An Introduction to the SuperDARN Data Visualization Toolkit in Python

Ashton Reimer and the DaViT-py Team:
Angeline Burrell, Kevin Sterne,
Xueling Shi, and Muhammad Rafiq

SRI International
Virginia Tech
University of Texas at Dallas

21/06/2017

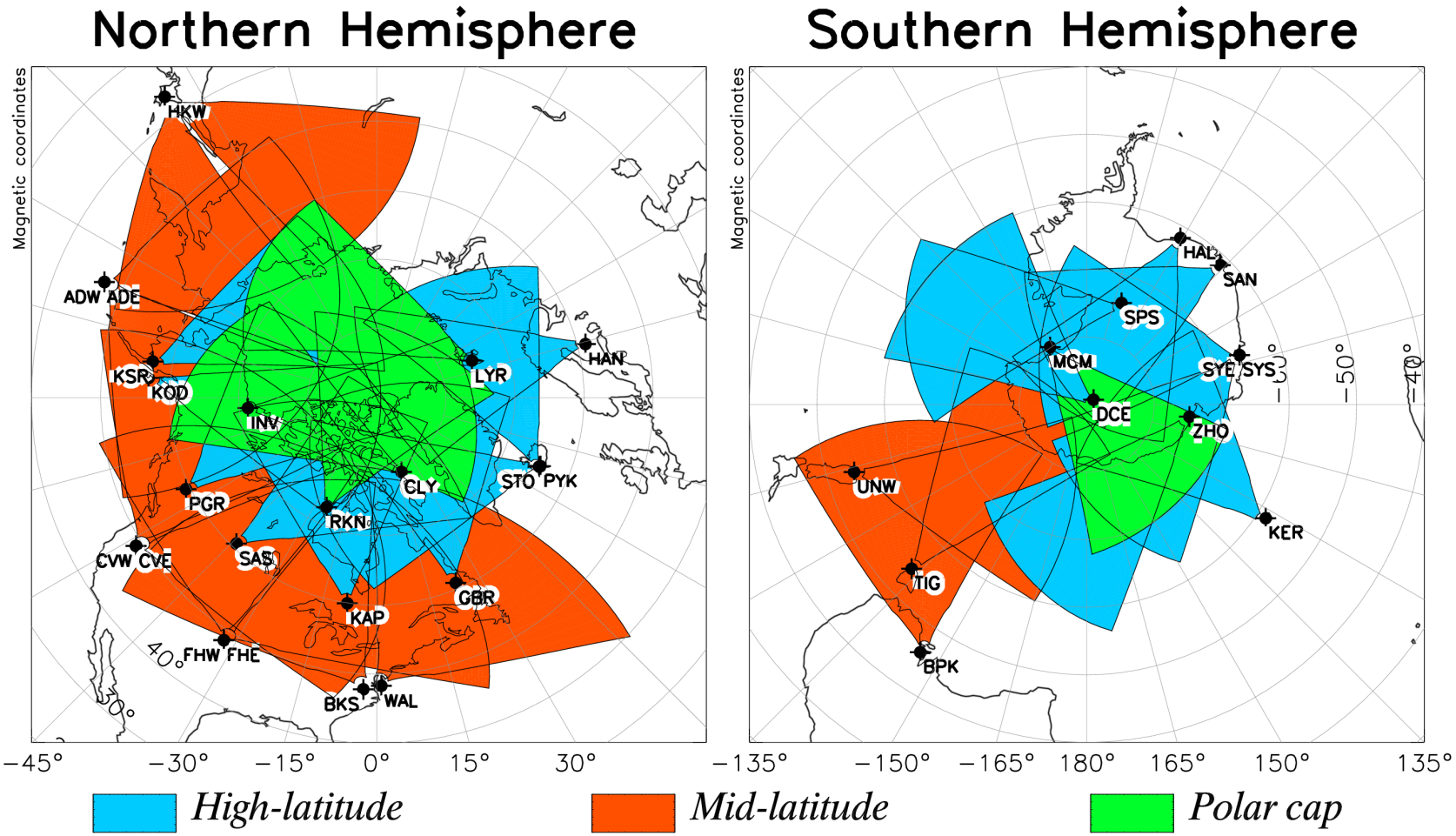
What is DaViT-py?

* Data Visualization Toolkit in Python for SuperDARN
* Open source Python project (some C and Fortran)
* Created by VT SuperDARN group in 2012
* International collaboration
* Version controlled (Git and Github)
* Branch and Pull development model
* Numpy Docstring Standard
* Enforce PEP-8 compliance
* Actively developed

Main Objective: Read and plot SuperDARN data products

What is SuperDARN?

* Super Dual Auroral Radar Network
* International network of 35 high frequency radars
* Primarily used to measure F region plasma convection



Why DaViT-py?

SuperDARN software:

* Most originally written in the 1990s
* Typically dependent on IDL ($) or C for visualization

VT SuperDARN Graduate Students:

