MetPy 1.1.0 Milestones: Code Fixes and Verification

LYDIA BUNTING

SUMMER 2021

Acknowledgements

- Unidata and UCAR
- Ryan May and Drew Camron
- Connor Cozad and Izzy Pfander

About MetPy

What it is:

A collection of tools in Python for reading, visualizing, and performing calculations with weather data.

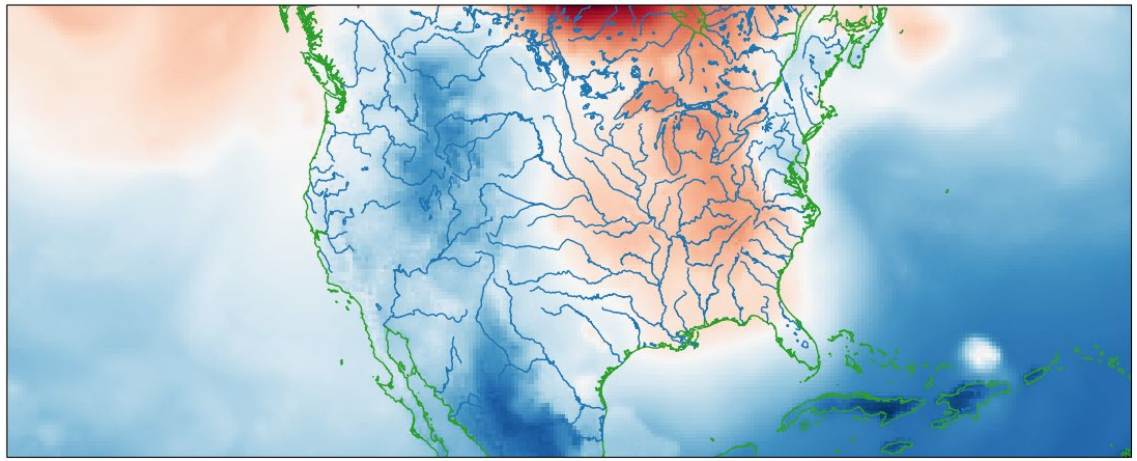
Development is supported by the National Science Foundation..

