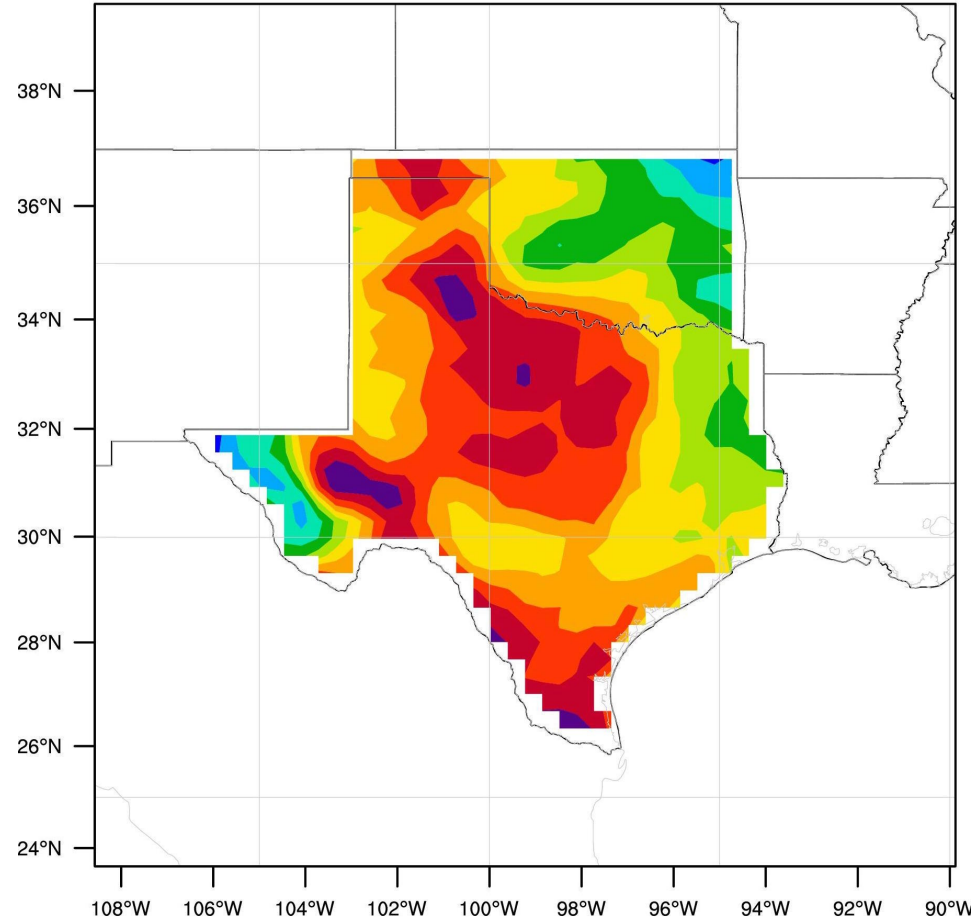


Post-Processing Tools: NCL and Python

Abby Jaye
jaye@ucar.edu

July 2023

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.

NCL

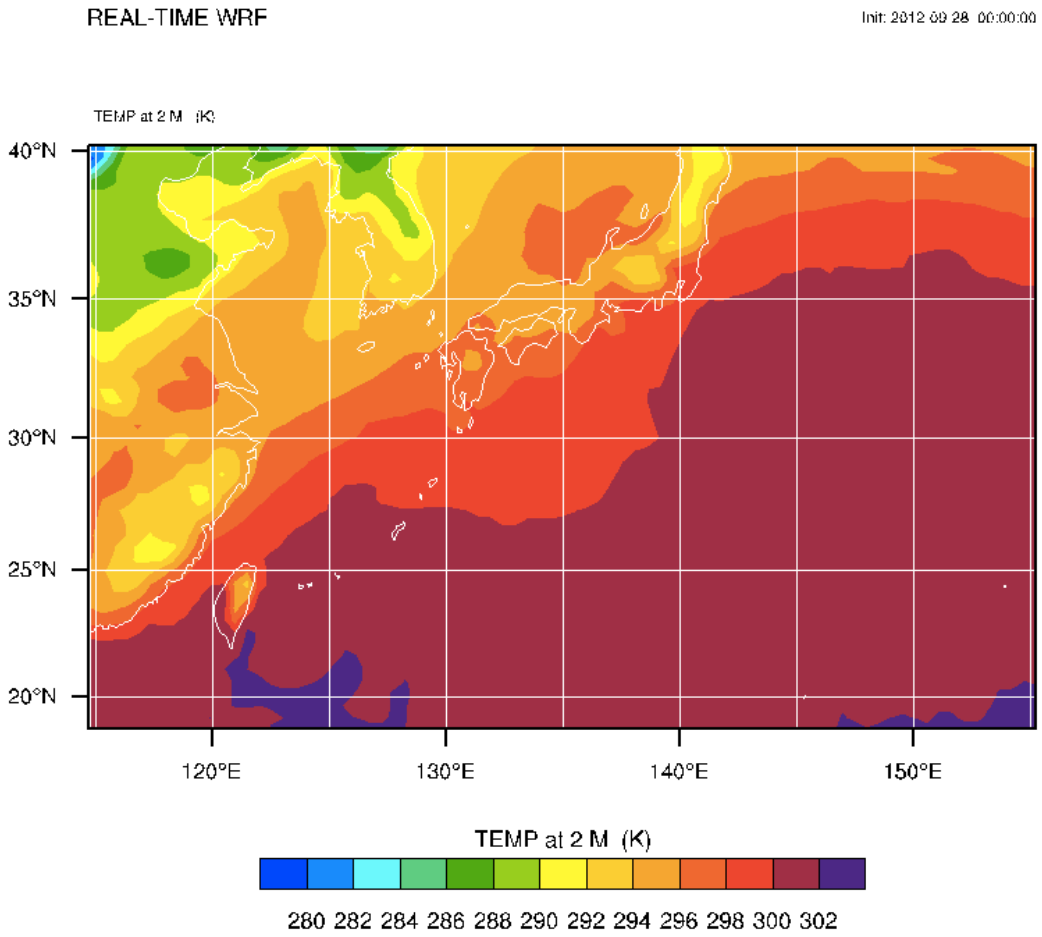
- NCAR Command Language
- Website: http://www.ncl.ucar.edu
- Reads WRF-ARW data directly
- Can generate many types of graphical plots
  - Horizontal
  - Cross-section
  - SkewT
  - Meteogram
  - Panel

Creating a Plot: NCL script

```
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl"
load "$NCARG_ROOT/lib/ncarg/nclscripts/wrf/WRFUserARW.ncl”

begin

 a = addfile("./wrfout_d01_2012-09-28_00:00:00.nc","r")
 wks = gsn_open_wks(“X11”,"plt_Surface")

 T2 = wrf_user_getvar(a,"T2",0)

 pltres = True
 mpres = True
 opts = True
 opts@cnFillOn = True
 contour_t2 = wrf_contour(a,wks,T2,opts)
 plot= wrf_map_overlays(a,wks,(/contour_t2/),pltres,mpres)

end
```

Creating a Plot: NCL script

```
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn
load "$NCARG_ROOT/lib/ncarg/nclscripts/wrf/WRF

begin

 a = addfile("./wrfout_d01_2012-09-28_00:00:00
 wks = gsn_open_wks("X11","plt_Surface")

 T2 = wrf_user_getvar(a,"T2",0)

 pltres = True
 mpres = True
 opts = True
 opts@cnFillOn = True
 contour_t2 = wrf_contour(a,wks,T2,opts)
 plot= wrf_map_overlays(a,wks,(/contour_t2/),pltres,mpres)

end
```

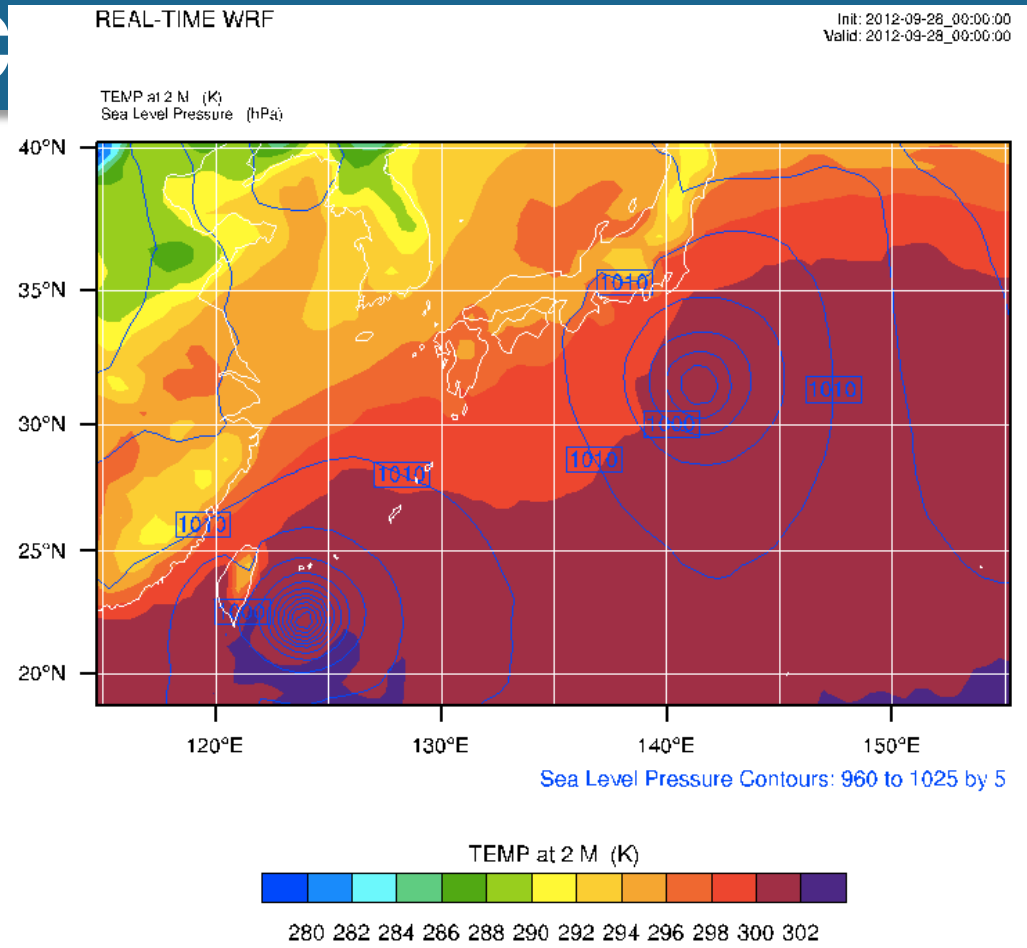


Creating a Plot: NCL script

```
  T2 = wrf_user_getvar(a,"T2",0)
  slp = wrf_user_getvar(a,"slp",0)

  pltres = True
  mpres = True

  opts = True
  opts@cnFillOn = True
  contour_t2 = wrf_contour(a,wks,T2,opts)
  delete(opts)

  opts = True
  opts@cnLineColor = "Blue"
  contour_slp = wrf_contour(a,wks,slp,opts)
  delete(opts)

  plot = wrf_map_overlays(a,wks,(/contour_t2,contour_slp/),
  pltres,mpres)


end
```