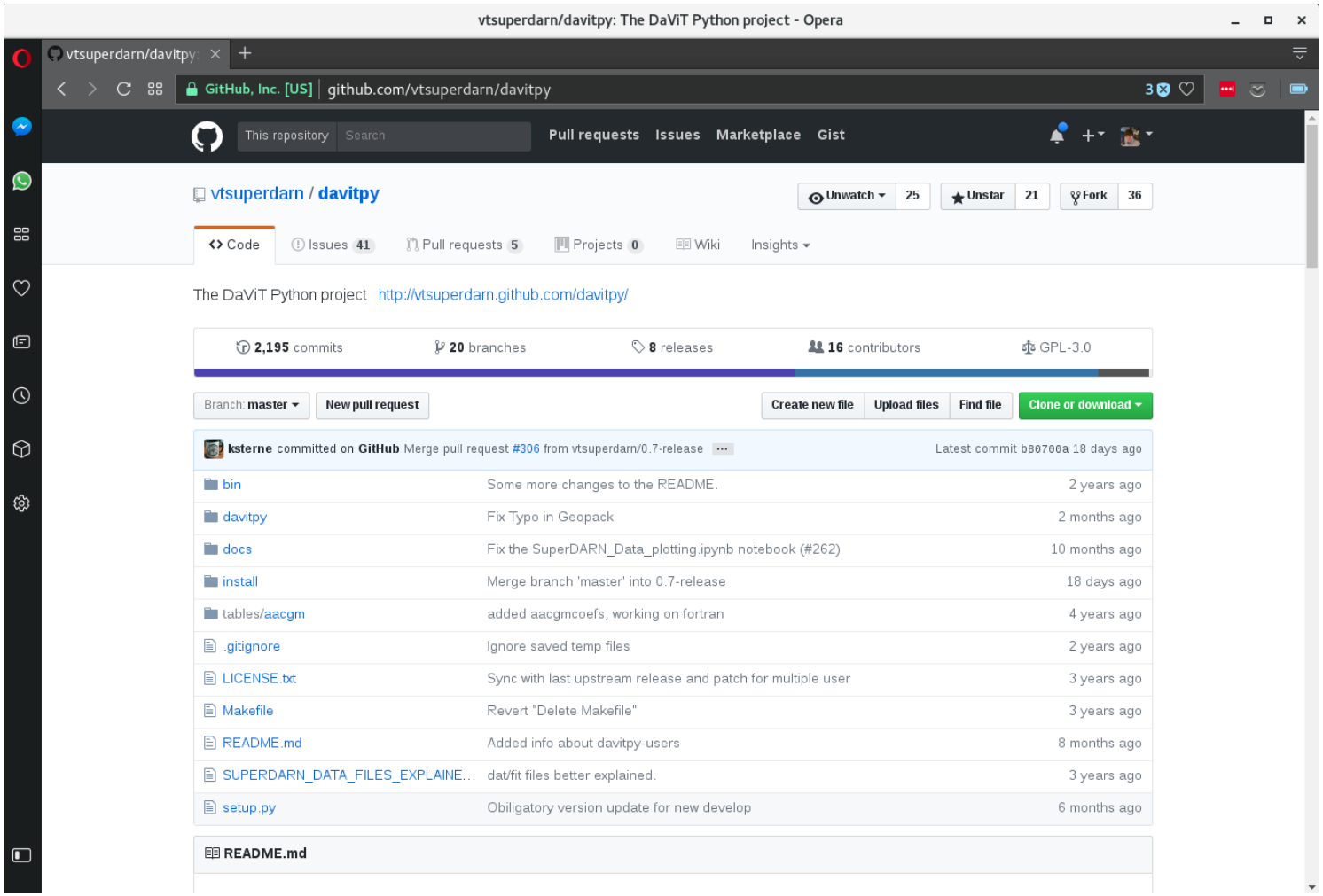
* Sébastien de Larquier and AJ Ribeiro
* Python is everywhere
* DaViT IDL code converted to Python in 2012-2013

DaViT-py will be capable of reading and plotting standard SuperDARN data files (ie. IQ, rawacf, fitted files, grid files, and map files).

How do I get DaViT-py?

Available on Github, use:

```
git clone https://github.com/vtsuperdarn/davitpy.git
```



All code is documented using Numpy docstring convention:

```
24  .. module:: pydarn.sdio.fetchUtils
25     :synopsis: Routines to fetch SuperDARN radar data locally and remotely
26
27  .. moduleauthor:: Angeline G. Burrell, UoL
28
29  ************************************
30  **Module**: pydarn.sdio.fetchUtils
31  ************************************
32
33  Functions
34  -----------
35    * :func:`pydarn.sdio.fetchUtils.uncompress_file`
36    * :func:`pydarn.sdio.fetchUtils.fetch_local_files`
37    * :func:`pydarn.sdio.fetchUtils.fetch_remote_files`
38  """
39
40  import logging
41  import datetime as dt
42  from dateutil.relativedelta import relativedelta
43
```

All code is documented using Numpy docstring convention:

```
44  def uncompress_file(filename, outname=None):
45      """
46      A function to perform an appropriate type of uncompression on a specified
47      file.  Current extensions include: bz2, gz, zip. This function does not
48      removed the compressed file. Requires bunzip2, gunzip, and unzip to be
49      installed.
50
51      Parameters
52      ----------
53      filename : (str)
54          Name of the compressed file
55      outname : (NoneType/str)
56          Compressed name of the desired output file (allows uncompressed file to
57          be placed in a different location) or None (if the uncompressed file
58          will stay in the same directory).  (default=None)
59
60      Returns
61      ---------
62      outname : (NoneType/str)
63          name of uncompressed file or None if the command was unsuccessful or
64          the compression method could not be determined
65      """
66      import os
```

DaViT-py ships with example usage code in Jupyter Notebooks:

vtsuperdarn / **davitpy**

Unwatch 25 | Unstar 21 | Fork 36

Code | Issues 41 | Pull requests 5 | Projects 0 | Wiki | Insights

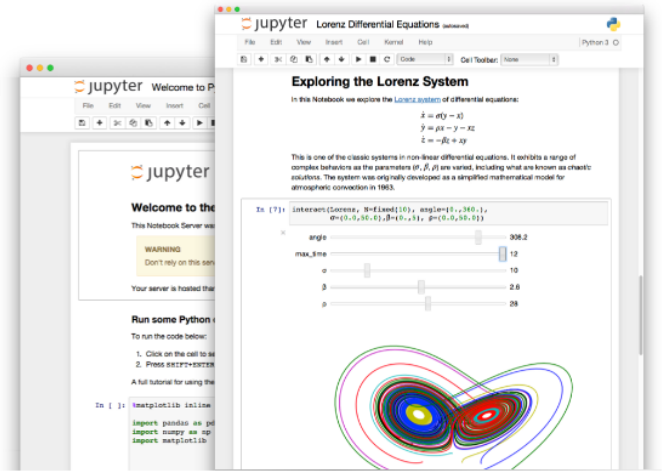
Branch: **master** — **davitpy** / docs / **notebook** /

Create new file | Upload files | Find file | History

MuhammadVT committed with **asreimer** Fix the SuperDARN_Data_plotting.ipynb notebook (#262) ··· Latest commit a4ccaab on Aug 18 2016

..		
GOES.ipynb	rerun of GOES notebook	2 years ago
MUSIC-Simulate.ipynb	rerun MUSIC notebooks	a year ago
MUSIC.ipynb	rerun MUSIC notebooks	a year ago
README.md	more clean-up	4 years ago
SuperDARN Data Plotting.ipynb	Fix the SuperDARN_Data_plotting.ipynb notebook (#262)	10 months ago
geoPack.ipynb	updated GeoPack notebook for ipython3 and davitpy_rc	2 years ago
maps.ipynb	changed all kw to plot_all	2 years ago
models.ipynb	updated models notebook to for ipython3 and davitpy_rc	2 years ago
plotGMagIndices.ipynb	rerun geomag ind notebook	a year ago
radDataRead.ipynb	updated radDataRead notebook for ipython3 and davitpy_rc	2 years ago
radarStruct.ipynb	updated radarStruct notebook for ipython3 and davitpy_rc	2 years ago
rayTracing.ipynb	raytracing notebook updated for davitpy_rc and ipython3	2 years ago

jupyter Install About Resources Documentation NBViewer Widgets Blog Donate



The Jupyter Notebook

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and explanatory text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, machine learning and much more.

