Python Science Analysis Toolkit and Data Interpolating Empirical Orthogonal Functions

Pysat and DINEOFs: A System for System Science

Russell Stoneback
Jeff Klenzing
Angeline Burrell
Rod Heelis

UTD

Instrument Object

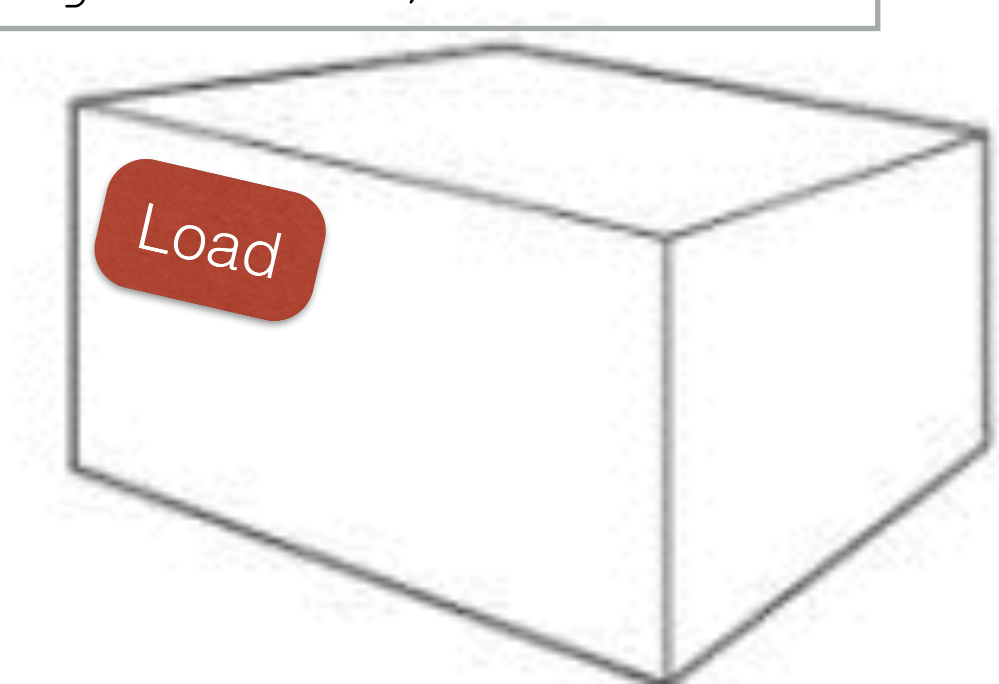
```
ivm = pysat.Instrument(platform='cnofs',
                       name='ivm')
```

```
vefi = pysat.Instrument(platform='cnofs',
                        name='vefi',
                        tag='dc_b')
```

```
cosmic = pysat.Instrument(platform='cosmic2013',
                          name='gps',
                          tag='ionprf',
                          clean_level='clean')
```

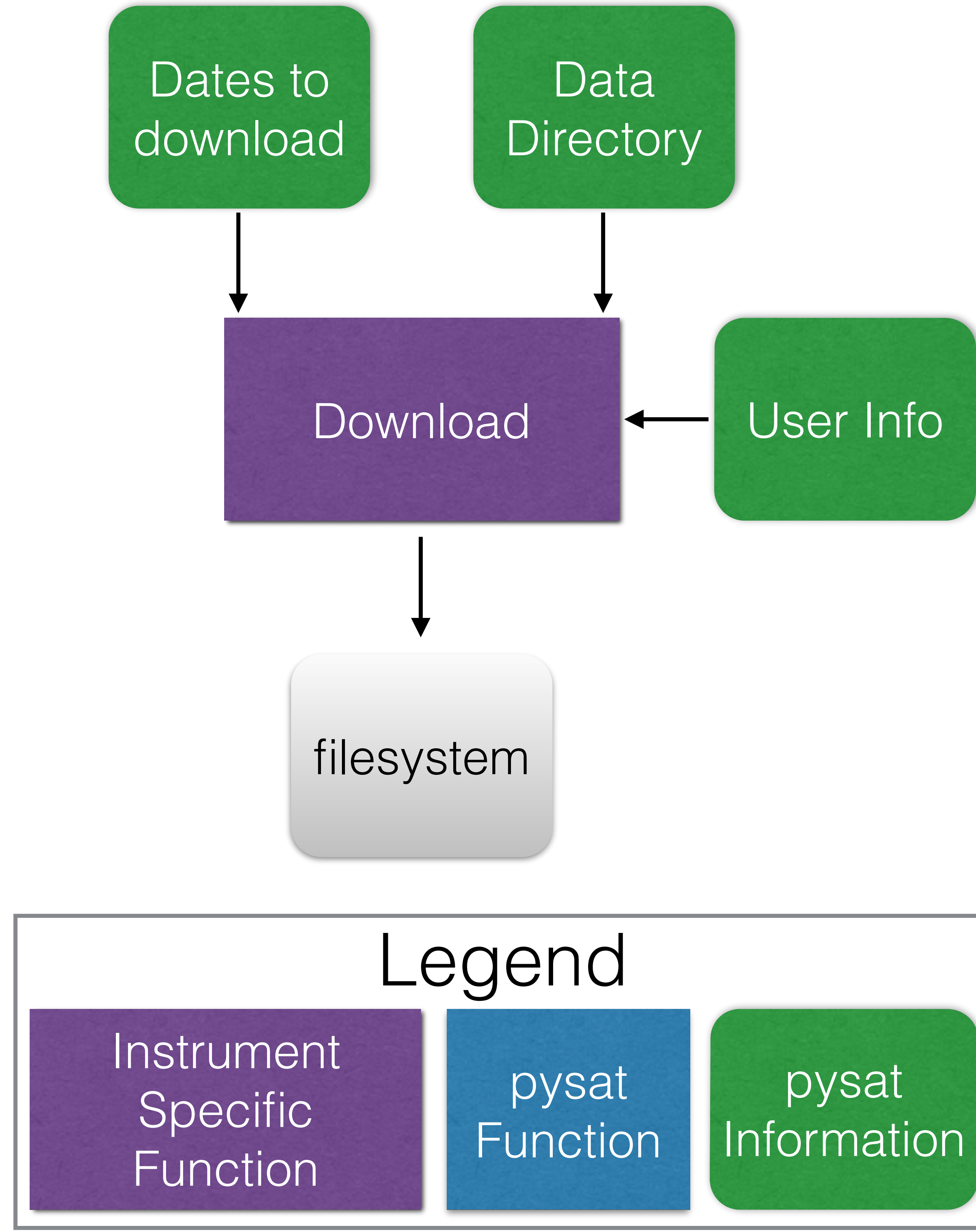
```
darn = pysat.Instrument(platform='superdarn',
                        name='grdex',
                        tag='north')
```

```
ivm.load(2012,1)
vefi.load(2012,1)
cosmic.load(2012,1)
darn.load(2012,1)
```



Supports Data Downloads

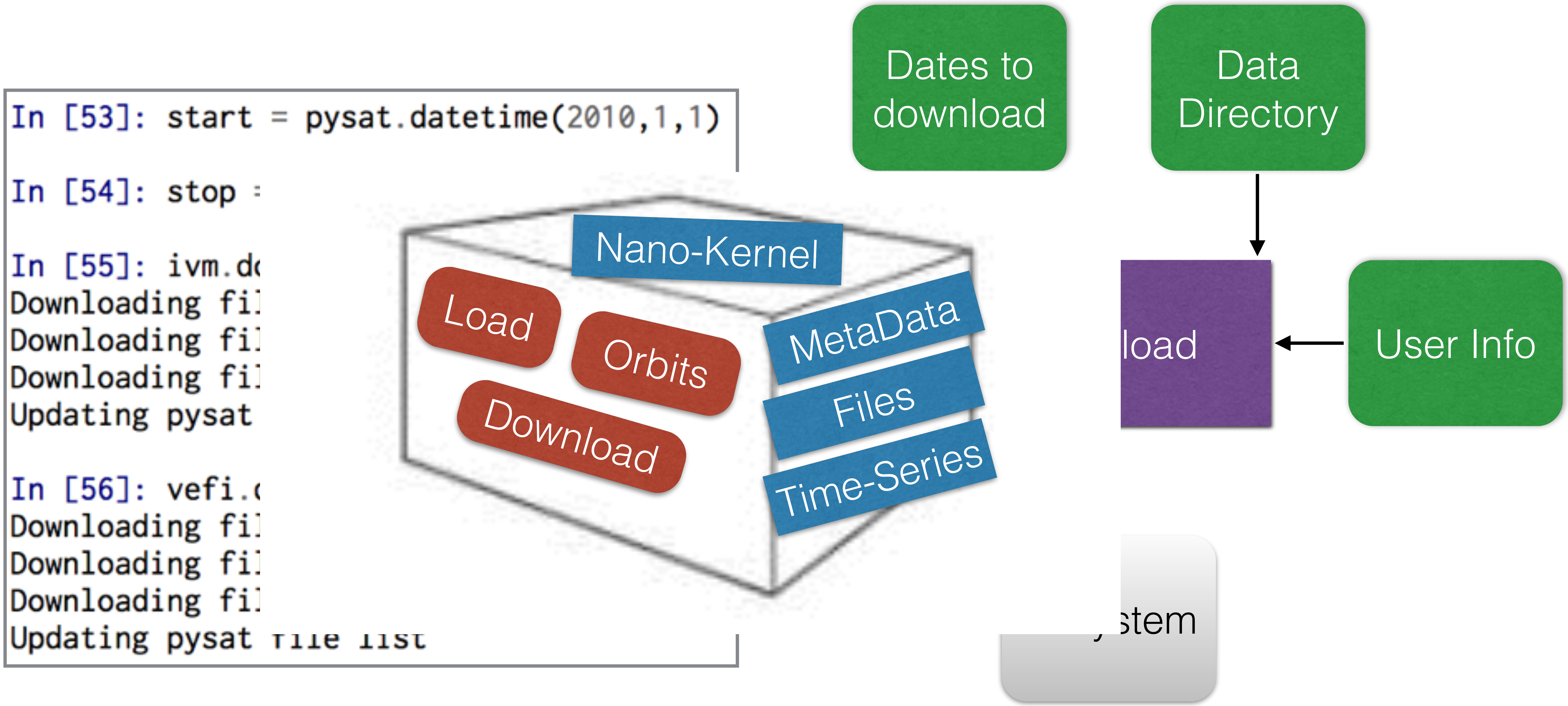
```
In [53]: start = pysat.datetime(2010,1,1)

In [54]: stop = pysat.datetime(2010,1,3)

In [55]: ivm.download(start, stop)
Downloading file for 01/01/10
Downloading file for 01/02/10
Downloading file for 01/03/10
Updating pysat file list

In [56]: vefi.download(start, stop)
Downloading file for 01/01/10
Downloading file for 01/02/10
Downloading file for 01/03/10
Updating pysat file list
```