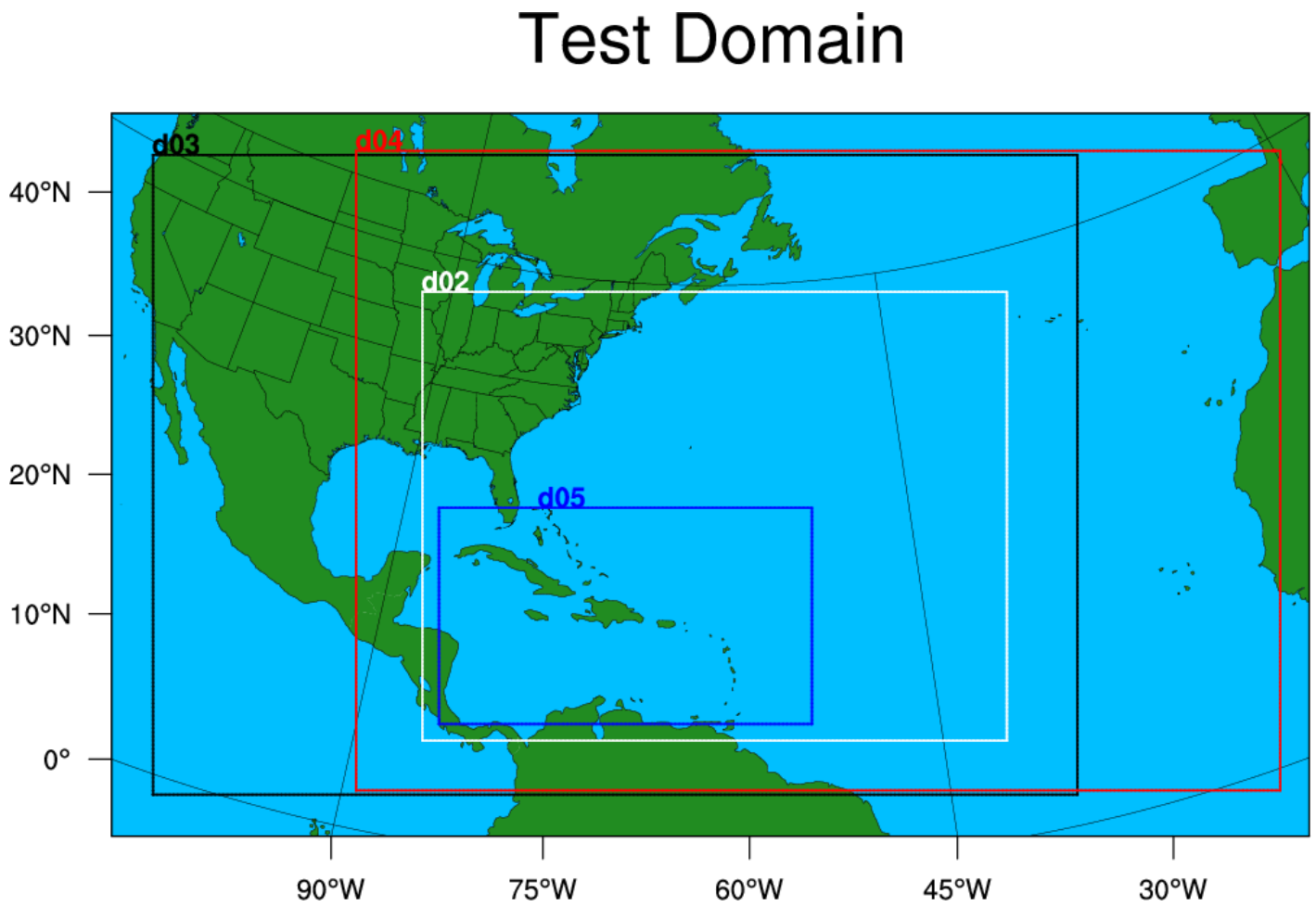
Creating a Plot: NC

```
T2 = wrf_user_getvar(a,"T2",0)
slp = wrf_user_getvar(a,"slp",0)

pltres = True
mpres = True

opts = True
opts@cnFillOn = True
contour_t2 = wrf_contour(a,wks,T2,opts)
delete(opts)

opts = True
opts@cnLineColor = "Blue"
contour_slp = wrf_contour(a,wks,slp,opts)
delete(opts)

plot = wrf_map_overlays(a,wks,(/contour_t2,contour_slp/),
pltres,mpres)


end
```



Domain Design

mp = wrf_wps_dom (wks, mpres, lnres, txres)
WPS/util/plotgrids.ncl



Python and WRF

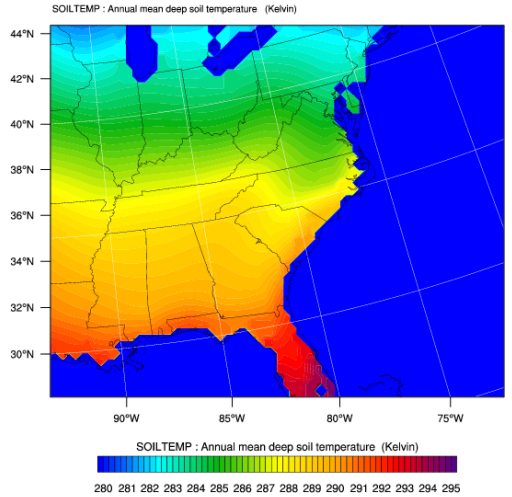
- NCAR has transitioned to focusing primarily on visualization development in Python as opposed to NCL
- NCL will still be supported and bugs will be fixed, but development is done
- There are many packages that exist where the same features exist and are easier to use than with NCL

Python and WRF

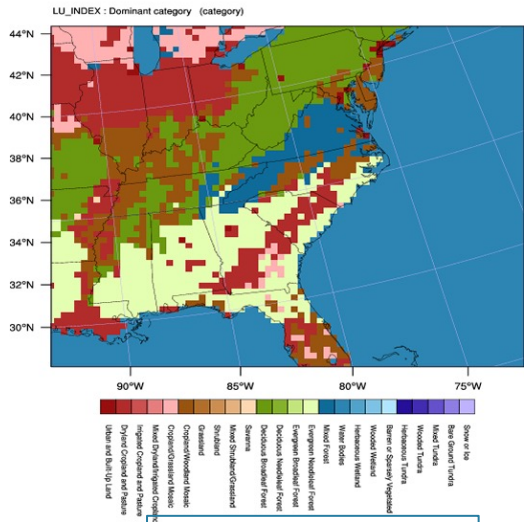
- Packages you should have:
  - WRF-Python
    - Analyzes WRF-ARW data directly
    - Has much of the same functionality of NCL
  - xarray
    - Supports multi-dimensional arrays
  - matplotlib
    - Great for making plots quickly
  - cartopy
    - Makes maps!
  - netCDF4
    - To read in your data for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

Example Plots

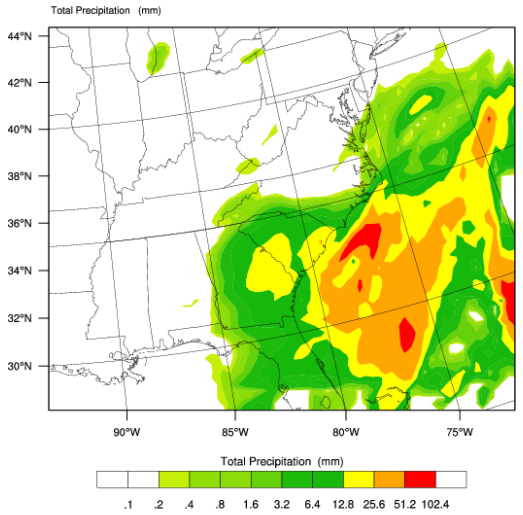
Soil Temperature



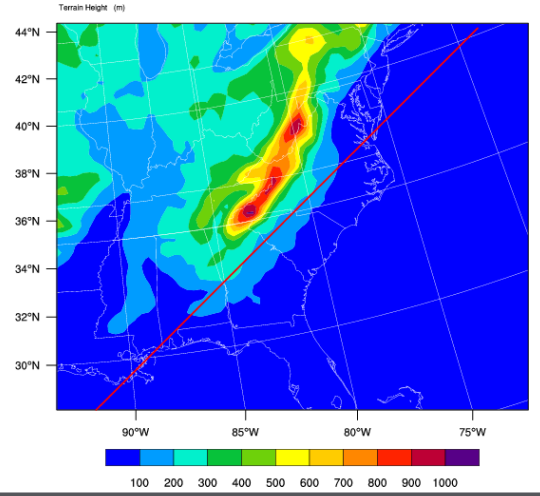
LU-INDEX



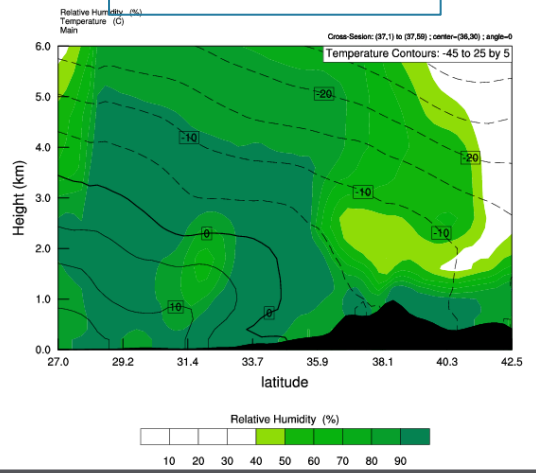
Total Precipitation



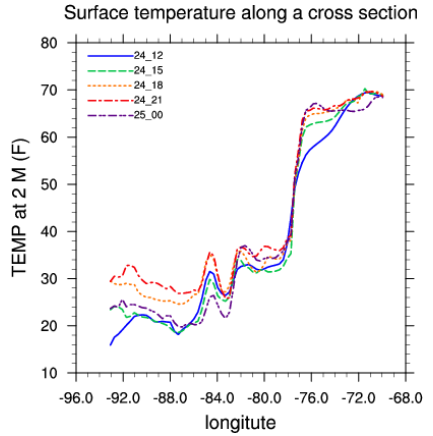
Terrain Height



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
          12.138855 , 12.136261 ],
        [ 7.0884705,  7.142578 ,  7.1893005, ..., 12.08303
          12.071808 , 12.177673 ],
        [ 7.1588135,  7.214508 ,  7.258423 , ..., 12.15008
          12.139008 , 12.19812  ],
        ...,
        [26.292694 , 26.272308 , 26.338745 , ..., 28.00772
          28.044281 , 28.109161 ],
        [26.23529  , 26.23883  , 26.294556 , ..., 27.99585
          28.043427 , 28.072021 ],
        [26.196228 , 26.23233  , 26.263397 , ..., 27.93350
          27.984406 , 28.033752 ]],
```

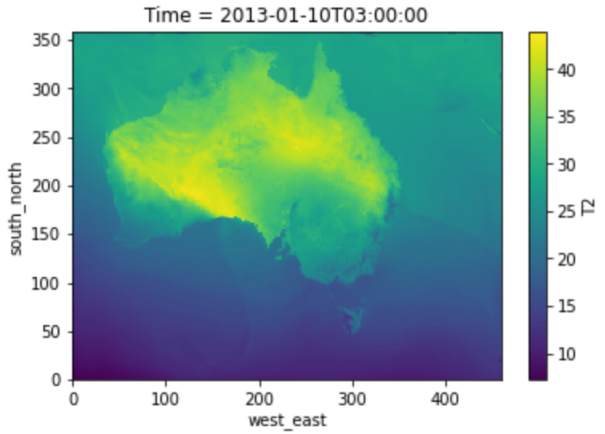
```
        27.314148 , 27.418915 ],
       [27.508087 , 27.595734 , 27.602173 , ..., 27.250366 ,
        27.329071 , 27.420776 ],
       [27.546448 , 27.484894 , 27.431732 , ..., 27.263641 ,
        27.342804 , 27.419586 ]]], dtype=float32)
```

▾ 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,
```

```
[299.84048, 299.8701 , 299.8973 , ..., 300.1016 , 300.1478 ,
 300.1926 ]]], dtype=float32)
```

