CROSS SECTIONS IN METPY

(WITH XARRAY...AND XKCD)

UNIDATA SUMMER INTERNSHIP 2018

JON THIELEN (WITH RYAN MAY AND JOHN LEEMAN)
WORKING ON METPY

- STARTED CONTRIBUTING IN NOV 17'
- APPLIED IN JAN 18'
- STARTED INTERNSHIP IN MAY
INITIAL PLANS

- AUTOMATIC FIELD CALCULATION
- INCOMPLETE STATE OF XARRAY INTEGRATION
ANOTHER PLAN: CROSS SECTIONS...
...WITH XARRAY!

THAT WOULD SOLVE MY PROBLEMS!

THAT WOULD GIVE ME EXCITING NEW PROBLEMS!
WHAT IS XARRAY?

• PYTHON PACKAGE FOR N-DIMENSIONAL LABELED ARRAYS
• "AN IN-MEMORY REPRESENTATION OF A NETCDF FILE"
• THE FUTURE OF METPY (FROM A DATA MODEL POINT-OF-VIEW)
import xarray as xr

data = xr.open_dataset('irma_gfs_example.nc')

print(data)
<xarray.Dataset>

Dimensions:  
(isobaric1: 21, isobaric3: 1)

Coordinates:
  * time1       (time1) datetime64[ns]
  * reftime     datetime64[ns] ...
  * latitude    (latitude) float32 50
  * isobaric3   (isobaric3) float64 1
  * isobaric1   (isobaric1) float64 1
  * longitude   (longitude) float32 2

Data variables:
  Vertical_velocity_pressure_isobaric  (time1, isobaric1, latitude) ...
  Relative_humidity_isobaric          (time1, isobaric3, latitude) ...
  Temperature_isobaric                (time1, isobaric3, latitude) ...
  u-component_of_wind_isobaric        (time1, isobaric3, latitude) ...
  v-component_of_wind_isobaric        (time1, isobaric3, latitude) ...

```python
heights = data['Geopotential_height_isobaric']
heights.sel(time1='2017-09-06T00:00Z',
            isobaric3=50000.)
```
<xarray.DataArray 'Geopotential_height_isobaric' (latitude: 81)
array([[5880.9595, 5878.8394, 5876.5195, ... , 5807.9194, 5809.
[5884.5596, 5882.3994, 5879.9194, ... , 5821.2393, 5822.
[5888.4395, 5885.7993, 5883.2393, ... , 5833.3193, 5834.
... ,
[5871.679 , 5871.719 , 5871.5996, ... , 5894.8394, 5894.
[5871.2393, 5871.159 , 5871.159 , ... , 5892.8794, 5893.
[5870.5195, 5871.119 , 5870.7993, ... , 5891.119 , 5891.
dtype=float32)
Coordinates:
    time1         datetime64[ns] 2017-09-06
    reftime       datetime64[ns] ...
* latitude     (latitude) float32 50.0 49.5 49.0 48.5 48.0 47.5
    isobaric3    float64 5e+04
* longitude    (longitude) float32 250.0 250.5 251.0 251.5 252.0
data['Temperature_isobaric'].mean(
('time1', 'latitude', 'longitude'))
<xarray.DataArray 'Temperature_isobaric' (isobaric3: 31)>
array([[259.4611, 255.66313, 248.79195, 240.33334, 235.54622,
       223.24205, 218.46536, 213.04253, 208.66364, 204.48088,
       219.08775, 229.3075 , 238.7891 , 247.04654, 253.94908,
       264.77377, 269.32214, 273.50998, 277.24646, 280.57074,
       286.40454, 289.0842 , 291.59814, 292.82648, 294.26868,
       297.45053],
      dtype=float32)
Coordinates:
  * reftime   datetime64[ns] ...
  * isobaric3 (isobaric3) float64 100.0 200.0 300.0 500.0 700
(data['Temperature_isobaric'] - 
data['Temperature_isobaric'].mean( 
  ('time1', 'latitude', 'longitude')))

<xarray.DataArray 'Temperature_isobaric' (time1: 9, isobaric3: 9) array:

```
[[[  -1.46109 ,  ...,  -1.161102],
   ...,
   [  1.738922,  ...,  -1.161102]],
   ...
   ...
   [[[  -7.250519,  ...,  -10.750519],
      ...,
      [  1.749481,  ...,  1.849457]]],
   ...
```

SO WHAT'S MISSING?

