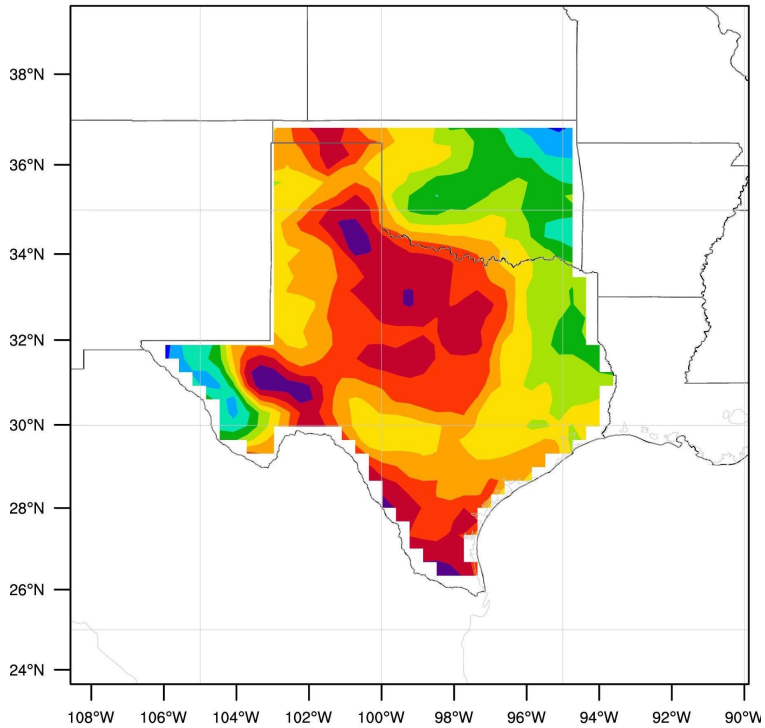


Post-Processing Tools: Python

Abby Jaye
jaye@ucar.edu

February 2025

This material is based upon work supported by the National Center for Atmospheric Research, which is a major facility sponsored by the National Science Foundation under Cooperative Agreement No. 1852977.

Python and WRF

- NSF NCAR has transitioned to focusing primarily on visualization development in Python as opposed to NCL
- NCL will still be supported and bugs will be fixed, but development is done
- There are many packages that exist where the same features exist and are easier to use than with NCL

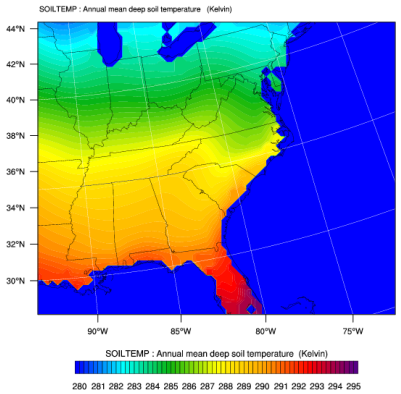
https://wrf-python.readthedocs.io/en/latest/

Python and WRF

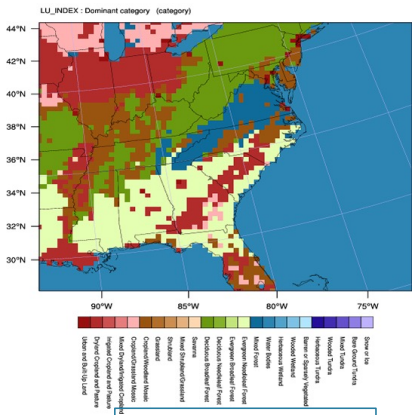
- Packages you should have:
  - WRF-Python
    - Analyzes WRF-ARW data directly
    - Has much of the same functionality of NCL
  - xarray
    - Supports multi-dimensional arrays
  - matplotlib
    - Great for making plots quickly
  - cartopy
    - Makes maps!
  - netCDF4
    - To read in your data for use in WRF-Python