Language of choice

The Notebook has support for over 40 programming languages, including those popular in Data Science such as Python, R, Julia and Scala.

Share notebooks

Notebooks can be shared with others using email, Dropbox, GitHub and the Jupyter Notebook Viewer.

Interactive widgets

Code can produce rich output such as images, videos, LaTeX, and JavaScript. Interactive widgets can be used to manipulate and visualize data in realtime.

Big data integration

Leverage big data tools, such as Apache Spark, from Python, R and Scala. Explore that same data with pandas, scikit-learn, ggplot2, dplyr, etc.

* Project born out of IPython in 2014.
* Interactive, plots, code, documentation

Data access provided by default from VT servers.

Provided by davitpy.pydarn.utils.fetchUtils:

Accessing data

In [26]:
```
from davitpy.pydarn.sdio import fetchUtils

fitex = fetchUtils.fetch_remote_files(datetime(2013,11,30,22),datetime(2013,12,1,2),'sftp',
                                      'sd-data.ece.vt.edu','data/{year}/{ftype}/{radar}/',
                                      {'ftype':'fitex','radar':'mcm','channel':'a'},
                                      '/tmp/sd/','{date}.{hour}......{radar}.{channel}.{ftype}',
                                      username=davitpy.rcParams['DBREADUSER'],
                                      password=davitpy.rcParams['DBREADPASS'])

print(fitex)
```

```
['/tmp/sd/20131130.2201.00.mcm.a.fitex', '/tmp/sd/20131201.0000.04.mcm.a.fitex', '/tmp/sd/20131201.0201.00.mcm.a.fitex']
```

Provided by davitpy.pydarn.sdio.radDataOpen

Reading data

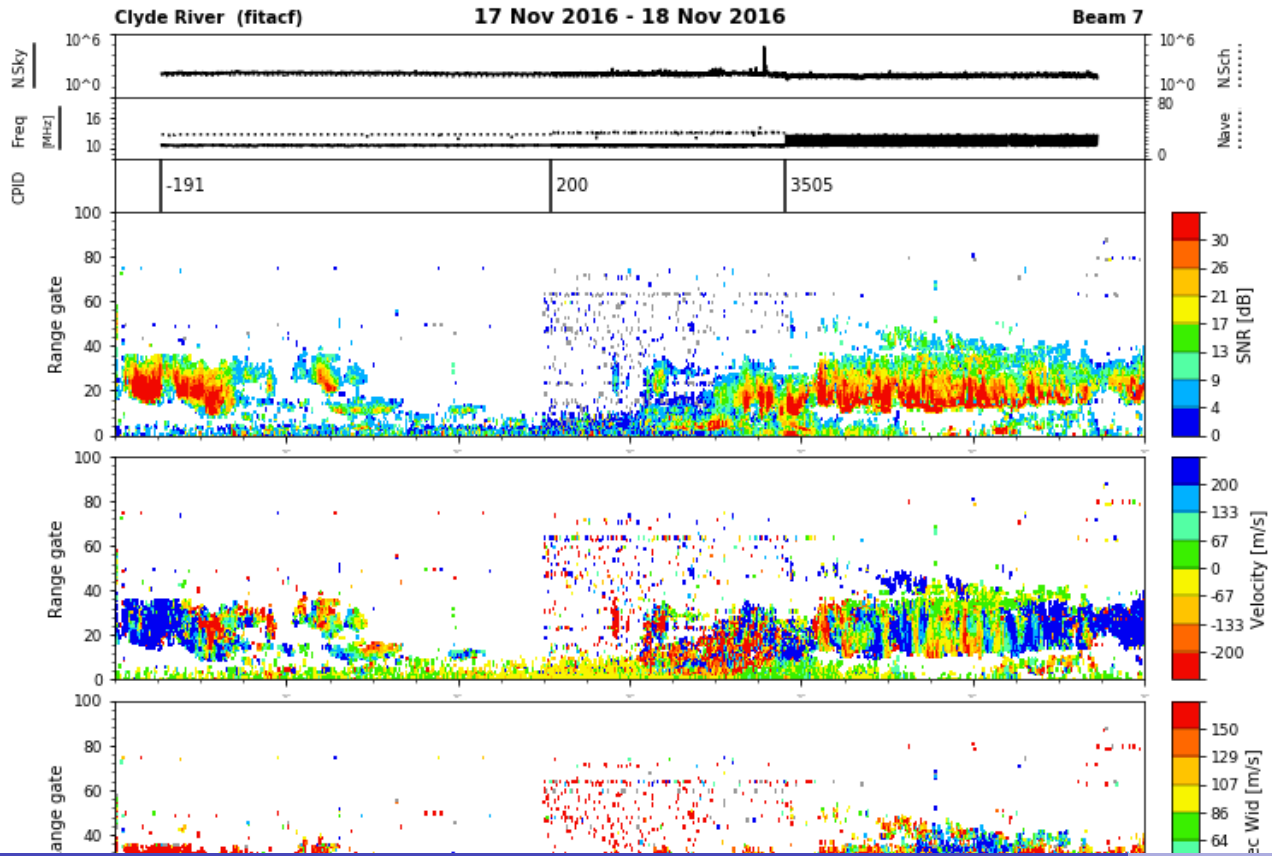
In [27]:
```
sdate = datetime(2013,11,30,22)
myPtr = pydarn.sdio.radDataOpen(sdate,'mcm', fileName='/tmp/sd/20131130.2201.00.mcm.a.fitex',fileType='fitex')
myData = myPtr.readRec()

print myData
```

```
Beam record FROM: 2013-11-30 22:01:00.057000
bmnum  = 0
fPtr  = object
fitex  = None
fit  = object
prm  = object
recordDict  = object
stid  = 20
lmfit  = None
exflg  = None
iqflg  = None
offset  = 0
rawacf  = object
lmflg  = None
rawflg  = None
fType  = fitex
time  = 2013-11-30 22:01:00.057000
acflg  = None
cp  = 151
iqdat  = object
fitacf  = None
channel  = 0
```