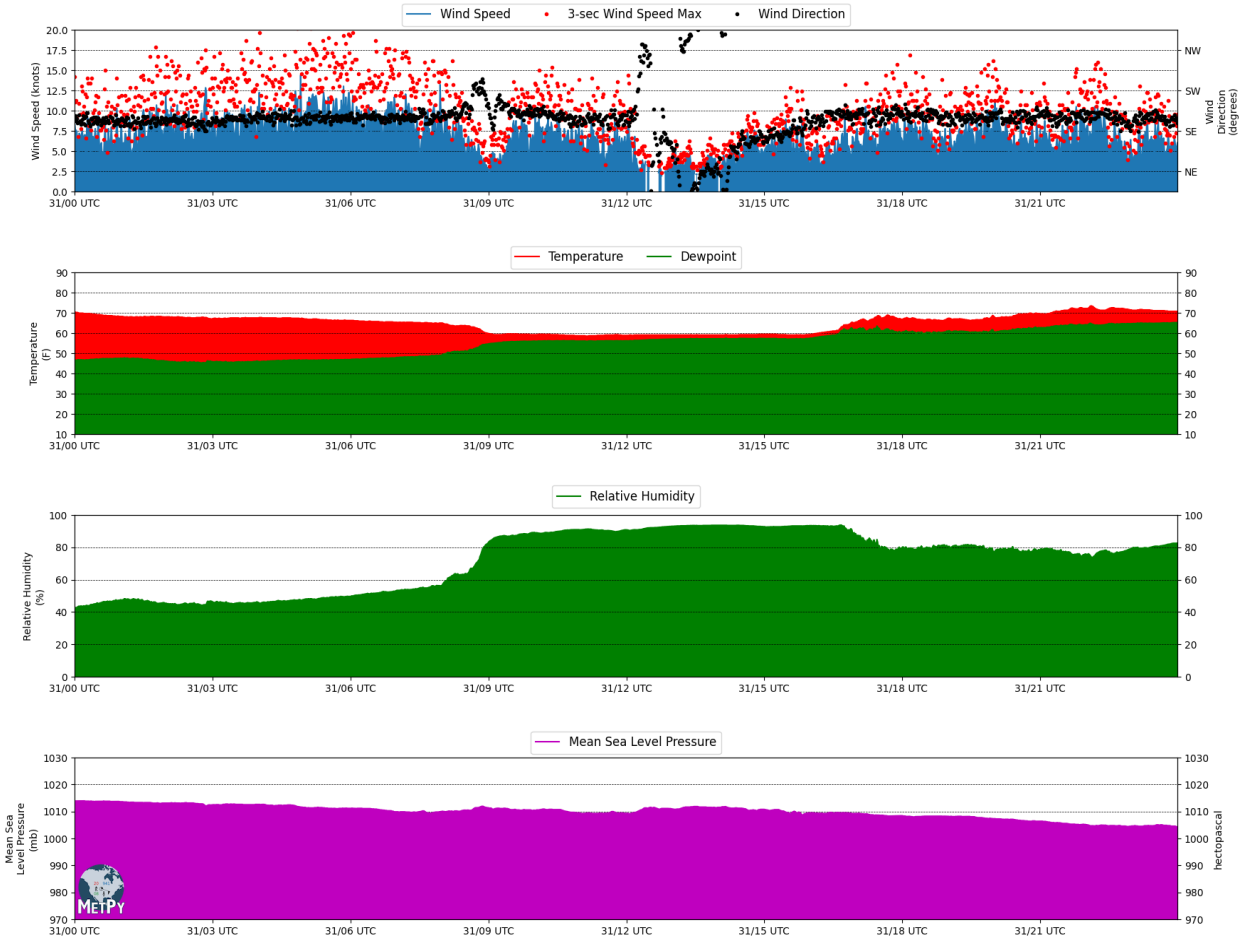
Primary Uses:

Meteorological research, including performing calculations, reading data, and plotting.

MetPy Usage Examples

1. Plotting time series data as a meteogram (right)
2. Plotting data on a map using XArray and CartoPy (below)





1.1.0 Milestones

- Code enhancements or bug fixes to be addressed for the 1.1.0 update.
- Presented as “issues” in GitHub to be addressed before the update is implemented.

Issue 1844

Initial problem:

- **pyproj** CF (climate and forecasting) output not accepted by **metpy.assign_crs().**
  - The function **Metpy.assign_crs()** assigns a coordinate reference system to the MetPy data array based on CF projection attributes.

Initial fix:

- Adding **earth_radius** to the input directory.

Issue 1844

New problem:

- Latitude of projection center missing in CF listing.
  - The value of lat_0 is lost.

Cause:

- Conversion from PyProj to CF results in a value 0 for the attribute **inverse_flattening.**

New fix:

- Interpret the 0 inverse_flattening as a spherical datum and do not pass that value on.

Addressing error

- To address the issue, added an ‘if’ statement to address the case where inverse_flattening = 0

```
63        # interpret the 0 inverse_flattening as a spherical datum
64        # and don't pass the value on.
65        if kwargs.get('inverse_flattening', None) == 0:
66            kwargs['ellipse'] = 'sphere'
67            kwargs.pop('inverse_flattening', None)
```

Code Verification

- Before fixes are merged with MetPy, need to verify it works as expected.
- This is done through **unit testing**.
- Starts with the smallest components first:
  - Ensures they work properly before integrating them with larger portions of code.

DEFINITION: *Unit Testing*

A piece of code that "activates" a piece of a system to ensure it behaves as expected by developers.

Code Verification

Goal

- Isolate each part of the program and show it is correct.

Importance

- Finds problems early as code is developed.
- Forces developers to think through code thoroughly.
- Neglecting tests can lead to broken code and problems for users.