Example Plots

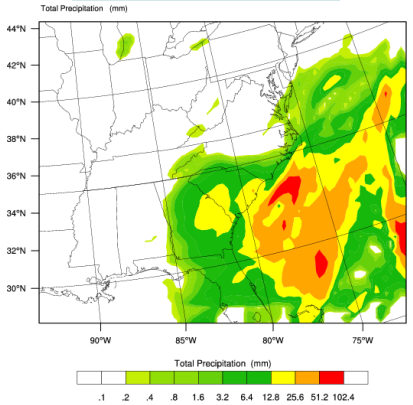
Soil Temperature



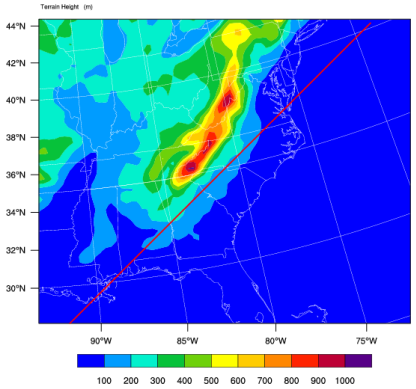
LU-INDEX



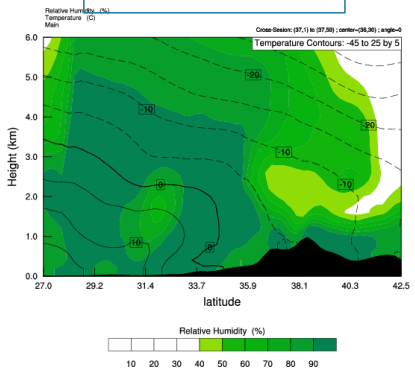
Total Precipitation



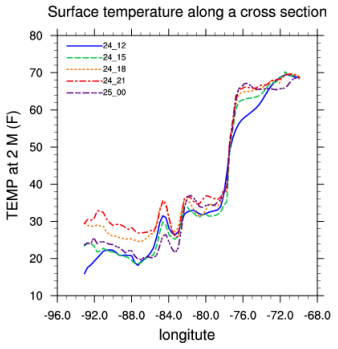
Terrain Height



Relative Humidity Vertical X-section



Sfc Temperature Horizontal X-section



Special WRF-Python Functions

- getvar
  - *Get native and diagnostic variables*
- interplevel
  - *Returns the 3D field interpolated to a horizontal plane at the specified vertical level*
- vertlevel
  - *Returns the vertical cross section for a 3D field*
- interpline
  - *Returns the 2D field interpolated along a line*
- vinterp
  - *Returns the field vertically interpolated to the given type of surface and a set of new levels*
- ll_to_xy / xy_to_ll
  - *Returns the x,y coordinates for a specified lat/lon… and the other way around*
- destagger
  - *Returns the variable on the unstaggered grid*

getvar

- avo
- eth/theta_e
- cape_2d (MCAPE/MCIN/LCL/LFC)
- cape_3d (3D cape and cin)
- ctt
- cloudfrac
- dbz
- mdbz
- geopotential
- geopt_stag
- helicity
- lat
- lon
- omega
- pres
- pressure
- pvo
- pw
- rh
- rh2
- slp
- T2
- ter
- td2
- td
- th/theta
- temp
- tk
- times
- tv
- twb
- updraft_helicity
- ua/va/wa
- uvmet10
- uvmet
- wspd_wdir
- wspd_wdir10
- z
- height_agl
- zstag
- uvmet_wspd_wdir
- uvmet10_wspd_wdir

```
tc2 = getvar(a, "T2", timeidx=ALL_TIMES)-273.15
```

Easy plotting!

```
[1]: from netCDF4 import Dataset
     import numpy as np
     import xarray as xr
     import os
     import matplotlib.pyplot as plt
     from matplotlib.cm import get_cmap
     from matplotlib.backends.backend_pdf import PdfPages
     import cartopy.crs as crs
     from cartopy.feature import NaturalEarthFeature
     from wrf import getvar, get_pyngl, latlon_coords, to_np, ALL_TIMES
```

```
[3]: # Open wrfout file with multiple times
     a = Dataset("/glade/work/jaye/ncl2py/wrfout_all.nc","r")
```

```
[14]: tc2 = getvar(a, "T2", timeidx=ALL_TIMES, meta=True)-273.15
```

```
[15]: tc2
```

```
[15]: xarray.DataArray 'T2' (Time: 16, south_north: 359, west_east: 461)
```

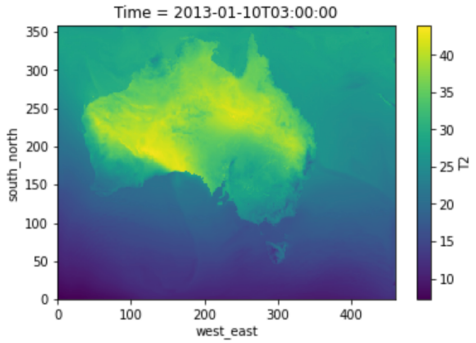
```
array([[[ 7.037018 ,  7.0637817,  7.1075745, ..., 12.13278 ],
        [ 7.0884705,  7.142578 ,  7.1893005, ..., 12.08303 ],
        [ 7.1588135,  7.214508 ,  7.258423 , ..., 12.15008 ],
        ...,
        [26.292694 , 26.272308 , 26.338745 , ..., 28.00772 ],
        [26.23529  , 26.23883  , 26.294556 , ..., 27.99585 ],
        [26.196228 , 26.23233  , 26.263397 , ..., 27.93350 ]]],
      dtype=float32)
Coordinates:
```

XLONG	(south_north, west_east)	float32	109.33436 109.45899 ... 166.6...
XLAT	(south_north, west_east)	float32	-47.66589 -47.66589 ... -9.695...
Time	(Time)	datetime64[ns]	2013-01-10 ... 2013-01-10T15:0...