Provided by davitpy.pydarn.plotting.plot_rti

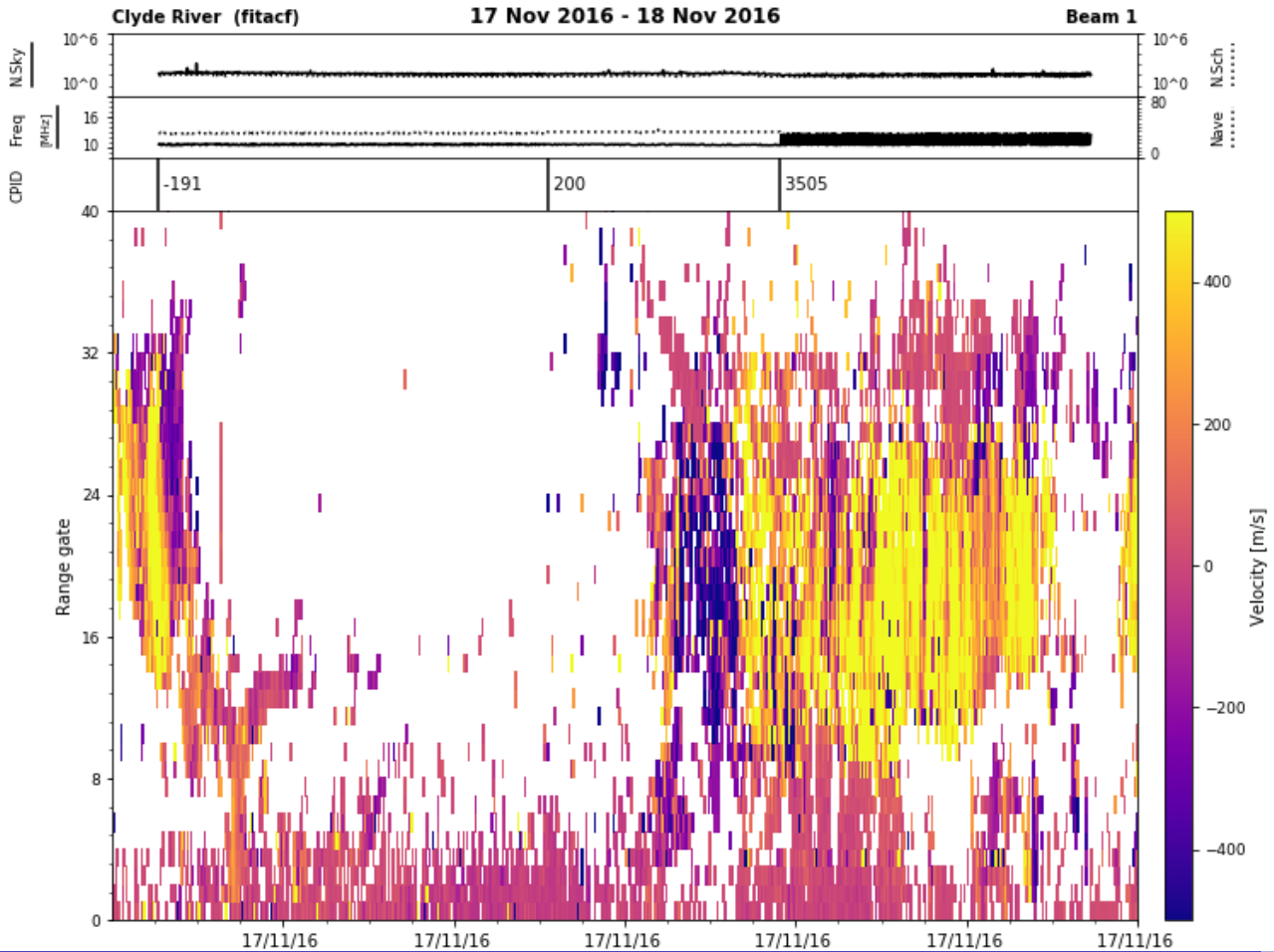
RTI Plots

In [14]:
```
sTime = datetime(2016,11,17)
eTime = datetime(2016,11,18)
radar = 'cly'

fig = pydarn.plotting.rti.plot_rti(sTime, radar, eTime=eTime, bmnum=7,fileName='20161117.cly.fitacf')

#Now save as a PNG
home = '/home/ashtonsethreimer/davitpy_tutorial'
filename = os.path.join(home,'rti1.png')
fig.savefig(filename)
```



Provided by davitpy.pydarn.plotting.plot_rti

In [15]:
```
fig = pydarn.plotting.rti.plot_rti(sTime,radar,eTime=eTime,bmnum=1,
                                   params=['velocity'],colors=['plasma'],
                                   yrng=[0,40],scales=[[-500,500]],
                                   fileName='20161117.cly.fitacf')

#Now save as a PNG
home = '/home/ashtonsethreimer/davitpy_tutorial'
filename = os.path.join(home,'rti2.png')
fig.savefig(filename)
```



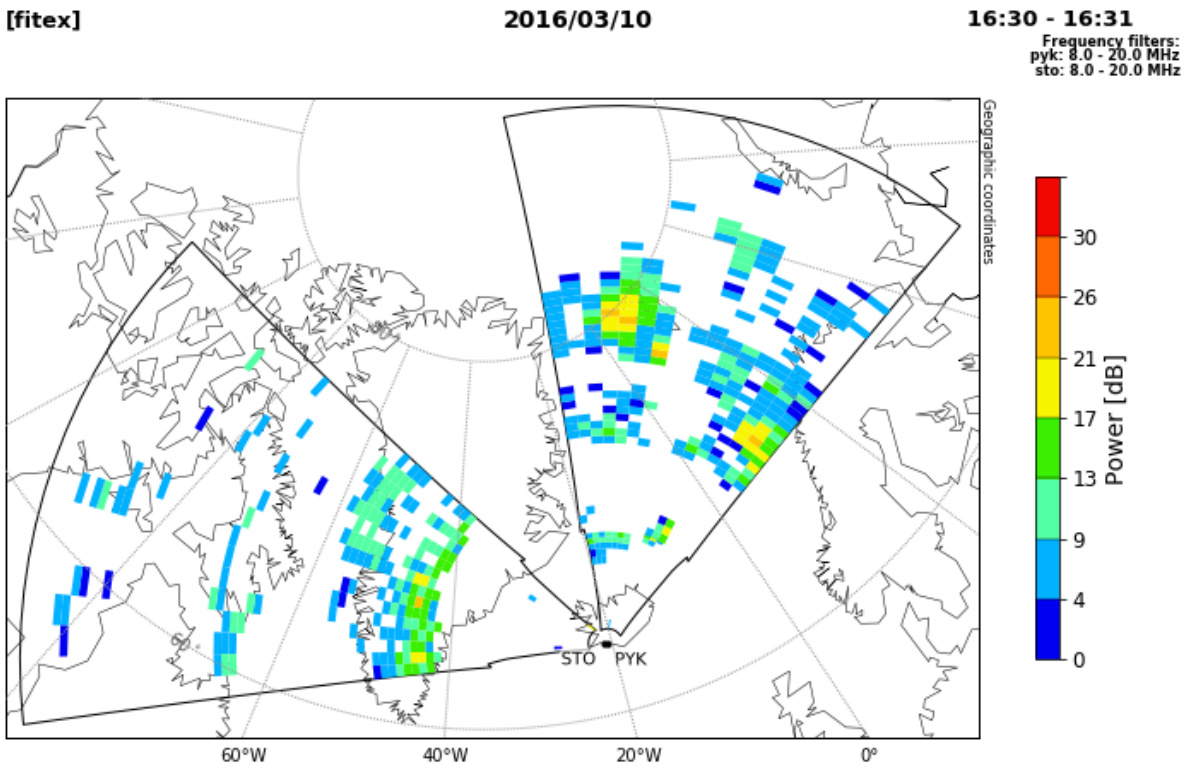
Provided by davitpy.pydarn.plotting.fan.plotFan

