MetPy: Community-driven Meteorological Analysis Tools in Python

Ryan May and John Leeman

UCAR/Unidata

28 November 2017

What is MetPy?

- Started in 2008
- Set of tools for meteorological analysis in Python
- Goal is to replace legacy tools, like GEMPAK (GEneral Meteorology PAcKage), for scripted analysis
- Provide building blocks for applications and scripts

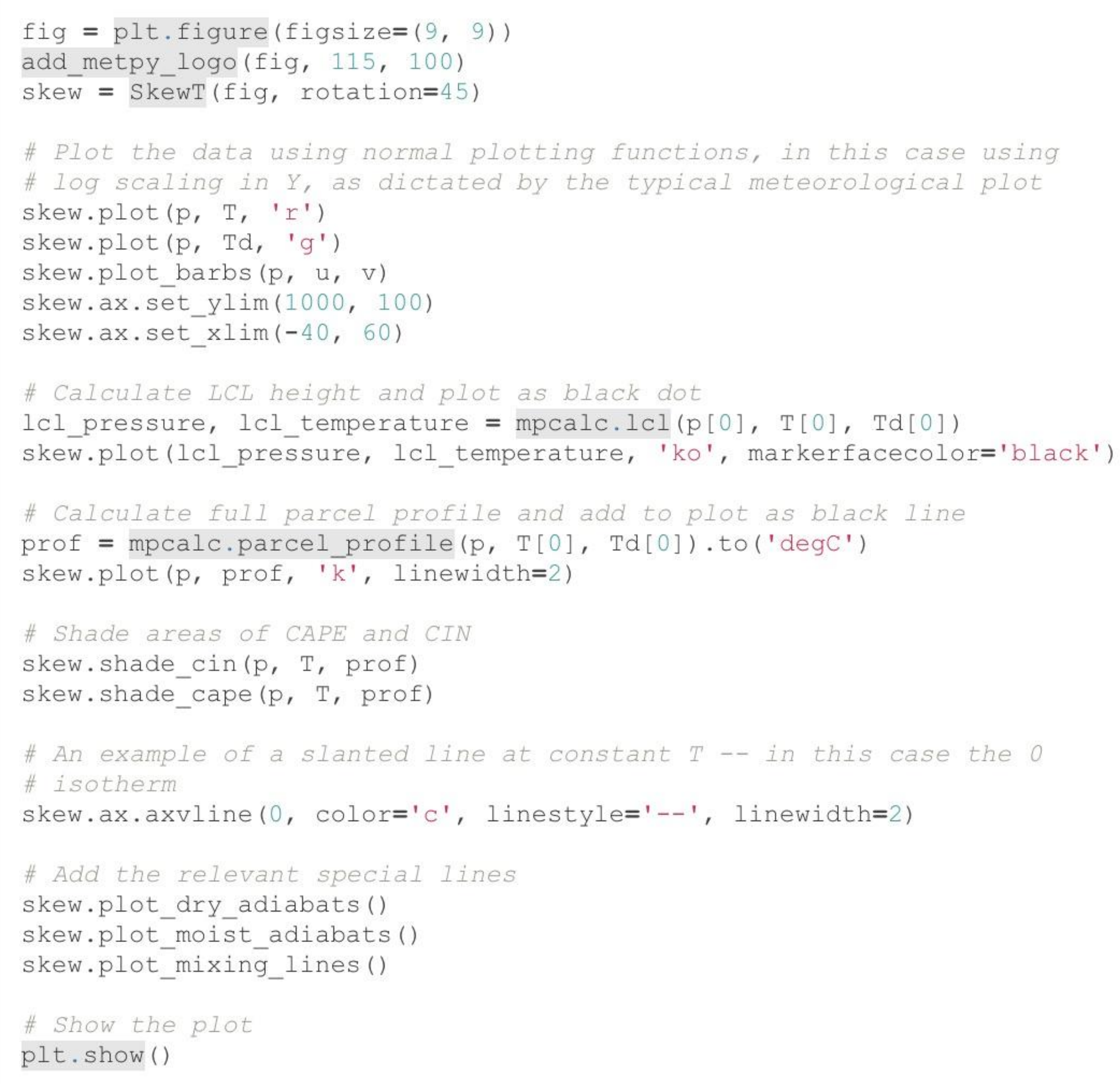
Design Philosophy

- Fit well with scientific Python ecosystem (NumPy, SciPy, Matplotlib, CartoPy, etc.)
- Unit-correctness built-in (using pint)
- Simple to use with your own data
- Good online documentation, with citations to literature when appropriate

