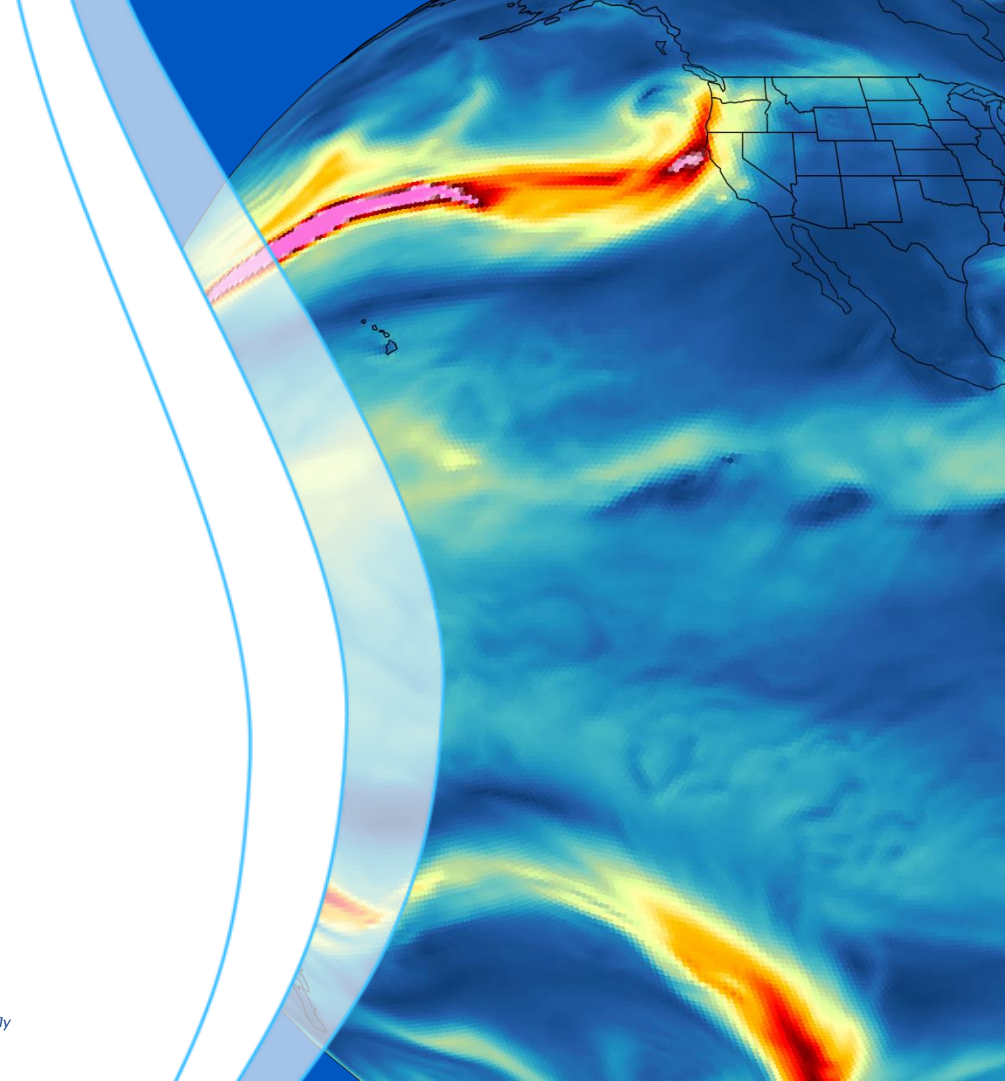
August 11, 2026

Post-processing and visualizing MPAS-Atmosphere output

Abby Jaye
Scientist V – NSF-NCAR/MMM

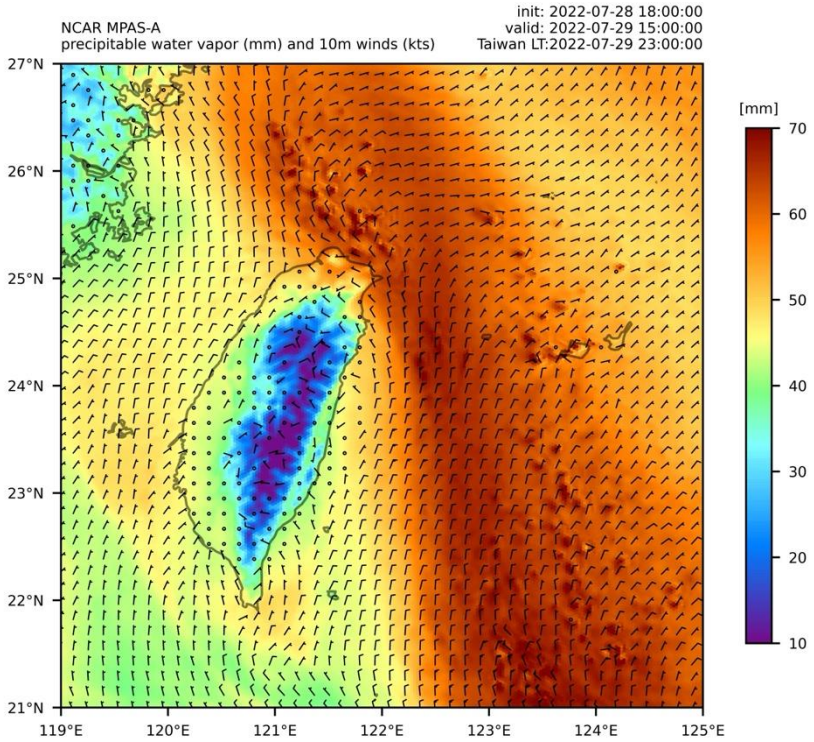


This material is based upon work supported by the NSF National Center for Atmospheric Research, a major facility sponsored by the U.S. National Science Foundation and managed by the University Corporation for Atmospheric Research. Any opinions, findings and conclusions or recommendations expressed in this material do not necessarily reflect the views of NSF.

MPAS Native Grid is Great!

Motivation

