pysat : A Bridge Between Worlds

Russell A. Stoneback

Center for Space Sciences, University of Texas at Dallas

Photon Audio Research Lab

Angeline G. Burrell

Naval Research Lab, Space Science Division

Jeff Klenzing

NASA GSFC

June 2019

UT DALLAS Python Satellite Data Analysis Toolkit

- PySat implements the general process of space science data analysis
- Instrument independent data, metadata, and processing support
- Design evolved over ~10 years
- Contains solutions to ~all of the processing problems I've encountered so far
- Began with Ion Velocity Meter (IVM) and Vector Electric Field Instrument (VEFI) on the Communications/Navigation Outage Forecasting System (C/NOFS)

- Package with support for common problems
  - Downloading
  - Organizing Files
  - Loading
  - Cleaning
  - Modifying/Processing
  - Exploit routines from other packages
  - Instrument specific analysis
  - Instrument independent analysis
  - Orbit iteration

Supported Instruments

- SuperDARN
- SuperMag
- C/NOFS IVM, PLP, VEFI
- OMNI
- JRO ISR

- DMSP IVM
- ISS-FPMU
- SGP4
- TIMED/SEE
- ICON (all instruments)

- ROCAST-1
- COSMIC-1, COSMIC-2
- Dst, Kp, F10.7 (Actual and predicted)
- CHAMP-STAR
- And more!

June 2019

Downloads no longer available

pysat community package use

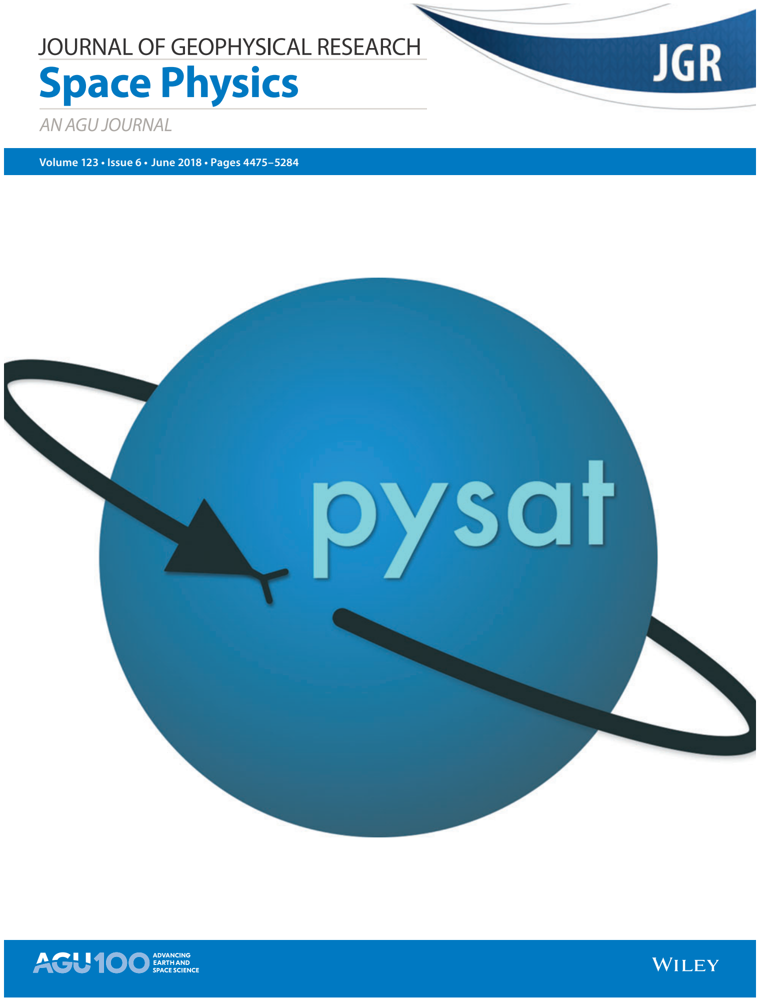
- DaViTPy
- AACGMv2
- apexpy
- pysatCDF
- pyglow
- pyEphem
- pysgp4
- pysatMagVect
- netCDF4

- IGRF reference Fortran
- scipy
- Pandas and xarray
- OCBPy (pysat integration not public)

pysat: Python Satellite Data Analysis Toolkit

build passing | docs passing | coverage 84% | DOI 10.5281/zenodo.1245182

Publication



JOURNAL OF GEOPHYSICAL RESEARCH

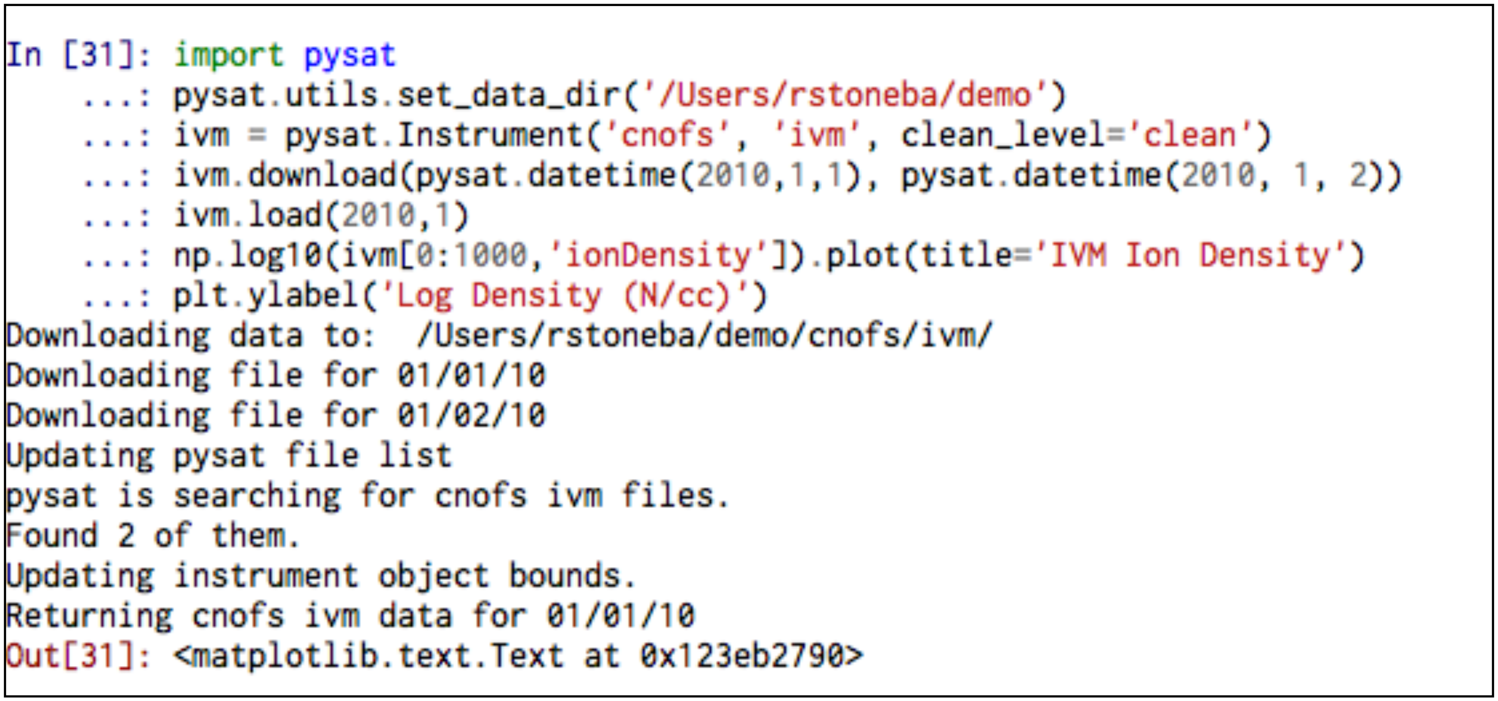
Space Physics

AN AGU JOURNAL

Volume 123 • Issue 6 • June 2018 • Pages 4475–5284

Process of Space Science Data Analysis
Implemented like a music recording signal chain
Constellation Support implemented by UTD CS Seniors