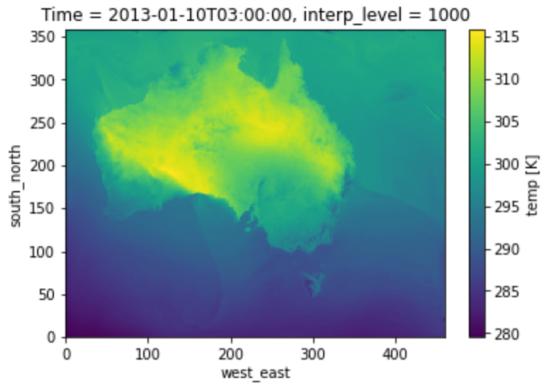
▾ Coordinates:

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

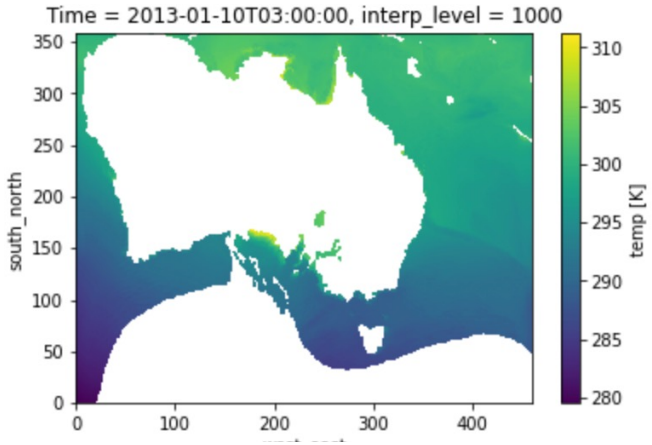
```
[30]: vert_coords = "pressure"
      interp_levels = [200,300,500,1000]

[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)

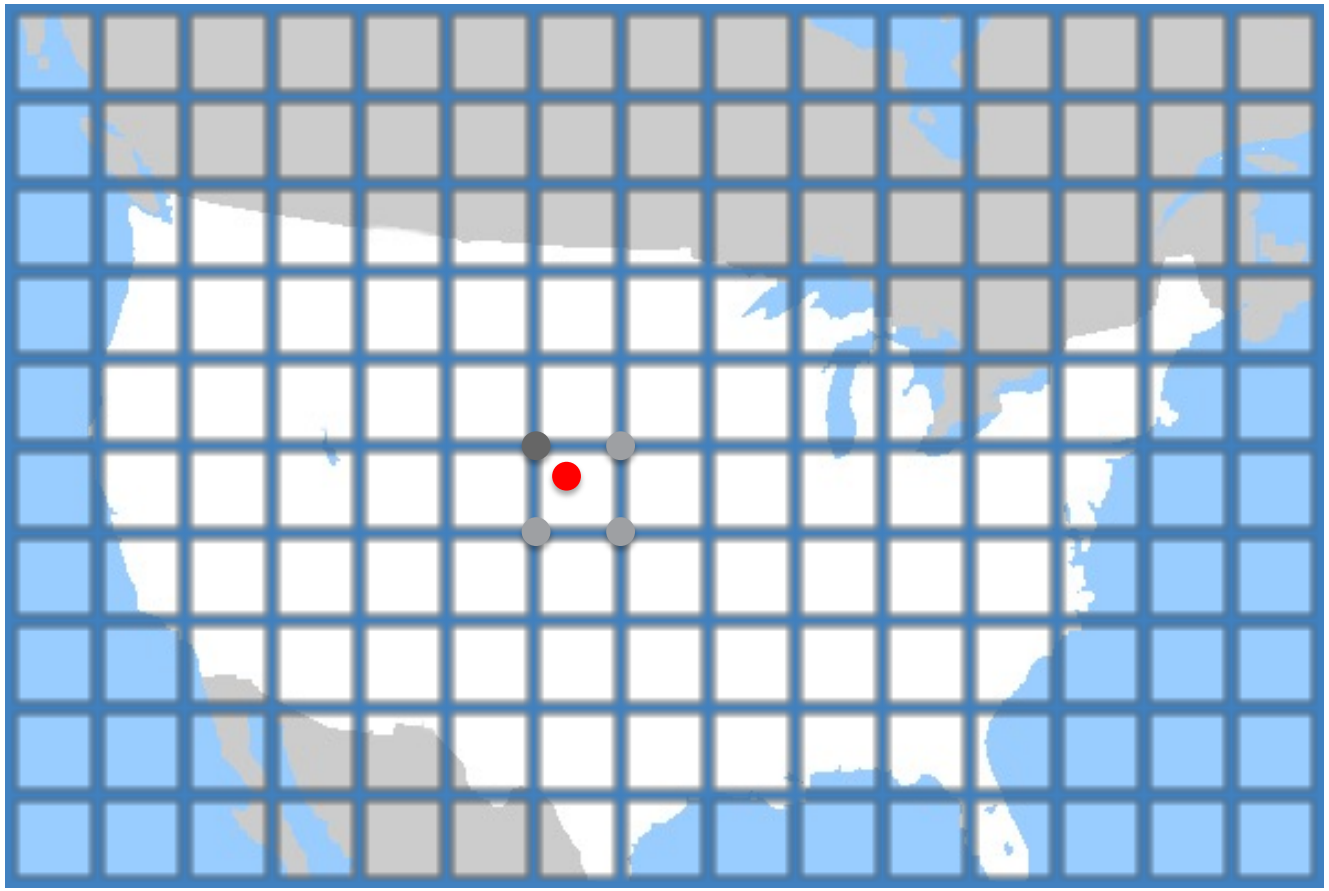
```
array([[[225.81506, 225.79605, 225.73921, ..., 229.932  , 229.8977 ,
         229.87909],
        [225.805  , 225.66464, 225.62424, ..., 229.86417, 229.78374,
         229.8116 ],
        [225.75153, 225.622  , 225.57852, ..., 229.82507, 229.76176,
         229.80647],
        ...,
        [222.08827, 222.10077, 222.09315, ..., 222.354
```

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

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



ll_to_xy



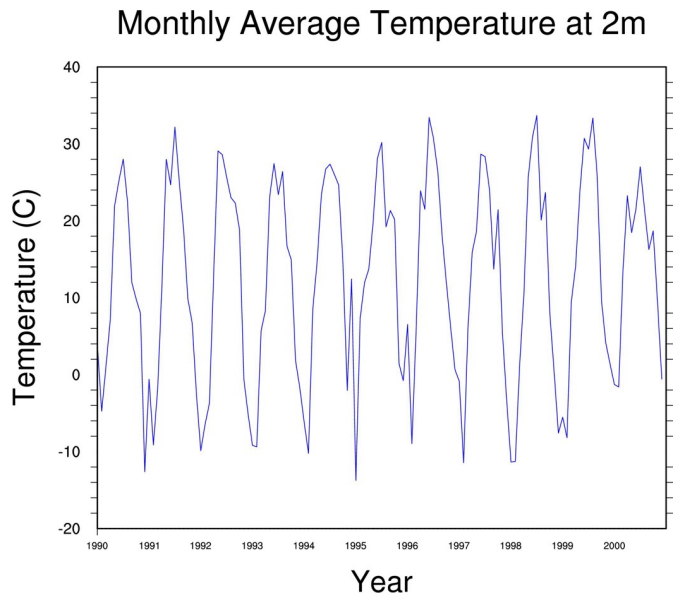
- Your point of interest will likely not coincide with a model grid point
- Most researchers pick the closest point
- Alternatively, pick all points around point of interest and interpolate

```
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

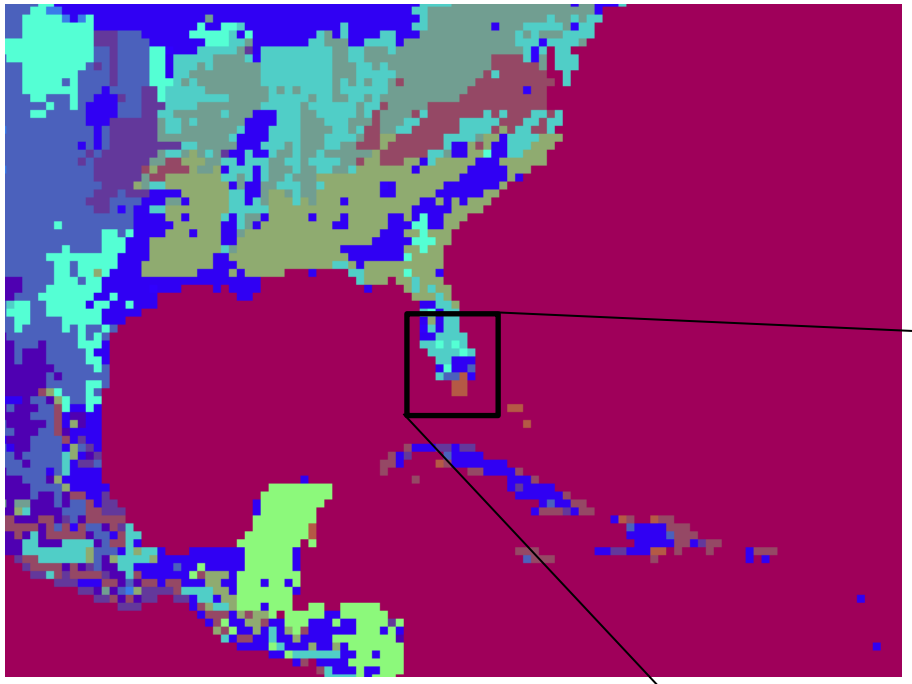
```
[76]: # Open wrfout file with multiple times
      a = xr.open_dataset("/glade/work/jaye/ncl2py/wrfout_all.nc")

[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 :
```

