG magnetometer data, creates Pandas DataFrames and provide statistical + GIC related information.



SAM-III Mag. Data Analysis Package requires:

- Clear description of data cleaning steps.

A quick look at software releases (!)

Property	HIME	ML-SoWGap	SAM-III Magnetometer Data Analysis
On GitHub	Yes	Yes	Yes
Lead career stage	Postdoc	Undergraduate	Undergraduate
Developed as a part of proposal	Yes – [HSR]	No – [EPSCoR]	No – [EPSCoR]
Published as a journal article	Yes	No	No
New Technology Report Submitted	Yes	Yes	No
Functionality separated	Yes	No	Yes
License	Yes	Yes	No

SCAN ME

Please go to www.menti.com and use code 7785 7850

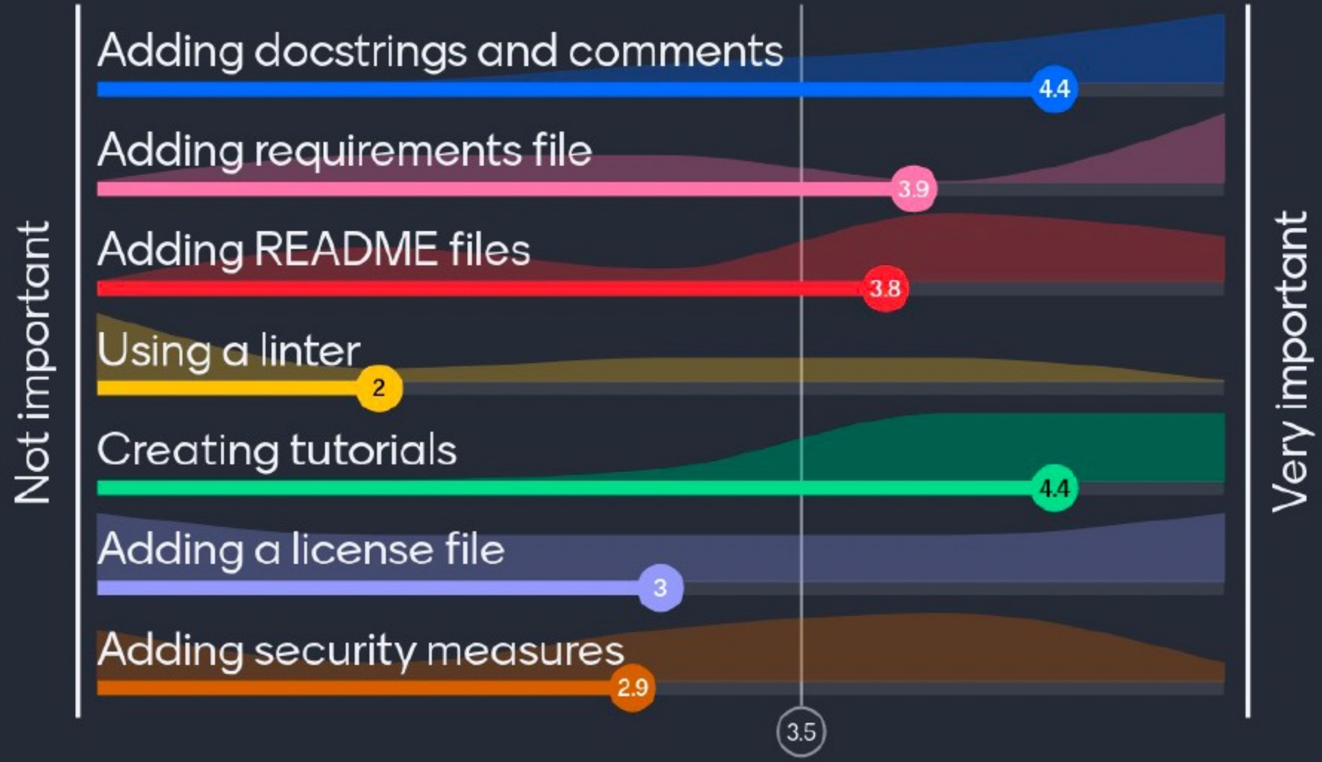
What are some of the challenges you face when releasing software packages?

push-pull rates
knowing the framework nee
conda dependencies
trainiing
data format
bureaucracy
complexity
tutorial
doing both pip and conda
time
new to the field
documentation
usability
lawyers copyright
procastination
readability
correct journal
bugs
upgradability
testing
fortran
can i lock
unit test
installation
software
how to package
writing good documentatio
upload speeds

13

Mentimeter

In your opinion how important are the following for a software package?