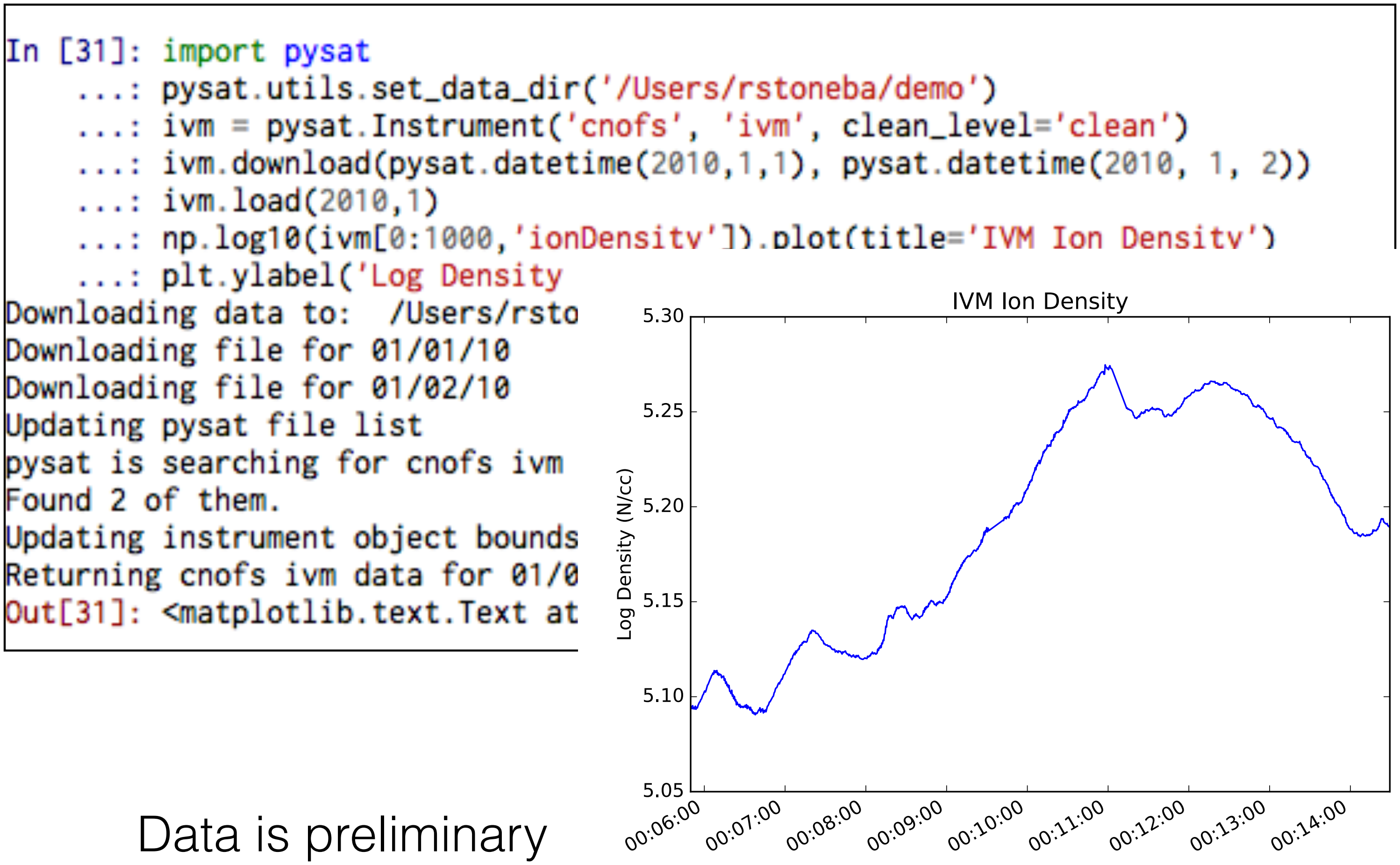
C/NOFS IVM



```
In [31]: import pysat
    ...: pysat.utils.set_data_dir('/Users/rstoneba/demo')
    ...: ivm = pysat.Instrument('cnofs', 'ivm', clean_level='clean')
    ...: ivm.download(pysat.datetime(2010,1,1), pysat.datetime(2010, 1, 2))
    ...: ivm.load(2010,1)
    ...: np.log10(ivm[0:1000,'ionDensity']).plot(title='IVM Ion Density')
    ...: plt.ylabel('Log Density (N/cc)')
Downloading data to:  /Users/rstoneba/demo/cnofs/ivm/
Downloading file for 01/01/10
Downloading file for 01/02/10
Updating pysat file list
pysat is searching for cnofs ivm files.
Found 2 of them.
Updating instrument object bounds.
Returning cnofs ivm data for 01/01/10
Out[31]: <matplotlib.text.Text at 0x123eb2790>
```

Data is preliminary

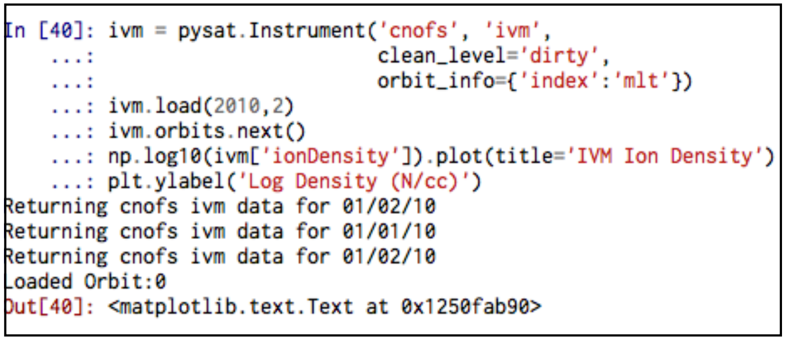
C/NOFS IVM



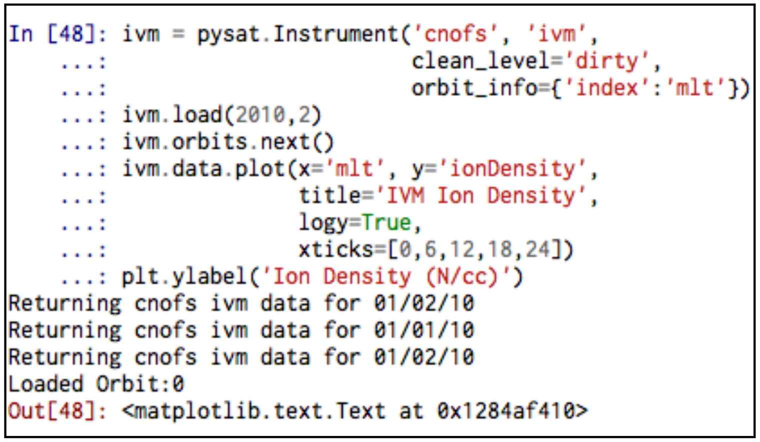
```
In [31]: import pysat
    ...: pysat.utils.set_data_dir('/Users/rstoneba/demo')
    ...: ivm = pysat.Instrument('cnofs', 'ivm', clean_level='clean')
    ...: ivm.download(pysat.datetime(2010,1,1), pysat.datetime(2010, 1, 2))
    ...: ivm.load(2010,1)
    ...: np.log10(ivm[0:1000,'ionDensity']).plot(title='IVM Ion Density')
    ...: plt.ylabel('Log Density
Downloading data to:  /Users/rsto
Downloading file for 01/01/10
Downloading file for 01/02/10
Updating pysat file list
pysat is searching for cnofs ivm
Found 2 of them.
Updating instrument object bounds
Returning cnofs ivm data for 01/0
Out[31]: <matplotlib.text.Text at
```