Data Organization

User Set Dir/platform/name/tag/

Supports Data Downloads



Data Organization

User Set Dir/platform/name/tag/

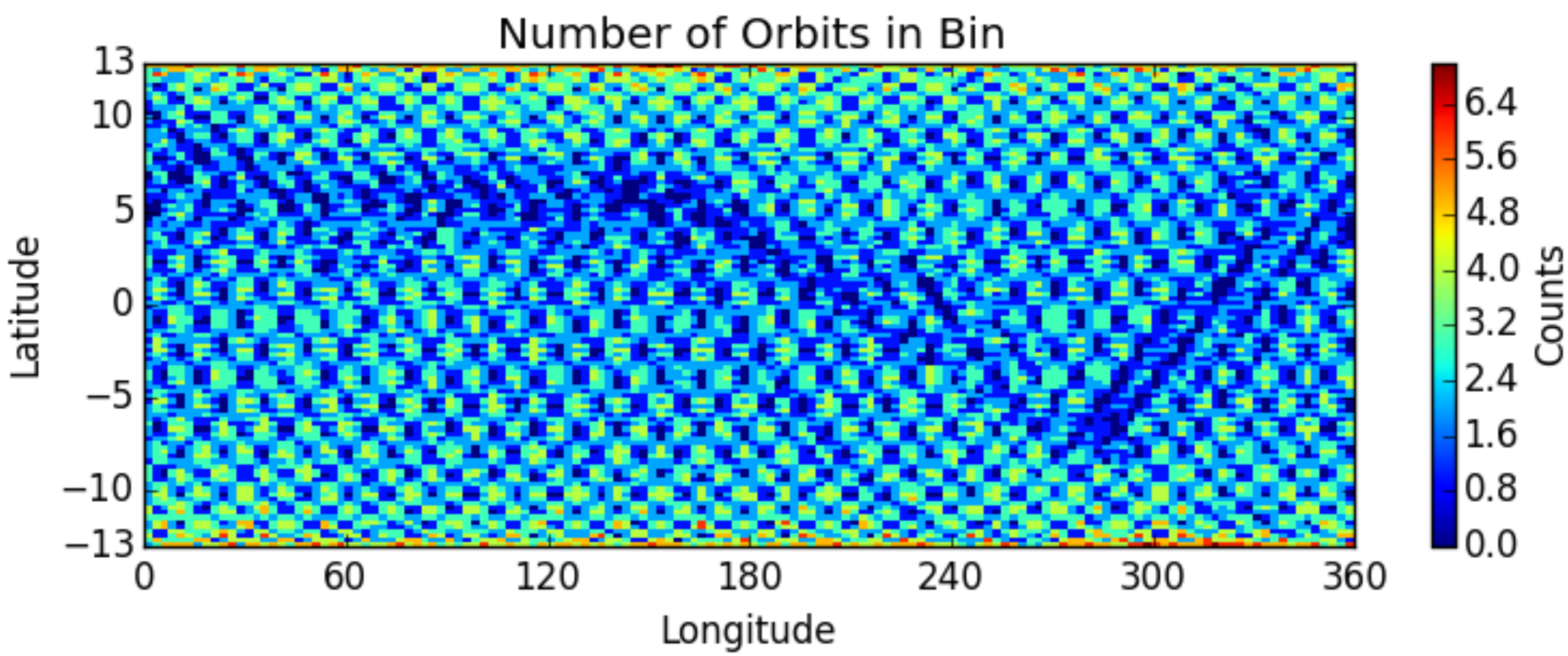
Legend

Instrument Specific Function	pysat Function	pysat Information

Instrument Independent Analysis

Attach a queue of functions to be applied whenever data is loaded, set and forget. (nano-kernel)

```
vefi = pysat.Instrument(platform='cnofs', name='vefi', tag='dc_b',
                        clean_level=None)
# define function to remove torque-rod firings
def filter_vefi(inst):
    idx, = np.where(vefi['B_flag']==0)
    vefi.data = vefi.data.iloc[idx]
    return
vefi.custom.add(filter_vefi,'modify')
ans = pysat.ssnl.occur_prob.by_orbit2D(vefi, [0,360,144],
'longitude', [-13,13,104], 'latitude', ['dB_mer'], [0.])
# plot commands go here
```

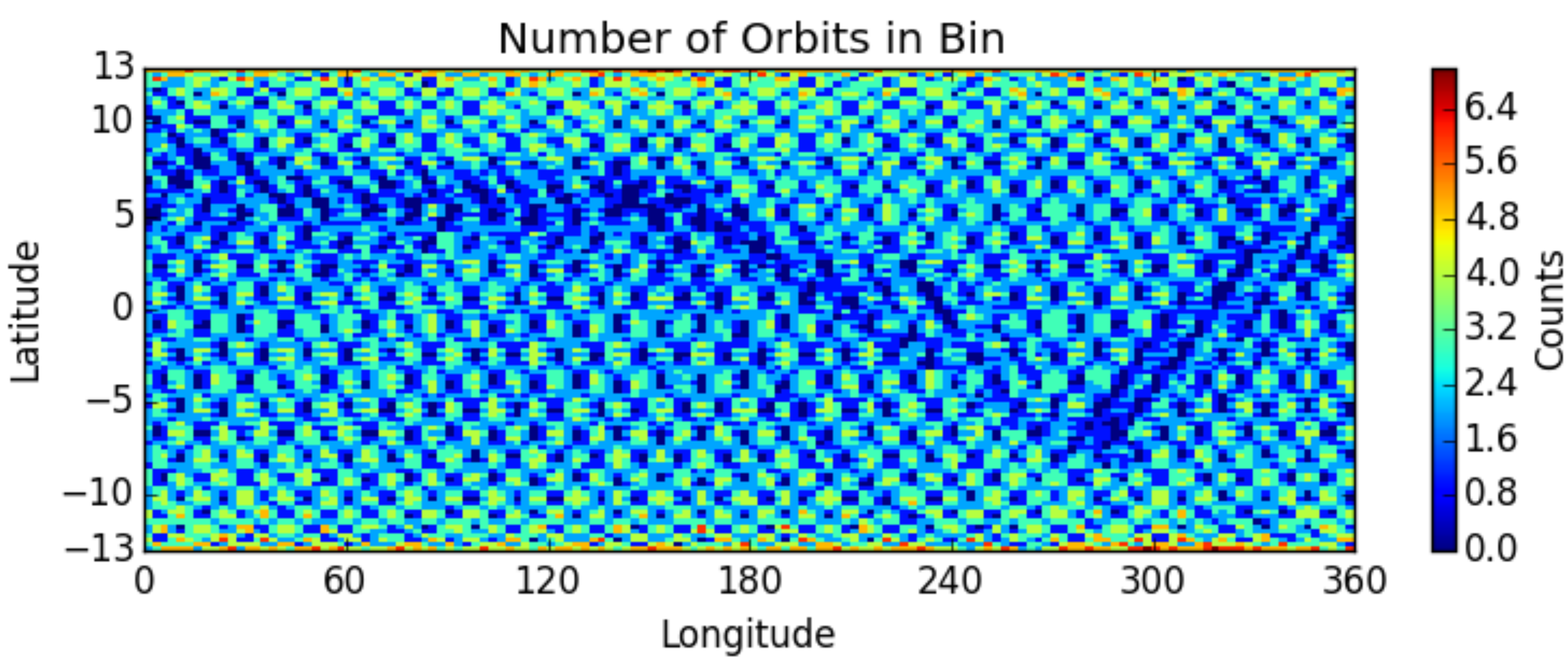


Instrument Independent Analysis

Attach a queue of functions to be applied whenever data is loaded, set and forget. (nano-kernel)

```
vefi = pysat.Instrument(platform='cnofs', name='vefi', tag='dc_b',
                        clean_level=None)
# define function to remove torque-rod firings
def filter_vefi(inst):
    idx, = np.where(vefi['B_flag']==0)
    vefi.data = vefi.data.iloc[idx]
    return
vefi.custom.add(filter_vefi,'modify')
ans = pysat.ssnl.occur_prob.by_orbit2D(vefi, [0,360,144],
'longitude', [-13,13,104], 'latitude', ['dB_mer'], [0.])
# plot commands go here
```