14

Mentimeter

In your opinion, what improvements could be applied to Heliophysics software packages?

5 Answers

Accessibility through central access point

Follow the Python in Heliophysics Community (PyHC) standards!

Barrier to entry — how easy is it to use

Giving people a starting point- how to use gut, what are unit tests, how to deal with the bureaucracy, how to chose a license, how to get a doi for the package, how to advertise your package etc

How do you gain contributors and users, how do you maintain the code, how do you set up a container, how do you teach others to use the container

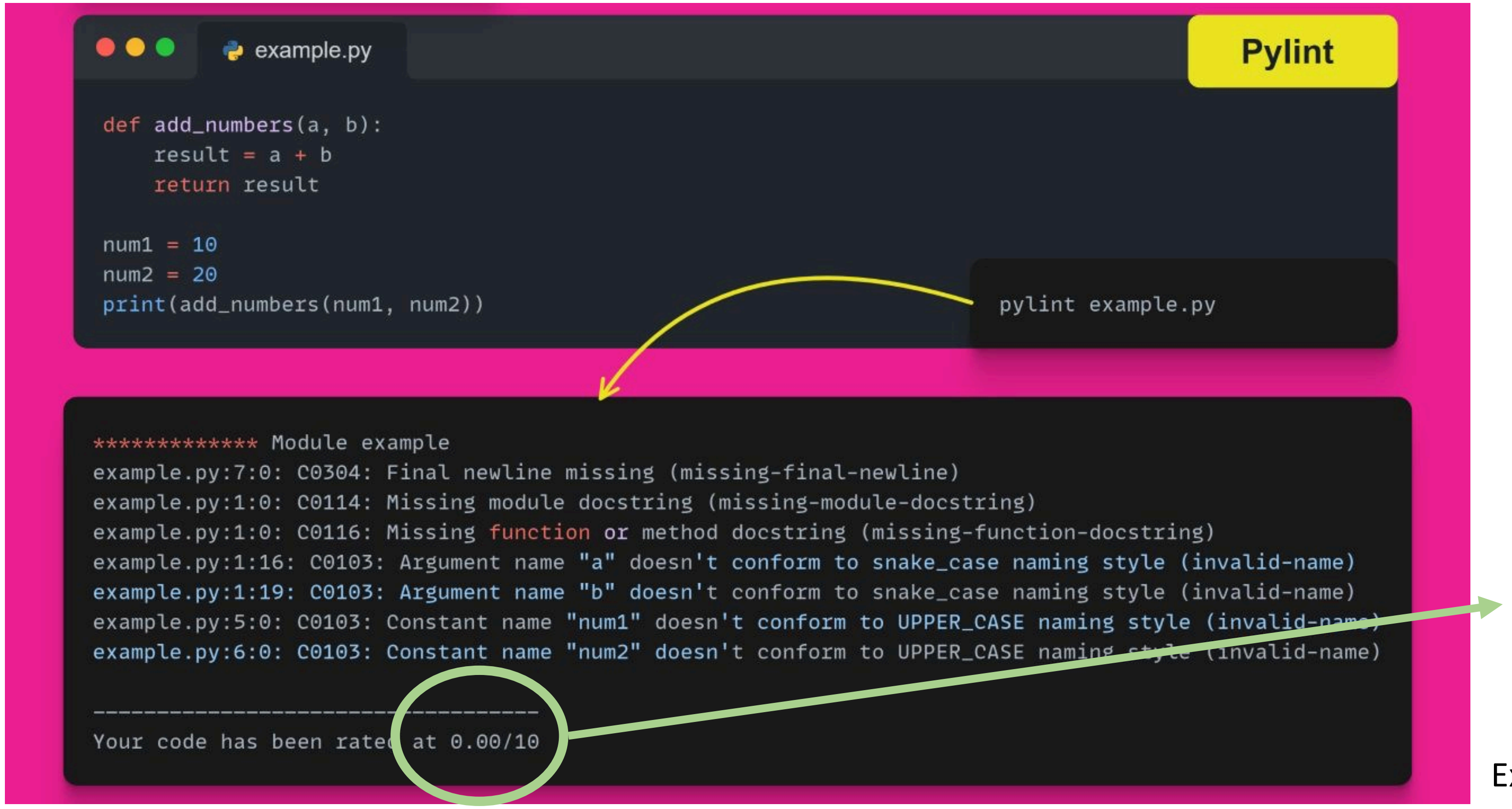
4

Software Package Release [a Minimal] Checklist

1. Add docstrings and documentation
2. Follow PEP 8 – Style Guide for Python Code
3. Create a layout for the code
4. Determine dependencies and create a requirements.txt file
5. Create a setup.py file
6. Use a formatting tool
7. Use a static code analysis tool
8. Create README file
9. Create License file
10. Create Tutorials
11. Upload to GitHub
12. Maintain GitHub
    - Test on GitHub
    - Update dependencies.txt as needed (automate)
    - Version control
13. Upload code to PyPI

Sample Static Code Analysis Tool: Pylint

- Can be used for detecting coding errors, bugs, and style violations.
- Makes suggestions to improve code readability.
- Can be used with integrated development environments.



Easy installation and usage:
>> pip install pylint
>> pylint example.py

Code grading helps with evaluating.

Example taken from Codiga.io

How can ChatGPT help with Software Release?

1. Package Structure: I can guide you on structuring your Python package properly, including organizing files, folders, and modules. This includes setting up the necessary files such as **'setup.py' and 'README.md'.**
2. Packaging Tools: I can help you understand and utilize packaging tools **like 'setuptools' and 'wheel'**. These tools assist in creating distribution packages for your Python code.
3. Versioning: I can provide guidance on versioning your software packages following best practices. This includes choosing appropriate version numbers and managing version dependencies.
4. Dependencies and Requirements: I can assist you in managing package dependencies and creating a **'requirements.txt'** file. I can also help you understand and work with tools like 'pip' and 'virtualenv'.