Features

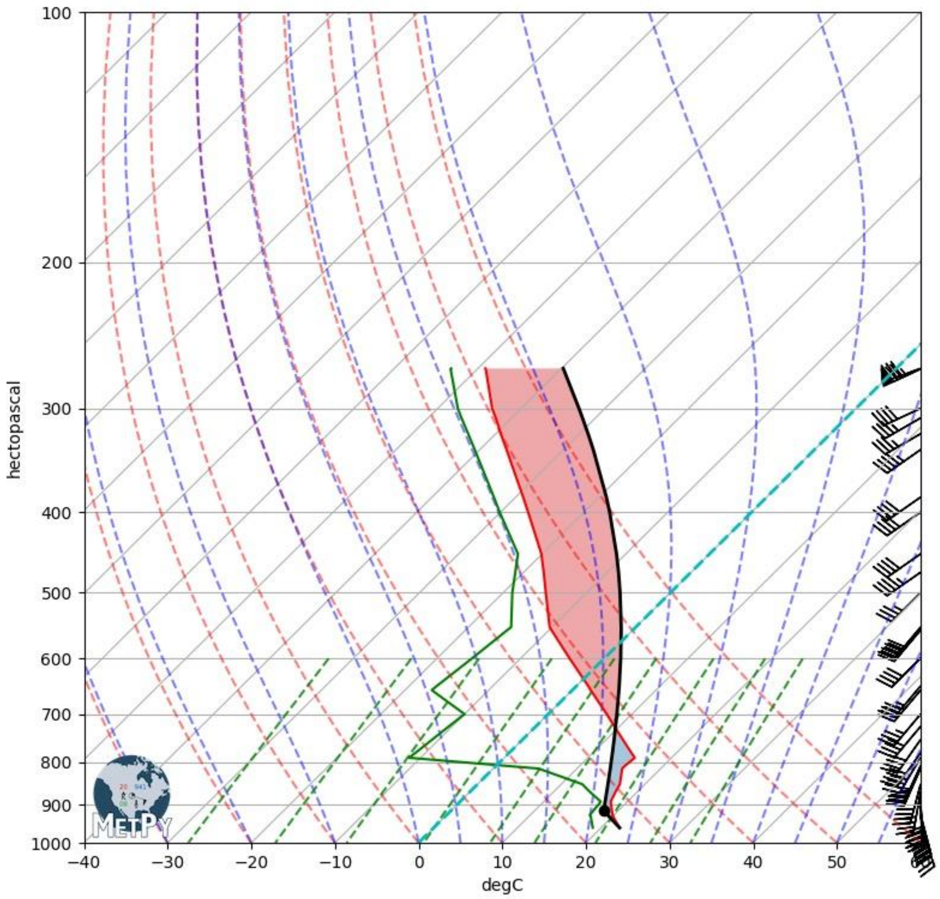
- Functionality breaks into three main areas:
  - Plotting (using matplotlib)
    - Skew-T, Station Plot
  - Reading data files
    - GINI, NEXRAD data and products
  - Calculations
    - Gridding, thermodynamics, etc...
- No compiled code in MetPy itself