- I will go over a variety of tools to help analyze and plot data from MPAS-A on the native grid
  - Command line tools for 'quick looks'
  - Simple plotting with **xarray**, **uxarray** and **matplotlib**
  - Make much nicer plots of MPAS-A output variables and adding wind barbs
  - Vertical cross-sections
  - Calculating basic averages, max, min, etc with **xarray**
- I will cover basic regridding tools if it is absolutely necessary to regrid
  - Command line with **CDO**

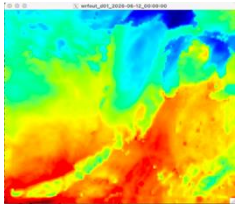


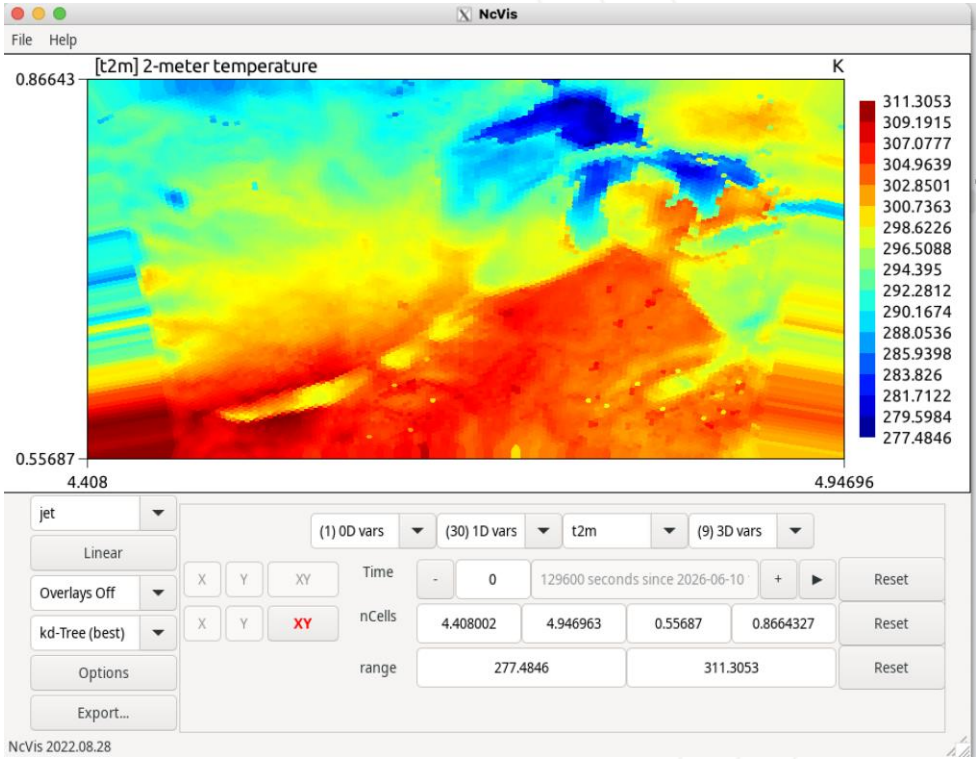
Tools for MPAS

How do I deal with the weird “grid cells”?