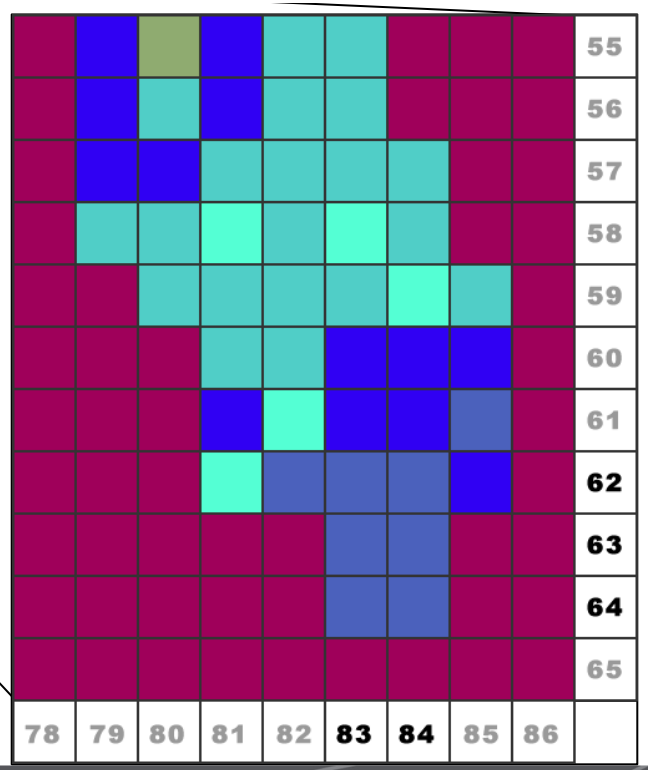
```
[79]: sst_new = sst+1

[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

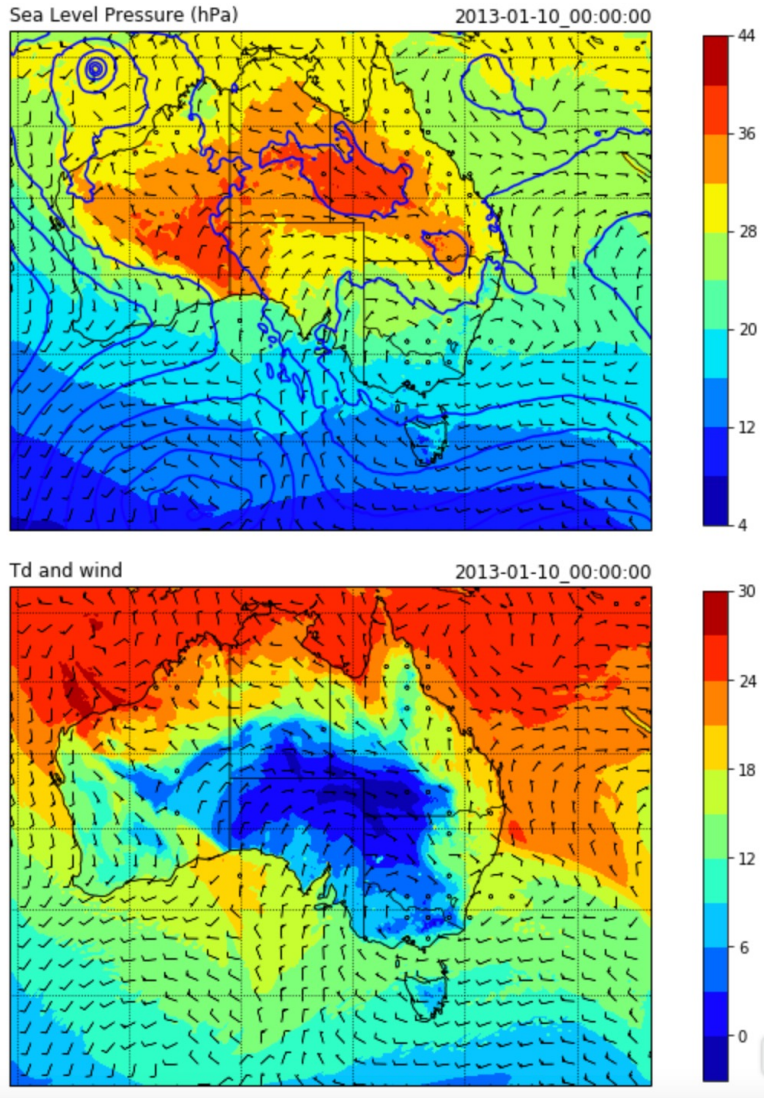
[1]:
```
from netCDF4 import Dataset
import numpy as np
import xarray as xr
import os
import matplotlib.pyplot as plt
from matplotlib.cm import get_cmap
from matplotlib.backends.backend_pdf import PdfPages
import cartopy.crs as crs
from cartopy.feature import NaturalEarthFeature
from wrf import getvar,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

169]:
```
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()
```



Masking with a shapefile

[1]:
```
# imports
from netCDF4 import Dataset
import matplotlib.pyplot as plt
import matplotlib.path as mplPath
import xarray as xr
import numpy as np
import shapefile as shp
import pandas as pd
from wrf import (to_np, getvar, cartopy_ylim, latlon_coords, ALL_TIMES)
```

[2]:
```
def read_shapefile(sf):
    """
    Read a shapefile into a Pandas dataframe with a 'coords'
    column holding the geometry information. This uses the pyshp
    package
    """
    fields = [x[0] for x in sf.fields][1:]
    records = sf.records()
    shps = [s.points for s in sf.shapes()]
    df = pd.DataFrame(columns=fields, data=records)
    df = df.assign(coords=shps)
    return df
```

[3]:
```
# read the file
dloc = '/glade/campaign/mmm/c3we/jaye/CONUS/wrf_del/wrfout_post/WY2001/'
file = dloc+'wrfout_tot_prec_WY2001.nc'
file2 = dloc+'wrfout404_tot_prec_WY2001.nc'
```

[4]:
```
ds = xr.open_dataset(file)
ds2 = xr.open_dataset(file2)
p = ds.PREC_ACC_NC
p = p.isel(west_east=slice(0,493))
p2 = ds2.PREC_ACC_NC
p2 = p2.isel(south_north=slice(334,334+526),west_east=slice(874,874+554))
```

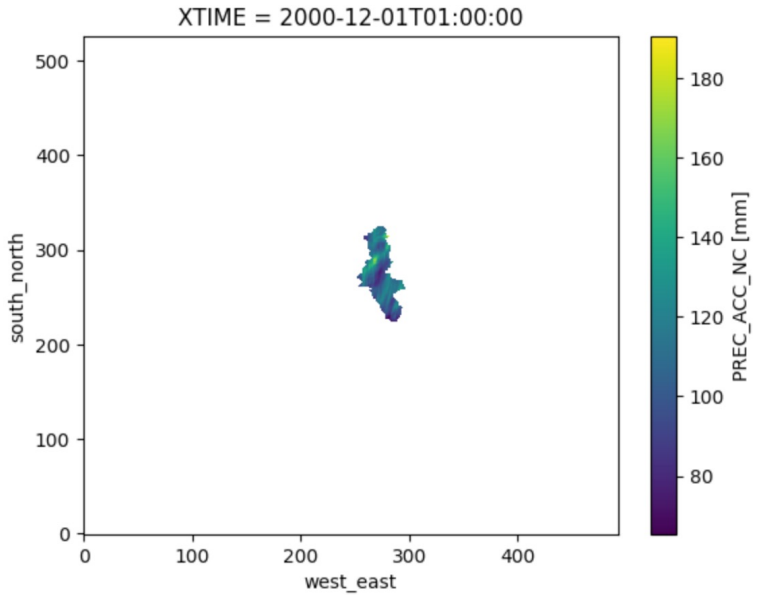
[5]:
```
# Using wrfpython read in file and convert 2d latlon
#   to 1d lat lon for shapefile masking
testds = Dataset(file,'r')
testp = getvar(testds,"PREC_ACC_NC",timeidx=0)
latd, lond = latlon_coords(testp)
```

[6]:
```
# Read the Delaware catchment area
Delaware='/glade/u/home/prein/projects/2020_CONUS404/Shapefiles/Delaware'
rgrGridCells=[(np.array(lond).ravel()[ii],np.array(latd).ravel()[ii]) \
              for ii in range(len(np.array(lond).ravel()))]
rgrSRactP=np.zeros((lond.shape[0]*lond.shape[1]))
sf = shp.Reader(Delaware)
df = read_shapefile(sf)
for sf in range(df.shape[0]):
    ctr = df['coords'][sf]
    if len(ctr) > 10000:
        ctr=np.array(ctr)[::100,:]
    else:
        ctr=np.array(ctr)
    grPRregion=mplPath.Path(ctr)
    TMP=np.array(grPRregion.contains_points(rgrGridCells))
    rgrSRactP[TMP == 1]=1
rgrSRactP=np.reshape(rgrSRactP, (latd.shape[0], latd.shape[1]))
```

[7]:
```
rgrSRactP = rgrSRactP[:,0:493]
```

[8]:
```
p.where(rgrSRactP).isel(Time=2).plot()
```

[8]:
```
<matplotlib.collections.QuadMesh at 0x2b067e450a90>
```



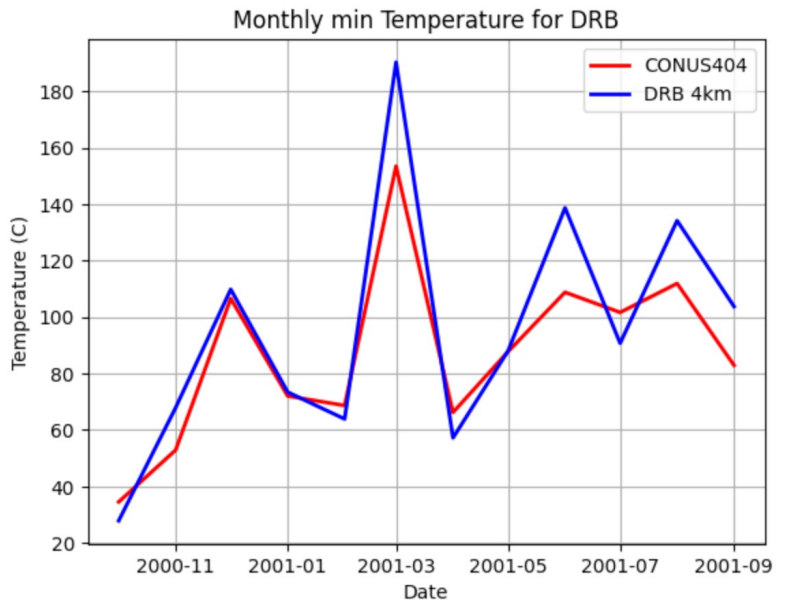
[9]:
```
pavg = p.where(rgrSRactP).mean(("south_north","west_east"))
```

[10]:
```
pavg
```

[10]:
```
xarray.DataArray 'PREC_ACC_NC' (Time: 12)

array([ 27.87684 ,  67.932526, 109.86341 ,  73.44908 ,  63.926727,
       190.33395 ,  57.23651 ,  88.0802  , 138.75372 ,  90.750046,
       134.19954 , 103.78523 ], dtype=float32)

▾ Coordinates:
    XTIME          (Time)  datetime64[ns] ...
```

[13]:
```
plt.plot(p.XTIME,p2avg,marker='',color='red',linewidth=2,label='CONUS404')
plt.plot(p.XTIME,pavg,marker='',color='blue',linewidth=2,label='DRB 4km')
plt.legend()
plt.grid()
plt.xlabel("Date")
plt.ylabel("Temperature (C)")
plt.title("Monthly min Temperature for DRB")
plt.savefig("t2_mon_min_drb.jpg",dpi=300,bbox_inches="tight")
```