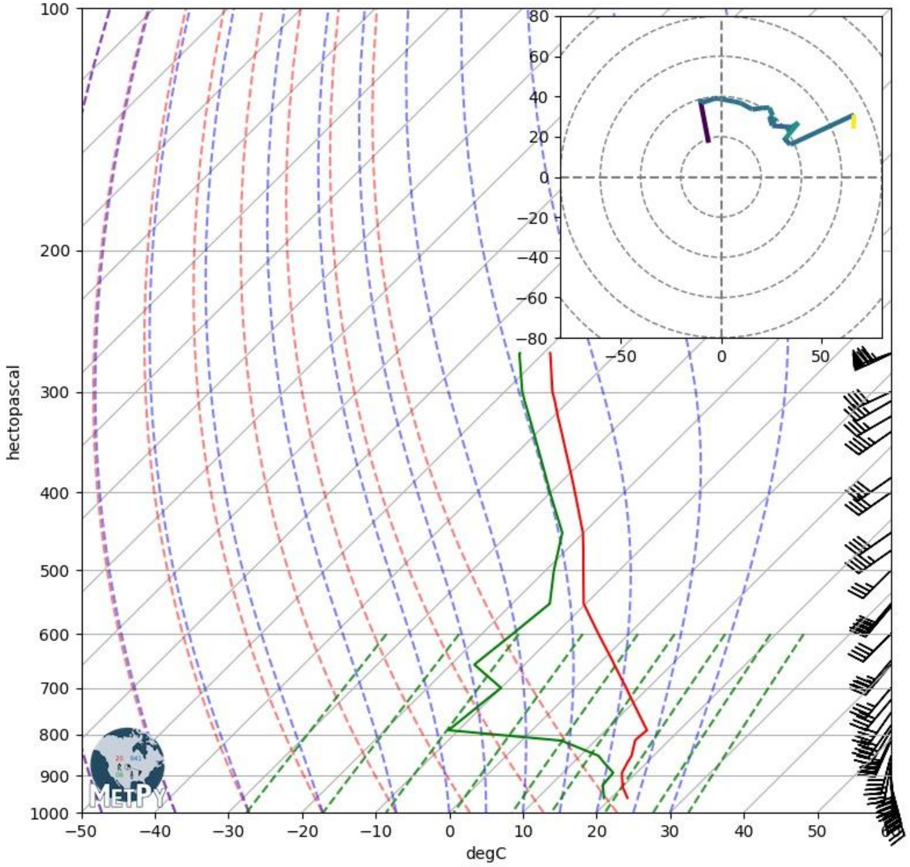
Code Example



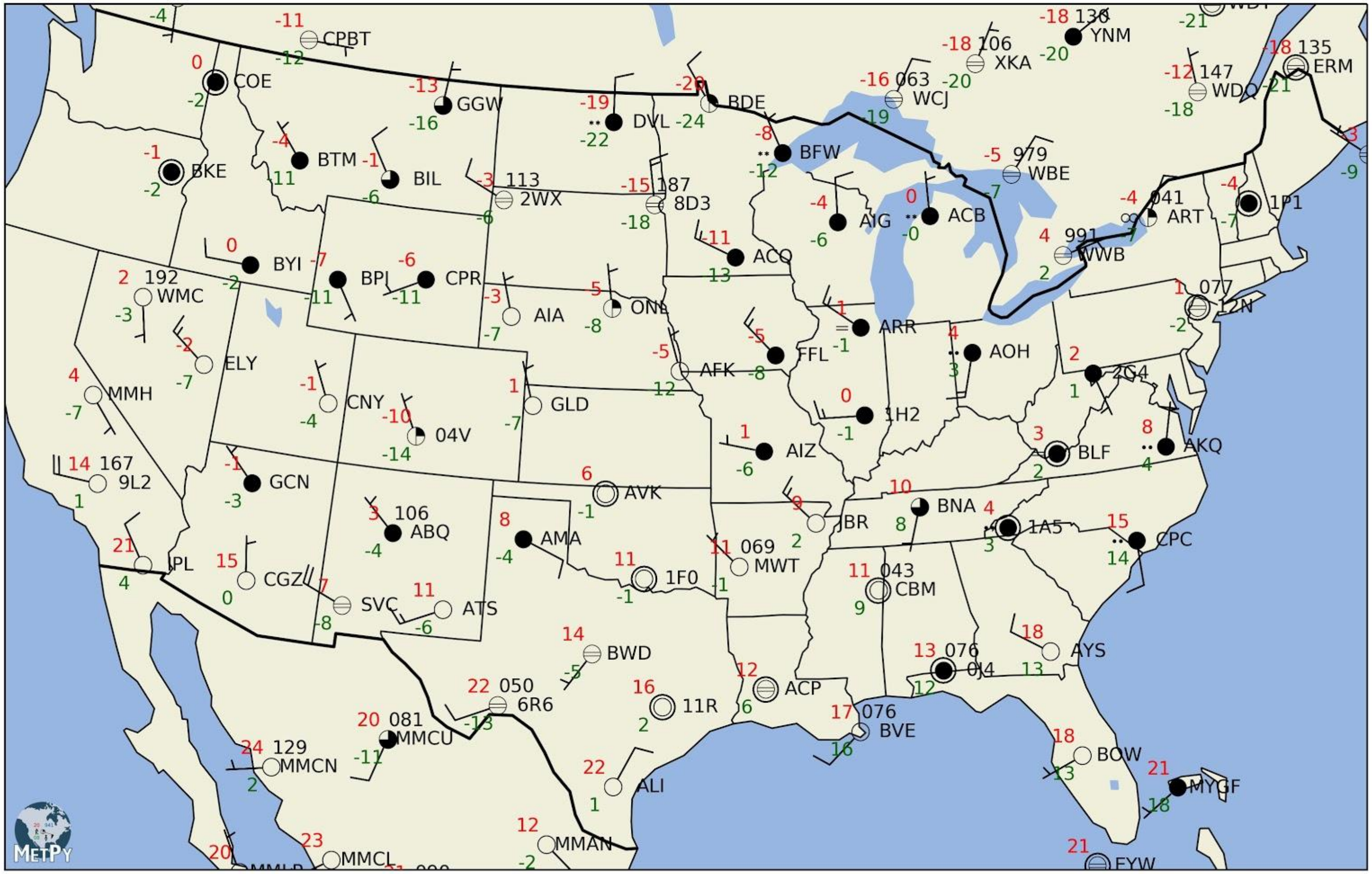
```python
fig = plt.figure(figsize=(9, 9))
add_metpy_logo(fig, 115, 100)
skew = SkewT(fig, rotation=45)

# Plot the data using normal plotting functions, in this case using
# log scaling in Y, as dictated by the typical meteorological plot
skew.plot(p, T, 'r')
skew.plot(p, Td, 'g')
skew.plot_barbs(p, u, v)
skew.ax.set_ylim(1000, 100)
skew.ax.set_xlim(-40, 60)

# Calculate LCL height and plot as black dot
lcl_pressure, lcl_temperature = mpcalc.lcl(p[0], T[0], Td[0])
skew.plot(lcl_pressure, lcl_temperature, 'ko', markerfacecolor='black')

# Calculate full parcel profile and add to plot as black line
prof = mpcalc.parcel_profile(p, T[0], Td[0]).to('degC')
skew.plot(p, prof, 'k', linewidth=2)

# Shade areas of CAPE and CIN
skew.shade_cin(p, T, prof)
skew.shade_cape(p, T, prof)

# An example of a slanted line at constant T -- in this case the 0
# isotherm
skew.ax.axvline(0, color='c', linestyle='--', linewidth=2)

# Add the relevant special lines
skew.plot_dry_adiabats()
skew.plot_moist_adiabats()
skew.plot_mixing_lines()

# Show the plot
plt.show()
```