- How do I look at it? It doesn’t work with ncview!
- **ncvis** is here to save the day! Works with unstructured meshes and easy to use
  - module load ncvis
- If you are plotting from *static.nc, *grid.nc, *init.nc or a history*nc file all the grid information is already there
- If you are plotting from a diag*nc file you need to include a file that has all the grid information:

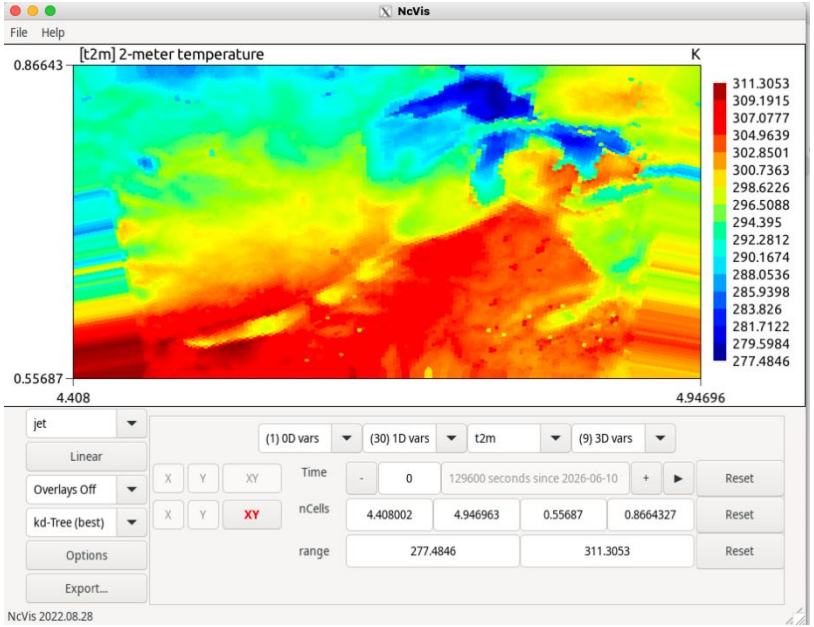
```
ncvis diag.2000-01-01_00.nc domain.grid.nc
```





Tools for MPAS

- For analysis and plotting in python **uxarray** is a great tool!




```
[1]: import uxarray as ux

[2]: mpas_20_files = "/glade/derecho/scratch/jaye/valpo/mpas_runs/20260610/rundir_20km/diag*"
     grid_20 = "/glade/derecho/scratch/jaye/valpo/mpas_runs/20260610/rundir_20km/x1.11981.grid.valpo.nc"
     uxds_20 = ux.open_mfdataset(grid_20,mpas_20_files,combine="by_coords")

[3]: uxds_20

[3]: uxarray.UxDataset
```

▸ Dimensions: (**Time**: 25, n_face: 11981, nVertLevels: 55, n_node: 24390, nSoilLevels: 4, nVertLevelsP1: 56)

▾ Coordinates:

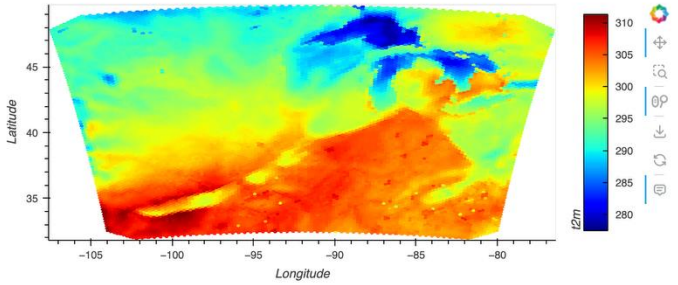
Time	(Time)	datetime64[ns]	2026-06-10T12:00:00 ... 2...