▸ Attributes: (0)

```
[16]: tc2.isel(Time=3).plot()
```

```
[16]: <matplotlib.collections.QuadMesh at 0x2aed0684bc90>
```




```
[ ]:
```

Screenshot

vinterp

- Interpolate to:
  - "pressure", "pres" - pressure [hPa]
  - "ght_msl" - grid point height msl [km]
  - "ght_agl" - grid point height agl [km]
  - "theta" - potential temperature [K]
  - "theta-e" - equivalent potential temperature [K]

- Extrapolate below the ground
  - **extrapolate=True**

vinterp

[17]:
```
from netCDF4 import Dataset
import numpy as np
import xarray as xr
import os
import matplotlib.pyplot as plt
from matplotlib.cm import get_cmap
from matplotlib.backends.backend_pdf import PdfPages
import cartopy.crs as crs
from cartopy.feature import NaturalEarthFeature
from wrf import getvar, get_pyngl, vinterp, latlon_coords, to_np, ALL_TIMES, smooth2d, get_
```

[18]:
```
# Open wrfout file with multiple times
a = Dataset("/glade/work/jaye/ncl2py/wrfout_all.nc","r")
```

[20]:
```
tk = getvar(a,"tk",timeidx=3)
```

[21]:
```
tk
```

[21]: xarray.DataArray 'temp' (bottom_top: 50, south_north: 359, west_east: 461)

```
array([[[279.82364, 279.8241 , 279.8463 , ..., 284.83   , 284.85657,
         284.86838],
```

[22]:
```
vert_coords = "pressure"
interp_levels = [200,300,500,1000]
```

[26]:
```
tk_vint = vinterp(a,tk,vert_coords,interp_levels,extrapolate=True)
```

[27]:
```
tk_vint
```

[27]: xarray.DataArray 'temp' (**interp_level**: 4, south_north: 359, west_east: 461)

```
array([[[225.81506, 225.79605, 225.73921, ..., 229.932  , 229.8977 ,
         229.87909],
        [225.805  , 225.66464, 225.62424, ..., 229.86417, 229.78374,
         229.8116 ],
        [225.75153, 225.622  , 225.57852, ..., 229.82507, 229.76176,
         229.80647],
        ...,
        [222.08827, 222.10077, 222.09315, ..., 222.35434, 222.37094,
         300.1926 ]]], dtype=float32)
```

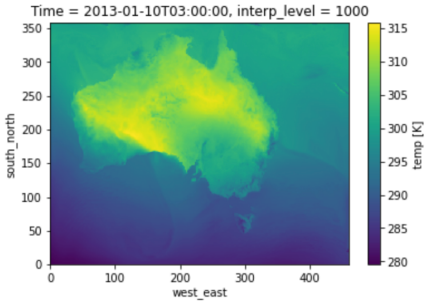
▾ Coordinates:

XLONG	(south_north, west_east)	float32	109.33436 109.45899 ... 166.6...
XLAT	(south_north, west_east)	float32	-47.66589 -47.66589 ... -9.695...
Time	()	datetime64[ns]	2013-01-10T03:00:00
interp_level	(interp_level)	int64	200 300 500 1000

▸ Attributes: (10)

[33]:
```
tk_vint.sel(interp_level=1000).plot()
```

[33]:
```
<matplotlib.collections.QuadMesh at 0x2aed0d896590>
```



vinterp

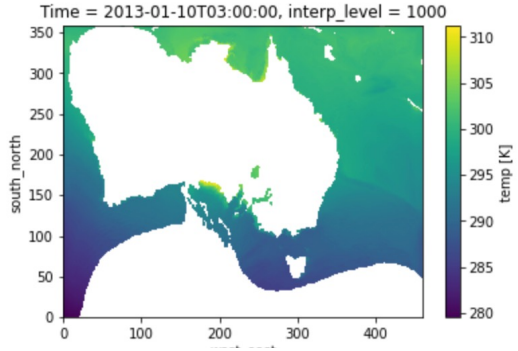
```
[30]: vert_coords = "pressure"
      interp_levels = [200,300,500,1000]

[34]: tk_vint = vinterp(a,tk,vert_coords,interp_levels,extrapolate=False)

[35]: tk_vint
```

[35]: xarray.DataArray 'temp' (**interp_level**: 4, south_north: 359, west_east: 461)