Skew-T

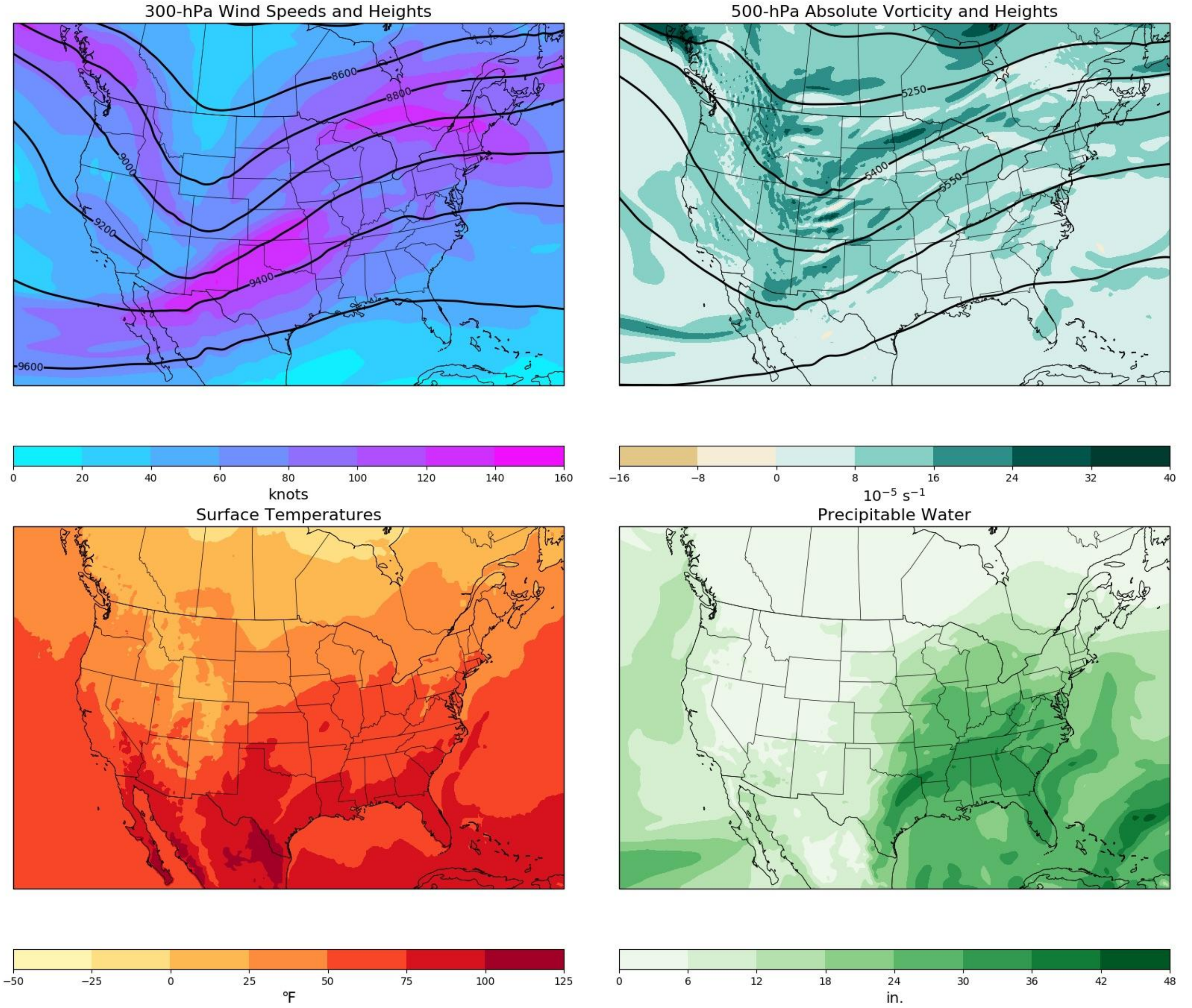

100
200
300
400
500
600
700
800
900
1000
hectopascal
−40
−30
−20
−10
0
10
20
30
40
50
60
degC


100
200
300
400
500
600
700
800
900
1000
hectopascal
−50
−40
−30
−20
−10
0
10
20
30
40
50
60
degC
80
60
40
20
0
−20
−40
−60
−80
−50
0
50

