TEXAS TECH UNIVERSITY™

A Python wrapper for NASA's Radar Software Library

Eric Bruning
TTU Department of Geosciences
Atmospheric Science Group

2011 AMS Annual Meeting, Seattle
Python Symposium Paper 4.4
27 January 2011, 4:15 pm

Thanks:
- RSL developers David Wolff, John Merritt & Bart Kelley (NASA/GSFC)
- Scott Collis (DOE/ANL/ARM) for feedback and contributions

Rationale

It should be trivial to…

- … access gate-level weather radar data
- … manipulate and plot data in a modern high-level language
  - *many existing tools are for non-extensible, canned analysis tasks*

Ideal radar data manipulation in Python:

```
for sweep in volume.sweeps:
   for ray in sweep.rays:
      ranges = ray.ranges
      az    = ray.azimuth
      data = ray.data
```

Why Python

Use Python for everything:

- Versatile data manipulation and plotting:
  - *NumPy: array-based numerics*
  - *Matplotlib: publication-quality plots*
- Everything except file formats!

Radar I/O is tedious. Let’s go shopping.

NASA's Radar Software Library

- Developed to deal with diverse collection of radars in the Tropical Rainfall Measuring Mission's Ground Validation dataset.
  - *http://trmm-fc.gsfc.nasa.gov/trmm_gv/software/rsl/*
- Reads many formats
  - *WSR-88D, UF, Sigmet, EDGE, McGill, Lassen, RAPIC, HDF, RADTEC*
- Cleanly designed object-oriented shared library in C
  - *Provides API for dealing with volumes, sweeps, rays, and metadata*
- Nested structs of radar, volume (field), sweep, ray, gate arrays
  - *Each has associated metadata header*
- Ideal for wrapping in Python

USING PYTHON'S `ctypes` MODULE

`ctypes`: part of Python's standard library

```
>>> from ctypes import *
>>> rsl_lib = CDLL('/usr/local/trmm/GVBOX/lib/librsl.dylib')
```

If it's in RSL.h, you can wrap it in Python:

```
>>> RSL_anyformat_to_radar = rsl_lib.RSL_anyformat_to_radar
>>> RSL_anyformat_to_radar.restype = POINTER(Radar)
>>> RSL_anyformat_to_radar.argtypes = [STRING]

>>> radar = RSL_anyformat_to_radar('KOUN_20040526_230320.uf')
>>> print radar
<RSL.LP_Radar object at 0x3910d40>
>>> print radar.contents
<RSL.Radar object at 0x452a3a0>
```

Pointer to a radar structure

`radar.contents` = Radar structure itself

WRAPPING RSL'S RADAR STRUCTURE

```
typedef struct {
  Radar_header h;
  Volume **v;  /* Array 0..nvolumes-1 of pointers to Volumes.
                    * 0 = DZ_INDEX = reflectivity.
                    * 1 = VR_INDEX = velocity.
                        ...
                    *24 = SQ_INDEX = Signal Quality Index */
} Radar;
```

RSL.h

```
class Radar(Structure):
    pass
Radar._fields_ = [
    ('h', Radar_header),
    ('v', POINTER(POINTER(Volume))),
]
```

RSL.py

```
>>> print radar.contents.h.latd
35
>>> print radar.contents.h.lond
-97
```

Not entirely Pythonic:
Radar attributes should be directly accessible

Convenient, Pythonic API

It's a regular Python class, so override attribute access to get the syntax we want:

```
class Radar(Structure):
    def __getattr__(self, attr):
        if attr == 'volumes':
            nvolumes = self.h.nvolumes
            vols = []
            for i in range(nvolumes):
                try: vols.append(self.v[i].contents)
                except: vols.append(None)
            return vols
        else:
            return self.h.__getattribute__(attr)
Radar._fields_ = [
    ('h', Radar_header),
    ('v', POINTER(POINTER(Volume))),
]
```