Fan Plots

Geopgraphic Coordinates

In [16]:
```
pydarn.plotting.fan.plotFan(datetime(2016,3,10,16,30),['pyk','sto'],param='power',gsct=False)
```

```
<matplotlib.figure.Figure at 0x7ff8dffc4a10>
```

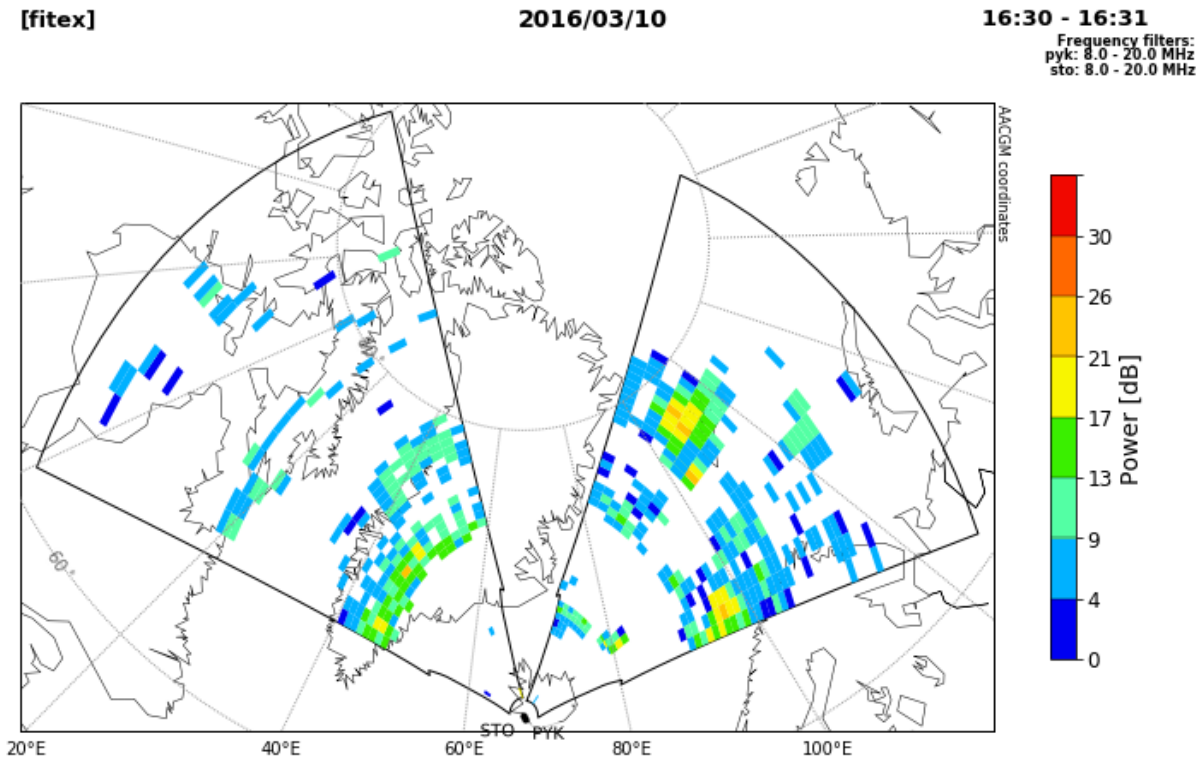


Provided by davitpy.pydarn.plotting.fan.plotFan

Magnetic Coordinates

In [17]:
```
pydarn.plotting.fan.plotFan(datetime(2016,3,10,16,30),['pyk','sto'],param='power',gsct=False,coords='mag')
```

```
<matplotlib.figure.Figure at 0x7ff8dfdcf050>
```



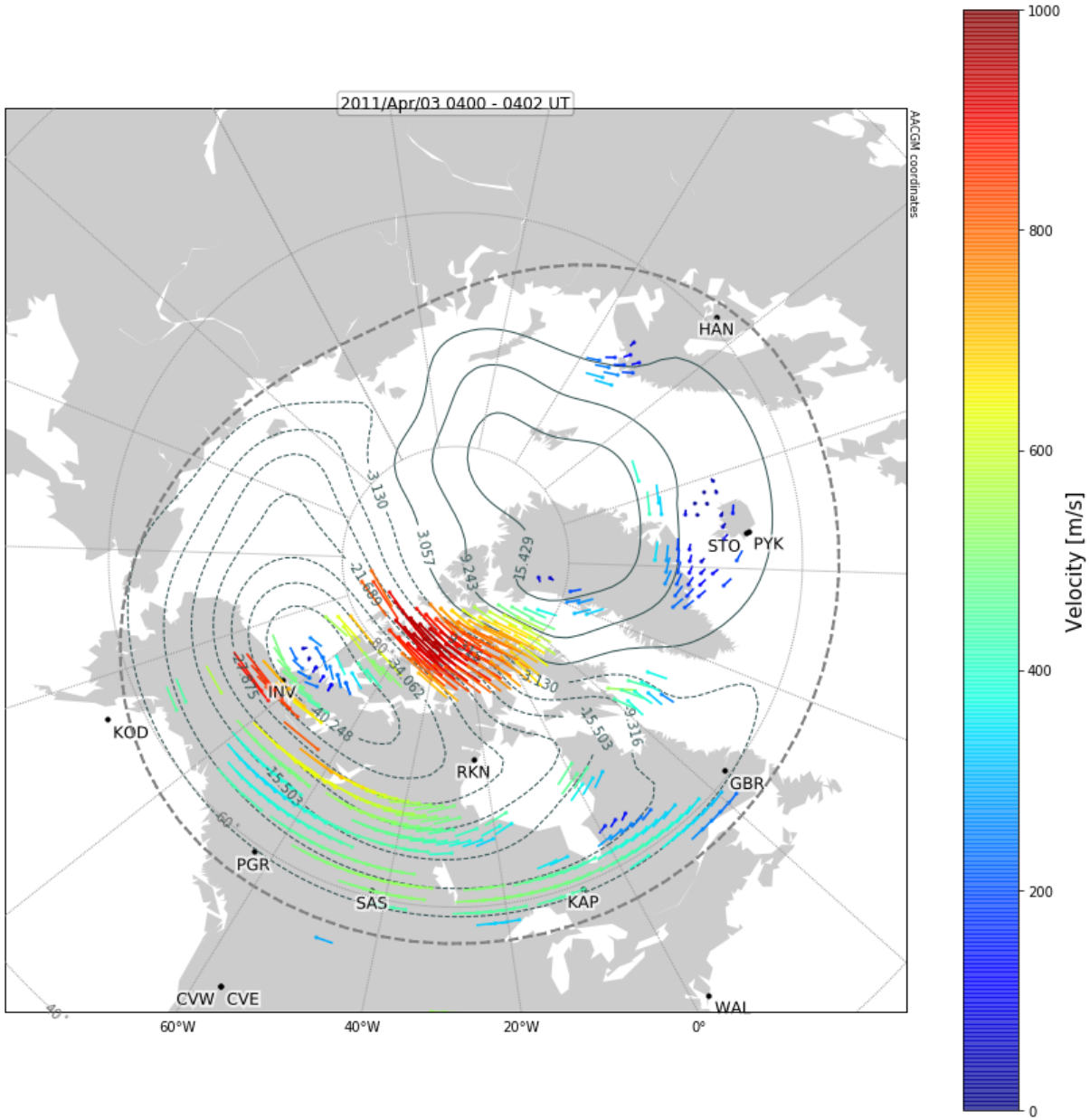
Provided by davitpy.pydarn.plotMapGrd.MapConv