Data is preliminary

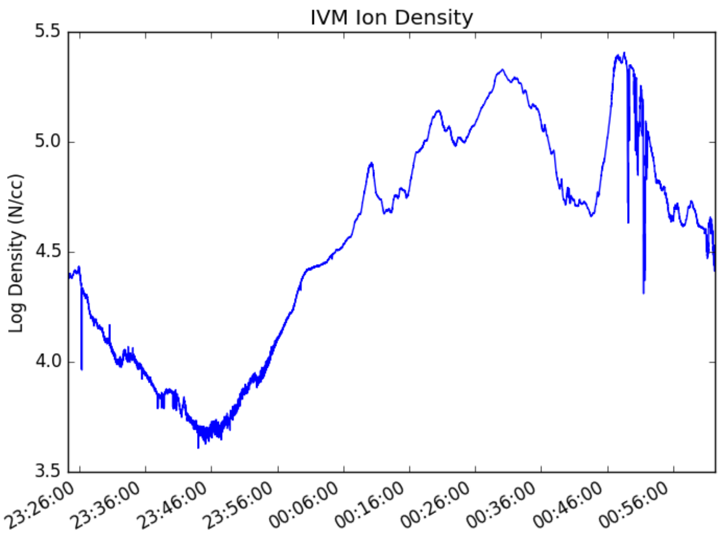
C/NOFS IVM by Orbit

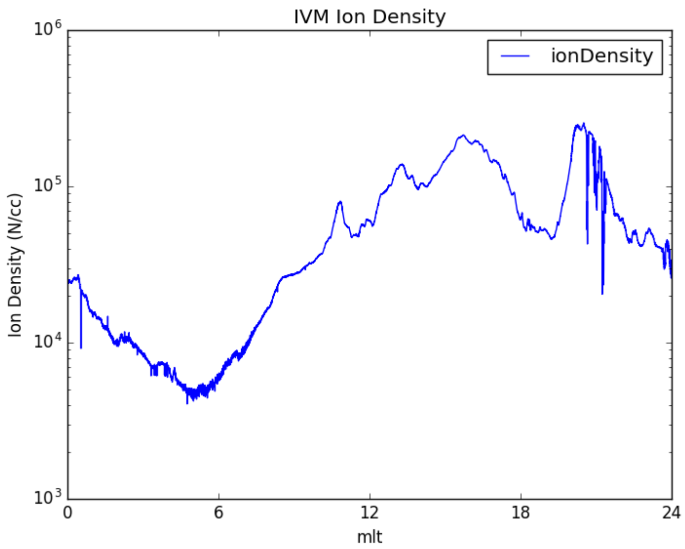


```
In [40]: ivm = pysat.Instrument('cnofs', 'ivm',
    ...:                          clean_level='dirty',
    ...:                          orbit_info={'index':'mlt'})
    ...: ivm.load(2010,2)
    ...: ivm.orbits.next()
    ...: np.log10(ivm['ionDensity']).plot(title='IVM Ion Density')
    ...: plt.ylabel('Log Density (N/cc)')
Returning cnofs ivm data for 01/02/10
Returning cnofs ivm data for 01/01/10
Returning cnofs ivm data for 01/02/10
Loaded Orbit:0
Out[40]: <matplotlib.text.Text at 0x1250fab90>
```

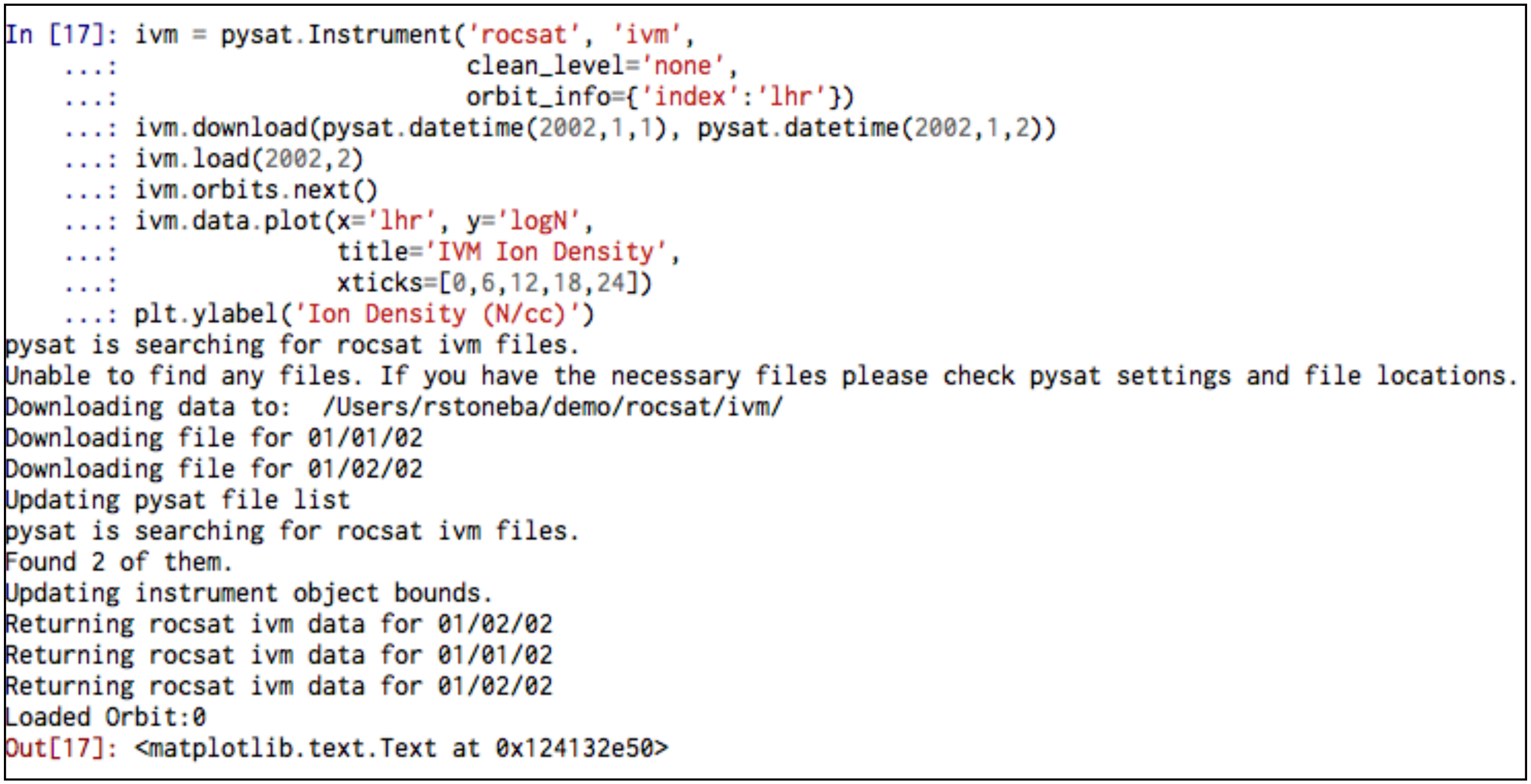


```
In [48]: ivm = pysat.Instrument('cnofs', 'ivm',
    ...:                          clean_level='dirty',
    ...:                          orbit_info={'index':'mlt'})
    ...: ivm.load(2010,2)
    ...: ivm.orbits.next()
    ...: ivm.data.plot(x='mlt', y='ionDensity',
    ...:               title='IVM Ion Density',
    ...:               logy=True,
    ...:               xticks=[0,6,12,18,24])
    ...: plt.ylabel('Ion Density (N/cc)')
Returning cnofs ivm data for 01/02/10
Returning cnofs ivm data for 01/01/10
Returning cnofs ivm data for 01/02/10
Loaded Orbit:0
Out[48]: <matplotlib.text.Text at 0x1284af410>
```