[13]:
```
Text(0.5, 1.0, 'Monthly min Temperature for DRB')
```



Masking with a shapefile

Add some observational data to compare to

```
14]: pris = '/glade/work/jaye/prism/prec_prism_monthly_2000.nc'
     pris2 = '/glade/work/jaye/prism/prec_prism_monthly_2001.nc'
     dp = xr.open_dataset(pris)
     dp2 = xr.open_dataset(pris2)
```

```
15]: wypris = xr.concat([dp.isel(time=slice(9,12)),dp2.isel(time=slice(0,9))],\
                        dim='time')
     wypris['lon'] = wypris['lon'] - 360
```

```
16]: wypris = wypris.sel(lat=slice(35,50),lon=slice(-85,-70))
```

Rerun the masking over the prism data

```
21]: pravg = wypris.prec.where(rgrSRactPpr).mean(("lat","lon"))
```

```
22]: pravg
```

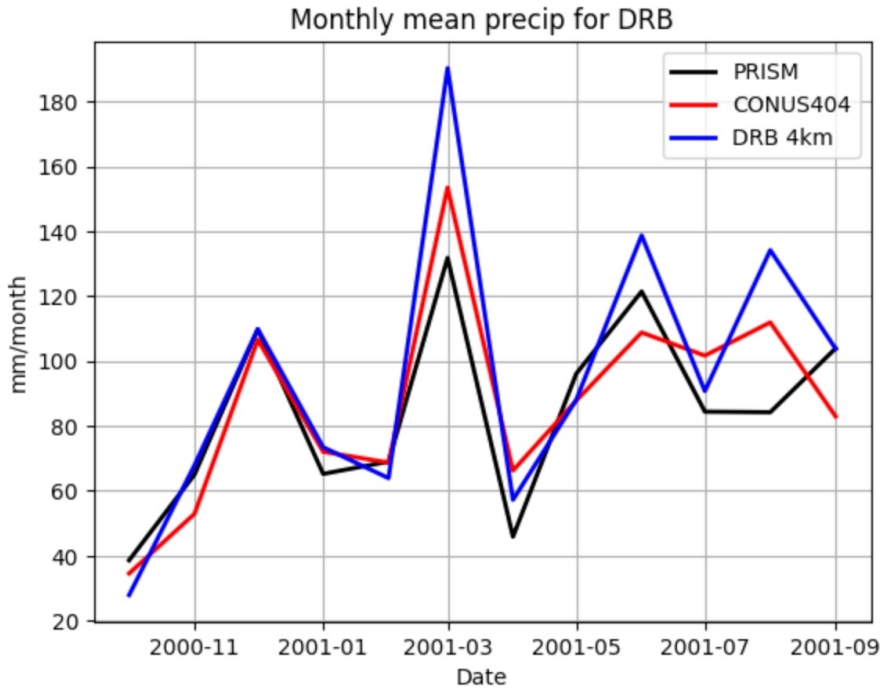
```
22]: xarray.DataArray 'prec' (time: 12)

     array([1.2454468, 2.1681194, 3.541397 , 2.1043456, 2.4647028, 4.252
            1.5292548, 3.1007044, 4.0490026, 2.7230759, 2.717378 , 3.464
           dtype=float32)
     ▾ Coordinates:
       time    (time)  datetime64[ns]  2000-10-16 ... 2001-09-15T12:00:00
     ▸ Attributes: (0)
```

```
23]: monthday = [31,30,31,31,28,31,30,31,30,31,31,30]
```

```
24]: pravg = pravg*monthday
```

```
25]: plt.plot(p.XTIME,pravg,marker='',color='black',linewidth=2,label='PRISM')
     plt.plot(p.XTIME,p2avg,marker='',color='red',linewidth=2,label='CONUS404')
     plt.plot(p.XTIME,pavg,marker='',color='blue',linewidth=2,label='DRB 4km')
     plt.legend()
     plt.grid()
     plt.xlabel("Date")
     plt.ylabel("mm/month")
     plt.title("Monthly mean precip for DRB")
     #plt.savefig("tp_mon_mean_drb.jpg",dpi=300,bbox_inches="tight")
```

```
25]: Text(0.5, 1.0, 'Monthly mean precip for DRB')
```



Plotting with a shapefile

[1]:
```
# imports
from netCDF4 import Dataset
import xarray as xr
xr.set_options(keep_attrs=True)
import matplotlib.pyplot as plt
import cartopy.feature as cfeature
import cartopy.crs as ccrs
from cartopy.feature import NaturalEarthFeature
from cartopy.io.shapereader import Reader
from cartopy.feature import ShapelyFeature
import matplotlib.ticker as mticker
import matplotlib.colors as mcolors
import matplotlib.path as mplPath
import numpy as np
import shapefile as shp
import pandas as pd
import pyproj
from cartopy.mpl.gridliner import LONGITUDE_FORMATTER, LATITUDE_FORMATTER
from mpl_toolkits.axes_grid1 import make_axes_locatable
from wrf import (to_np, getvar, smooth2d, get_cartopy, cartopy_xlim,
                 cartopy_ylim, latlon_coords)
```

[2]:
```
#load constants file so we can have xlat, xlong coordinates :/
filec = '/glade/campaign/ncar/USGS_Water/CONUS404/wrfconstants_d01_1979-10-01_00:00:00.nc4'
const = Dataset(filec,'r')
lm = getvar(const,'LANDMASK',timeidx=0)
lm = lm.isel(south_north=slice(334,334+526),west_east=slice(874,874+554))
constxr = xr.open_dataset(filec)
```

[3]:
```
yr = '2010'
```

[4]:
```
# read the file
file = '/glade/campaign/ncar/USGS_Water/CONUS404/WY'+yr+'/wrf2d_d01_'+yr+'-09-30_00:00:00'
ds = xr.open_dataset(file)
ds = ds.assign_coords(constxr.coords)
ds = ds.assign_attrs(constxr.attrs)
```

[5]:
```
ds
```

[5]: xarray.Dataset

▸ Dimensions: (Time: 1, south_north: 1015, west_east: 1367, soil_layers_stag: 4, snow_layers_stag: 3, snso_layers_stag: 7, west_east_stag: 1368, south_north_stag: 1016)

▾ Coordinates:

XTIME	(Time)	datetime64[ns]	1979-10-01
XLONG	(Time, south_north, west_east)	float32	-122.6 -122.5 ... -57....
XLAT	(Time, south_north, west_east)	float32	17.65 17.66 17.67 ... ...
XLONG_U	(Time, south_north, west_east_stag)	float32	-122.6 -122.6 ... -57....
XLAT_U	(Time, south_north, west_east_stag)	float32	17.64 17.65 17.66 ... ...
XLONG_V	(Time, south_north_stag, west_east)	float32	-122.6 -122.5 ... -57....
XLAT_V	(Time, south_north_stag, west_east)	float32	17.63 17.64 17.65 ... ...

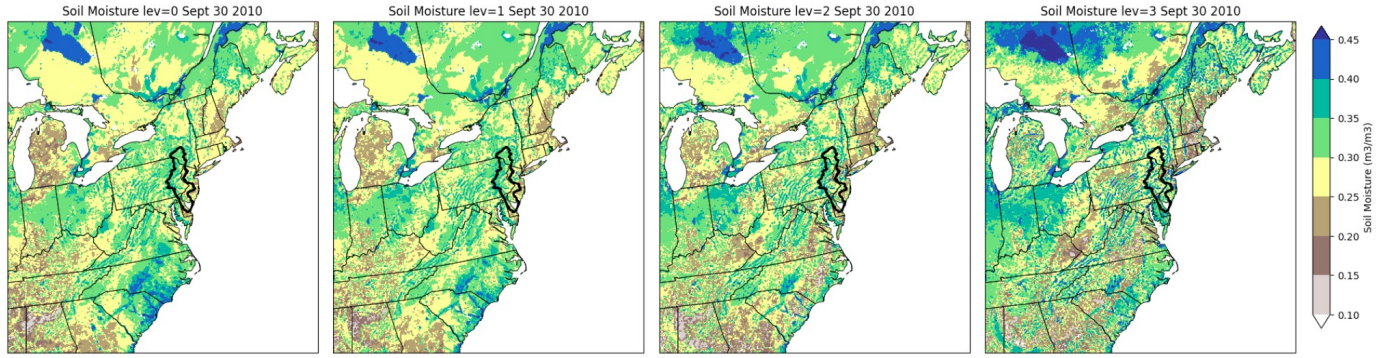
▸ Data variables: (191)

▸ Attributes: (150)

[6]:
```
sm = ds.SMOIS
sm = sm.isel(Time=0,south_north=slice(334,334+526),west_east=slice(874,874+554))
sm = sm.where(lm==1)
```

[7]:
```
# get the lat, lon grid
lats, lons = latlon_coords(lm)
cart_proj = get_cartopy(lm)
```

[9]:
```
fig, axs = plt.subplots(ncols=4,nrows=1,figsize=(24,8),sharex=True,sharey=True,subplot_kw=dict(projection=cart_proj))
fig.subplots_adjust(hspace=.05,wspace=.05)
axs = axs.ravel()

clevs = [0.1,0.15,0.2,0.25,0.3,0.35,0.4,0.45]

Delaware='/glade/u/home/prein/projects/2020_CONUS404/Shapefiles/Delaware'
shape_feature = ShapelyFeature(Reader(Delaware+'.shp').geometries(),
                               ccrs.PlateCarree(), edgecolor='black')

for ii in range(0,4):
    cont=axs[ii].contourf(to_np(lons),to_np(lats),to_np(sm.isel(soil_layers_stag=ii)),clevs,
                          transform=ccrs.PlateCarree(),cmap='terrain_r',extend='both')
    axs[ii].add_feature(shape_feature, facecolor='none',edgecolor='black',linewidth=2)
    axs[ii].add_feature(cfeature.STATES,linewidth=.6,edgecolor='black')
    axs[ii].set_xlim(cartopy_xlim(lm)); axs[ii].set_ylim(cartopy_ylim(lm))
    axs[ii].set_title('Soil Moisture lev='+str(ii)+' Sept 30 ' +yr)

fig.subplots_adjust(right=0.87)
cbar_ax = fig.add_axes([0.88, 0.25, 0.01, 0.525]) #[left, bottom, width, height]
fig.colorbar(cont,cax=cbar_ax,label='Soil Moisture (m3/m3)')

#plt.savefig('CONUS404_SMOIS_sept30'+yr+'.jpg',dpi=300,bbox_inches='tight')
plt.show()
```



GeoCAT

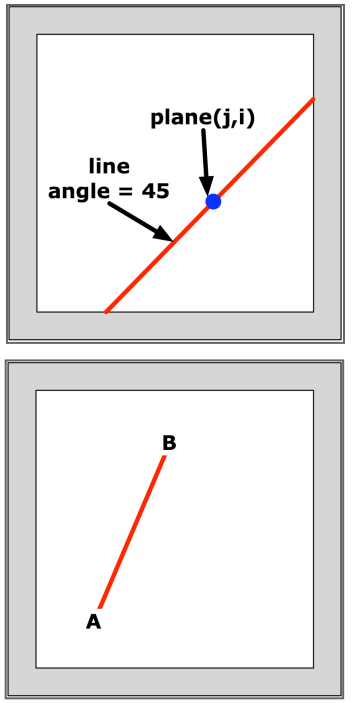
- GeoCAT is the Geoscience Community Analysis Toolkit
- A collection of Python tools related to NCL developed at NCAR
- Examples page:

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

  https://geocat.ucar.edu/blog/

vertcross

- Returns the vertical cross-section of a 3d field
- Cross section is defined by a horizontal line through the domain
  - You can either have a pivot point and angle
  - Or a start point and end point
  - Pivot point, start point and end point can be input as i,j grid cell number or as a lat/lon point

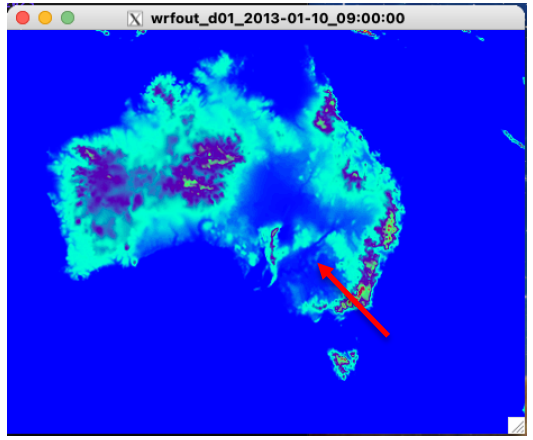


GeoCAT and vertcross

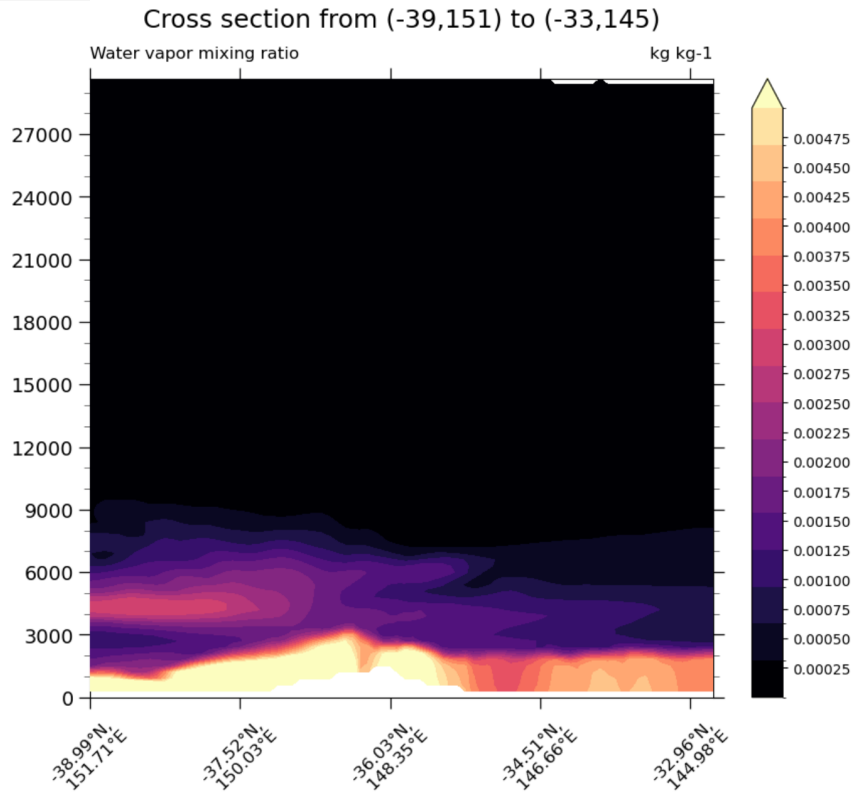
[1]:
```
from netCDF4 import Dataset
import numpy as np
import matplotlib.pyplot as plt
import xarray as xr
import os

from wrf import (to_np, getvar, CoordPair, vertcross, latlon_coords)
import geocat.viz as gv
```

[2]:
```
# Specify the necessary variables needed from the data set in order to
#   use 'z' and 'QVAPOR'
toinclude = ['PH', 'P', 'HGT', 'PHB', 'QVAPOR']

# Read in the data and extract variables
wrfin = Dataset('/glade/work/jaye/ncl2py/wrfout_d01_2013-01-10_07:00:00')

z = getvar(wrfin, "z")
qv = getvar(wrfin, "QVAPOR")
# Pull lat/lon coords from QVAPOR data using wrf-python tools
lats, lons = latlon_coords(qv)
```

[3]:
```
# Define start and stop coordinates for cross section
start_point = CoordPair(lat=-38.98, lon=151.71)
end_point = CoordPair(lat=-32.72, lon=144.73)

qv_cross = vertcross(qv,z,wrfin=wrfin,start_point=start_point,\
                     end_point=end_point,latlon=True)
```



[4]:
```
fig = plt.figure(figsize=(10, 8))
ax = plt.axes()

# Set the x-ticks to use latitude and longitude labels.
coord_pairs = to_np(qv_cross.coords["xy_loc"])
x_ticks = np.arange(coord_pairs.shape[0])

# Plot filled contours
qv_contours = qv_cross.plot.contourf(ax=ax,levels=17,cmap='magma',vmin=0,\
                                     vmax=0.005,zorder=4,add_labels=False,\
                                     add_colorbar=False,yticks=np.arange(0, 30000, 3000),\
                                     xticks=x_ticks[::20])
# Add colorbar
plt.colorbar(qv_contours, ax=ax, ticks=np.arange(0.00025, 0.005, .00025))

# Add minor ticks to the yaxis
gv.add_major_minor_ticks(ax=ax,x_minor_per_major=1,y_minor_per_major=3,labelsize=14)

# Format the xtick labels
x_labels = [
    pair.latlon_str(fmt="{:.2f}\N{DEGREE SIGN}N, \n {:.2f}\N{DEGREE SIGN}E")
    for pair in to_np(coord_pairs)
]
ax.set_xticklabels(x_labels[::20], rotation=45, fontsize=12)

# Set the plot titles
plt.title("Cross section from (-39,151) to (-33,145)", fontsize=18, y=1.07)
plt.title('Water vapor mixing ratio', loc='left', y=1.02)
plt.title('kg kg-1', loc='right', y=1.02)

plt.show()
```



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

MetPy and GeoCAT (skewT)

[1]:
```
from netCDF4 import Dataset
import matplotlib.pyplot as plt
import matplotlib.lines as mlines
import numpy as np
from metpy.plots import SkewT
from metpy.units import units
import metpy.calc as mpcalc

from wrf import (to_np, getvar, ll_to_xy)
import geocat.viz as gv
```

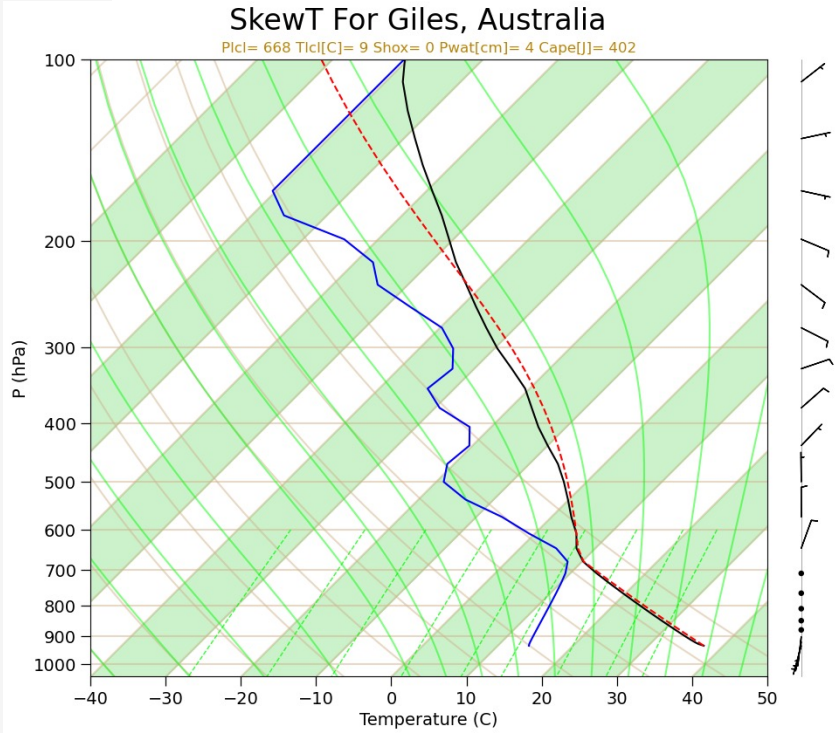
[2]:
```
# Read in data
wrfin = Dataset('/glade/work/jaye/ncl2py/wrfout_d01_2013-01-13_04:00:00')
```

[3]:
```
# Find the xy location of Giles, Australia
locij = ll_to_xy(wrfin,-25.033,128.3,True)
x = locij[0]; y = locij[1]
```

[4]:
```
# Read in necessary variables
tc = getvar(wrfin,'tc')[:,y,x]
td = getvar(wrfin,'td')[:,y,x]
p = getvar(wrfin,'pressure')[:,y,x]
uvm = getvar(wrfin,'uvmet')[:,:,y,x]
u = uvm[0,:].metpy.convert_units('m/s')
v = uvm[1,:].metpy.convert_units('m/s')
```

[5]:
```
tc0 = tc[0]  # Temperature of surface parcel
td0 = td[0]  # Dew point temperature of surface parcel
pro = mpcalc.parcel_profile(p, tc0, td0)  # Temperature profile of parcel
subtitle = gv.get_skewt_vars(p, tc, td, pro)  # Create subtitle
```

[6]:
```
fig = plt.figure(figsize=(10, 12))

# The rotation keyword changes how skewed the temperature lines are. MetPy has
# a default skew of 30 degrees
skew = SkewT(fig, rotation=45)
ax = skew.ax

# Plot temperature and dew point
skew.plot(p, tc, color='black')
skew.plot(p, td, color='blue')

# Draw parcel path
parcel_prof = mpcalc.parcel_profile(p, tc[0], td[0]).metpy.convert_units('degC')
skew.plot(p, parcel_prof, color='red', linestyle='--')
p = np.where(p.values >= 100, p.values, np.nan)  # unitless for plot_barbs

# Add wind barbs
skew.plot_barbs(pressure=p[::2],u=u[::2],v=v[::2],xloc=1.05,fill_empty=True,\
                sizes=dict(emptybarb=0.075, width=0.1, height=0.2))

# Draw line underneath wind barbs
line = mlines.Line2D([1.05,1.05],[0, 1],color='gray',linewidth=0.5,\
                     transform=ax.transAxes,clip_on=False,zorder=1)
ax.add_line(line)

# Shade every other section between isotherms
x1 = np.linspace(-100, 40, 8)  # The starting x values for the shaded regions
x2 = np.linspace(-90, 50, 8)  # The ending x values for the shaded regions
y = [1050, 100]  # The range of y values that the shades regions should cover
for i in range(0, 8):
    skew.shade_area(y=y,x1=x1[i],x2=x2[i],color='limegreen',alpha=0.25,zorder=1)

skew.plot_dry_adiabats(linestyles='solid',colors='tan',linewidths=1.5)
skew.plot_moist_adiabats(linestyles='solid',colors='lime',linewidth=1.5)
skew.plot_mixing_lines(linestyle='dashed',colors='lime',linewidths=1)

# Use geocat.viz utility function change ticks, add labels, etc
gv.add_major_minor_ticks(ax=ax,x_minor_per_major=1,y_minor_per_major=1,labelsize=14)
ax.tick_params('both', which='both', top=False, right=False)
gv.set_titles_and_labels(ax=ax,xlabel='Temperature (C)',ylabel='P (hPa)',labelfontsize=14)

# Manually add suptitle and subtitle for appropriate positioning
fig.suptitle('SkewT For Giles, Australia', fontsize=24, y=0.84)
ax.set_title(subtitle, color='darkgoldenrod')

# Change the style of the gridlines
plt.grid(True,which='major',axis='both',color='tan',linewidth=1.5,alpha=0.5)

plt.show()
```