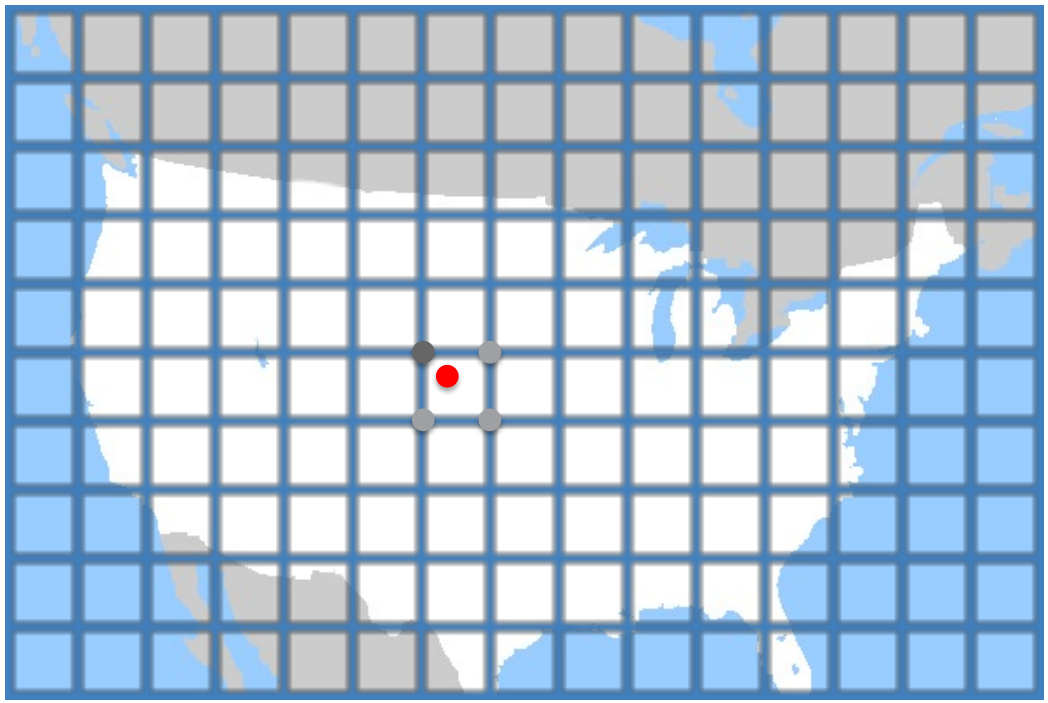
```
array([[[225.81506, 225.79605, 225.73921, ..., 229.932  , 229.8977 ,
         229.87909],
        [225.805  , 225.66464, 225.62424, ..., 229.86417, 229.78374,
         229.8116 ],
        [225.75153, 225.622  , 225.57852, ..., 229.82507, 229.76176,
         229.80647],
        ...,
        [222.08827, 222.10077, 222.09315, ..., 222.3543
```

```
[36]: tk_vint.sel(interp_level=1000).plot()
```

[36]: <matplotlib.collections.QuadMesh at 0x2aed0d95bb10>



ll_to_xy



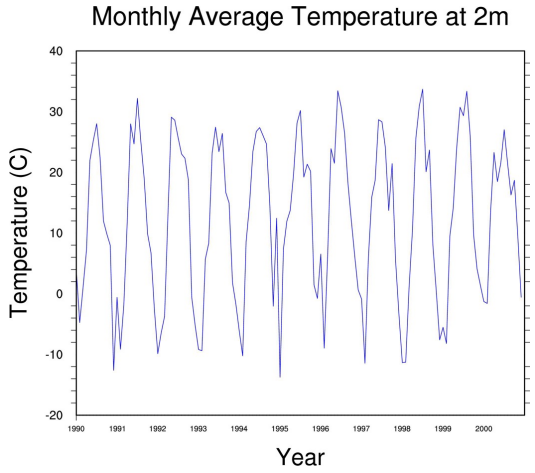
- Your point of interest will likely not coincide with a model grid point
- Most researchers pick the closest point
- Alternatively, pick all points around point of interest and interpolate

```
locij = ll_to_xy(a, lat, lon, True)
```

ll_to_xy

```
locij = ll_to_xy(a,40.02,-105.27,True)
locY = locij(0)
locX = locij(1)
t2_point = a.T2[:,locY,locX]
```