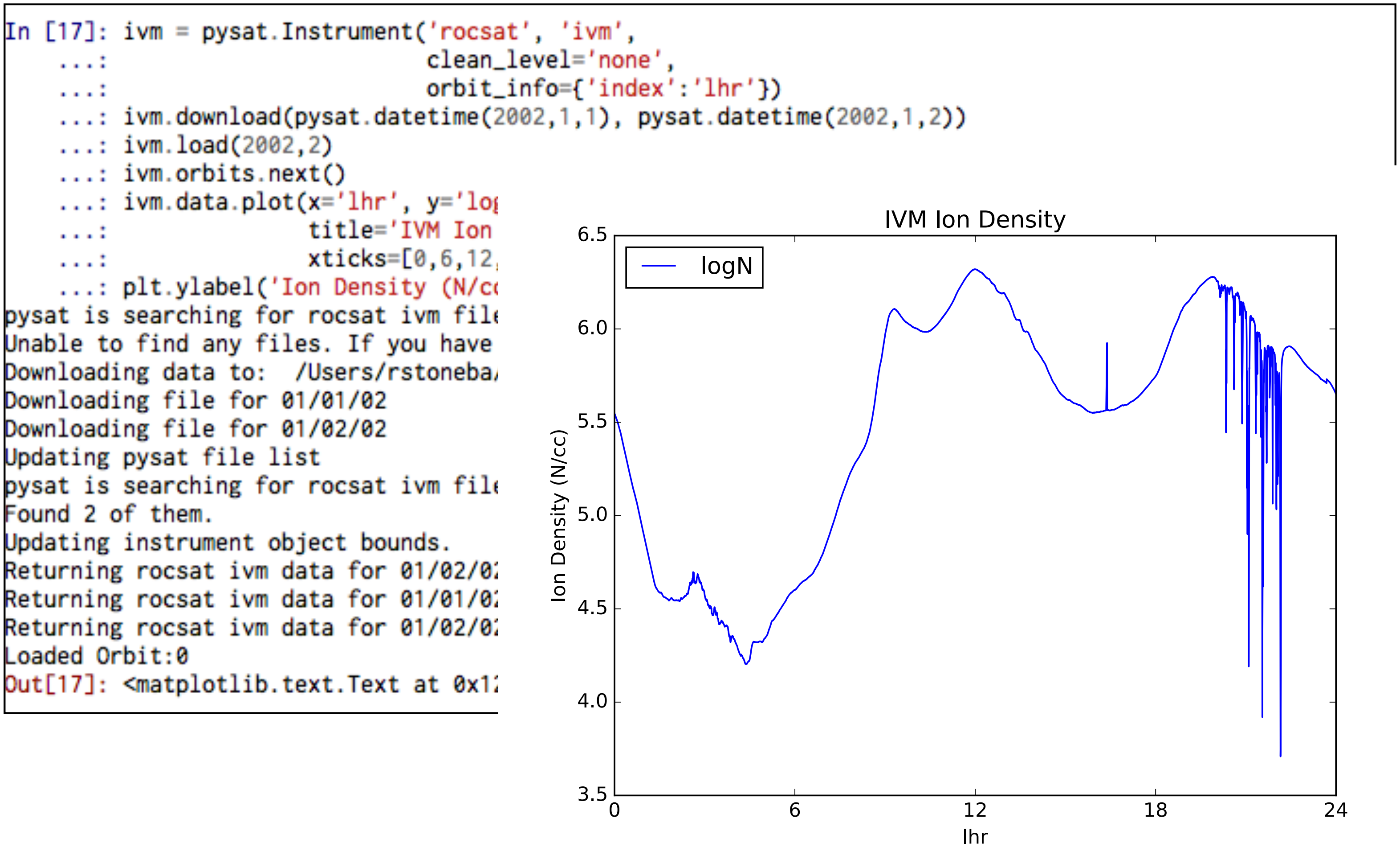


ROCSAT



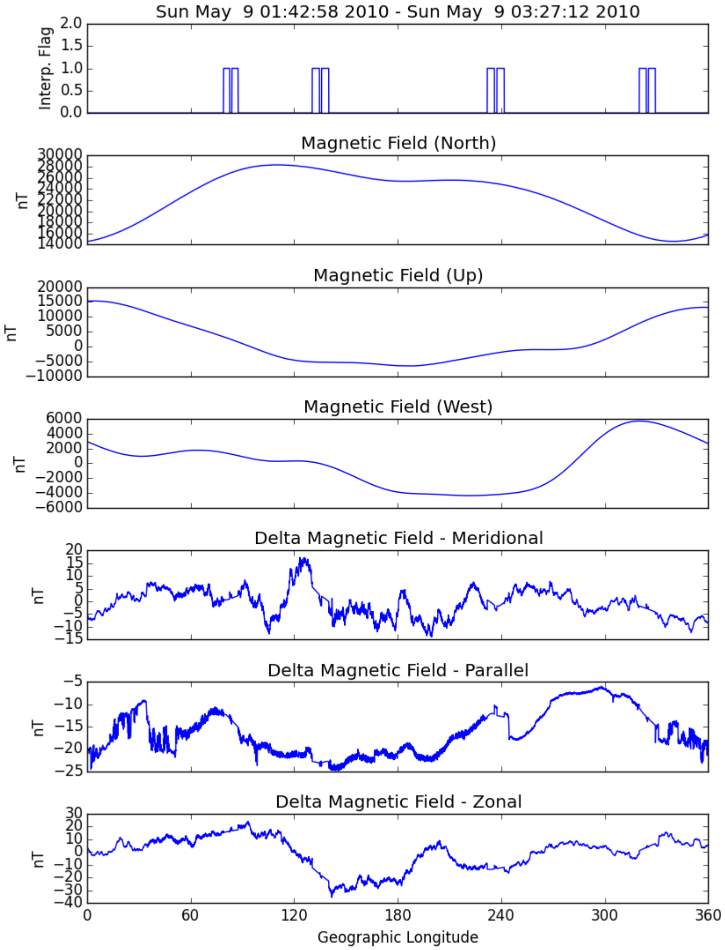
```
In [17]: ivm = pysat.Instrument('rocsat', 'ivm',
    ...:                        clean_level='none',
    ...:                        orbit_info={'index':'lhr'})
    ...: ivm.download(pysat.datetime(2002,1,1), pysat.datetime(2002,1,2))
    ...: ivm.load(2002,2)
    ...: ivm.orbits.next()
    ...: ivm.data.plot(x='lhr', y='logN',
    ...:               title='IVM Ion Density',
    ...:               xticks=[0,6,12,18,24])
    ...: plt.ylabel('Ion Density (N/cc)')
pysat is searching for rocsat ivm files.
Unable to find any files. If you have the necessary files please check pysat settings and file locations.
Downloading data to:  /Users/rstoneba/demo/rocsat/ivm/
Downloading file for 01/01/02
Downloading file for 01/02/02
Updating pysat file list
pysat is searching for rocsat ivm files.
Found 2 of them.
Updating instrument object bounds.
Returning rocsat ivm data for 01/02/02
Returning rocsat ivm data for 01/01/02
Returning rocsat ivm data for 01/02/02
Loaded Orbit:0
Out[17]: <matplotlib.text.Text at 0x124132e50>
```

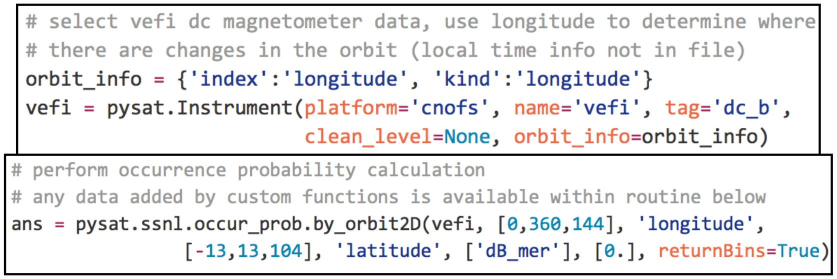
ROCSAT



```
In [17]: ivm = pysat.Instrument('rocsat', 'ivm',
    ...:                        clean_level='none',
    ...:                        orbit_info={'index':'lhr'})
    ...: ivm.download(pysat.datetime(2002,1,1), pysat.datetime(2002,1,2))
    ...: ivm.load(2002,2)
    ...: ivm.orbits.next()
    ...: ivm.data.plot(x='lhr', y='log
    ...:               title='IVM Ion
    ...:               xticks=[0,6,12,
    ...: plt.ylabel('Ion Density (N/cc
pysat is searching for rocsat ivm file
Unable to find any files. If you have
Downloading data to:  /Users/rstoneba/
Downloading file for 01/01/02
Downloading file for 01/02/02
Updating pysat file list
pysat is searching for rocsat ivm file
Found 2 of them.
Updating instrument object bounds.
Returning rocsat ivm data for 01/02/02
Returning rocsat ivm data for 01/01/02
Returning rocsat ivm data for 01/02/02
Loaded Orbit:0
Out[17]: <matplotlib.text.Text at 0x12
```