Null pointers to empty volumes throw an error, so make them Python's `None`

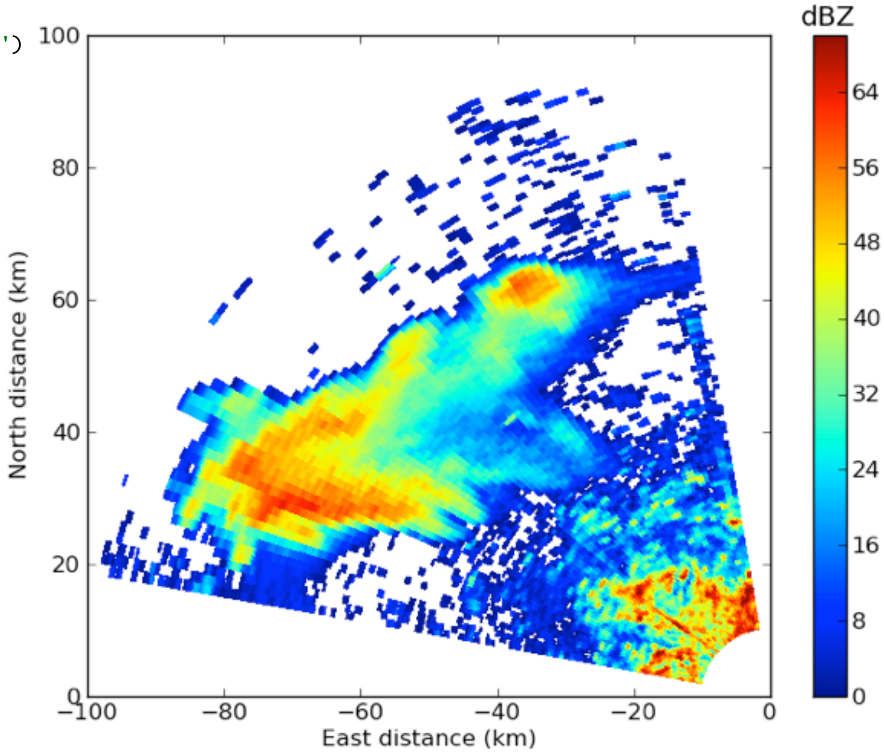
Grab radar attributes from the header

```
>>> print radar.contents.latd
35
>>> print radar.contents.volumes
[  <RSL.Volume object at 0x452a3f8>,   # Reflectivity
   <RSL.Volume object at 0x452a660>,   # Velocity
   None,                               # Other fields are absent
   None, ... ]
>>> for swp in radar.contents.volumes[0].sweeps:
        print swp.elev
0.0
0.5
1.5
2.5
3.5
```

Now it's easy to loop over sweeps and print the VCP

Using PyRSL: Complete PPI plotting script

```
import numpy as np
flt = np.float32

from RSL import RSL_anyformat_to_radar, getAllRays, fieldTypes
from LMAtools.coordinateSystems import RadarCoordinateSystem, MapProjection
import matplotlib.pyplot as plt

kounLat = 35.23833
kounLon = -97.46028
kounAlt = 377.0
mapProj = MapProjection(projection='eqc', ctrLat=kounLat, ctrLon=kounLon, lat_ts=kounLat,
lon_0=kounLon)
radarProj = RadarCoordinateSystem(kounLat,kounLon,kounAlt)

radar = RSL_anyformat_to_radar('/data/20040526/KOUN/0526/KOUN_20040526_230320.uf')

DZ_volume = fieldTypes().DZ
sweep = radar.contents.volumes[DZ_volume].sweeps[0]

dtype = [('r', flt),('az', flt),('el',flt),('DZ',flt)]
swp_data = np.zeros([sweep.nrays, sweep.rays[0].nbins], dtype=dtype)
for raynum, ray in enumerate(sweep.rays):
    swp_data[raynum, :]['DZ'] = ray.data
    swp_data[raynum, :]['r']  = ray.dists
    swp_data[raynum, :]['az'] = ray.azimuth
    swp_data[raynum, :]['el'] = ray.elev

good = ((swp_data['r'] > 10e3) & (swp_data['r'] < 100e3) &
        (swp_data['az']> 280)  & (swp_data['az']< 350) &
        (swp_data['DZ']< 100)  & (swp_data['DZ']> 0) )
data = swp_data.view(np.ma.MaskedArray)
data[good==False] = np.ma.masked

x, y, z = mapProj.fromECEF(*radarProj.toECEF(data['r'],data['az'],data['el']))
x.shape = y.shape = z.shape = data['DZ'].shape

mesh = plt.pcolormesh(x/1e3, y/1e3, data['DZ'], vmin=0, vmax=70)
plt.axis((-100,0,0,100))
plt.xlabel('East distance (km)')
plt.ylabel('North distance (km)')
cbar = plt.colorbar(mesh)
cbar.ax.set_title('dBZ')
```