Instrument Independent Analysis Routine



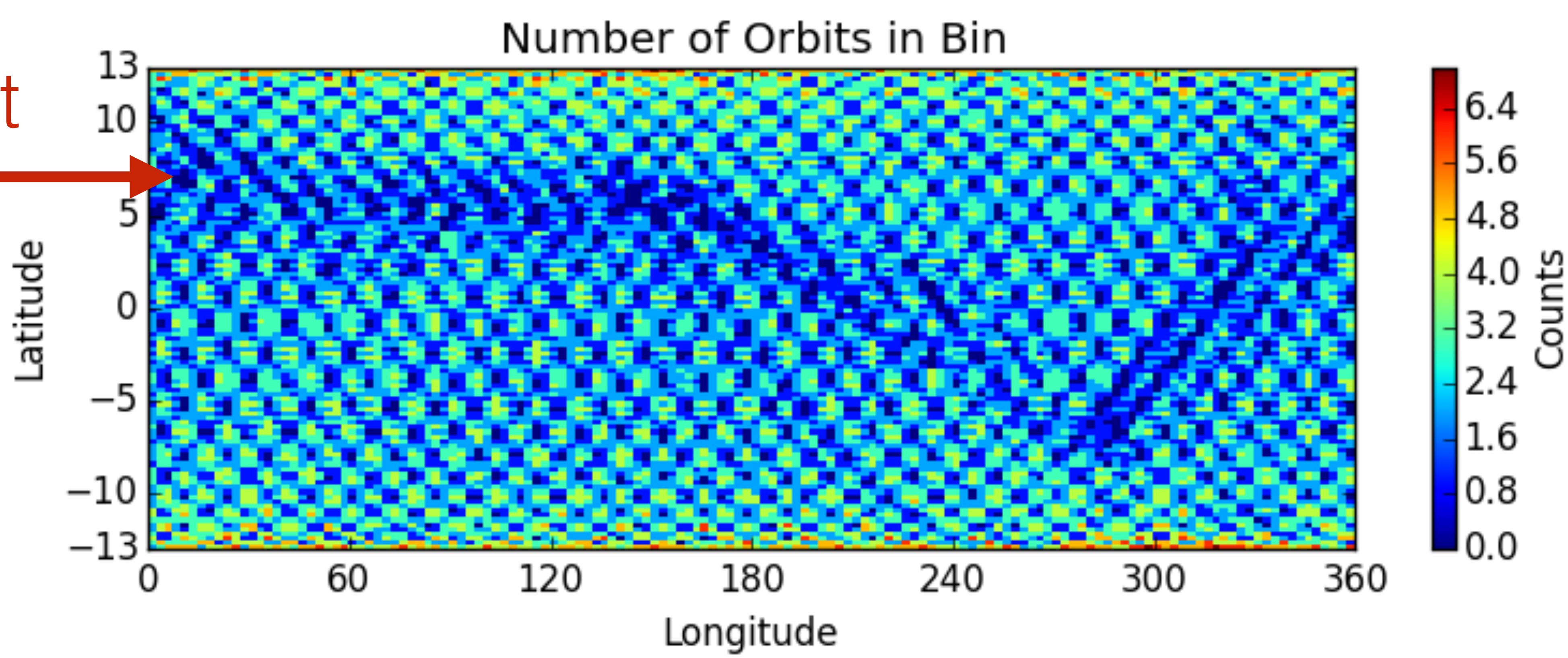
Instrument Independent Analysis

Attach a queue of functions to be applied whenever data is loaded, set and forget. (nano-kernel)

```
vefi = pysat.Instrument(platform='cnofs', name='vefi', tag='dc_b',
                        clean_level=None)
# define function to remove torque-rod
def filter_vefi(inst):
    idx, = np.where(vefi['B_flag']==0)
    vefi.data = vefi.data.iloc[idx]
    return
vefi.custom.add(filter_vefi,'modify')
ans = pysat.ssnl.occur_prob.by_orbit2D(vefi, [0,360,144],
'longitude', [-13,13,104], 'latitude', ['dB_mer'], [0.])
# plot commands go here
```

This filter function modifies the vefi data available in the probability routine

Firings at Mag Eq.



Averaging Drifts and Profiles

```
# instantiate IVM Object
ivm = pysat.Instrument('cnofs','ivm','','clean')
ivm.custom.add(restrictMLAT, 'modify', maxMLAT=25.)
ivm.bounds(startDate, stopDate)
ivmResults = pysat.ssnl.median2D(ivm, [0,360,24], 'apex_long',
                     [0,24,24], 'mlt', ['iv_mer'])

# create CODMIC instrument object
cosmic = pysat.Instrument('cosmic2013', 'gps','ionprf',
                          'clean', altitude_bin=3)
# apply custom functions to all data that is loaded through cosmic
cosmic.custom.add(addApexLong, 'add')
cosmic.custom.add(filterMLAT, 'modify', mlatRange=(0.,10.) )
cosmic.custom.add(addlogNm, 'add')
cosmic.custom.add(addTopsideScaleHeight, 'add')

# do an average of multiple COSMIC data products from startDate
through stopDate
cosmic.bounds(startDate, stopDate)
cosmicResults = pysat.ssnl.median2D(cosmic, [0,360,24], 'apex_long',
     [0,24,24],'edmaxlct', ['profiles', 'edmaxalt', 'lognm', 'thf2'])
# plot commands
```

COSMIC profiles are aligned by altitude before averaging

Averaging Drifts and Profiles

```
# instantiate IVM Object
ivm = pysat.Instrument('cnofs','ivm','','clean')
ivm.custom.add(restrictMLAT, 'modify', maxMLAT=25.)
ivm.bounds(startDate, stopDate)
ivmResults = pysat.ssnl.median2D(ivm, [0,360,24], 'apex_long',
                     [0,24,24], 'mlt', ['iv_mer'])

# create CODMIC instrument object
cosmic = pysat.Instrument('cosmic2013', 'gps','ionprf',
                          'clean', altitude_bin=3)
# apply custom functions to all data that is loaded through cosmic
cosmic.custom.add(addApexLong, 'add')
cosmic.custom.add(filterMLAT, 'modify', mlatRange=(0.,10.) )
cosmic.custom.add(addlogNm, 'add')
cosmic.custom.add(addTopsideScaleHeight, 'add')

# do an average of multiple COSMIC data products from startDate
through stopDate
cosmic.bounds(startDate, stopDate)
cosmicResults = pysat.ssnl.median2D(cosmic, [0,360,24], 'apex_long',
     [0,24,24],'edmaxlct', ['profiles', 'edmaxalt', 'lognm', 'thf2'])
# plot commands
```

Instrument Independent
Analysis Routine

Instrument Independent
Analysis Routine

COSMIC profiles are aligned by altitude before averaging

Averaging Drifts and Profiles

Instrument Independent Analysis Routine

```
# instantiate IVM Object
ivm = pysat.Instrument('cnofs','ivm','','clean')
ivm.custom.add(restrictMLAT, 'modify', maxMLAT=25.)
```

Custom functions adds magnetic coordinates, filters data, and adds log density and topside scale height. All the processing needed to compare with IVM.

```
# apply custom functions to all data that is loaded through cosmic
cosmic.custom.add(addApexLong, 'add')
cosmic.custom.add(filterMLAT, 'modify', mlatRange=(0.,10.) )
cosmic.custom.add(addlogNm, 'add')
cosmic.custom.add(addTopsideScaleHeight, 'add')

# do an average of multiple COSMIC data products from startDate
through stopDate
cosmic.bounds(startDate, stopDate)
cosmicResults = pysat.ssnl.median2D(cosmic, [0,360,24], 'apex_long',
    [0,24,24],'edmaxlct', ['profiles', 'edmaxalt', 'lognm', 'thf2'])
# plot commands
```

Instrument Independent Analysis Routine

COSMIC profiles are aligned by altitude before averaging

Averaging Drifts and Profiles

Instrument Independent Analysis Routine

```
# instantiate IVM Object
ivm = pysat.Instrument('cnofs','ivm','','clean')
ivm.custom.add(restrictMLAT, 'modify', maxMLAT=25.)
ivm.bounds(startDate, stopDate)
ivmResults = pysat.ssnl.median2D(ivm, [0,360,24], 'apex_long',
                    [0,24,24], 'mlt', ['iv_mer'])
```

1D

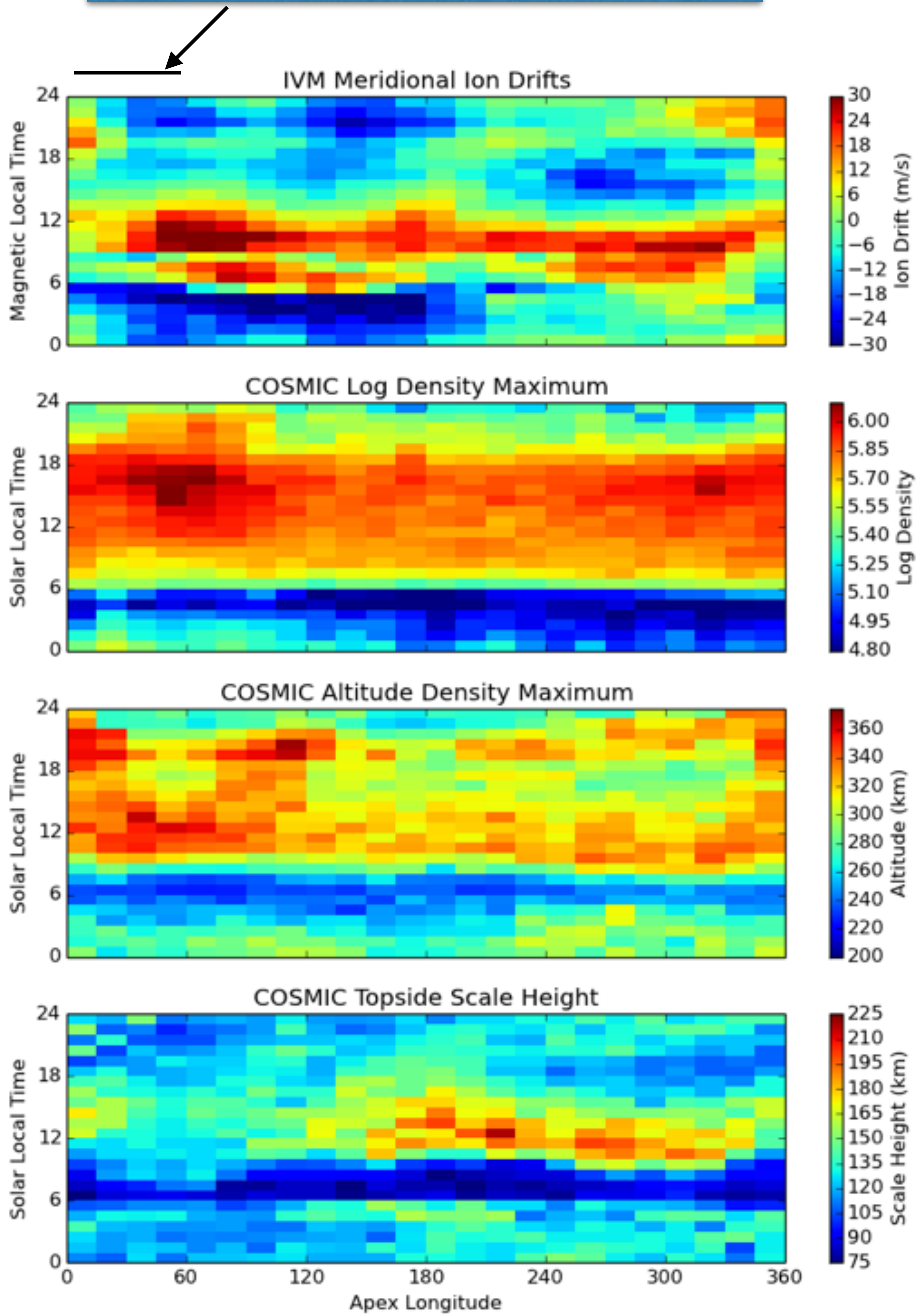
```
# create CODMIC instrument object
cosmic = pysat.Instrument('cosmic2013', 'gps','ionprf',
                         'clean', altitude_bin=3)
# apply custom functions to all data that is loaded through cosmic
cosmic.custom.add(addApexLong, 'add')
cosmic.custom.add(filterMLAT, 'modify', mlatRange=(0.,10.) )
cosmic.custom.add(addlogNm, 'add')
cosmic.custom.add(addTopsideScaleHeight, 'add')

# do an average of multiple COSMIC data products from startDate
through stopDate
cosmic.bounds(startDate, stopDate)
cosmicResults = pysat.ssnl.median2D(cosmic, [0,360,24], 'apex_long',
     [0,24,24],'edmaxlct', ['profiles', 'edmaxalt', 'lognm', 'thf2'])
# plot commands
```