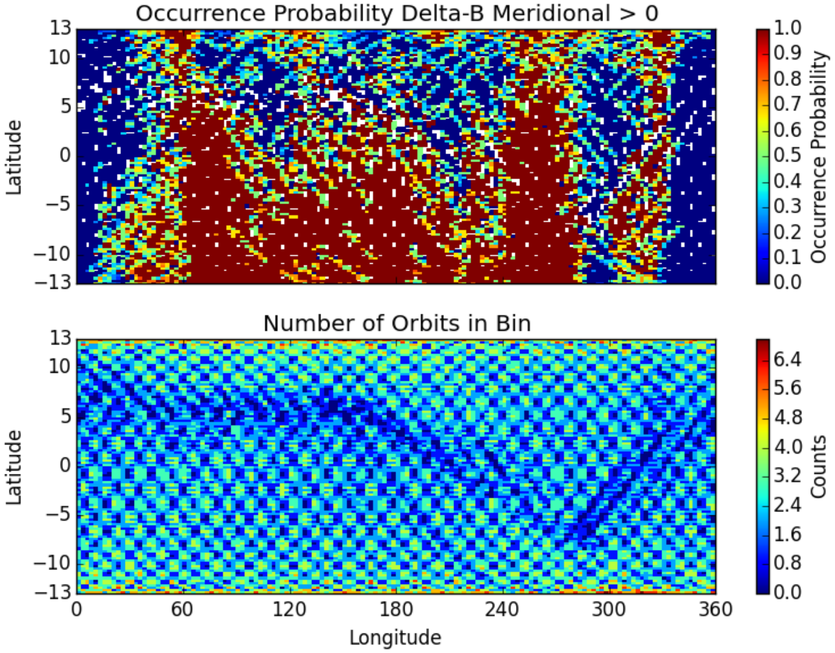
C/NOFS VEFI





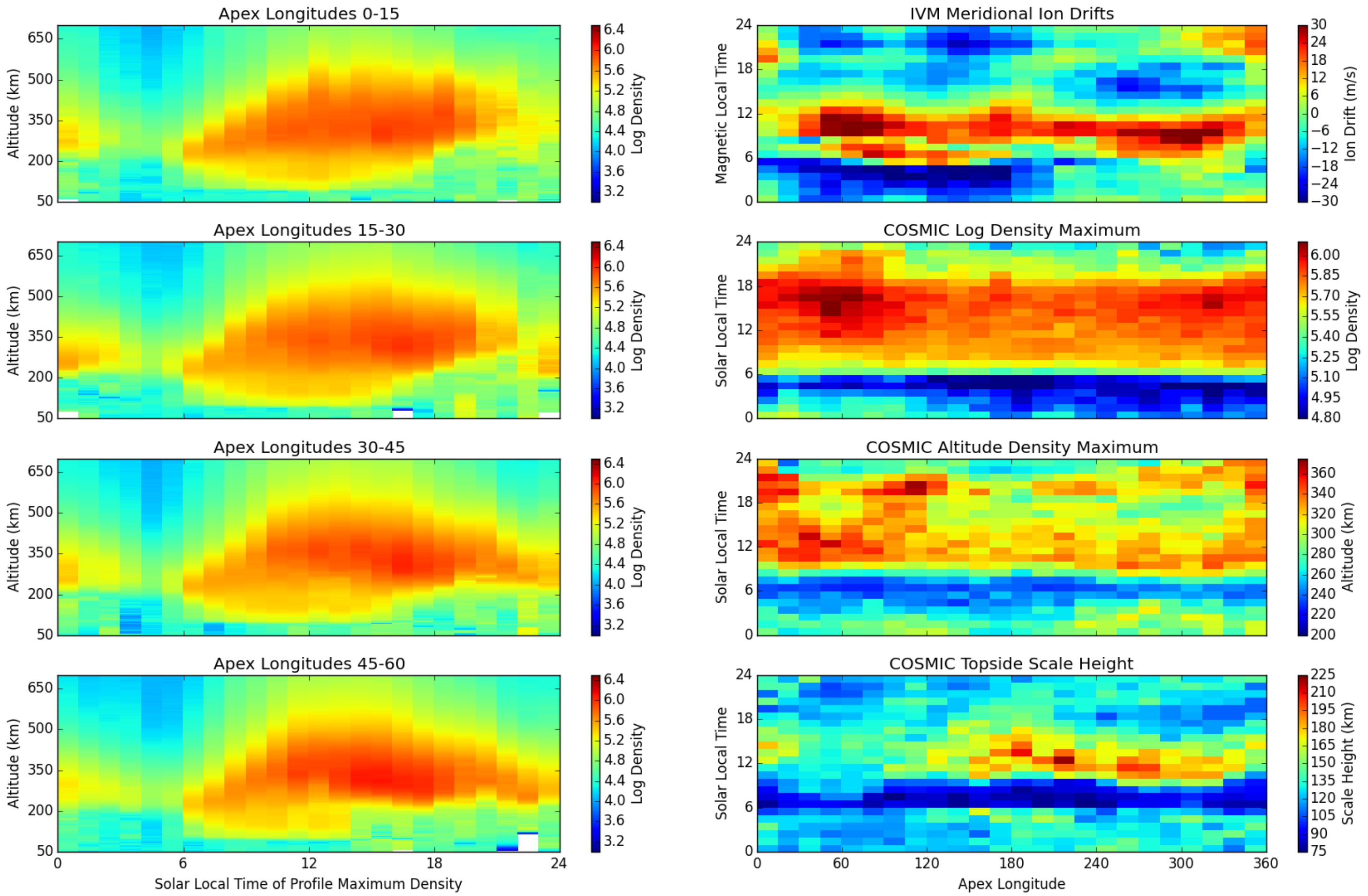
```
# select vefi dc magnetometer data, use longitude to determine where
# there are changes in the orbit (local time info not in file)
orbit_info = {'index':'longitude', 'kind':'longitude'}
vefi = pysat.Instrument(platform='cnofs', name='vefi', tag='dc_b',
                        clean_level=None, orbit_info=orbit_info)
```

```
# perform occurrence probability calculation
# any data added by custom functions is available within routine below
ans = pysat.ssnl.occur_prob.by_orbit2D(vefi, [0,360,144], 'longitude',
                [-13,13,104], 'latitude', ['dB_mer'], [0.], returnBins=True)
```