Convection Plotting

Magnetic Coordinates

In [18]:
```
fig = pyplot.figure(figsize=(15,15))
ax = fig.add_subplot(111)

sdate = datetime(2011,4,3,4,0)
mObj = utils.plotUtils.mapObj(boundinglat=50.,gridLabels=True, coords='mag',datetime=sdate)
mapDatObj = pydarn.plotting.plotMapGrd.MapConv(sdate, mObj, ax)
mapDatObj.overlayMapFitVel()
mapDatObj.overlayCnvCntrs()
mapDatObj.overlayHMB()
```

Provided by davitpy.pydarn.plotMapGrd.MapConv

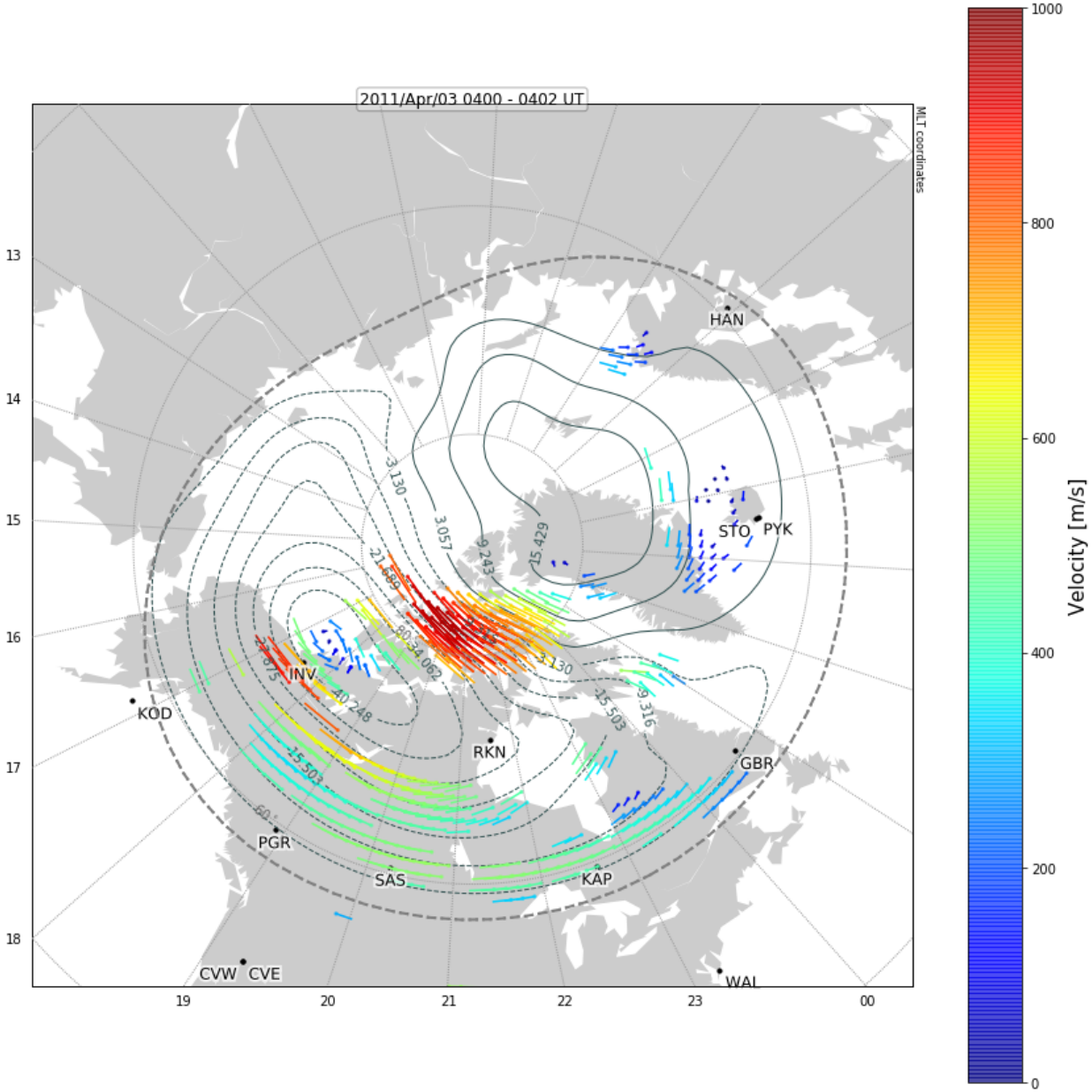


Provided by davitpy.pydarn.plotMapGrd.MapConv

MLT Coordinates

In [19]:
```
fig = pyplot.figure(figsize=(15,15))
ax = fig.add_subplot(111)

sdate = datetime(2011,4,3,4,0)
mObj = utils.plotUtils.mapObj(boundinglat=50.,gridLabels=True, coords='mlt',datetime=sdate)
mapDatObj = pydarn.plotting.plotMapGrd.MapConv(sdate, mObj, ax)
mapDatObj.overlayMapFitVel()
mapDatObj.overlayCnvCntrs()
mapDatObj.overlayHMB()
```