T2 taken from a lat/lon point in Boulder



Change Fields in a netCDF File

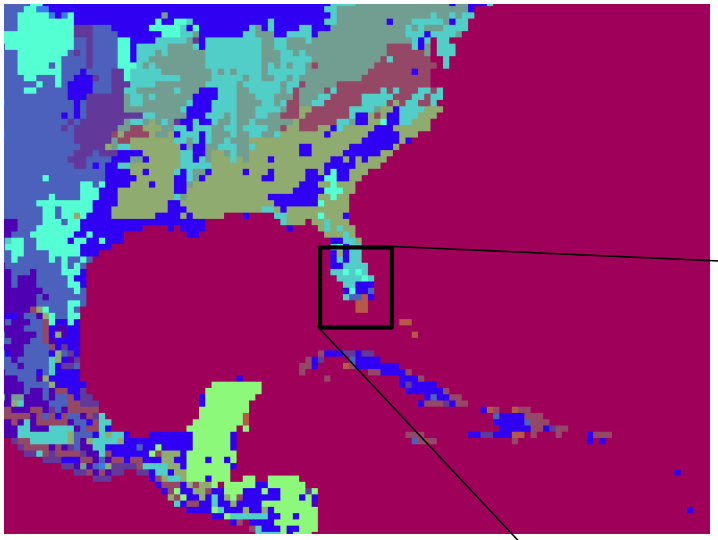
```
[76]: # Open wrfout file with multiple times
      a = xr.open_dataset("/glade/work/jaye/ncl2py/wrfout_all.nc")

[77]: sst = a.SST

[78]: sst
```

```
[78]: xarray.DataArray 'SST' (Time: 16, south_north: 359, west_east: 461)
      [2647984 values with dtype=float32]
      ▾ Coordinates:
          XLAT    (Time, south_north, west_east)  float32 ...
          XLONG   (Time, south_north, west_east)  float32 ...
      ▾ Attributes:
          FieldType :     104
          MemoryOrder :   XY
          description :   SEA SURFACE TEMPERATURE
          units :         K
          stagger :
```

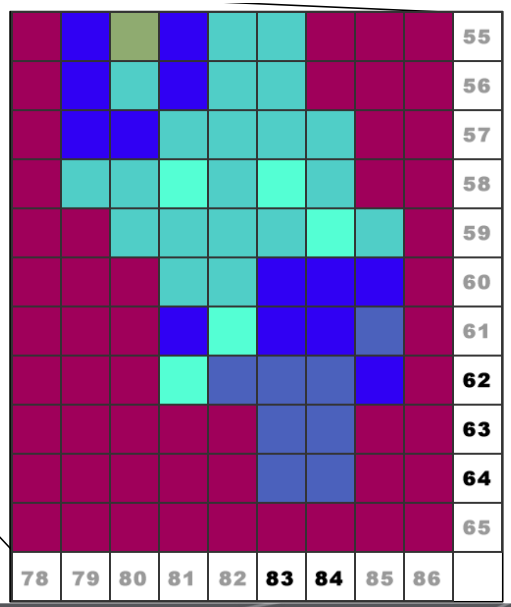
```
[79]: sst_new = sst+1

[81]: ds_new = a.assign(SST=sst_new)

[90]: ds_new.to_netcdf("new_sst.nc")

[83]: test = ds_new.SST - sst

[84]: test
```

```
[84]: xarray.DataArray 'SST' (Time: 16, south_north: 359, west_east: 461)
      array([[[1., 1., 1., ..., 1., 1., 1.],
              [1., 1., 1., ..., 1., 1., 1.],
              [1., 1., 1., ..., 1., 1., 1.],
              ...,
              [1., 1., 1., ..., 1., 1., 1.],
              [1., 1., 1., ..., 1., 1., 1.],
              [1., 1., 1., ..., 1., 1., 1.]],

             [[1., 1., 1., ..., 1., 1., 1.],
              [1.  1.  1.        1.  1.  1.]
```

Change Fields in a netCDF File



```
[3]: # Open wrfout file with multiple times
     a = xr.open_dataset("/glade/work/jaye/WPS/geo_em.d01.nc")

[4]: lu = a.LU_INDEX

[5]: lu[:,63:65,83:85] = 7
     lu[:,62,84] = 7

[6]: ds_new = a.assign(LU_INDEX=lu)
```