5. Publishing Packages: I can guide you on **publishing your Python packages to package indexes** such as PyPI (Python Package Index). This involves creating an account, preparing the package metadata, and uploading your package.

!!! Beware if using proprietary information or ITAR. Check with institution guidelines.

creating docstrings, writing tutorials, and generating HTML or PDF documentation.

7. Continuous Integration: I can assist you in setting up continuous integration (CI) systems like Travis CI or **GitHub Actions.** These systems automate tasks such as running tests, building packages, and publishing to package indexes whenever changes are made.
8. Licensing: I can provide guidance on choosing **an appropriate open-source license for your Python package** and help you understand licensing implications.
9. Troubleshooting: If you encounter any issues during the release process, I can help you **troubleshoot common problems and provide solutions.**

Further thoughts on releasing software packages.

- Data-associated software packages can prevent misuse.
- ML codes require further attention to ethical practices.
- Incentives like NTR, patent, inventions are very important for early-career scientists. (immediate CV entries)
- Special Journals/Software-X type Journals to separate software from science objectives could benefit the community.
- More funding for Tools and Methods, LWS-TM, H-ARD, etc. will help maintain and upgrade software.
- Special GitHub achievements for Heliophysics could further incentivize wider usage of commits, pull requests, and issues.
- Citing software rather than including packages in Acknowledgements helps early-career researchers more.
- Hackatons (AMGeo style) could further help these packages reach wider audiences.

Useful links and resources

- Python Developer's Guide: https://devguide.python.org/#contributing
- PEP 8 – Style Guide for Python Code: https://peps.python.org/pep-0008/#function-annotations
- Test Python Package Index: https://test.pypi.org/
- Pylint User Manual: https://docs.pylint.org/
- Ethical ML: https://github.com/EthicalML/awesome-production-machine-learning#model-and-data-versioning
- Continuous Integration to GitHub: https://docs.github.com/en/actions/automating-builds-and-tests/building-and-testing-python
- Semantic Versioning: https://semver.org/
- Choose a License: https://choosealicense.com/
- Software – X: https://www.sciencedirect.com/journal/softwarex

Contact: dsozturk@alaska.edu

Thank you!