Instrument Independent Analysis Routine

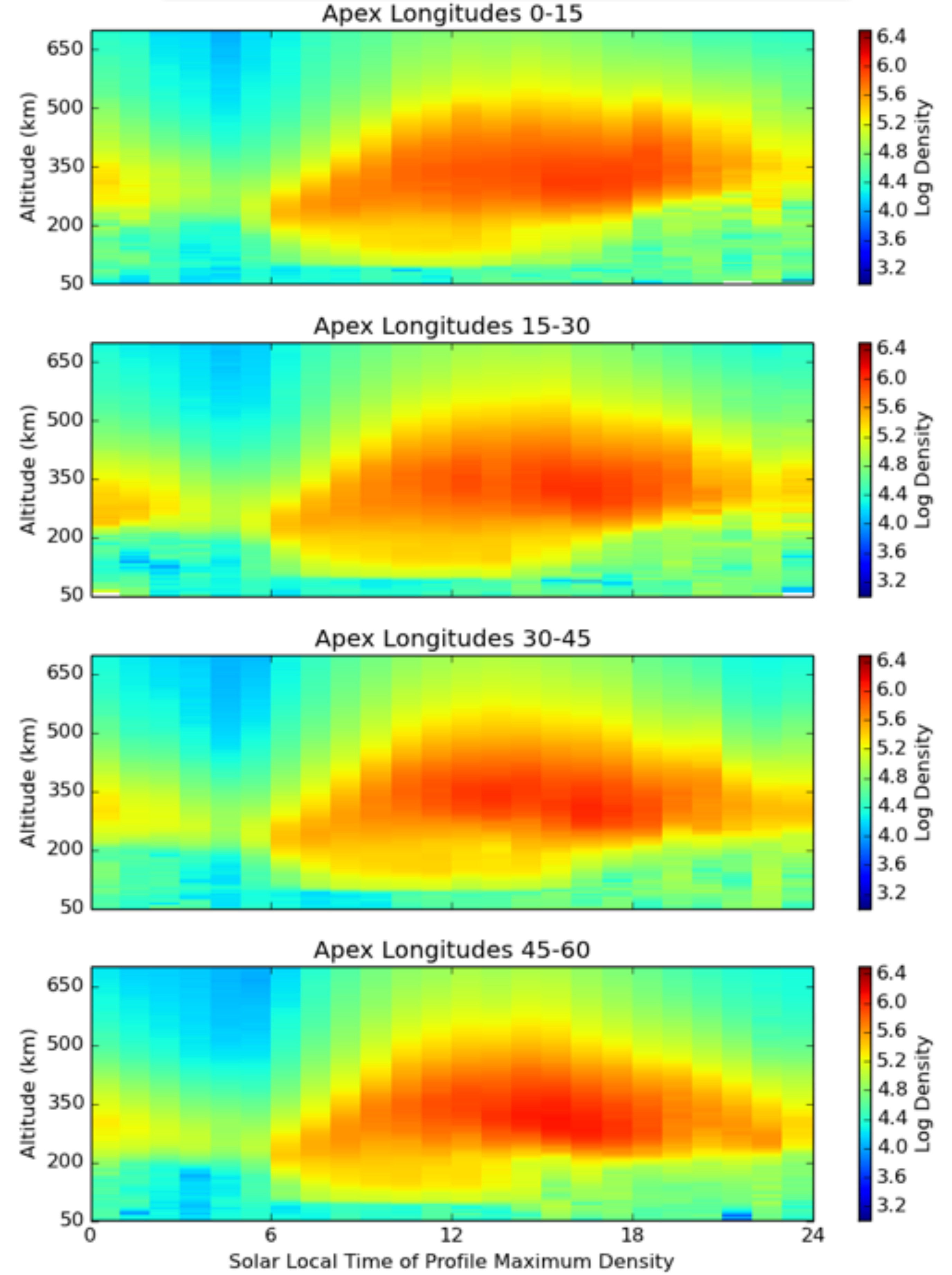
3D 1D 1D 1D

COSMIC profiles are aligned by altitude before averaging

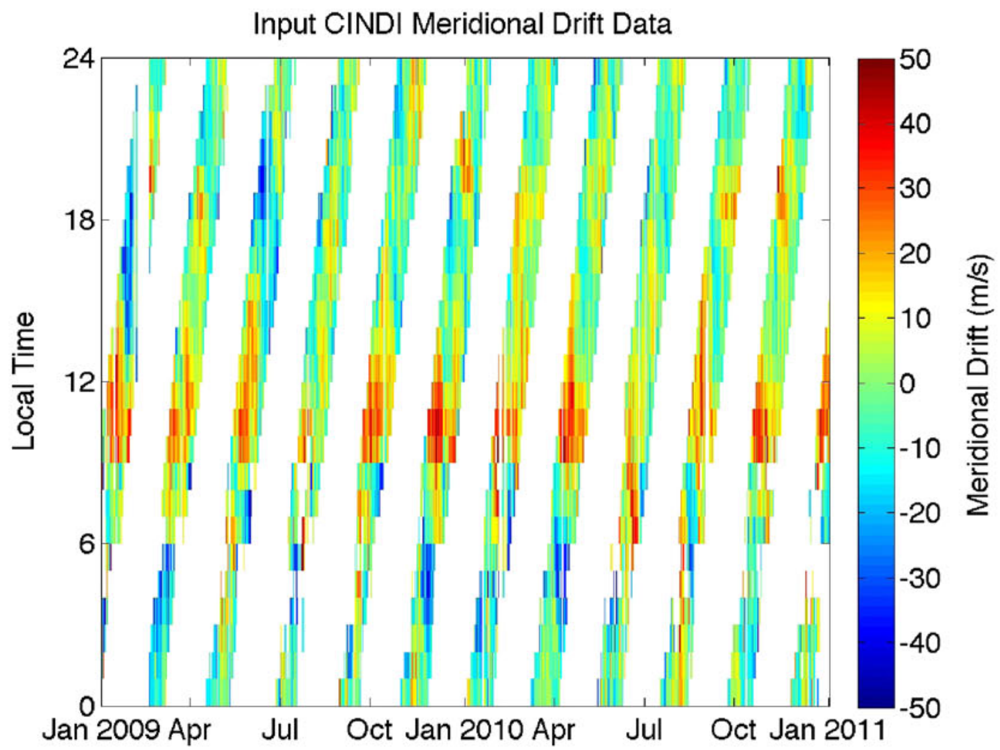
Upward to downward drifts



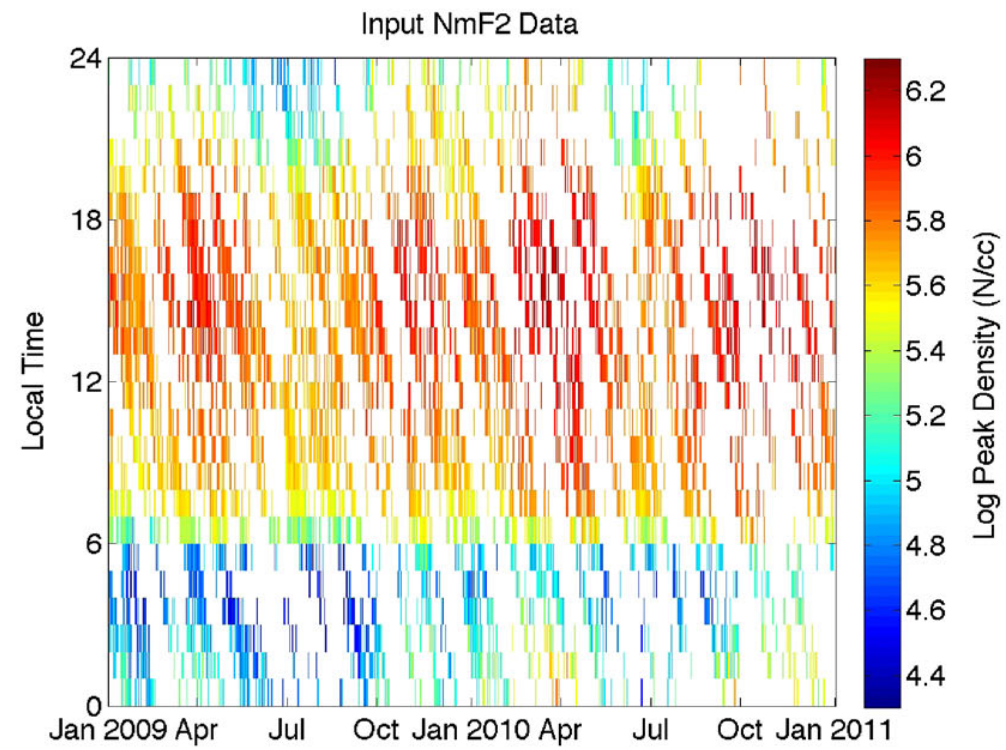
Influence of drifts observed in COSMIC density profiles