Mapping with cartopy

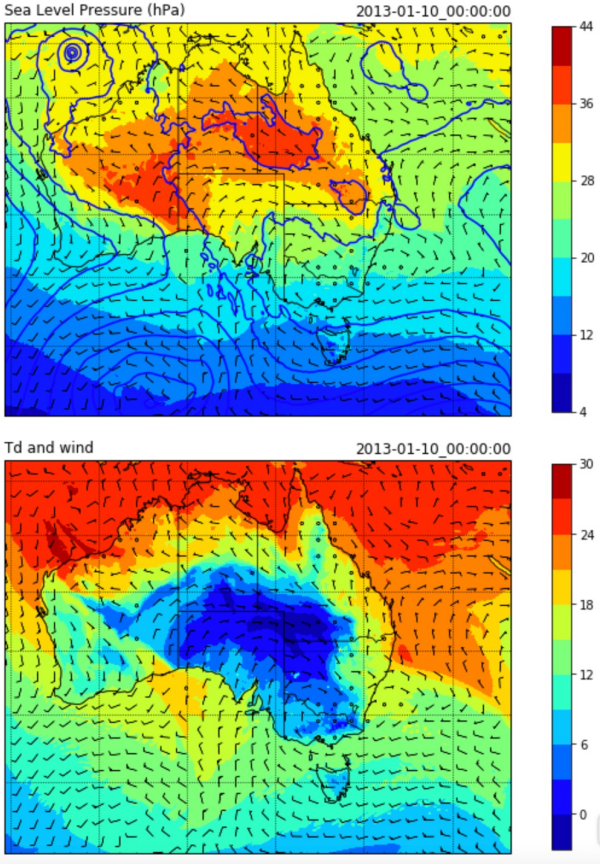
[1]:
```
from netCDF4 import Dataset
import numpy as np
import xarray as xr
import os
import matplotlib.pyplot as plt
from matplotlib.cm import get_cmap
from matplotlib.backends.backend_pdf import PdfPages
import cartopy.crs as crs
from cartopy.feature import NaturalEarthFeature
from wrf import getvar, get_pyngl, latlon_coords, to_np, ALL_TIMES, smooth2d, get_cartopy, cartopy_xlim, cartopy_ylim

# Open wrfout file with multiple times
b = xr.open_dataset("wrfout_all.nc")
a = Dataset("wrfout_all.nc","r")
```

[69]:
```
times = getvar(a,"times",timeidx=ALL_TIMES)
timesxr = b.Times
```

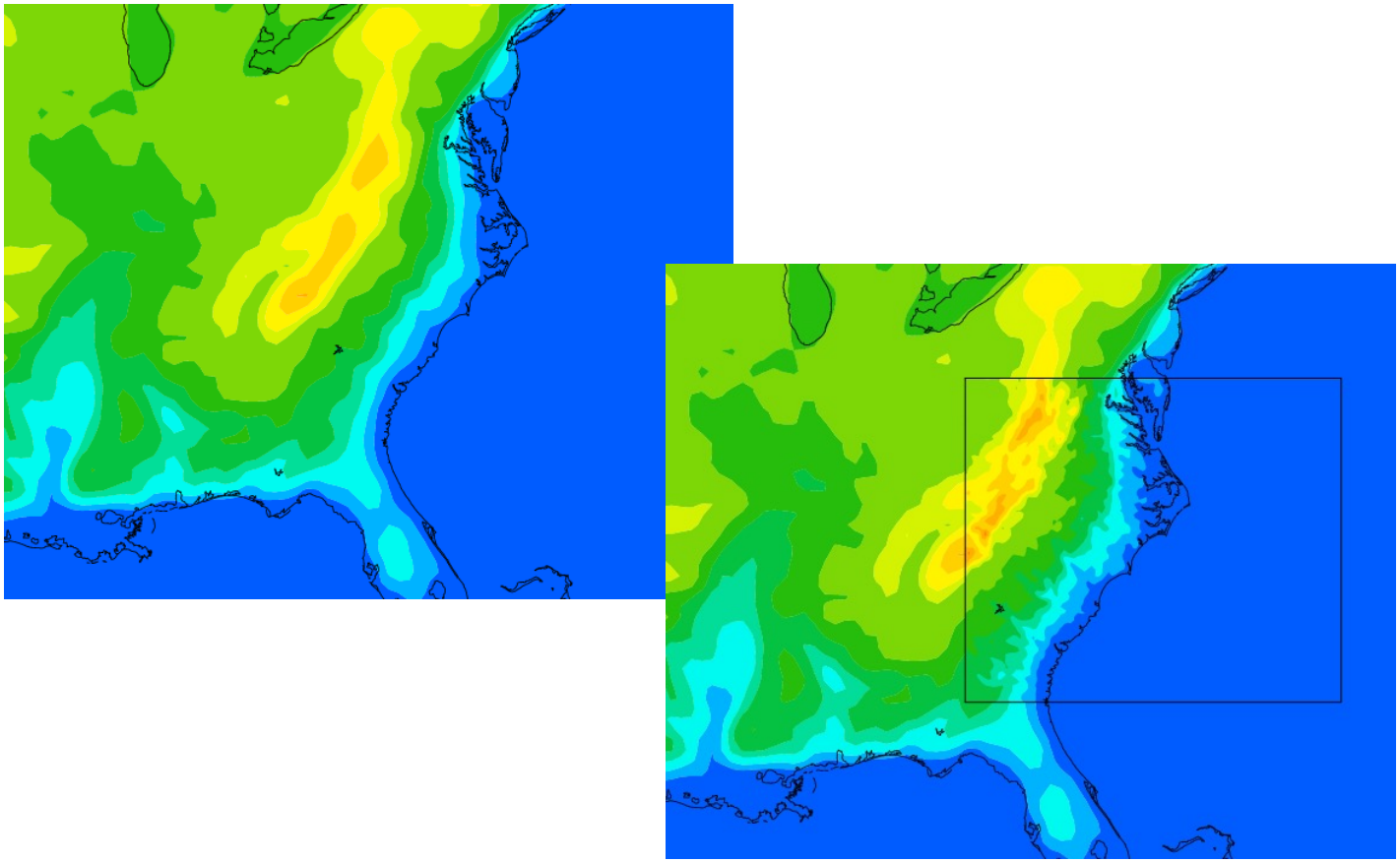
170]:
```
for i in range(len(times)):
    slp = getvar(a, "slp", timeidx=i, meta=True)
    smooth_slp = smooth2d(slp, 3, cenweight=4)
    tc = getvar(a, "tc", timeidx=i, meta=True)
    td = getvar(a, "td", timeidx=i, meta=True)
    u = getvar(a, "ua", timeidx=i, meta=True)
    v = getvar(a, "va", timeidx=i, meta=True)
    td2 = getvar(a, "td2", timeidx=i, meta=True)
    tc2 = getvar(a, "T2", timeidx=i, meta=True)-273.15
    u10 = getvar(a, "U10", timeidx=i, meta=True)
    v10 = getvar(a, "V10", timeidx=i, meta=True)

    # Get the latitude and longitude points
    lats, lons = latlon_coords(slp)

    # Get the cartopy mapping object
    cart_proj = get_cartopy(slp)
    # Create a figure
    fig = plt.figure(figsize=(12,6))
    # Set the GeoAxes to the projection used by WRF
    ax = plt.axes(projection=cart_proj)

    # Download and add the states and coastlines
    states = NaturalEarthFeature(category="cultural", scale="50m",
                                 facecolor="none",
                                 name="admin_1_states_provinces_shp")
    ax.add_feature(states, linewidth=.5, edgecolor="black")
    ax.coastlines('50m', linewidth=0.8)

    # Make the contour outlines and filled contours for the smoothed sea level
    # pressure.
    plt.contour(to_np(lons), to_np(lats), to_np(smooth_slp), 10,
                colors="blue", transform=crs.PlateCarree())
    plt.contourf(to_np(lons), to_np(lats), to_np(tc2), 10,
                 transform=crs.PlateCarree(),
                 cmap=get_cmap("jet"))
    plt.barbs(to_np(lons[::15,::15]), to_np(lats[::15,::15]),
          to_np(u10[::15, ::15]), to_np(v10[::15, ::15]),
          transform=crs.PlateCarree(), length=4, linewidth = 0.5)

    # Add a color bar
    plt.colorbar(ax=ax, shrink=.98)

    # Set the map bounds
    ax.set_xlim(cartopy_xlim(smooth_slp))
    ax.set_ylim(cartopy_ylim(smooth_slp))

    # Add the gridlines
    ax.gridlines(color="black", linestyle="dotted")

    plt.title("Sea Level Pressure (hPa)", loc="left")
    plt.title((timesxr.values[i]).decode(), loc="right")

    #pdftest.savefig()
    plt.show()
```

