2019 Workshop: Python for Space Science

Long title
Snakes on a Spaceship and the Goblet of Plots
Conveners
Russell Stoneback
Angeline G. Burrell
Jeff Klenzing
Description

'Snakes on a Spaceship' is focused on introducing the Python language, associated tools, and science software packages developed for the CEDAR and GEM community. This year, we have a focus on presenting new and/or interesting plots from across the field.

The pursuit of system science requires integrating measurements from multiple platforms into a coherent system for analysis. The variety of instrument types and data formats makes this a challenge. Typically these challenges are solved separately by different research teams leading to duplicated efforts. The study of the magnetosphere and the ionosphere as a system would be enhanced if solutions to these problems were made broadly available to the community. The use of community developed software has found acceptance in astronomy (astropy) and solar science (sunpy). ‘Snakes on a Spaceship’ is dedicated to fostering the same collaborative and open development practices within CEDAR and GEM.

Agenda

0) Introduction (5 min)

1) Ashton Reimer -- Resen: Towards a Reproducible Software Environment using Python and Docker (pdf)

2) Chih-Ting Hsu -- Comparison of TIE-GCM and C/NOFS ion velocity using PysatMagVect (pdf)

3) Marina Schmidt -- pydarn (pdf)

4) Leslie Lamarche -- Visualization of Diverse Geospace Data in Python (pdf)

5) Asher Pembroke -- Kamodo Analysis Suite (pdf)

6) Discussion (40 min)

Justification

CEDAR justification: Strategic thrust #5: Fuse the Knowledge Base across Disciplines Strategic thrust #6: Manage, Mine, and Manipulate Geoscience Data and Models

1) How the questions will be addressed: The challenge of performing system science across and within disciplines is addressed by teaching the community about the existence and use of enabling open source science software.

2) What resources exist, are planned, or are needed: Science python software already exists that helps the community achieve these goals, pysat, davitpy, spacepy, madrigal, etc.

3) How progress should be measured: Participation rates in open source science python software. Publications that use community tools can also be tracked.

Summary

Resen: Towards a Reproducible Software Environment using Python and Docker

Asthon Reimer

InGeO: An EarthCube project supported by the NSF Cyberinfrastructure for sustained and scientific innovation with two goals: provide tools to facilitate collaborations and reproducibility and to help educate the geospace community on best practices. Resen provides a portable environment for reproducing analysis. Installation is hard, accessing data can be difficult, and even more. Resen uses a cross platform containerized environment with software pre-installed. This allows work on linux to be reproduced as easily in windows or mac. You can save a completed analysis and share it with colleagues.

There are other tools that solve the installation and reproducibility problems: e.g., anaconda and scientific linux operating systems. Nothing does everything we want it to, though. To solve this, Resen was developed.

Resen is a command line tool for creating, importing and exporting software environments. It provides a simplified and abstracted interface. A beta version is

available. It has numpy, matplotlib, scipy, pandas, and more installed. It also has community tools such as apexpy, davitpy, madrigalweb, spacepy, etc. To use this you need to:

1) Install docker

2) use pip to install

3) Type resen from the command line

Within Resen you:

1) create a bucket. This creates a jupyter interface with notebooks or command line. It saves your work as you go as long as you have the bucket open.

2) export and import bucket commands are being developed

Other packages can also be installed within a bucket. This can effectively encapsulate the work for a paper, making an environment accessible even after those versions of the code are no longer supported.

On Wednesday at 19:00 at 109 N Guadalupe St there will be a guided tutorial.

Comparison of TIE-GCM and C/NOFS ion velocity using PysatMagVect

Chih-Ting Hsu

The motivation was to improve the understanding of the variability of the dayside, low-latitude, and global-scale ionospheric electrodynamics. COSMIC and C/NOFS are assimilated into TIE-GCM by assimilative mapping, using ensemble methods (DART, the Data Assimilation Research Testbed).

Complications: TIE-GCM covers 400-700 km depending on solar conditions, while C/NOFS measures between 400-860 km. PysatMagVect is used to map the C/NOFS observations along the magnetic field lines down to 120 km. This location is then converted into geographic coordinates and assimilated using the ensemble adjustment Kalman filter module in DART.