Using PyRSL

- Make a named array of data for the first sweep

```
sweep = radar.contents.volumes[DZ_volume].sweeps[0]

dtype = [('r', flt),('az', flt),('el',flt),('DZ',flt)]
swp_data = np.zeros([sweep.nrays, sweep.rays[0].nbins], dtype=dtype)
for raynum, ray in enumerate(sweep.rays):
    swp_data[raynum, :]['DZ'] = ray.data
    swp_data[raynum, :]['r']  = ray.dists
    swp_data[raynum, :]['az'] = ray.azimuth
    swp_data[raynum, :]['el'] = ray.elev
```

- Filter based on range, azimuth, and physical values of reflectivity.

```
good = ((swp_data['r'] > 10e3) & (swp_data['r'] < 100e3) &
        (swp_data['az']> 280)  & (swp_data['az']< 350) &
        (swp_data['DZ']< 100)  & (swp_data['DZ']> 0) )
data = swp_data.view(np.ma.MaskedArray)
data[good==False] = np.ma.masked
```

COORDINATE TRANSFORMS AND A PPI PLOT

- Need to go from radar coordinate space[1] (r, θ, φ) to some map projection's (x,y,z). Use pyproj[2] and some helper code[3].

```
from RSL import RSL_anyformat_to_radar, getAllRays, fieldTypes
from LMAtools.coordinateSystems import RadarCoordinateSystem, MapProjection
import matplotlib.pyplot as plt


kounLat = 35.23833
kounLon = -97.46028
kounAlt = 377.0
mapProj = MapProjection(projection='eqc', ctrLat=kounLat, ctrLon=kounLon,
                                              lat_ts=kounLat, lon_0=kounLon)
radarProj = RadarCoordinateSystem(kounLat,kounLon,kounAlt)

x, y, z = mapProj.fromECEF(*radarProj.toECEF(data['r'],data['az'],data['el']))
x.shape = y.shape = z.shape = data['DZ'].shape

mesh = plt.pcolormesh(x/1e3, y/1e3, data['DZ'], vmin=0, vmax=70)
```

[1] Doviak and Zrnic (1993) slant range formulation with (4/3)*R_{earth} refractivity

[2] http://code.google.com/p/pyproj/

[3] https://bitbucket.org/deeplycloudy/lmatools/src/c4606ce74556/coordinateSystems.py

DISCUSSION OF THE PPI PLOT

What we've done:

- Selected a sweep of data and made a plot
- Used non-domain-specific and familiar Python tools
  - *Radar data are just data*
- Treated two geophysical coordinate systems in a rigorous way

What we haven't had to do

- Convert radar data formats
- Write more than a few dozen lines of code
- Think about pointers

What else is possible?

Creative applications of x, y, z & data:

- Pseudo-RHIs are easy
- Scott Collis (DOE/ANL/ARM):
  - *"I took the University of Washington 4D Dealias code which actually uses RSL and wrapped it into the RSL library which allowed me to do in-situ dealiasing of files in Python (i.e., I could feed in first guess winds as a numpy array and no data was written to disk)"*

Create a radar structure from scratch in Python and write it to disk

- Synthetic data for ingest into REORDER for checking gridding techniques and limits of detectability (Collis)

Get the Code

PyRSL source available

- https://bitbucket.org/deeplycloudy/pyrsl/src
- PPI plot example is exampleAMS2011.py
- Wrapper is BSD licensed

TEXAS TECH UNIVERSITY™