▾ Data variables:

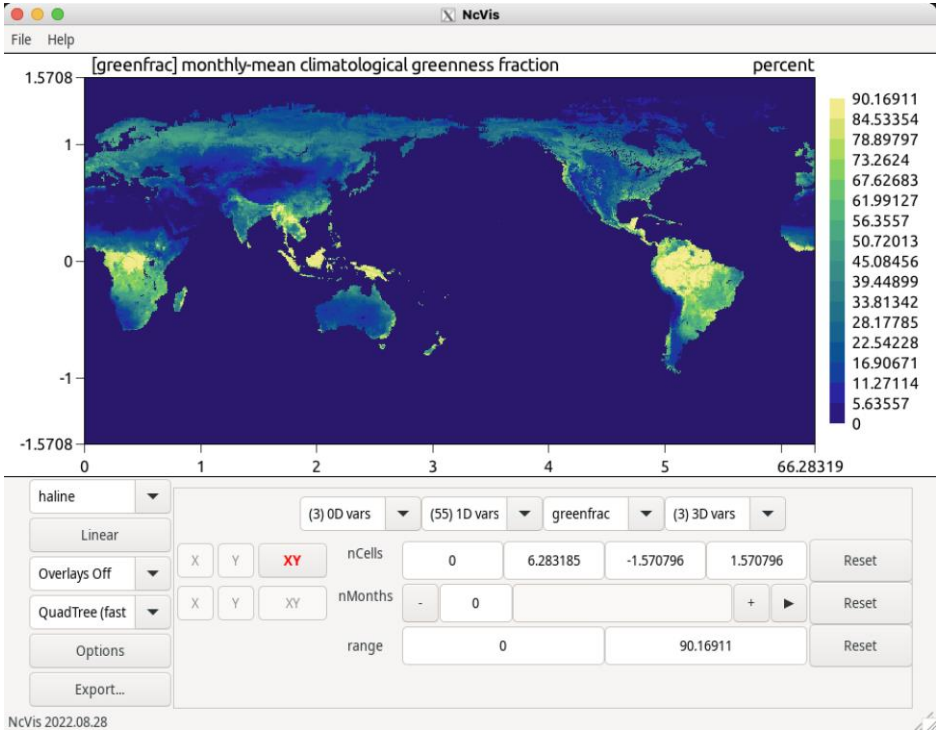
initial_time	(Time)	|S64	b'2026-06-10_12:00:00 ...
xtime	(Time)	|S64	dask.array<chunksize=(1,), ...
olrtoa	(Time, n_face)	float32	dask.array<chunksize=(1, 1...
rainc	(Time, n_face)	float32	dask.array<chunksize=(1, 1...
rainnc	(Time, n_face)	float32	dask.array<chunksize=(1, 1...
refl10cm	(Time, n_face, nVertLevels)	float32	dask.array<chunksize=(1, 1...

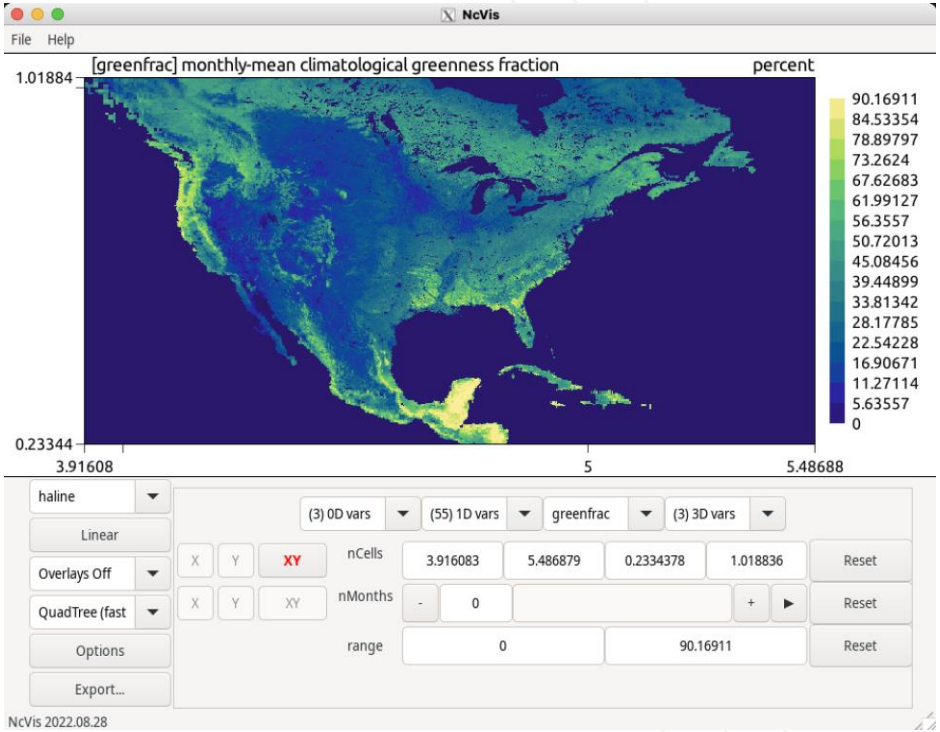
```
[8]: uxds_20["t2m"].sel(Time="2026-06-12 00:00:00").plot(cmap="jet")
```