Test for Issue 1844

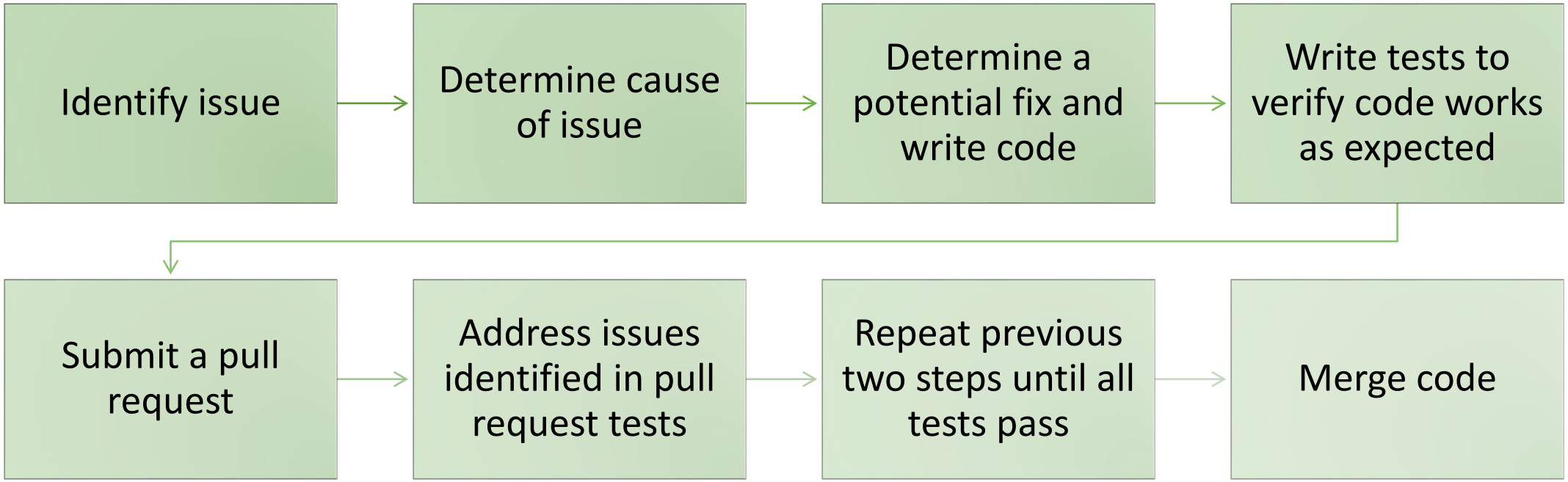
- Need to test new code by writing a test to “activate” it.
- For Issue 1844, introduce the case where inverse_flattening = 0.

```
13  def test_inverse_flattening_0():
14      """Test new code for dealing the case where inverse_flattening = 0."""
15      attrs = {'grid_mapping_name': 'lambert_conformal_conic', 'semi_major_axis': 6367000,
16               'semi_minor_axis': 6367000, 'inverse_flattening': 0}
17      proj = CFProjection(attrs)
18
19      crs = proj.to_cartopy()
20      globe_params = crs.globe.to_proj4_params()
21
22      assert globe_params['ellps'] == 'sphere'
23      assert globe_params['a'] == 6367000
24      assert globe_params['b'] == 6367000
```

Pull request process

1. Submit pull request
   - Submits the changed code for testing and review
2. Automated tests
   - Identify code that may have been missed by manual testing process.
   - Check for drops in code coverage and style variations.
3. Code review by Unidata staff
4. Merging
   - Performed once all tests and details of the pull request are addressed.

Complete Process



Summary

- Code verification is an essential component to code development.
  - Unit testing is the primary way this is achieved.
- Failing to perform code verification can lead to broken code and lack of functionality.
- MetPy is a program used for a variety of applications and by a variety of users:
  - This makes adequate testing even more important.
  - Broken code can have a lasting impact on research & user experience.

Thank you!