Data Manipulation and Testing

[1]:
```
# imports
from netCDF4 import Dataset
import xarray as xr
xr.set_options(keep_attrs=True)
import matplotlib.pyplot as plt
import cartopy.feature as cfeature
import cartopy.crs as ccrs
from cartopy.feature import NaturalEarthFeature
from cartopy.io.shapereader import Reader
from cartopy.feature import ShapelyFeature
import matplotlib.ticker as mticker
import matplotlib.colors as mcolors
import matplotlib.path as mplPath
import numpy as np
import shapefile as shp
import pandas as pd
import pyproj
from cartopy.mpl.gridliner import LONGITUDE_FORMATTER, LATITUDE_FORMATTER
from mpl_toolkits.axes_grid1 import make_axes_locatable
import wrf
from wrf import (to_np, getvar, smooth2d, get_cartopy, cartopy_xlim,
                 cartopy_ylim, latlon_coords, xy_to_ll)
```

[2]:
```
day = ['0701','0702','0703','0704','0705','0706','0707','0708','0709',\
       '0710','0711','0712','0713','0714','0715']
```

[3]:
```
# read in files from five wrf runs to compare
path = '/glade/campaign/ncar/USGS_Water/drb/tests/'
file = path+'wrf_drb60s_nourb/wrfout_all_nourb.nc'
ds = xr.open_dataset(file)
file2 = path+'wrf_drb60s_urbiri0/wrfout_all_urb0.nc'
ds2 = xr.open_dataset(file2)
file3 = path+'wrf_drb60s_urbiri1/wrfout_all_urb1.nc'
ds3 = xr.open_dataset(file3)
file4 = path+'wrf_drb60s_urbiri0_frcurb/wrfout_all_urb0_frc.nc'
ds4 = xr.open_dataset(file4)
file5 = path+'wrf_drb60s_urbiri0_frcurb_noahmpgeo/wrfout_all_urb0_frc_noahmpgeo.nc'
ds5 = xr.open_dataset(file5)

# Assign time attribute using getvar (I just prefer this method)
a = Dataset(file,"r"); times = getvar(a,"times",timeidx=wrf.ALL_TIMES)
ds = ds.assign_coords(Time=times); ds2 = ds2.assign_coords(Time=times)
ds3 = ds3.assign_coords(Time=times); ds4 = ds4.assign_coords(Time=times)
ds5 = ds5.assign_coords(Time=times)
lm = getvar(a,'LANDMASK',timeidx=0)
```

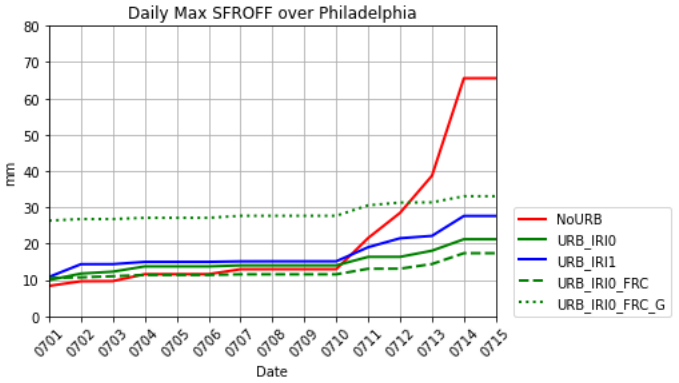
[4]:
```
# Read in surface runoff
sr = ds.SFROFF; sr2 = ds2.SFROFF; sr3 = ds3.SFROFF
sr4 = ds4.SFROFF; sr5 = ds5.SFROFF
```

[5]:
```
# Resample data from hourly to daily maximum
srm = sr.resample(Time='1D').max(); sr2m = sr2.resample(Time='1D').max()
sr3m = sr3.resample(Time='1D').max(); sr4m = sr4.resample(Time='1D').max()
sr5m = sr5.resample(Time='1D').max()
```

[6]:
```
# Drawing a box around central Philadephia
ns1 = 256; ns2 = 261
we1 = 279; we2 = 284
```

[7]:
```
# Take slices of the box and make a spatial average, then plot
srmts = srm.isel(south_north=slice(ns1,ns2),west_east=slice(we1,we2)).mean(dim=["south_north","west_east"])
sr2mts = sr2m.isel(south_north=slice(ns1,ns2),west_east=slice(we1,we2)).mean(dim=["south_north","west_east"])
sr3mts = sr3m.isel(south_north=slice(ns1,ns2),west_east=slice(we1,we2)).mean(dim=["south_north","west_east"])
sr4mts = sr4m.isel(south_north=slice(ns1,ns2),west_east=slice(we1,we2)).mean(dim=["south_north","west_east"])
sr5mts = sr5m.isel(south_north=slice(ns1,ns2),west_east=slice(we1,we2)).mean(dim=["south_north","west_east"])

plt.plot(day,srmts,marker='',color='red',linewidth=2,label='NoURB')
plt.plot(day,sr2mts,marker='',color='green',linewidth=2,label='URB_IRI0')
plt.plot(day,sr3mts,marker='',color='blue',linewidth=2,label='URB_IRI1')
plt.plot(day,sr4mts,marker='',color='green',linestyle='dashed',linewidth=2,label='URB_IRI0_FRC')
plt.plot(day,sr5mts,marker='',color='green',linestyle='dotted',linewidth=2,label='URB_IRI0_FRC_G')

plt.ylim(ymax=80,ymin=0); plt.xlim(xmin="0701",xmax="0715")
plt.legend(loc=(1.04,0))
plt.grid()
plt.xlabel("Date"); plt.ylabel("mm")
plt.xticks(rotation = 45)
locs, labels = plt.xticks()
plt.title("Daily Max SFROFF over Philadelphia")
#plt.savefig("SFROFFMAX_ts_phil_"+point+"_frcg.jpg",dpi=300,bbox_inches="tight")
```

[7]: Text(0.5, 1.0, 'Daily Max SFROFF over Philadelphia')



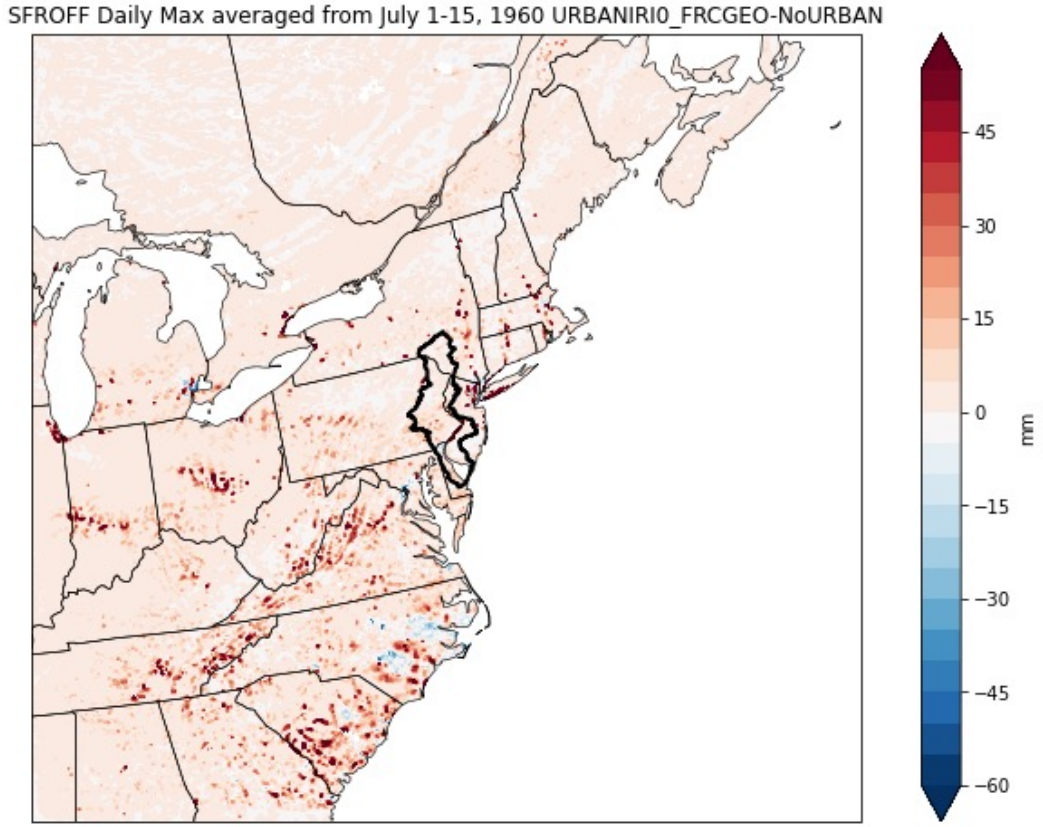
Data Manipulation and Testing

[8]:
```
# Average over entire 2 week period
srm = srm.mean("Time")
sr2m = sr2m.mean("Time")
sr3m = sr3m.mean("Time")
sr4m = sr4m.mean("Time")
sr5m = sr5m.mean("Time")
```

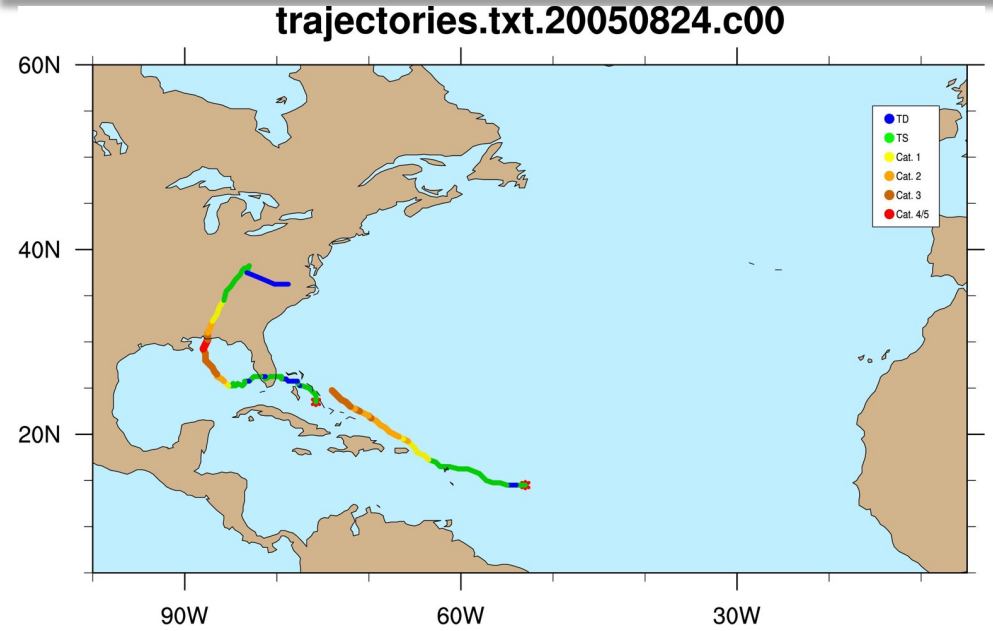
[9]:
```
srdiff_frc = sr5m-srm
```

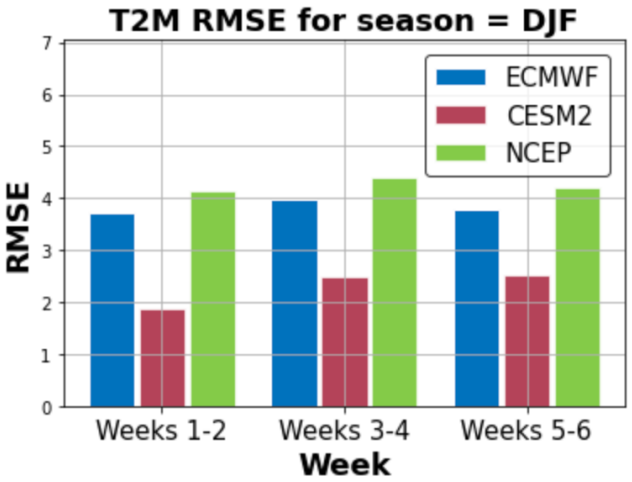
10]:
```
# get the lat, lon grid
lats, lons = latlon_coords(lm)
cart_proj = get_cartopy(lm)
```