Station Plot

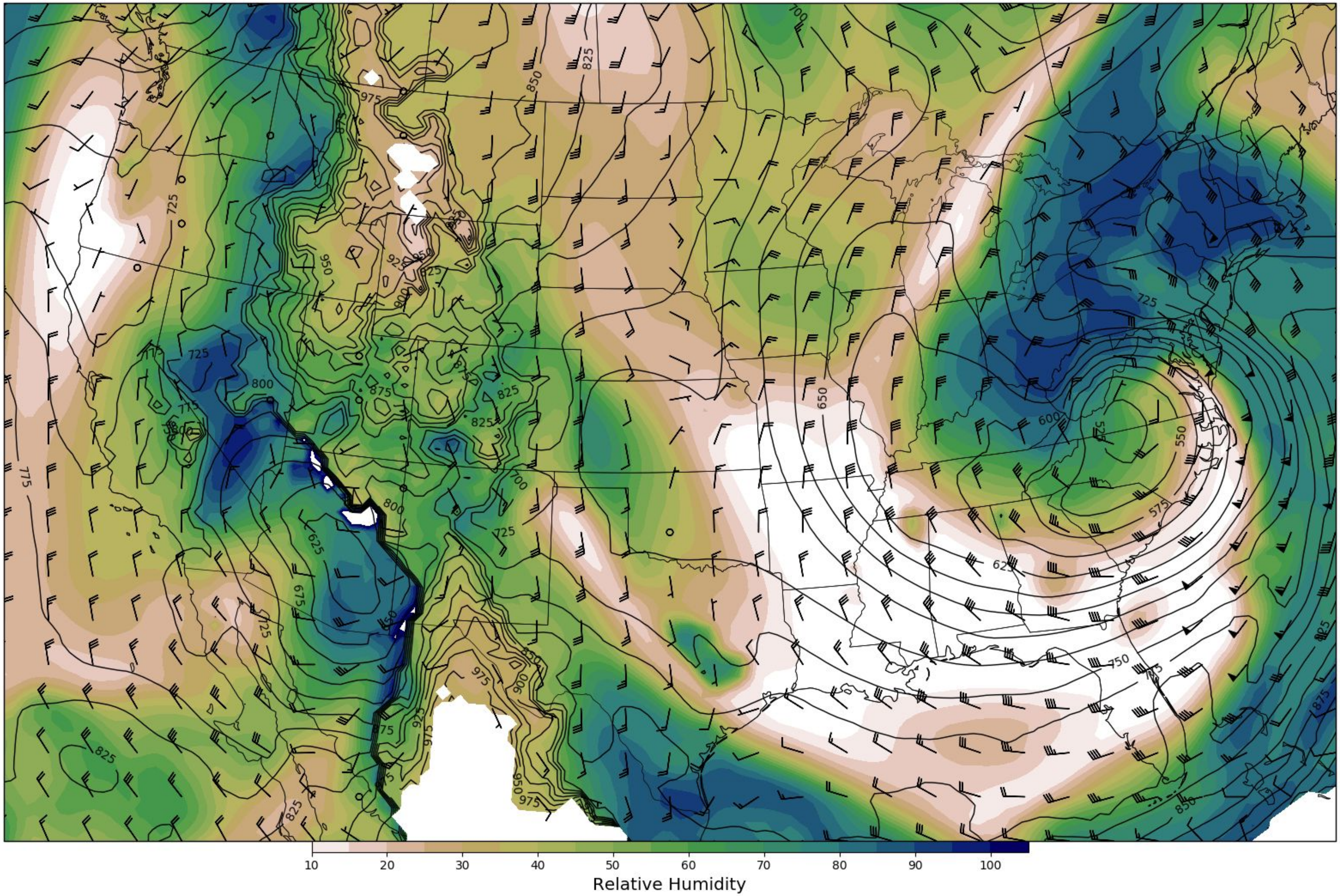


28 February 2017 21:00Z



New Features in 0.6

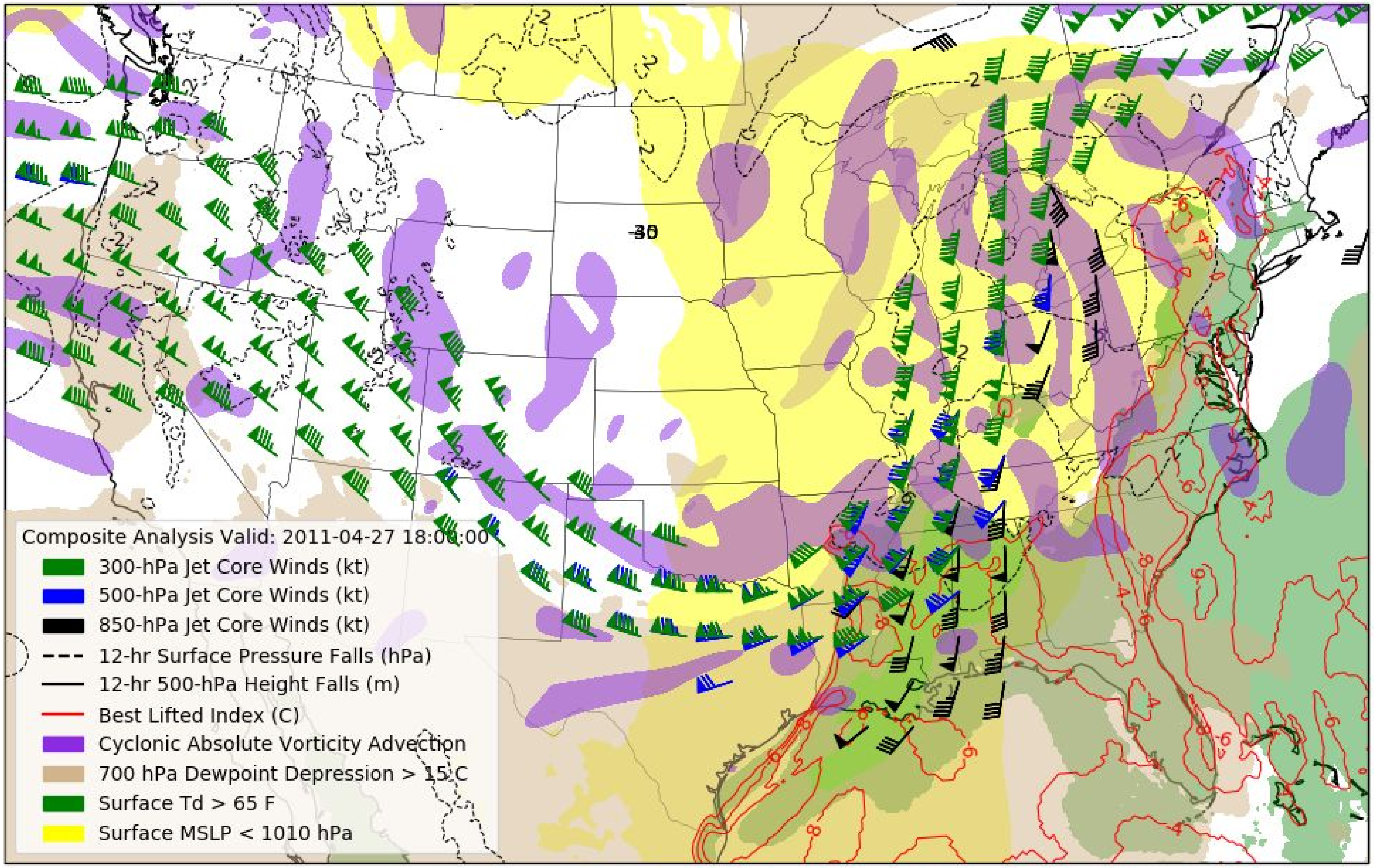
- Many new calculations
  - Isentropic interpolation
  - Severe weather indices
  - Sigma
  - Frontogenesis and deformation
- Weather symbol table
- Version-ed and devel docs
- 4 external contributors



296 K Isentropic Pressure (hPa), Wind (kt), Relative Humidity (percent)

VALID: 1987-04-04 18:00:00

Miller Composite


Composite Analysis Valid: 2011-04-27 18:00:00
300-hPa Jet Core Winds (kt)
500-hPa Jet Core Winds (kt)
850-hPa Jet Core Winds (kt)
12-hr Surface Pressure Falls (hPa)
12-hr 500-hPa Height Falls (m)
Best Lifted Index (C)
Cyclonic Absolute Vorticity Advection
700 hPa Dewpoint Depression > 15 C
Surface Td > 65 F
Surface MSLP < 1010 hPa

Upcoming 0.7

- End of December
- Calculations
  - Specific humidity
  - Thickness
- Internal gradient function for irregularly-spaced grids
- More bug fixes
- AMS short course

3-year Plan

- NSF Award to replace GEMPAK
- Standard data model using xarray and pandas
  - Simplifies use of library
  - Make some calculations easier
  - Need to make xarray play with pint
- Parity with GEMPAK’s calculation collection