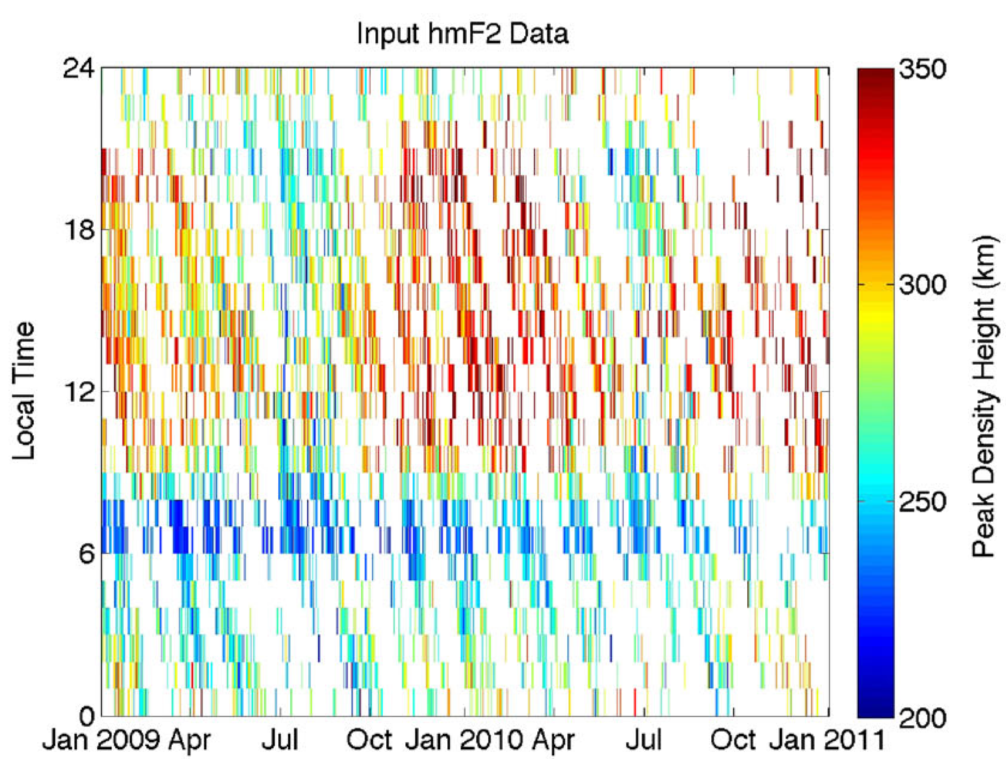
Equatorial Space Weather



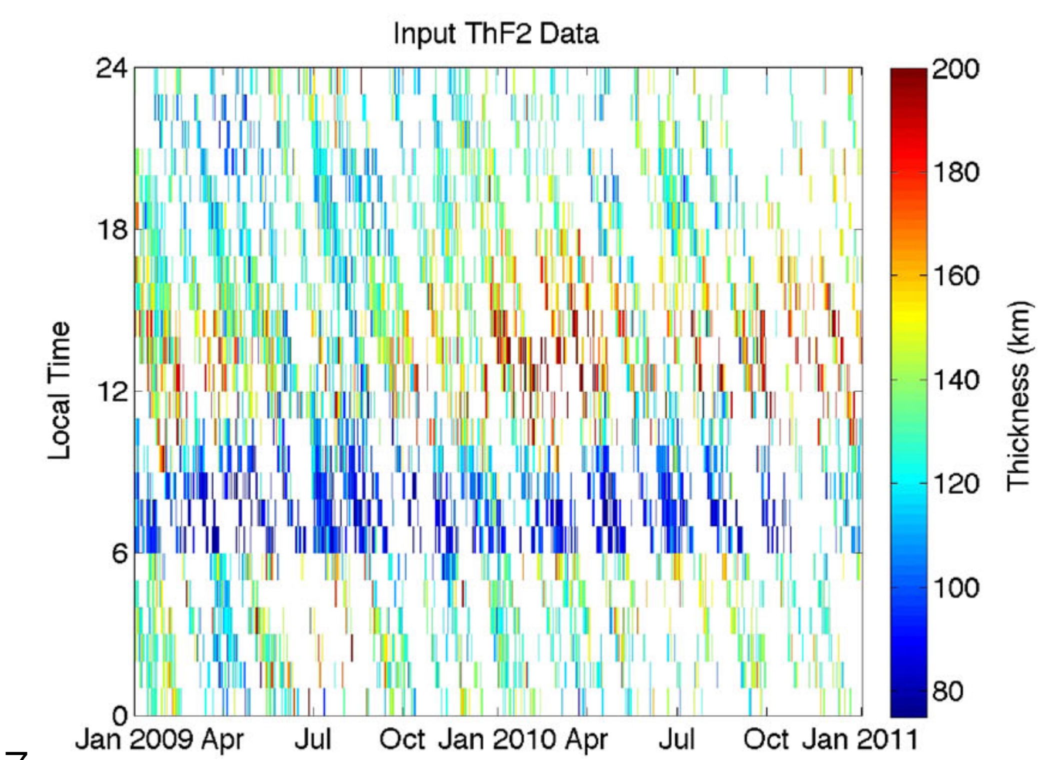
(a) CINDI Meridional Drifts



(b) COSMIC Peak Density

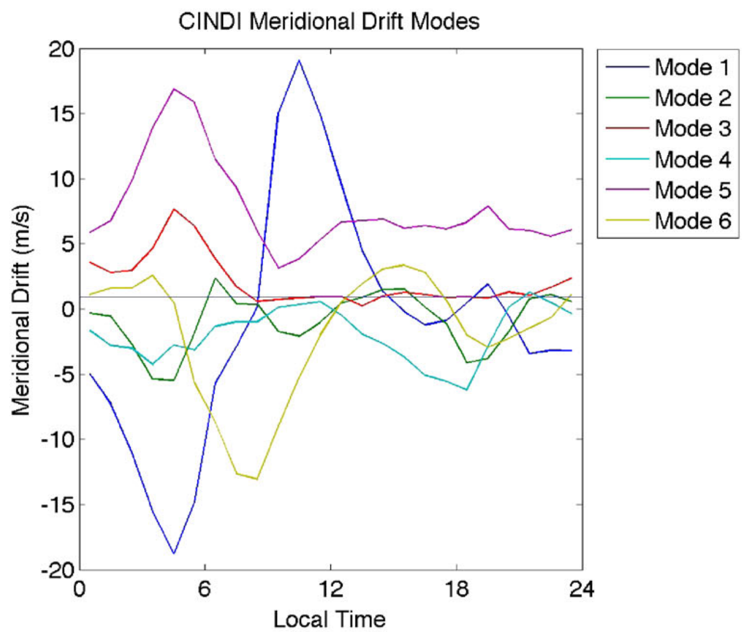


(c) COSMIC peak density height

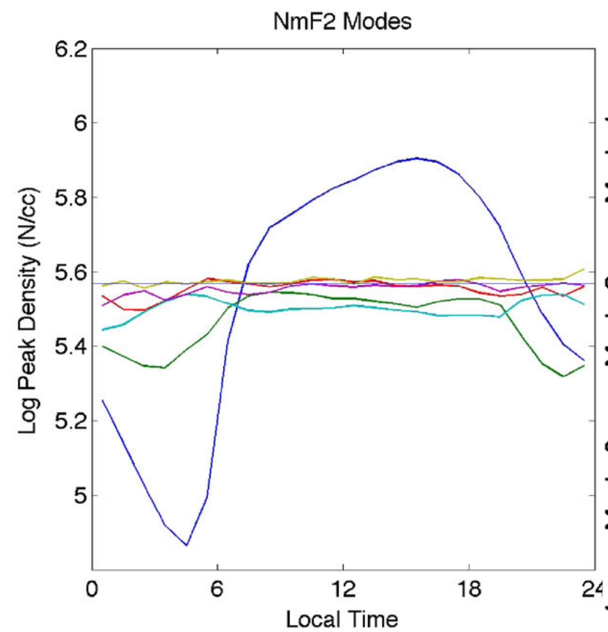


(d) COSMIC thickness

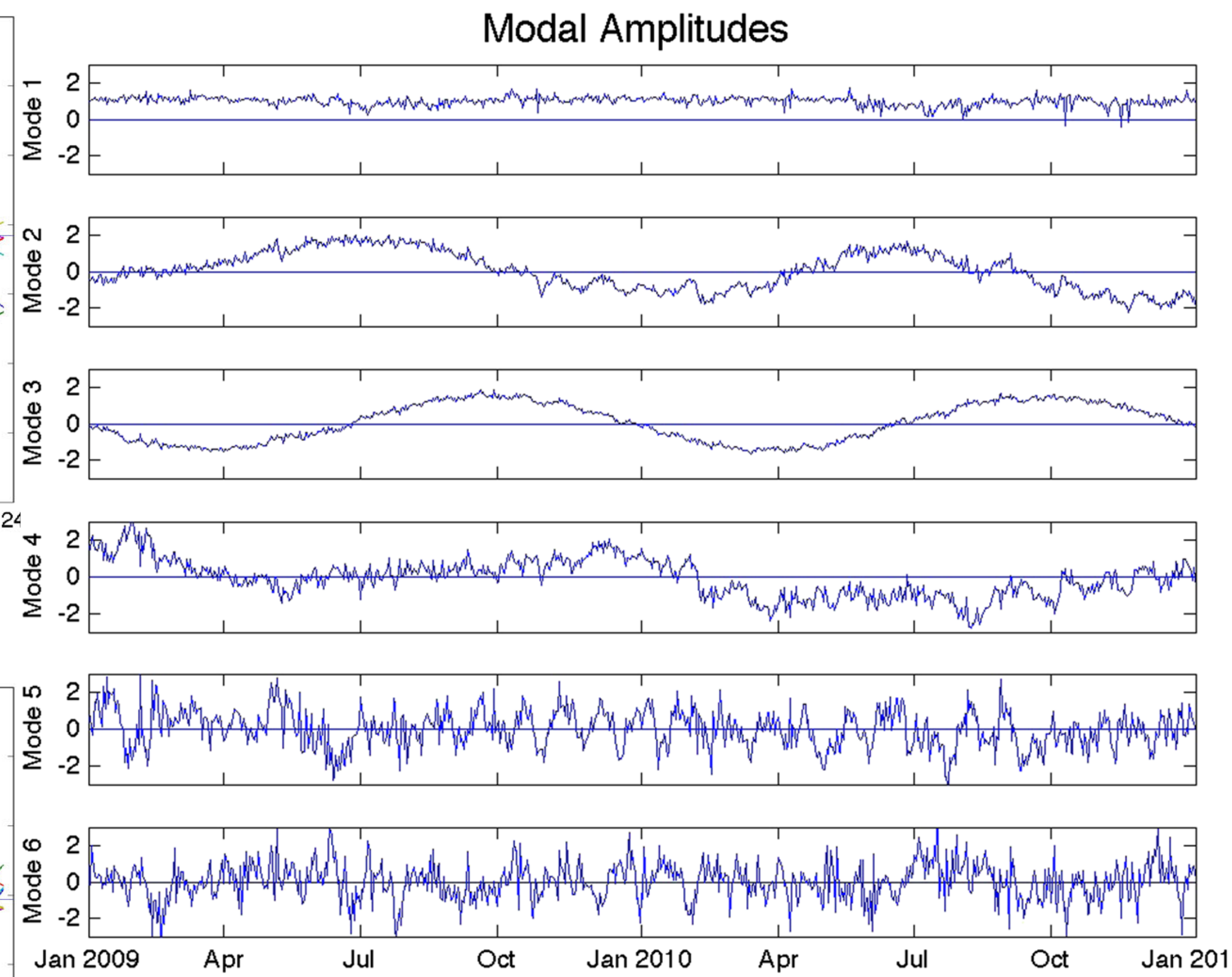
DINEOF Modes

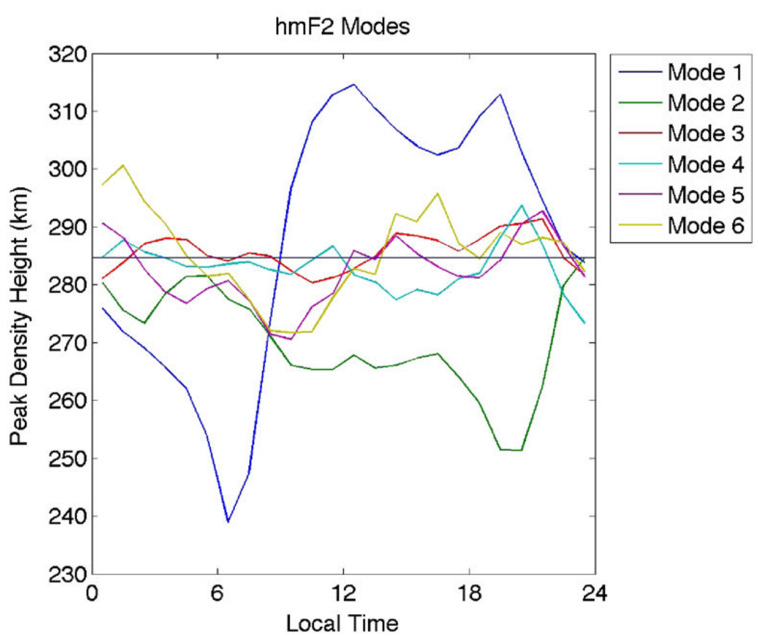


(a) CINDI Meridional Drifts

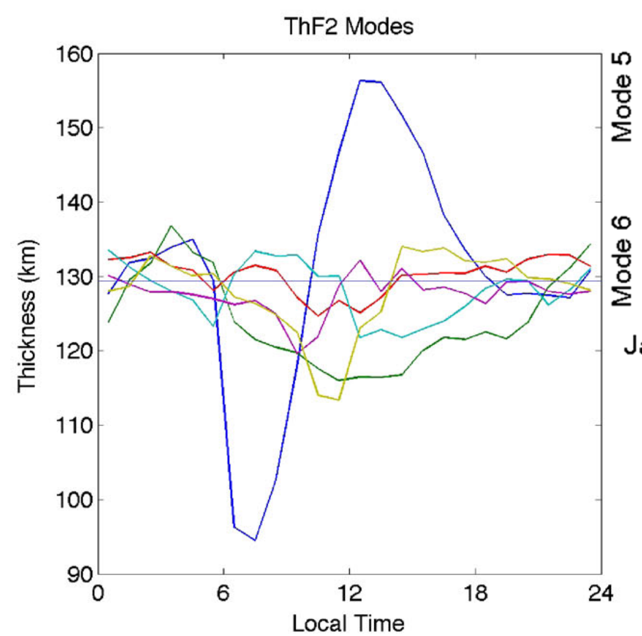


(b) COSMIC Peak Density



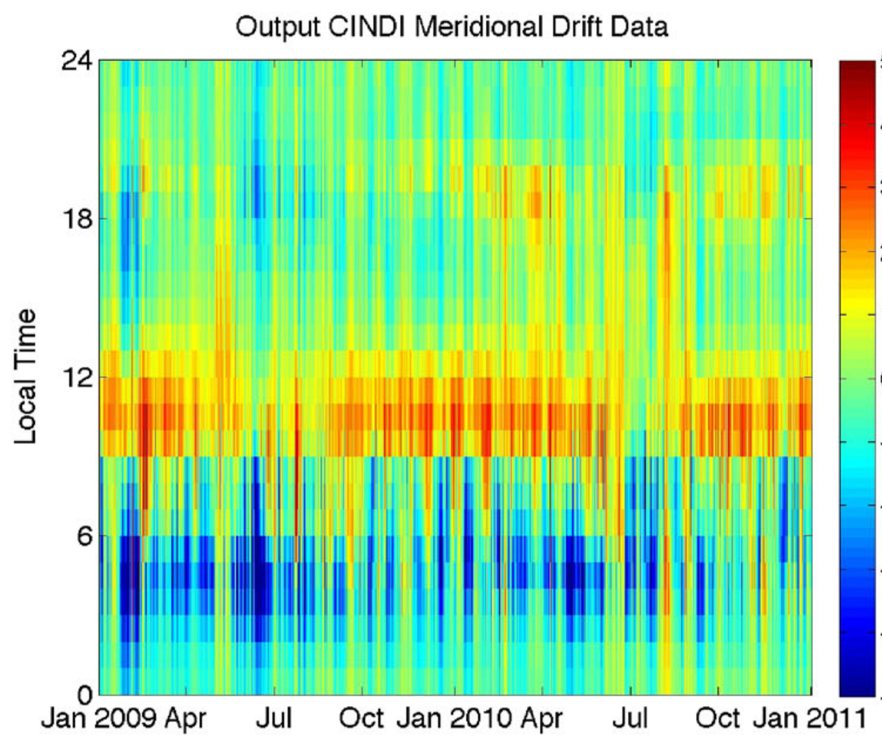


(c) COSMIC peak density height

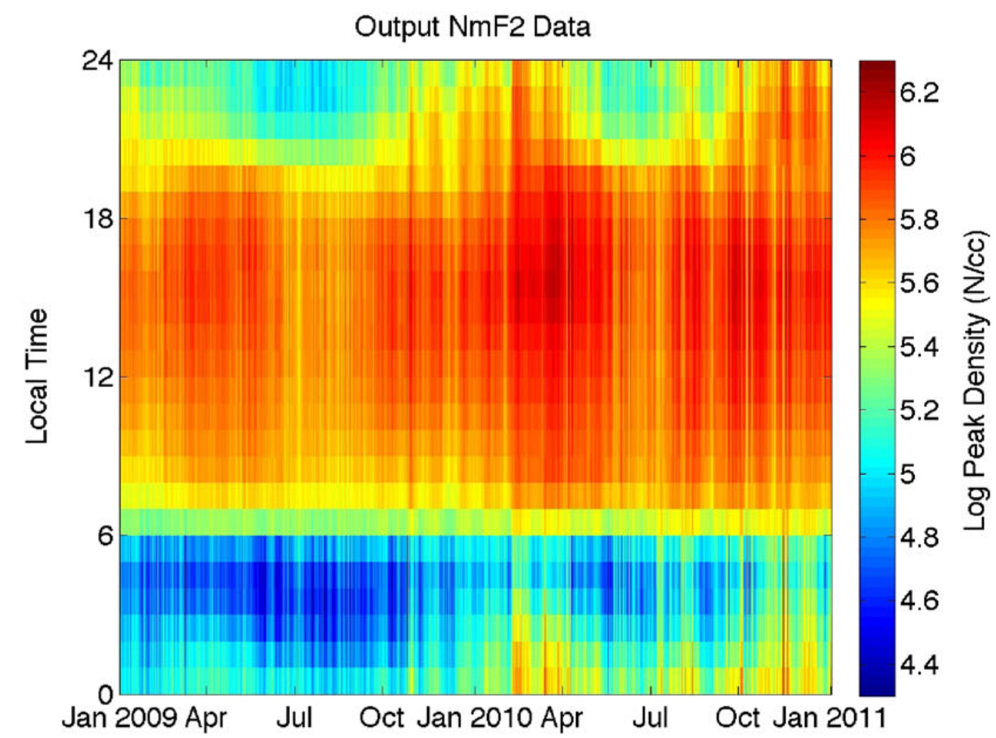