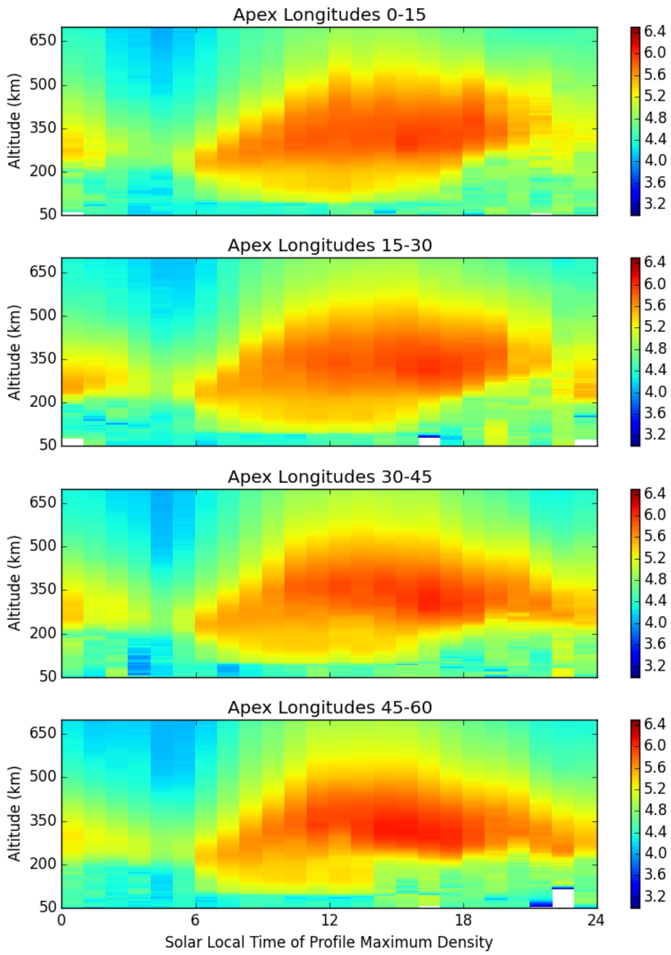


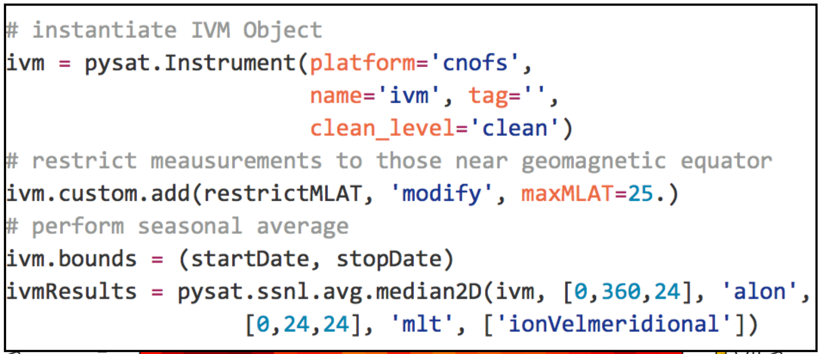
Full Code in Demo Area of Repo

COSMIC and IVM Demo

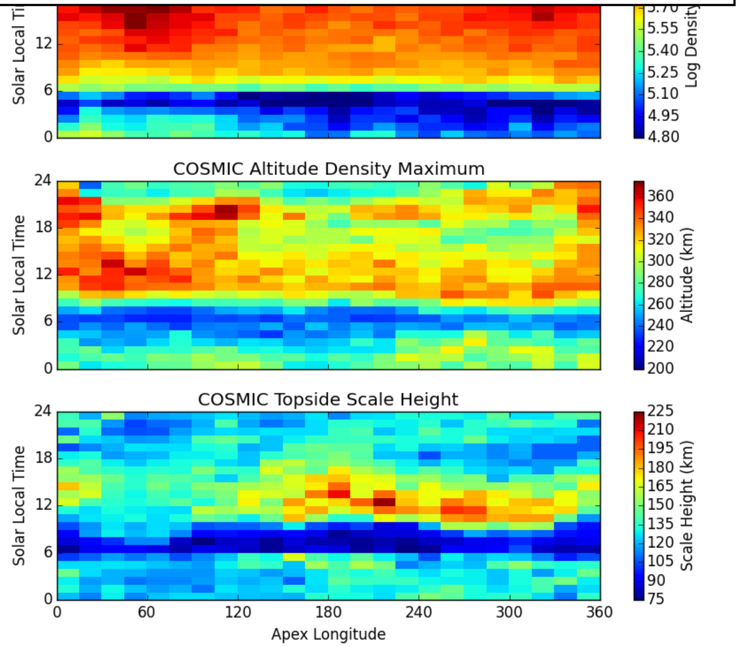


COSMIC and IVM Demo

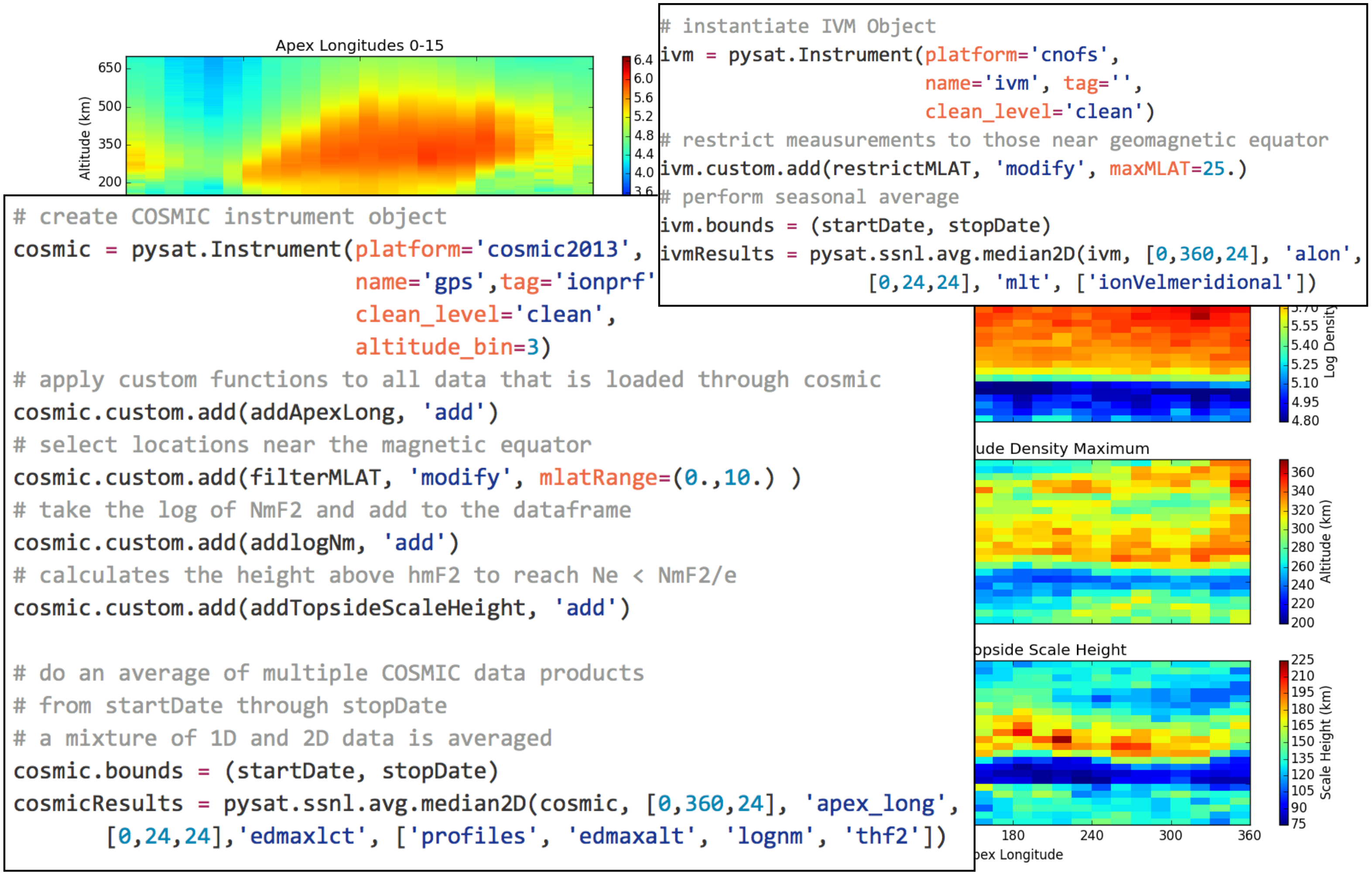




```
# instantiate IVM Object
ivm = pysat.Instrument(platform='cnofs',
                       name='ivm', tag='',
                       clean_level='clean')
# restrict meausurements to those near geomagnetic equator
ivm.custom.add(restrictMLAT, 'modify', maxMLAT=25.)
# perform seasonal average
ivm.bounds = (startDate, stopDate)
ivmResults = pysat.ssnl.avg.median2D(ivm, [0,360,24], 'alon',
                  [0,24,24], 'mlt', ['ionVelmeridional'])
```