Mapping with cartopy

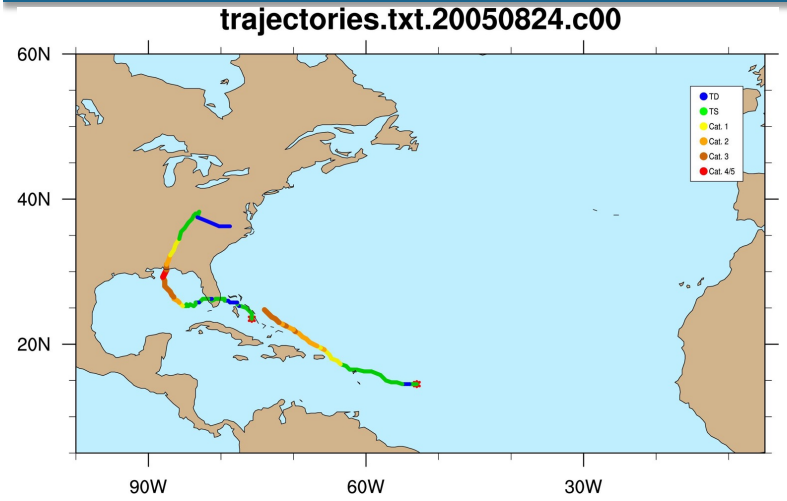
```
    # Get the cartopy mapping object
    cart_proj = get_cartopy(slp)
    # Create a figure
    fig = plt.figure(figsize=(12,6))
    # Set the GeoAxes to the projection used by WRF
    ax = plt.axes(projection=cart_proj)

    # Download and add the states and coastlines
    states = NaturalEarthFeature(category="cultural", scale="50m",
                                 facecolor="none",
                                 name="admin_1_states_provinces_shp")
    ax.add_feature(states, linewidth=.5, edgecolor="black")
    ax.coastlines('50m', linewidth=0.8)

    ab = plt.contourf(to_np(lons), to_np(lats), to_np(td2), 10,
                      transform=crs.PlateCarree(),
                      cmap=get_cmap("jet"))
    plt.barbs(to_np(lons[::15, ::15]), to_np(lats[::15, ::15]),
              to_np(u10[::15, ::15]), to_np(v10[::15, ::15]),
              transform=crs.PlateCarree(), length=4, linewidth = 0.5)
    # Add a color bar
    plt.colorbar(ax=ax, shrink=.98)

    # Set the map bounds
    ax.set_xlim(cartopy_xlim(smooth_slp))
    ax.set_ylim(cartopy_ylim(smooth_slp))

    # Add the gridlines
    ax.gridlines(color="black", linestyle="dotted")
    plt.title("Td and wind", loc="left")
    plt.title((timesxr.values[i]).decode(), loc="right")
    #plt.savefig("test"+str(i)+".png", dpi=300, bbox_inches="tight")
    #pdftest.savefig()
    plt.show()

#pdftest.close()
```