- PROJECTION HANDLING
- SYSTEMATIC IDENTIFICATION OF VARIABLES/COORDINATES
- UNITS
- METEOROLOGICAL CALCULATIONS
That would solve my problems!

That would give me exciting new problems!
XARRAY AND METPY
PROJECTION HANDLING

temperature = data.metpy.parse_cf('Temperature_isobaric')
temperature.metpy.cartopy_crs

<cartopy.crs.PlateCarree object at 0x7f86c780e3b8>
Xarray + CF + CartoPy Projection Handling #786

Merged  jruleman merged 10 commits into Unidata:master from dopplershift:xarray-projections on May 14

Conversation 34  Commits 10  Checks 0  Files changed 15

Changes from all commits  Jump to...  +683 −31

```yaml
- .codeclimate.yml

---
0  argument-count:
 0 config:
 11 threshold: 10

+ method-complexity:
+ config:
+ threshold: 15
```

Edit

Unidata / MetPy
**SYSTEMATIC IDENTIFICATION OF COORDINATES**

```python
temperature = data.metpy.parse_cf('Temperature_isobaric')
temperature.metpy.vertical

<xarray.DataArray 'isobaric3' (isobaric3: 31)>
array([ 100.,  200.,  300.,  500.,  700., 1000.,
       5000.,  7000., 10000., 15000., 20000., 25000.,
       40000., 45000., 50000., 55000., 60000., 65000.,
       80000., 85000., 90000., 92500., 95000., 97500.,
       100000.])

Coordinates:
  reftime     datetim64[ns] 2017-09-05T12:00:00
  crs    object Projection: latitude_longitude
  * isobaric3 (isobaric3) float64 100.0 200.0 300.0 500.0 700.0

Attributes:
  units: Pa
  positive: down
  axis: Z
```
IF YOU'RE HAVIN' PERL PROBLEMS I FEEL BAD FOR YOU, SON--

I GOT 99 PROBLEMS,

SO I USED REGULAR EXPRESSIONS,

NOW I HAVE 100 PROBLEMS.
def __getitem__(self, item):
    
    """Return a given attribute."""
    return self._attrs[item]

def __eq__(self, other):
    """Test equality (CFProjection with matching attrs)."""
    return self.__class__ == other.__class__ and self.to_dict() == other.to_dict()

def __ne__(self, other):
    """Test inequality (CFProjection with non-matching attrs)."""
    return self.__class__ != other.__class__ and self.to_dict() != other.to_dict()
SYSTEMATIC IDENTIFICATION OF COORDINATES
Systematic identification of variables from an xarray Dataset #886

jthielen commented on Jul 10

Corresponding to #860, it would seem useful to also be able to systematically identify variables from an xarray dataset. A simple use-case would be something like what motivated this issue, #662, where we want to identify each of the components of the 3D wind field and then do some calculations on those. This also would likely be a prerequisite for #3 (whenever enough pieces are in place for that to be implemented).

A initial approach could be simply searching for the standard_name attribute and strictly adhering to the CF Standard Name list, while giving some option to the user to supply a dictionary to fill standard names where they are missing. However, would there be cases where we don’t have a CF standard name for the quantity we want? Or, should there be some kind of automatic processing to fill in for missing standard_name attributes? But, then again, anything too much more flexible/complex would likely become even messier than systematic coordinate identification ended up being.
UNITS

Since the Celsius vs Fahrenheit debate has proven surprisingly hard to resolve, as a compromise I've started using Felsius (°F), the average of the two.