COSMIC and IVM Demo



```
# instantiate IVM Object
ivm = pysat.Instrument(platform='cnofs',
                       name='ivm', tag='',
                       clean_level='clean')
# restrict meausurements to those near geomagnetic equator
ivm.custom.add(restrictMLAT, 'modify', maxMLAT=25.)
# perform seasonal average
ivm.bounds = (startDate, stopDate)
ivmResults = pysat.ssnl.avg.median2D(ivm, [0,360,24], 'alon',
                   [0,24,24], 'mlt', ['ionVelmeridional'])
```

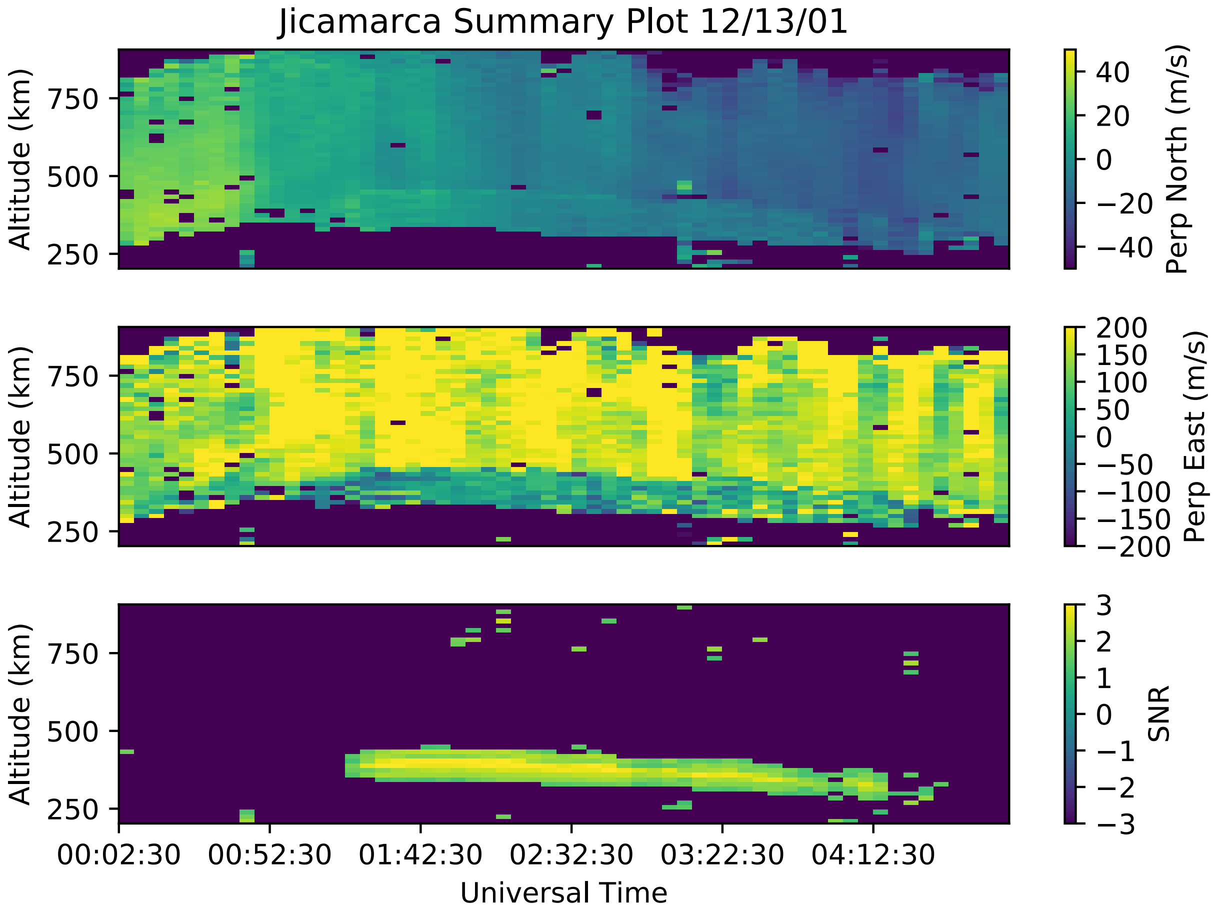
```
# create COSMIC instrument object
cosmic = pysat.Instrument(platform='cosmic2013',
                          name='gps',tag='ionprf',
                          clean_level='clean',
                          altitude_bin=3)
# apply custom functions to all data that is loaded through cosmic
cosmic.custom.add(addApexLong, 'add')
# select locations near the magnetic equator
cosmic.custom.add(filterMLAT, 'modify', mlatRange=(0.,10.) )
# take the log of NmF2 and add to the dataframe
cosmic.custom.add(addlogNm, 'add')
# calculates the height above hmF2 to reach Ne < NmF2/e
cosmic.custom.add(addTopsideScaleHeight, 'add')

# do an average of multiple COSMIC data products
# from startDate through stopDate
# a mixture of 1D and 2D data is averaged
cosmic.bounds = (startDate, stopDate)
cosmicResults = pysat.ssnl.avg.median2D(cosmic, [0,360,24], 'apex_long',
       [0,24,24],'edmaxlct', ['profiles', 'edmaxalt', 'lognm', 'thf2'])
```

Jicamarca Radar Observatory

- Interfaces with Madrigal

```
jro = pysat.Instrument('jro', ‘isr’, ‘drifts’)
jro.download(date1, date2)
jro.load(date=date1)
pysat.instruments.jro_isr.drifts_plot(jro)
```



DMSP

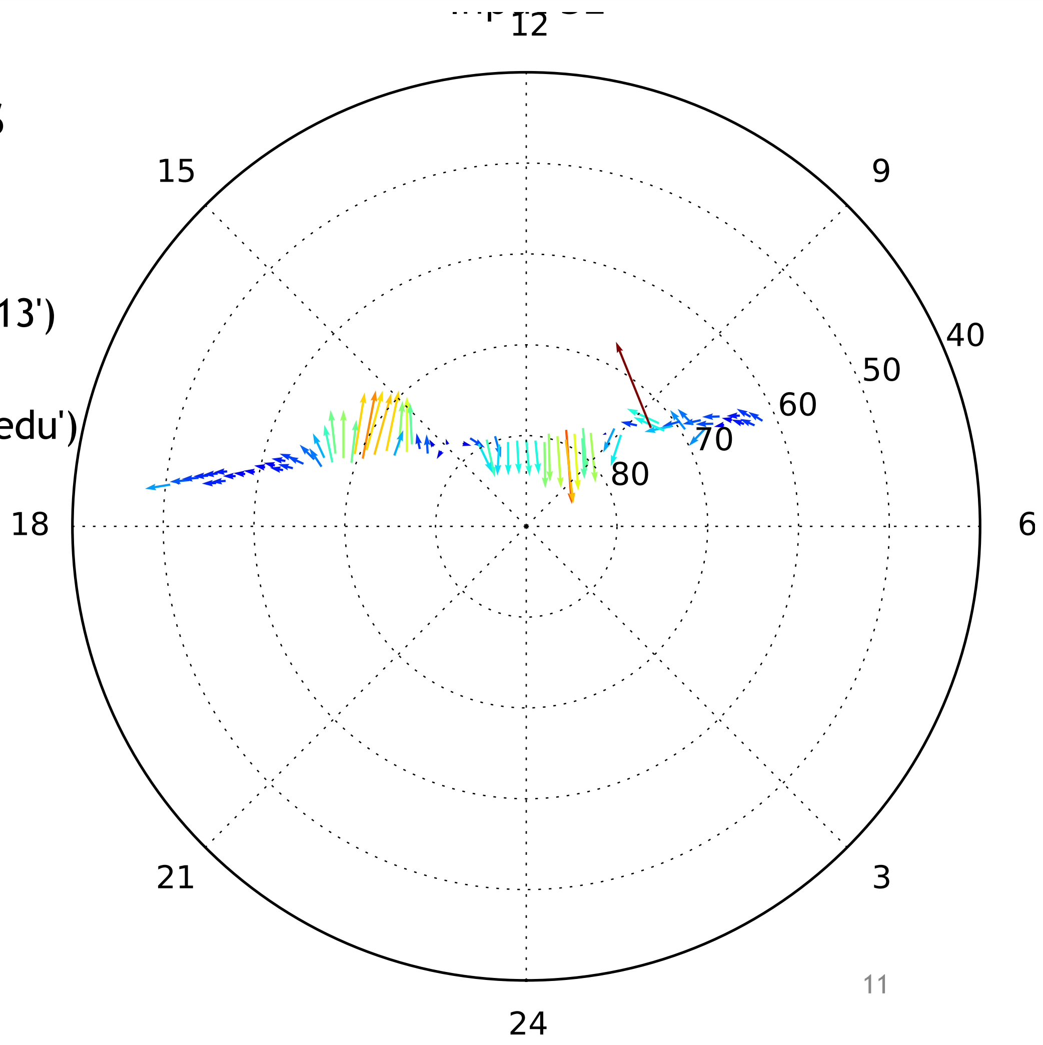
- Interfaces with Madrigal servers

```
dmsp = pysat.Instrument('dmsp', 'ivm', tag='utd', sat_id='f13')

dmsp.download(date1, date2,
  user='Russell+Stoneback', password='rstoneba@utdallas.edu')

dmsp.load(date=date1)

pysat.instruments.dmsp_ivm.polar_plot_by_orbit(dmsp,
                                                orbit=orbit)
```

