Enabling Declarative Syntax while using Matplotlib's pcolormesh in MetPy

Nathaniel Martinez; University Corporation for Atmospheric Research (UCAR), Unidata Program Center, Boulder, USA

Overview

- How does the software development process work?
- Why are raster plots and declarative syntax relevant in MetPy?
- How is documentation maintained once new code is merged into the GitHub repository?

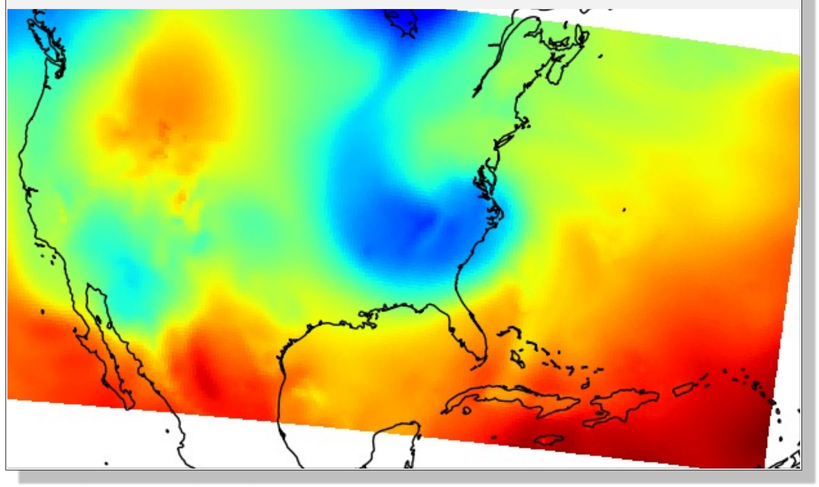
Software Development Process

- Planning
  - Identify the issue to be fixed or new functionality that needs to be added
  - Identify stakeholders' software needs
- Design
  - Compile a design that resolves the issues identified in the planning phase
- Implementation
  - Draw from the design to write the code to fulfill the identified criteria
- Testing
  - Create sufficient tests to ensure all new or edited code is verified to work properly
- Integration
  - Create a pull request for the new code into the code repository, resolving any conflicts

The case for Raster Plots

- Raster Plots help visualize key atmospheric data
  - MetPy does not currently support declarative syntax when trying to use Matplotlib's pcolormesh
- Declarative syntax allows for easy, flexible plotting of data
- Radar is a type of raster plot
  - Functionality may be extended to making radar plots
  - MetPy's current process for making radar plots is extensive
  - Declarative radar plots will remove the involved work required to plot

Temperature Raster Plot



```python
data = xr.open_dataset(get_test_data('narr_example.nc', as_file_obj=False))

raster = RasterPlot()
raster.data = data
raster.field = 'Temperature'
raster.level = 700 * units.hPa

panel = MapPanel()
panel.area = 'us'
panel.projection = 'lcc'
panel.layers = ['coastline']
panel.plots = [raster]

pc = PanelContainer()
pc.size = (8.0, 8)
pc.panels = [panel]
pc.draw()
```

Documentation Improvements

- After code is published, it needs to be maintained
- Documentation provides clarity on best practices and use cases for existing functions in MetPy
- As new functions are added, updates need to be reflected in documentation

Radar Documentation Improvements

```python
# Grab azimuths and calculate a range based on number of gates
az = np.array(datadict['start_az'] + [datadict['end_az'][-1]])
rng = np.linspace(0, f.max_range, data.shape[-1] + 1)
# Grab azimuths and calculate a range based on number of gates,
# both with their respective units
az = units.Quantity(np.array(datadict['start_az'] + [datadict['end_az'][-1]]), 'degrees')
rng = units.Quantity(np.linspace(0, f.max_range, data.shape[-1] + 1), 'kilometers')

# Extract central latitude and longitude from the file
cent_lon = f.lon
cent_lat = f.lat

# Convert az,range to x,y
xlocs = rng * np.sin(np.deg2rad(az[:, np.newaxis]))
ylocs = rng * np.cos(np.deg2rad(az[:, np.newaxis]))
xlocs, ylocs = azimuth_range_to_lat_lon(az, rng, cent_lon, cent_lat)
```

Acknowledgements

- Thank you to Unidata, UCAR, NCAR for having me!
- Thanks to the other interns for having a great summer together!
- Thank you to Drew and Ryan for their guidance and mentorship this summer!

Award # AGS-1901712