(d) COSMIC thickness

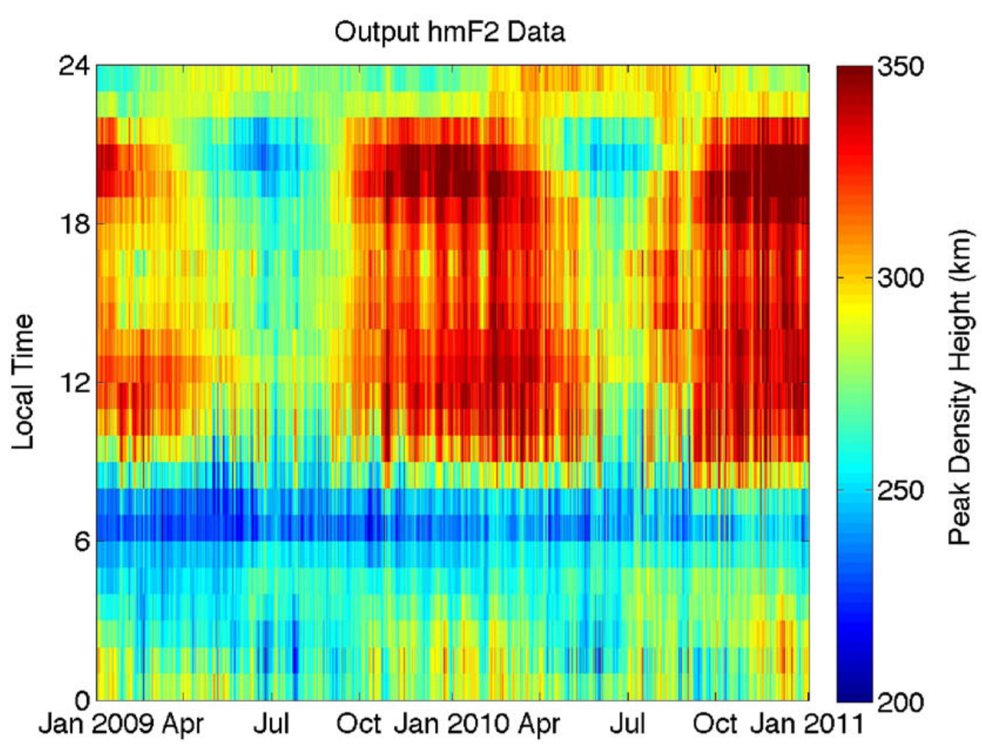
DINEOF Results



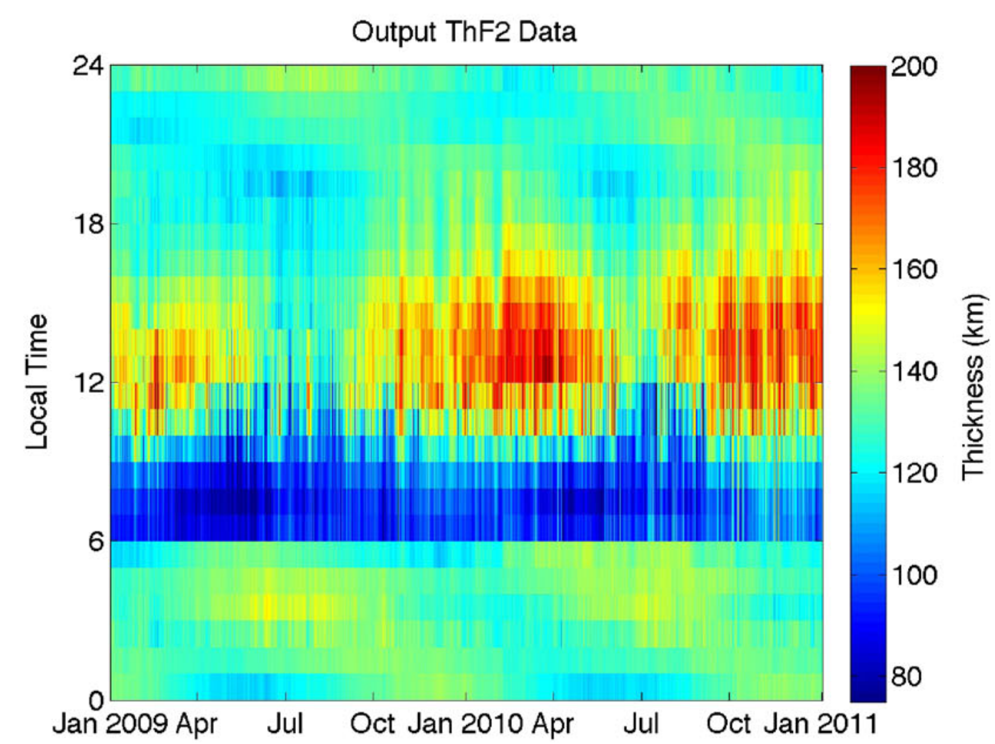
(a) CINDI Meridional Drifts



(b) COSMIC Peak Density

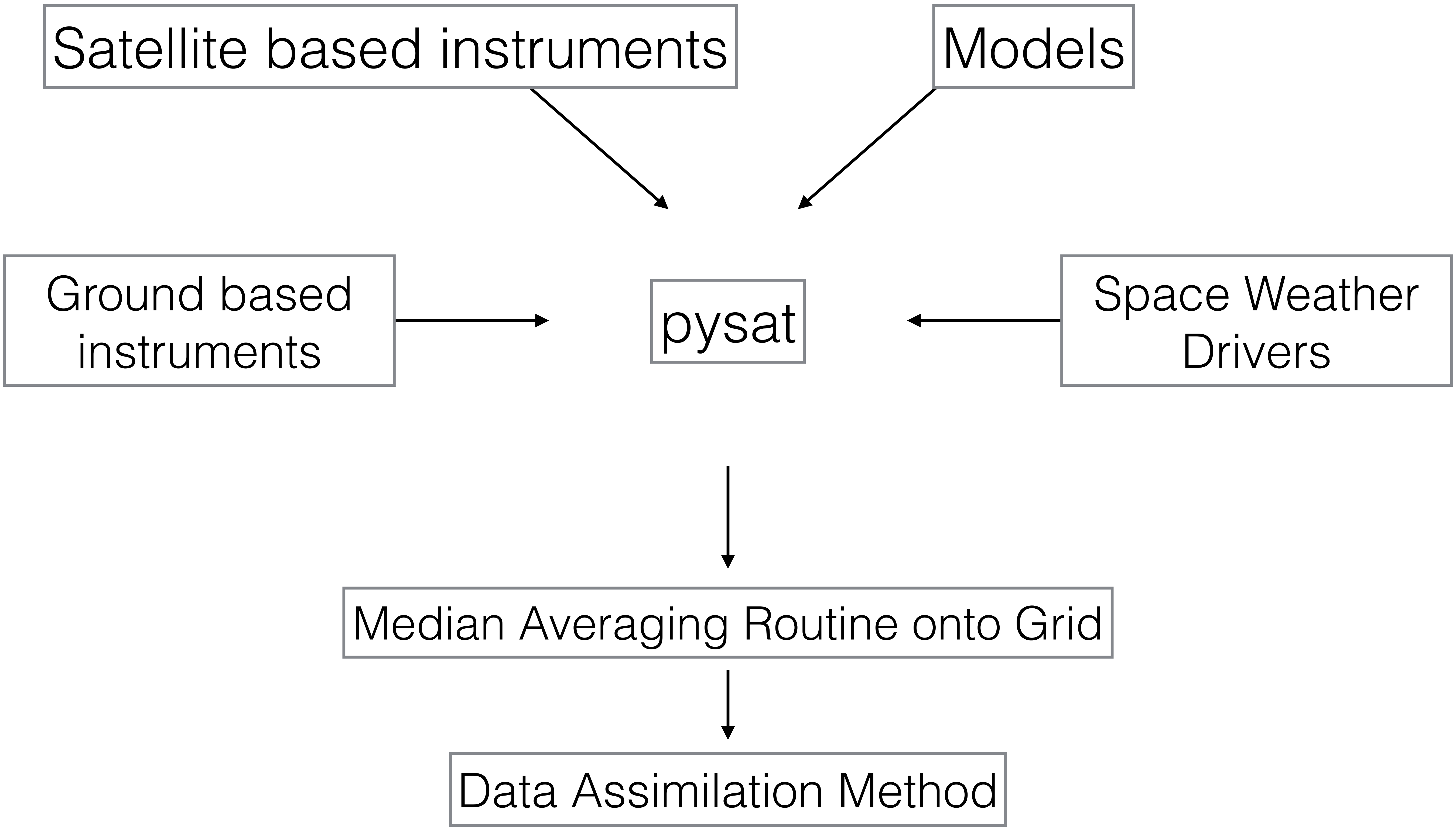


(c) COSMIC peak density height

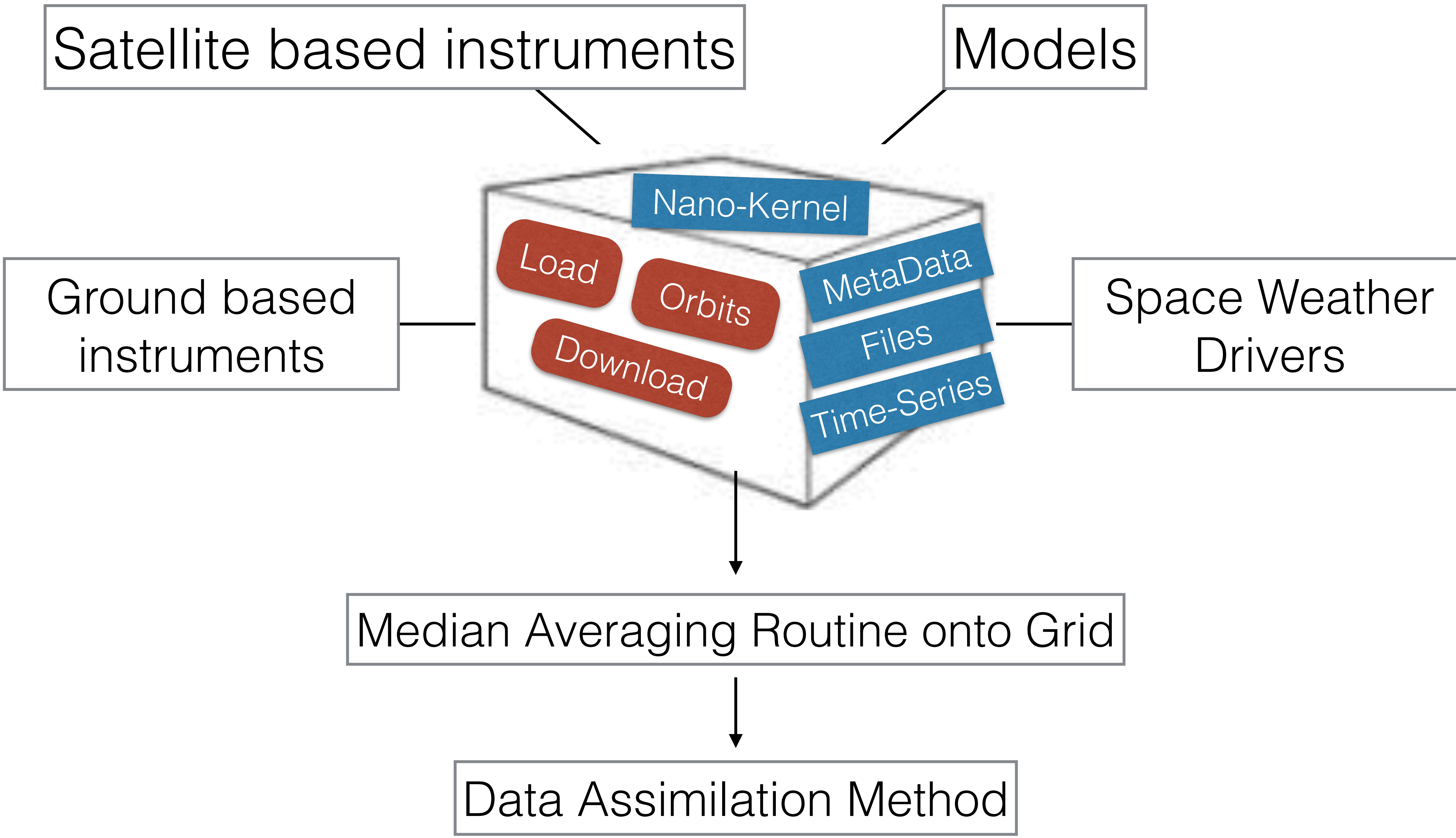


(d) COSMIC thickness

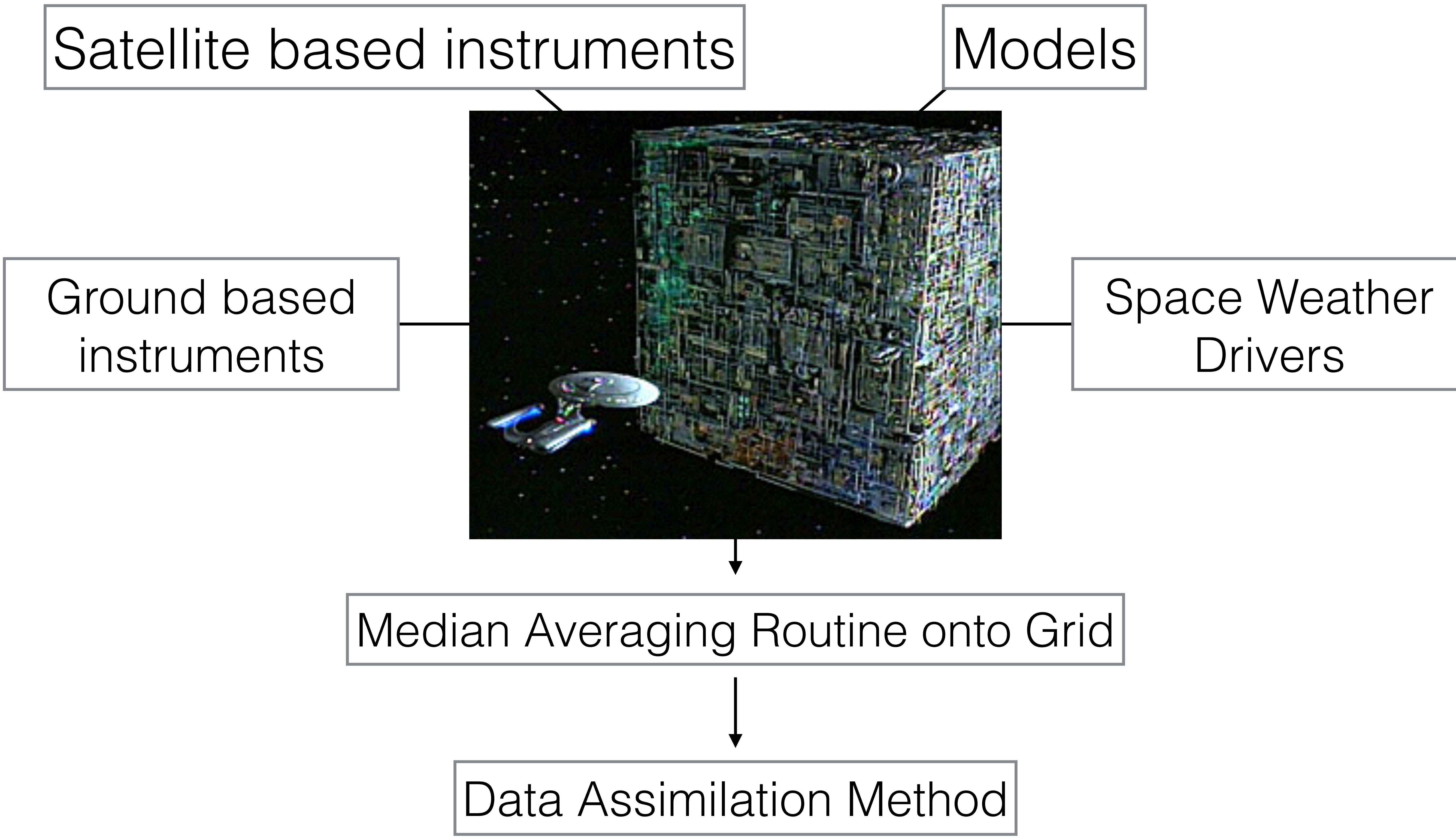
System for System Science



System for System Science



System for System Science

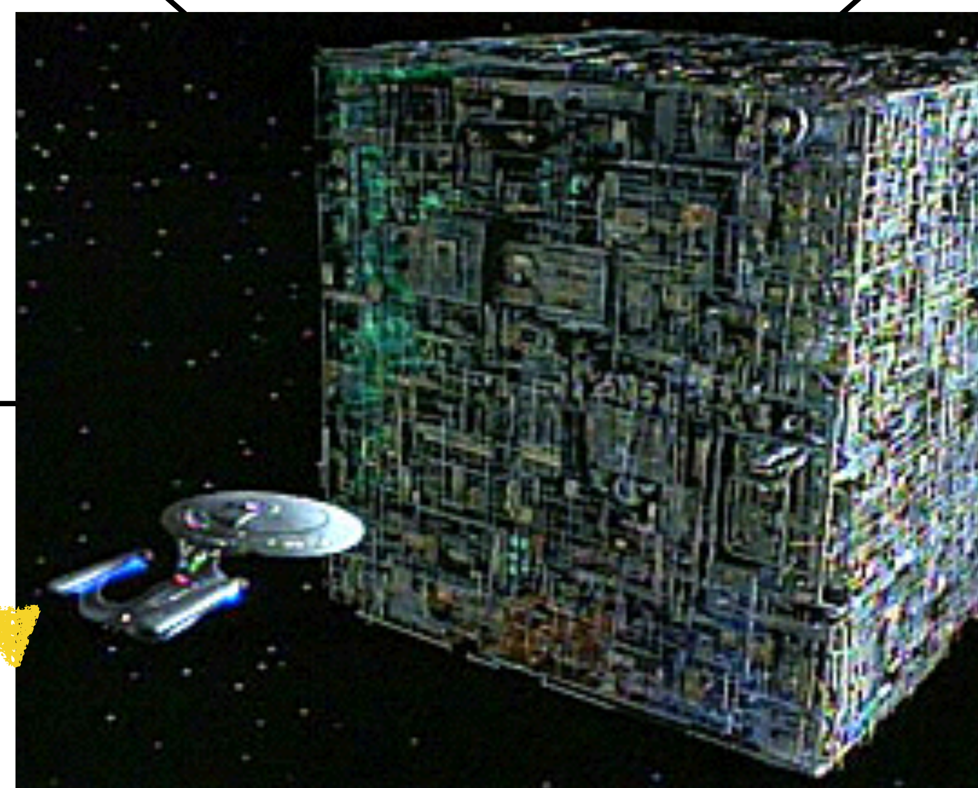


System for System Science

Satellite based instruments

Models

Ground based instruments



Space Weather Drivers

New Data Set

Median Averaging Routine onto Grid

Data Assimilation Method

Suggestion

Grad Students : You should use python. There are no good reasons to still use IDL (or even Matlab)

- Installation, at terminal command prompt:

  pip install pysat Requires CDF, netCDF, HDF libraries as appropriate

- Full Documentation, tutorials, installation, API reference:

  http://rstoneback.github.io/pysat/

- Stay up-to-date with latest code

  git clone https://github.com/rstoneback/pysat.git

- Getting science python

  Enthought (Canopy); Continuum Analytics (Anaconda);
  PyCharm (full featured IDE) from JetBrains free for education

Question

What unique qualities does your data have that need to be accounted for by a general system for system science?

- Installation, at terminal command prompt:

  pip install pysat Requires CDF, netCDF, HDF libraries as appropriate

- Full Documentation, tutorials, installation, API reference:

  http://rstoneback.github.io/pysat/

- Stay up-to-date with latest code

  git clone https://github.com/rstoneback/pysat.git

- Getting science python

  Enthought (Canopy); Continuum Analytics (Anaconda);
  PyCharm (full featured IDE) from JetBrains free for education