°C = \frac{7°\text{C}}{5} + 16 = \frac{7°\text{F} - 80}{9}
```python
temperature[0, -1].metpy.unit_array

[[290.2 289.7 289.2 ... 286.6 286.6 286.7]
 [289.7 289.7 289.6 ... 286.5 286.2 286.1]
 [289.6 289.4 289.6 ... 286.2 286.  286. ]
...
 [299.5 299.5 299.4 ... 299.9 300.  300.1]
 [299.2 299.2 299.1 ... 299.9 299.5 299.7]
 [299.2 298.9 298.8 ... 299.9 299.5 299.3]] kelvin
```
dew_point = mpcalc.dewpoint_rh(data['temperature'][0, -1],
data['relative_humidity'][0, -1]).to('degF')
print(dew_point)

[[32.106335  34.216534  37.958862 ...  54.41962  54.06663  53.707
[37.16378  37.290783  39.498154 ...  54.56085  53.85064  53.439
[36.5536  37.554565  39.08699  ...  54.3687  53.896233 53.551
...[73.24063  73.3862  73.355545 ... 70.961296 70.89378 71.066
[72.96811  73.04029 73.04425  ...  70.720695 70.78227 70.853
[73.148285 72.76404  72.409294 ...  69.40728  70.028336 70.317
(data['height'].sel(isobaric3=500) - data['height'].sel(isobaric3=1000))

```
xarray.DataArray 'height' (time1: 9, latitude: 81, longitude:
array([[5643.725, 5635.4604, ..., 5619.6846, 5617.981],
       [5639.421, 5635.045, ..., 5630.989, 5629.373],
       ...
       [5778.173, 5778.397, ..., 5761.6445, 5763.4287],
       [5777.149, 5776.445, ..., 5760.085, 5760.773]],
      [[5654.589, 5647.1167, ..., 5637.925, 5636.581],
       [5649.821, 5645.093, ..., 5646.637, 5645.485],
       ...
       [5776.485, 5775.877, ..., 5766.0366, 5766.7407],
       [5775.261, 5774.6606, ..., 5764.285, 5764.565]],
      ...
```

...
HOW TO CALCULATE THE “HEAT INDEX”:

MEASURE THE TEMPERATURE

DOES IT LOOK HOT ENOUGH?

YES → DONE

NO → ADD A FEW DEGREES
BUT WHAT ABOUT CROSS SECTIONS?
• CROSS SECTIONS—A FORM OF INTERPOLATION
• METPY'S INTERPOLATION WAS PREVIOUSLY
  ■ METPY.GRIDDING
  ■ A FEW FUNCTIONS IN METPY.CALC
• SO...
Changing gridding to interpolate #872

This PR is a major API change: it deprecates the `gridding` subpackage in favor of a more general `interpolate` subpackage, and brings over the current single-axis interpolation functions now in `calc` as well. This is motivated by the soon-to-arrive cross section implementation, as cross sections require more general point-/path-based horizontal interpolation, rather than strictly grid-based (what is currently implemented in the `gridding` subpackage).

Below is the list of changes I've made at this point:

- The internal module structure of `gridding` has been updated for the `interpolate` package (so that users using non-public functions from `gridding` will have to change more than just the subpackage name)
WITH THAT OUT OF THE WAY...
CROSS SECTIONS!
Copyright (c) 2018 MetPy Developers.
Distributed under the terms of the BSD 3-Clause License.

Tools for interpolating to a vertical slice/cross section through data.

```python
import cartopy.crs as ccrs
import numpy as np
import xarray as xr

from ..package_tools import Exporter
from ..xarray import CFConventionHandler

exporter = Exporter(globals())
```
LIVE DEMO TIME!
THANK YOU!

- RYAN AND JOHN
- HAILEY
- SEAN
- INKEN AND SHERI
- MATT
- ETHAN
- EVERYBODY!
WE NEED TO MAKE 500 HOLES IN THAT WALL, SO I'VE BUILT THIS AUTOMATIC DRILL. IT USES ELEGANT PRECISION GEARS TO CONTINUALLY ADJUST ITS TORQUE AND SPEED AS NEEDED.

GREAT; IT'S THE PERFECT WEIGHT! WE'LL LOAD 500 OF THEM INTO THE CANNON WE MADE AND SHOOT THEM AT THE WALL.

HOW SOFTWARE DEVELOPMENT WORKS