Provided by davitpy.pydarn.plotMapGrd.MapConv



Provided by davitpy and pyAMISR

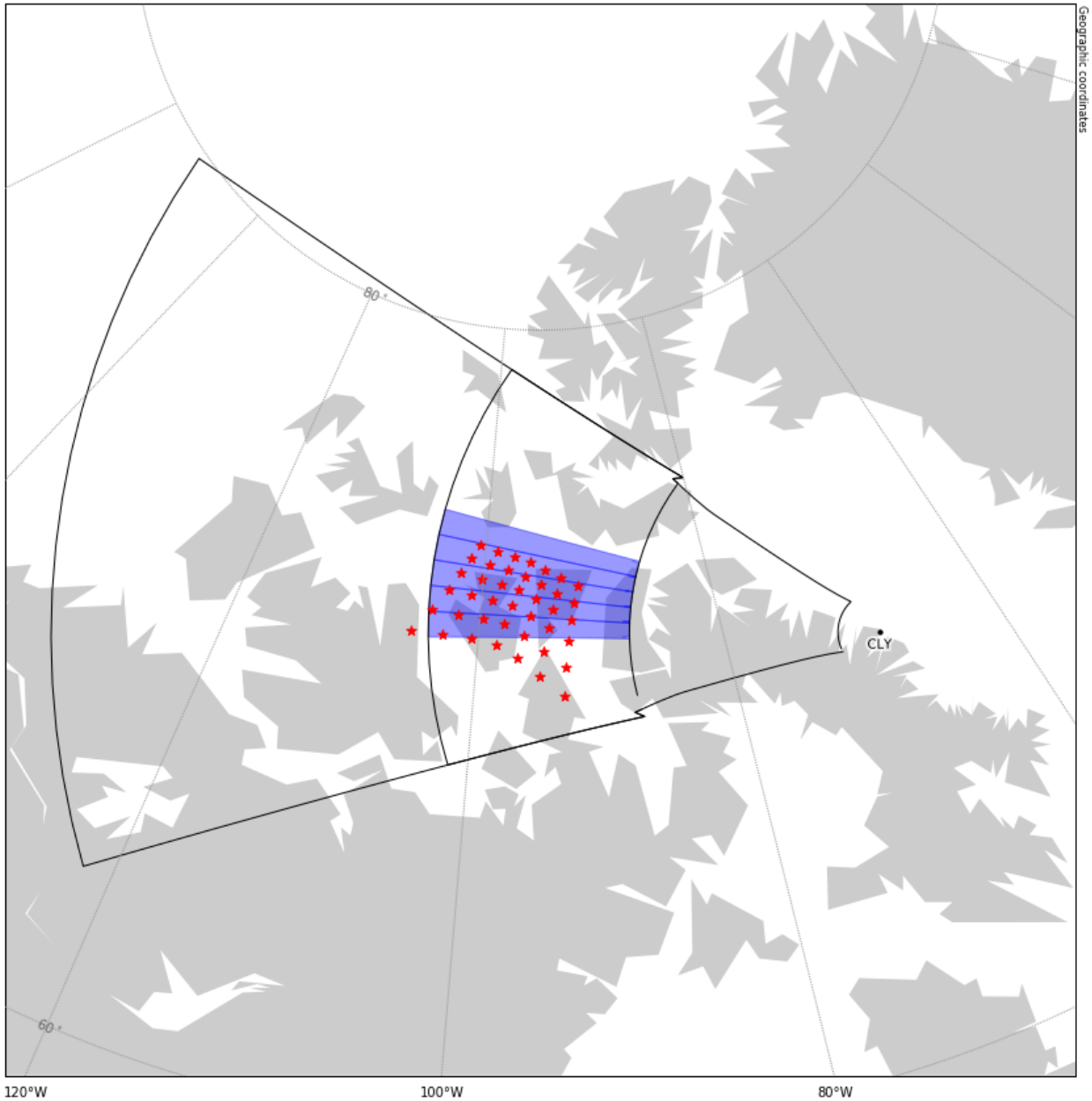
Plot SuperDARN data with other instruments!

In [20]:
```
import pyAMISR
import numpy as np
```

In [21]:
```
# Start analysis of RISR-C data
isr = pyAMISR.analyze('20160303.001_lp_1min.h5')

risrc_beam_grid_inds = np.array([[45,44,48,47,46,43,42], \
                                 [38,37,41,40,39,36,35], \
                                 [31,30,34,33,32,29,28], \
                                 [24,23,27,26,25,22,21], \
                                 [17,16,20,19,18,15,14], \
                                 [10, 9,13,12,11, 8, 7]])-1
#risrc_beam_grid_inds = risrn_beam_grid_inds[::-1]
isr.beamGridInds = risrc_beam_grid_inds
```

In [22]:
```
fig = pyplot.figure(figsize=(15,15))
ax = fig.add_subplot(111)

sdate = datetime(2016,3,3,4,0)
mObj = utils.plotUtils.mapObj(lat_0=isr.data['siteLatitude'],lon_0=isr.data['siteLongitude'],
                              width=3e6,height=3e6,gridLabels=True, coords='geo',
                              datetime=sdate,ax=ax)

pydarn.plotting.overlayRadar(mObj, codes=['cly'], dateTime=sdate)

site = pydarn.radar.site(code='cly',dt=sdate)

fov = pydarn.radar.fov(frang=180.0,rsep=30.0,site=site,nbeams=16,ngates=115,
                       altitude=300.0, model='IS', coords='geo',
                       date_time=sdate, coord_alt=300.0, fov_dir='front')

pydarn.plotting.overlayFov(mObj, codes=['cly'], fovObj=fov, rangeLimits=[0,75],
                           beamLimits=None, fovColor=None, fovAlpha=0.2,
                           beams=None, beamsColors=None,zorder=2,
                           lineColor='k', lineWidth=1)

pydarn.plotting.overlayFov(mObj, codes=['cly'], fovObj=fov, rangeLimits=[20,40],
                           beamLimits=None, fovColor=None, fovAlpha=0.2,
                           beams=range(5,10), beamsColors=5*['blue'],zorder=2,
                           lineColor='k', lineWidth=1)

isr.overlayBeamGrid(300.0, myMap=mObj, sym=['*',60], symColor='red', zorder=3, alpha=1)
```