PysatMagVect helps:

1) compute scalars for mapping

2) compute the direction of geomagnetic field lines

3) Determine the velocity at the desired location

The vertical drift matches the best, but work still needs to be done with the meridional and zonal drift.

pyDARN

Marina Schmidt

SuperDARN looks at both hemispheres with a network of High Frequency Radars. We can do a lot with SuperDARN data. There are four data products provided to users. Most scientists use the FitACF files, which allow you to examine phenomena like TIDs, ULF waves, and more. These are typically examined through RTI plots, but fan plots are useful for seeing the spatial extent. SuperDARN also provides map files, which allow you to investigate things like sub-storms as you examine the spatial extent and characteristics of plasma convection.

Previously a lot of data visualization was done with davitpy, but it was difficult to install. Last year, we decided to have a fresh start with pydarn. Marina began development a few months ago. The goal is to institute best practices and limit the scope to keep the package maintainable. Additionally will include testing for unit tests, integration tests, benchmarking performance, and coverage.

Flexibility and extendability are also important. One should be able to easily add and remove features and build off of other packages. Documentation is all important, even for simple code.

pyDARN will provide a data visualzation library for SuperDARN. SuperDARN files are not easy to read in, because we have a custom data binary format. Reading is done through the DarnRead class. This class reads fitacf, rawacf, map, grid, et cetera through their own functions. These are clearly named (i.e., .read_fitacf()).

Plotting for range-time plots is currently supported. through pydarn.RTP (range-time-parameter). It only plots the data and lets you declare the figure, axis, and formats outside of the plotting routine.

For data visualization, you can make your colormaps more colorblind friendly by having yellow at one end of the scale. Almost any other color (but blue) can go on the other end of the scale.

The current timeline has the development of fan plots and convection plots this year. Then the IO will be removed to a separate library to avoid scope creep. Finally, a SuperDARN FAC data product will be added to RST and then a visualization product will be created.

Visualization of Diverse Geospace Data in Python

Leslie Lamarche

- Mangopy* is a python package for using the data from the Midlatitude All-sky-imager Network for Geophysical Observations. It fetched data from the ftp server, reads the composite hdf5 files and retrieves data arrays. It is python 2/3 compatible and available for download.

A mosaic can consider all or only certain sites and provide a composite image of all the data. A single image can be produced as well. In the future, movie creation will be supported, keograms will be included, difference images will be availalble, and performance will be improved.

- MIVIT*: Multi-Instrument Visualization Toolkit. The goal of MIVIT is to make it easier to compare diverse geospatial data sets. Comparing linear plots is easy enough, but it’s not easy for data with a spatial extent.

You provide a data set and a plot method to create a data visualization object. You can add as many of these as you want. Then, you can include all of these in the same figure.

The DataSet object is consuming to write for many different instruments. Contributions to this are very welcome.

This is a very new project that is being actively developed. Improvements include: fixing labeling and color bar options, expanding the available helper scripts, incorporating common models, introduce 3D visualization options, add data gridding and interpolation, and optimize the performance.

Kamodo Analysis Suite

Asher Pembroke

SSMS enables, supports, and performs research for the space weather community. It gathers data and models from the solar, heliospheric, magnetospheric, thermosphere, and ionosphere communities. The endeavor to support a wide variety of users, models, and data sources means that a new tool deeded to be developed to:

- quickly integrate new models and data - provide API support for scientists and educators who don’t code - model agnostic API - format agnostic API - Transparent, permissive metadata - automatic unit conversion - support coordinate transformations - compatible with helio-python ecosystem - provide instant visualization

Kamodo architecture is designed to serve modelers (c/c++/fortran), data scientists (python), and physicists (LaTeX). It uses pysat, spacepy, sunpy, plasmapy, and sympy to deal with each of these things as it needs to.

Scientists work with models and data through Kamodo objects, which map symbols to interpolating functions or mathematical expressions. It converts each expression to a highly optimized python function capable of operating on large arrays.

Any python function can be decorated with the @kamodofy decorator, adding metadata to a function.

A visualization API allows users to make plots without doing any real programming.

A model or data source is considered ‘kamodofied’ when all scientifically relevant variables are exposed as Kamodo objects. This requires:

1) It must be accessible from python

2) It must provide an interpolating function for each variable (models)

3) Interpolating functionns should supply default values as arguements, indicating the valid domain for their inputs

4) Variable names should follow Kamodos standards

5) Several more...

NASA software release process is underway to make this open source.

View PDF