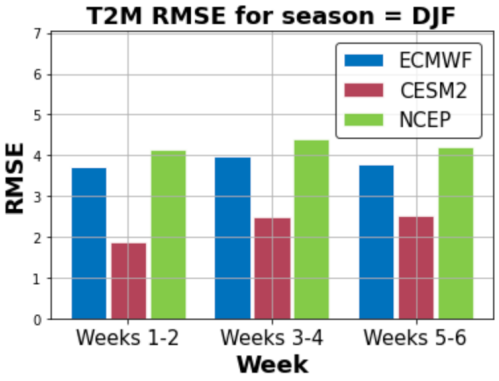


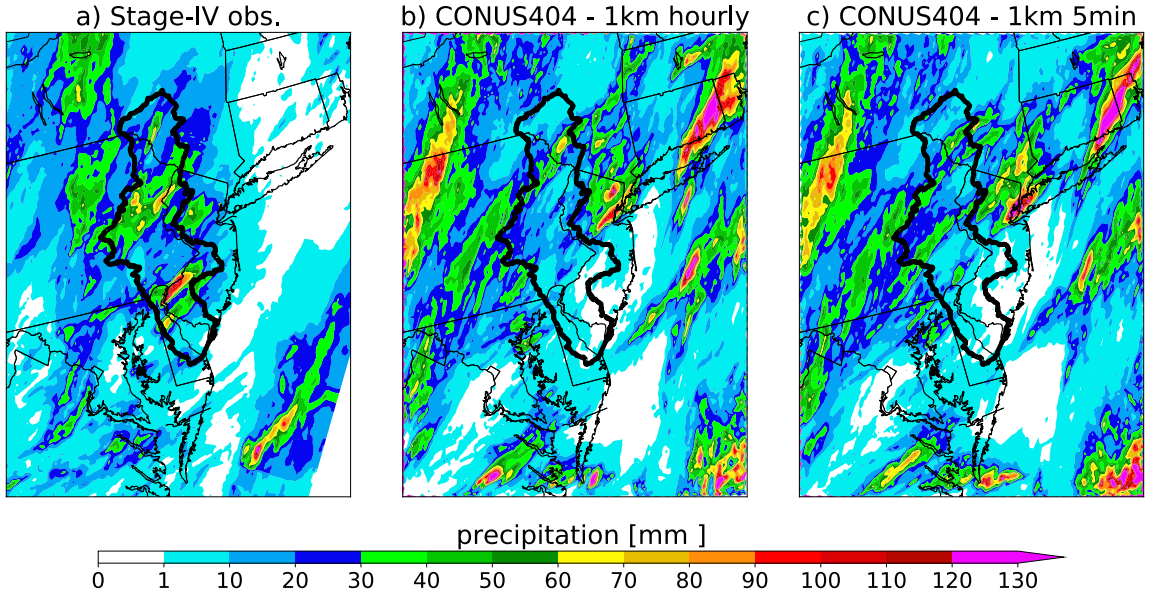
Overlay Domains



More examples







GeoCAT

- GeoCAT is the Geoscience Community Analysis Toolkit
- A collection of Python tools related to NCL developed at NCAR
- Examples page:

  https://geocat-examples.readthedocs.io/
- Updates:

  https://geocat.ucar.edu/news/

MetPy

- A collection of tools for reading, visualizing and performing calculations on weather data
- Developed at Unidata (Part of UCAR with NCAR)
- MetPy Mondays!
- Examples page:

  https://unidata.github.io/MetPy/latest/examples/index.html

  Github:

  https://github.com/Unidata/MetPy

Thank you!

Questions??