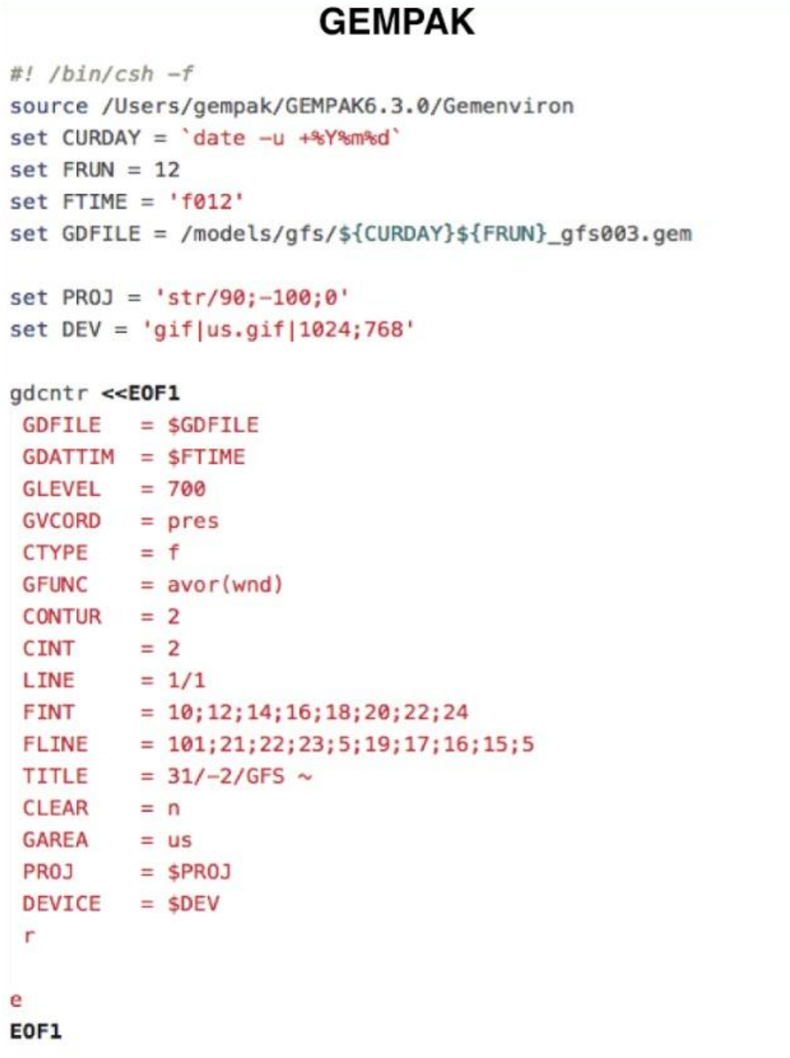
3-Year Plan (cont.)

- Automated field calculation
  - Large collection of calculations
  - Hard to search
  - Complex calculations require too many steps
  - Combine data model with graph-based solver to automatically calculate parameters from source data

3-Year Plan (cont.)

- Declarative plotting interface
  - Way too much boilerplate to make a script
  - Leverage data model to streamline plotting
  - Exploit traitlets and create GEMPAK-like declarative plotting