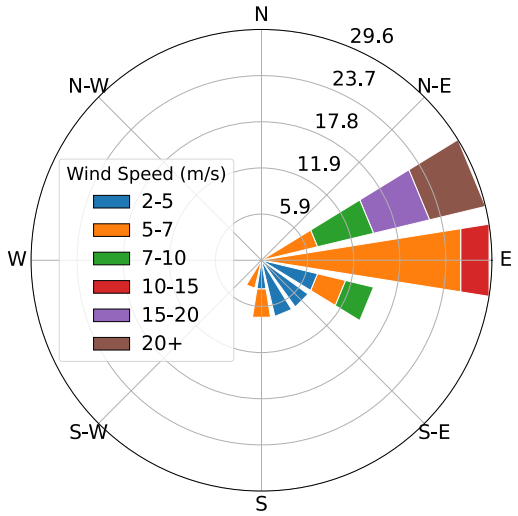
11]:
```
fig = plt.figure(figsize=(12,8))
ax = plt.axes(projection=cart_proj)
clevs = np.arange(-60,60,5).tolist()

Delaware='/glade/u/home/prein/projects/2020_CONUS404/Shapefiles/Delaware'
shape_feature = ShapelyFeature(Reader(Delaware+'.shp').geometries(),
                               ccrs.PlateCarree(), edgecolor='black')

cont = ax.contourf(to_np(lons),to_np(lats),to_np(srdiff_frc.where(lm==1)),clevs,
                   transform=ccrs.PlateCarree(),cmap='RdBu_r',extend='both')
ax.add_feature(shape_feature, facecolor='none',edgecolor='black',linewidth=2)
ax.add_feature(cfeature.STATES,linewidth=.6,edgecolor='black')
ax.set_xlim(cartopy_xlim(lm)); ax.set_ylim(cartopy_ylim(lm))
ax.set_title('SFROFF Daily Max averaged from July 1-15, 1960 URBANIRI0_FRCGEO-NoURBAN')
fig.colorbar(cont,label='mm')

#plt.savefig('DRB_URBIRI0FRCGDIFF_SFROFFMAX_196007.jpg',dpi=300,bbox_inches='tight')
plt.show()
```

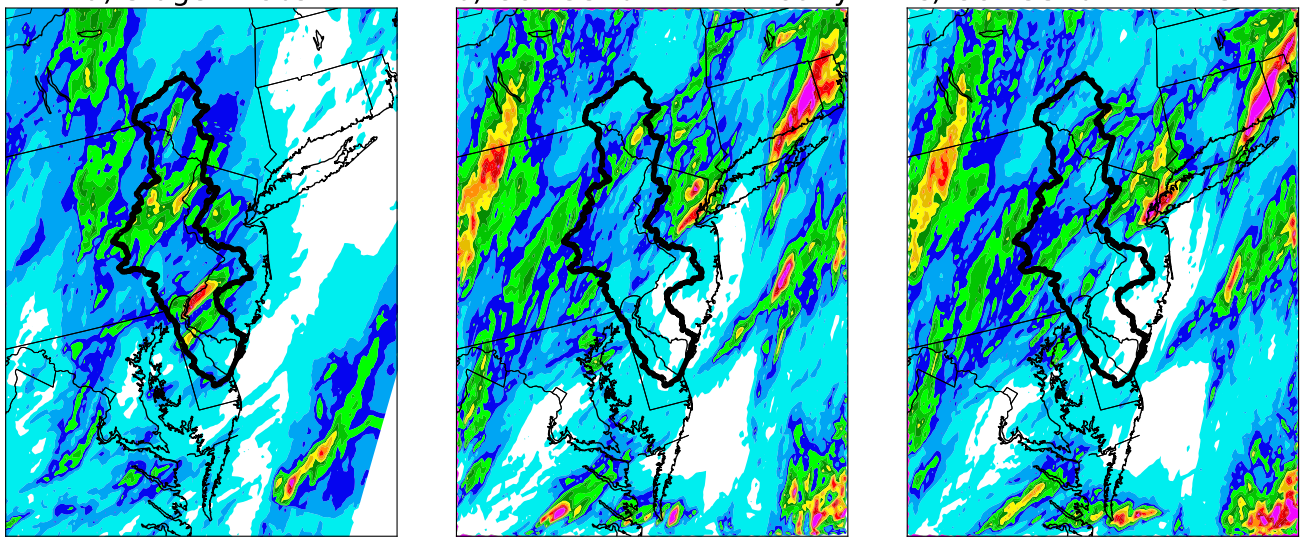


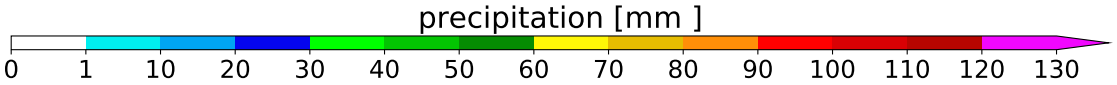
More examples



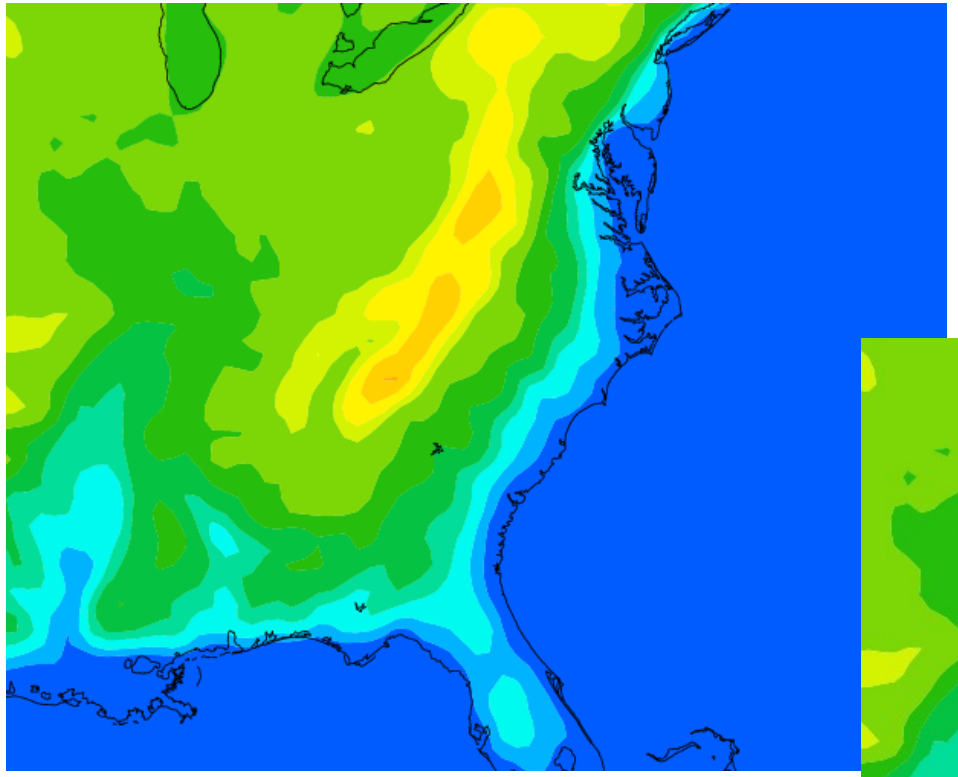


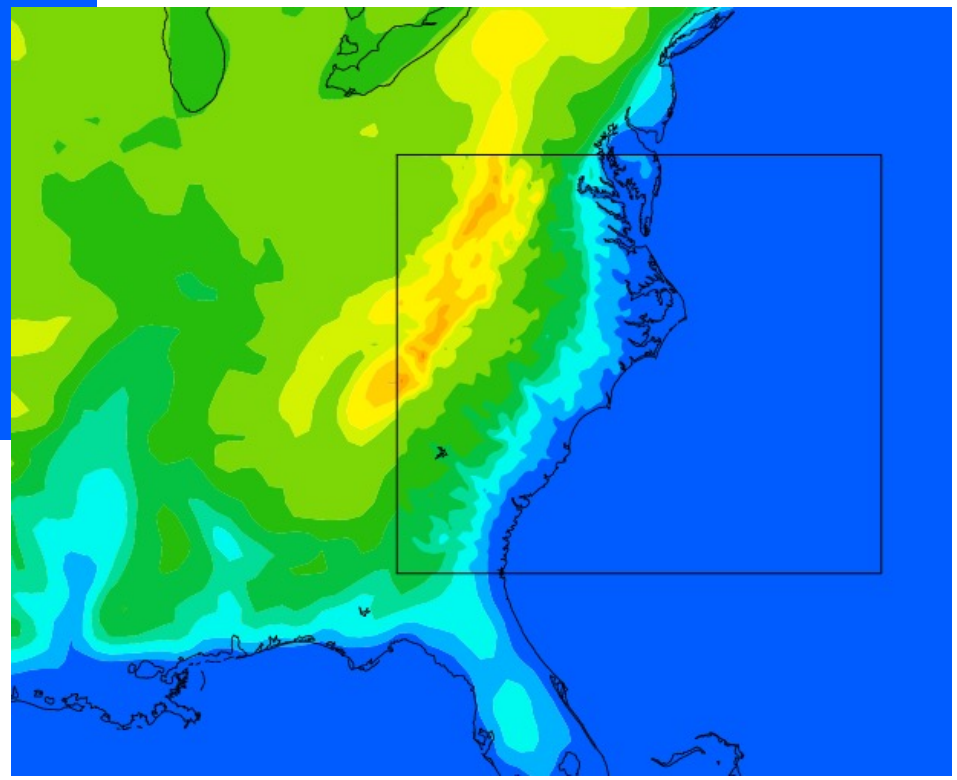






Overlay Domains





Thank you!

Questions??