Provided by davitpy and pyAMISR

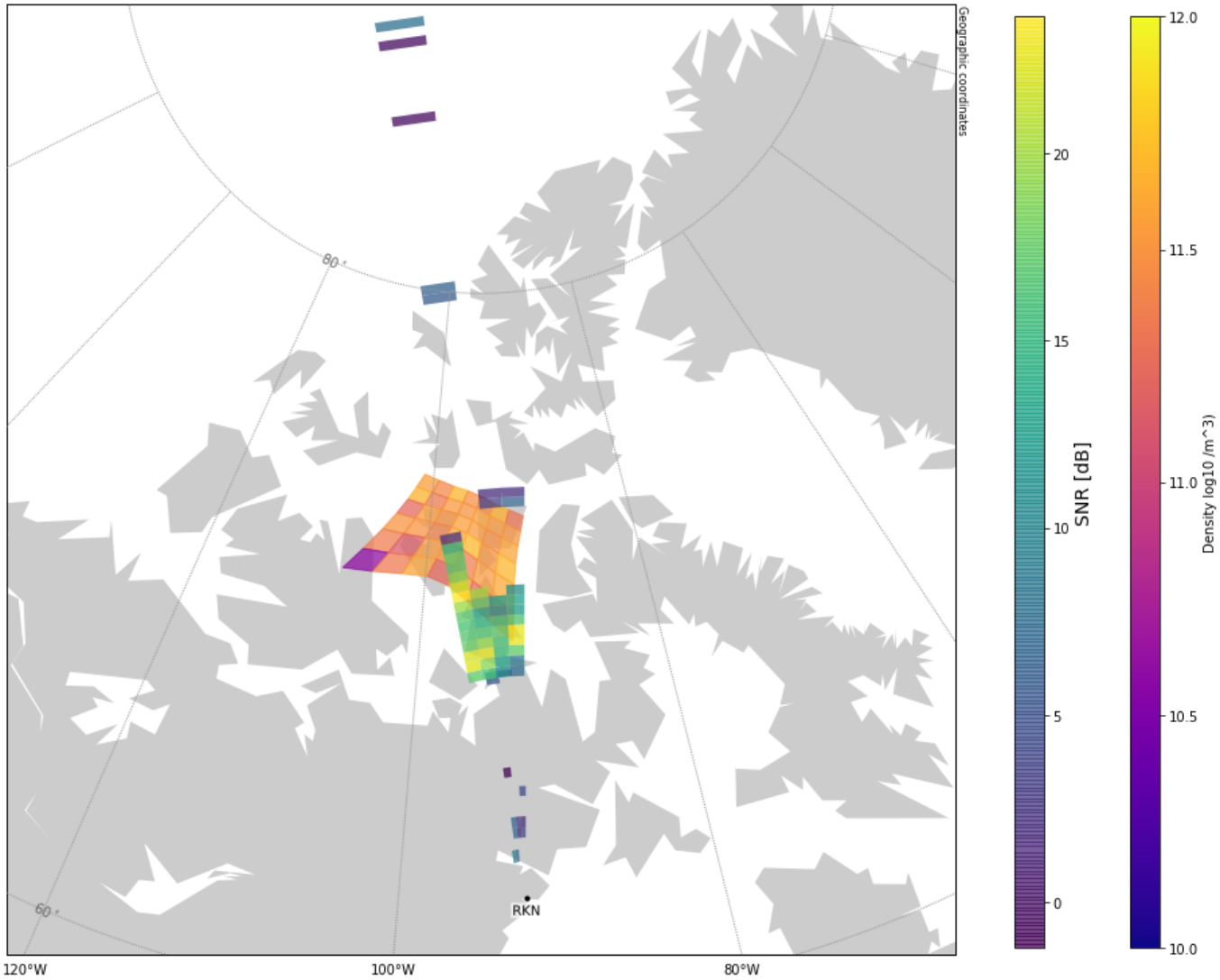


Provided by davitpy and pyAMISR

In [23]:
```
sdate = datetime(2016,3,3,16)
myPtr = pydarn.sdio.radDataOpen(sdate,'rkn', fileName='20160303.1600.00.rkn.fitacf',fileType='fitacf')
myData = myPtr.readScan()
myData = myPtr.readScan()
```

In [24]:
```
fig = pyplot.figure(figsize=(15,13))

mObj = utils.plotUtils.mapObj(lat_0=isr.data['siteLatitude'],lon_0=isr.data['siteLongitude'],
                              width=3e6,height=3e6,gridLabels=True, coords='geo',
                              datetime=sdate)

pydarn.plotting.overlayRadar(mObj, codes=['rkn'], dateTime=sdate)

site = pydarn.radar.site(code='rkn',dt=sdate)

fov = pydarn.radar.fov(frang=180.0,rsep=30.0,site=site,nbeams=16,ngates=115,
                       altitude=300.0, model='IS', coords='geo',
                       date_time=sdate, coord_alt=300.0, fov_dir='front')

intensities, pcoll = pydarn.plotting.overlayFan(myData, mObj, fig, 'power', coords='geo', gsct=0,
                           site=site, fov=fov, gs_flg=[], fill=True, velscl=1000.0,
                           dist=1000.0, cmap='viridis', norm=None, alpha=0.75)

cbax1 = fig.add_axes([0.88,0.13,0.02,0.74])
cbar = fig.colorbar(pcoll,cax=cbax1)
cbar.set_label('SNR [dB]', size=14)

cbax2 = fig.add_axes([0.96,0.13,0.02,0.74])
isr.overlayData('density',sdate, 300.0, myMap=mObj, zorder=3, alpha=0.75, cmap='plasma',colBar=cbax2,cLim=[10,12])
```

Provided by davitpy and pyAMISR



* IRI
* MSIS
* HWM
* IGRF
* Tsyganenko
* AACGM
* RayDARN

Development:

* Started 3 June 2012
* Current: Beta version 0.7 released 1 June 2017
* 41 issues/feature requests
* 5 pull requests
* 5 developers
* 1308 commits, 862,870 additions, 803,065 deletions

Future work (before 1.0 release):

* Rename project to pydarn
* Split codebase into pydarn and pydarn-toolkits
* Set up documentation on readthedocs.com
* Unit tests
* Reduce dependencies and simply installation process (pip?)

How do I develop for DaViT-py?

* Submit issues/feature requests on Github (https://github.com/vtsuperdarn/davitpy)
* Post on DaViT-py Google Group (https://groups.google.com/forum/#forum/davitpy-users)
* Email me (ashtonsethreimer@gmail.com)
* Email other developers: Angeline Burrell, Kevin Sterne, Xueling Shi, or Muhammad Rafiq

How do I develop for DaViT-py?

Development model: Branch and pull model

http://nvie.com/posts/a-succesful-git-branching-model