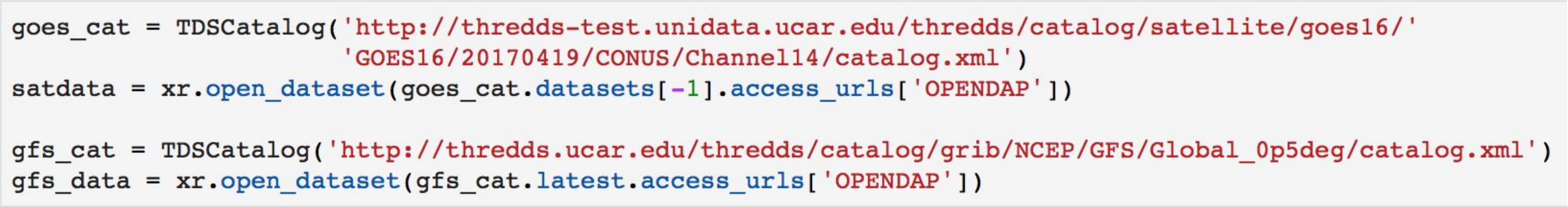
GEMPAK



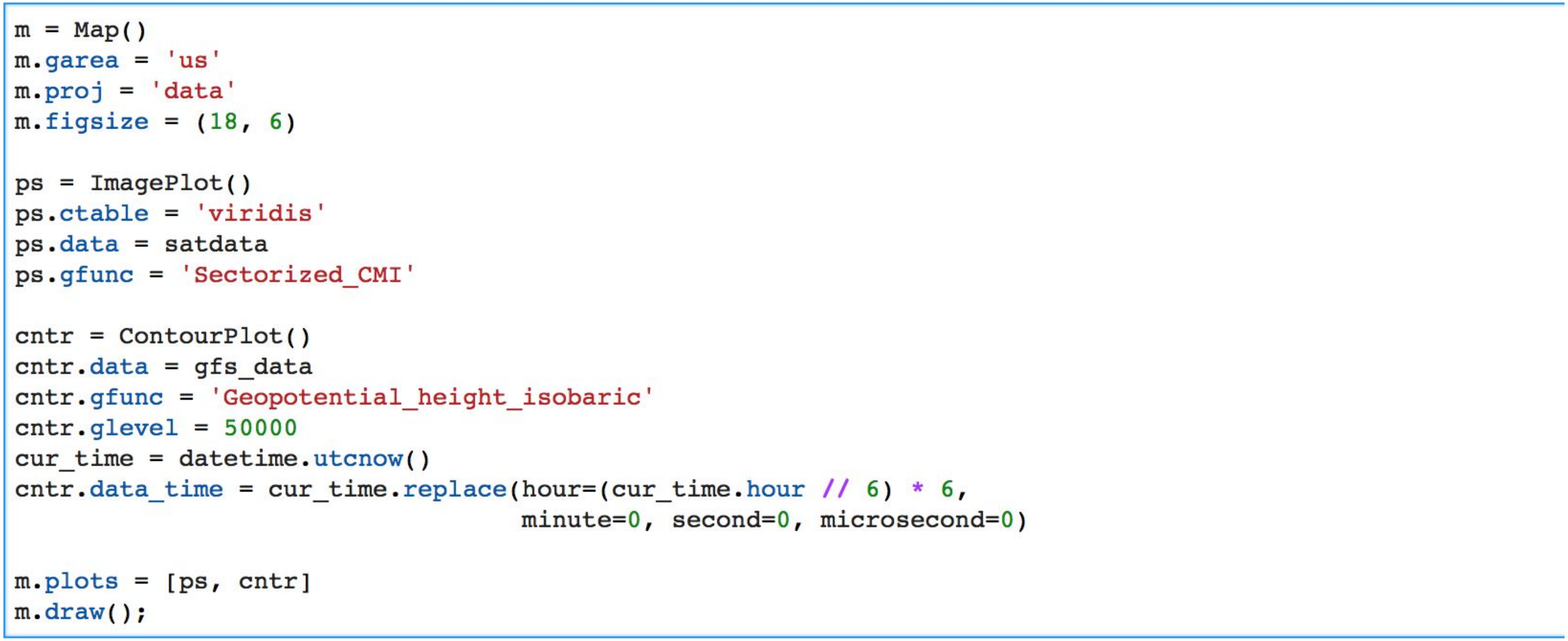
GEMPAK

```
#! /bin/csh -f
source /Users/gempak/GEMPAK6.3.0/Gemenviron
set CURDAY = `date -u +%Y%m%d`
set FRUN = 12
set FTIME = 'f012'
set GDFILE = /models/gfs/${CURDAY}${FRUN}_gfs003.gem

set PROJ = 'str/90;-100;0'
set DEV = 'gif|us.gif|1024;768'

gdcntr <<EOF1
 GDFILE    = $GDFILE
 GDATTIM   = $FTIME
 GLEVEL    = 700
 GVCORD    = pres
 CTYPE     = f
 GFUNC     = avor(wnd)
 CONTUR    = 2
 CINT      = 2
 LINE      = 1/1
 FINT      = 10;12;14;16;18;20;22;24
 FLINE     = 101;21;22;23;5;19;17;16;15;5
 TITLE     = 31/-2/GFS ~
 CLEAR     = n
 GAREA     = us
 PROJ      = $PROJ
 DEVICE    = $DEV
 r

e
EOF1
```

Current Prototype



```
goes_cat = TDSCatalog('http://thredds-test.unidata.ucar.edu/thredds/catalog/satellite/goes16/'
                      'GOES16/20170419/CONUS/Channel14/catalog.xml')
satdata = xr.open_dataset(goes_cat.datasets[-1].access_urls['OPENDAP'])

gfs_cat = TDSCatalog('http://thredds.ucar.edu/thredds/catalog/grib/NCEP/GFS/Global_0p5deg/catalog.xml')
gfs_data = xr.open_dataset(gfs_cat.latest.access_urls['OPENDAP'])
```



```
m = Map()
m.garea = 'us'
m.proj = 'data'
m.figsize = (18, 6)

ps = ImagePlot()
ps.ctable = 'viridis'
ps.data = satdata
ps.gfunc = 'Sectorized_CMI'

cntr = ContourPlot()
cntr.data = gfs_data
cntr.gfunc = 'Geopotential_height_isobaric'
cntr.glevel = 50000
cur_time = datetime.utcnow()
cntr.data_time = cur_time.replace(hour=(cur_time.hour // 6) * 6,
                                  minute=0, second=0, microsecond=0)

m.plots = [ps, cntr]
m.draw();
```

Community Driven

- BSD 3-clause license
- Continually soliciting participation
  - 23 contributors to repository
- Open development model
  - Everything goes through pull requests
  - Ideas and bugs become GitHub issues
  - Discussions on Gitter (chat)
  - GitHub milestones used for roadmap
- Contributor License Agreement
- Release early and often

Automate Everything

- Infrastructure in place to assure sustainability
- TravisCI and AppVeyor
  - 97% test coverage
  - Code style and lint checking
  - Examples all executed
  - Automated documentation deployment to GitHub Pages
  - Automated PyPI deployment
- Web-based static analysis

Resources

- GitHub
  - https://github.com/Unidata/MetPy
- Documentation
  - https://unidata.github.io/MetPy
- Twitter
  - https://twitter.com/MetPy
- Conda
  - conda install -c conda-forge metpy