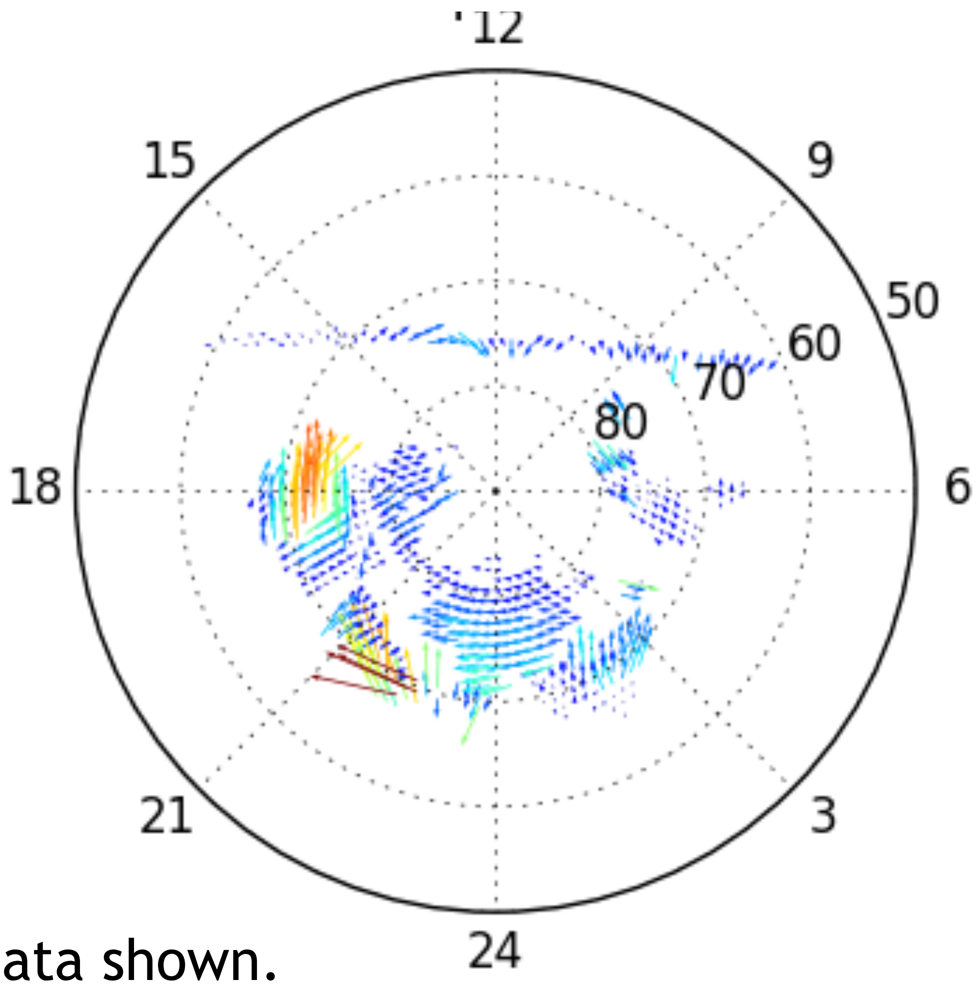


- Performs all DMSP processing needed to support this plot

SuperDARN

- Interfaces with VT servers

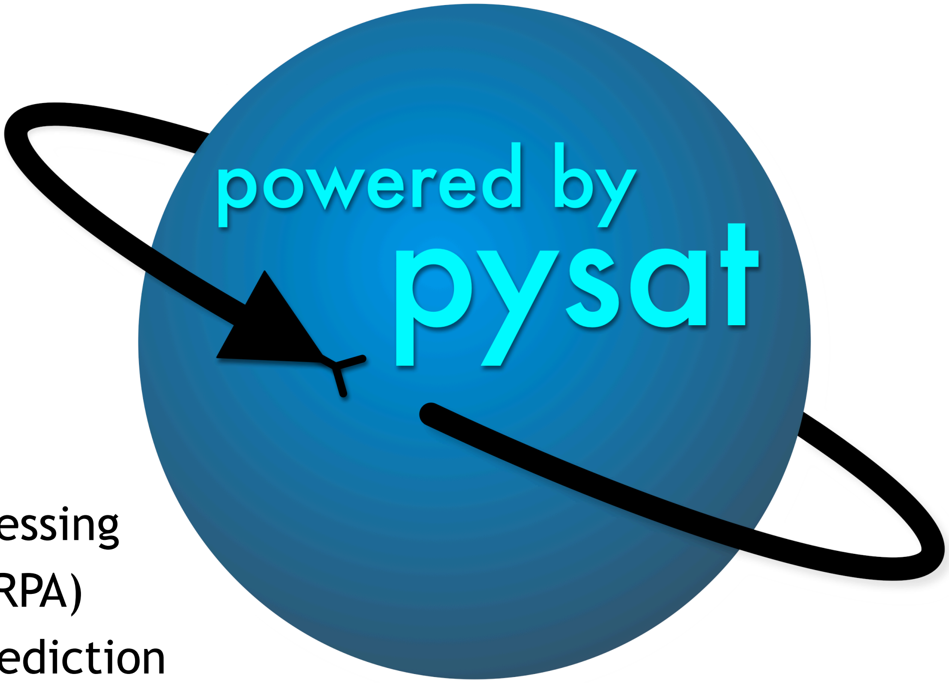
```
sdarn = pysat.Instrument('superdarn', ‘grdex’, ‘north’)
sdarn.download(date1, date2)
sdarn.load(date=date1)
pysat.instruments.superdarn_grdex.polar_plot(sdarn,
                                             time=time)
```



Plot including both DMSP and SuperDARN data shown.

Powered by pysat

- NASA Ionospheric Connections Explorer (ICON)
  - Ion Velocity Meter (IVM) processing
- COSMIC-2 Constellation
  - IVM Processing
- NASA SORTIE CubeSat
  - IVM Processing
- NASA SPORT CubeSat
  - Mission instrument simulation and data processing
- Next-Generation Space Weather Model (NRL/DARPA)
  - Data Assimilation Layer and Measurement Prediction
- UC - Boulder
  - Tomoko Matsuo (TIEGCM Data Assimilation Layer)
- Polar Cap Convection Specification
  - Combine SuperDARN, DMSP, and machine learning to produce full estimates of convection at 2-minute cadence.



Acknowledgements

The research performed by A.G. Burrell was funded by the Chief of Naval Research

pysat source code available at https://github.com/rstoneback/pysat

Or ‘pip install pysat’

Latest code: https://github.com/rstoneback/pysat/tree/develop

Version 2.0 coming soon!