[8]:



From Global to Regional

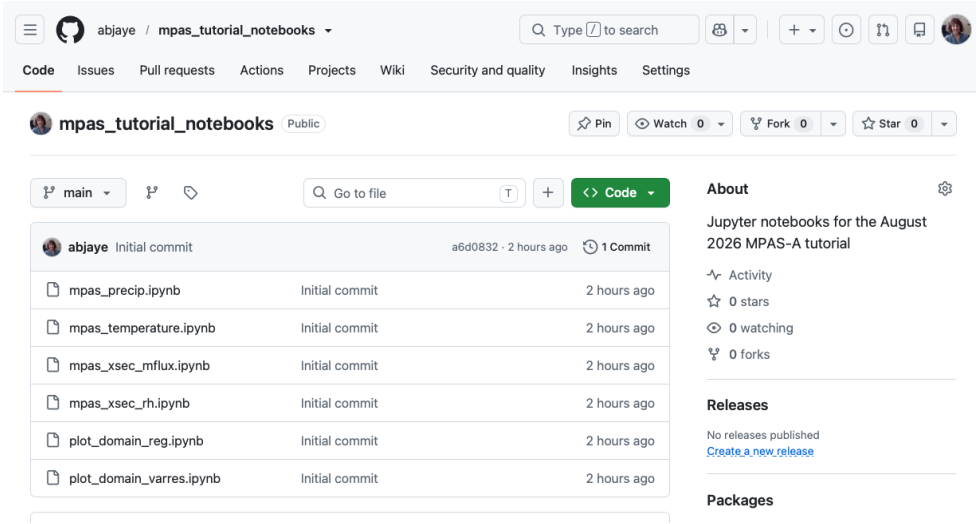




Example python notebooks

- First, grab this github repository with example notebooks
- In your mpas_tutorial directory:

git clone https://github.com/abjaye/mpas_tutorial_notebooks.git

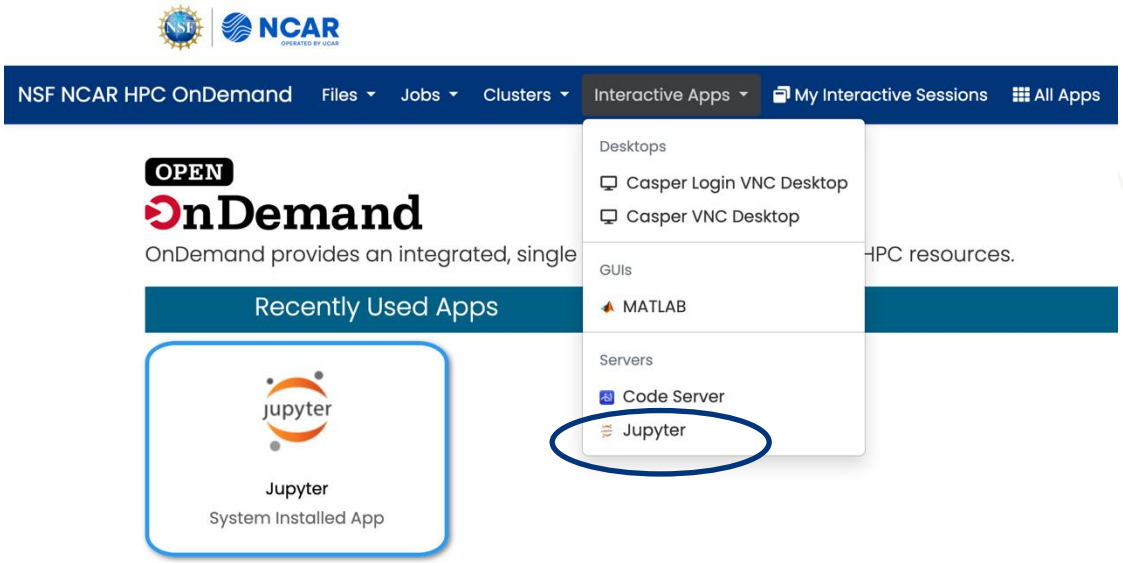


NCAR JupyterHub

- I'll step through some screenshots showing a jupyterhub login

ondemand.hpc.ucar.edu

Click on 'Jupyter'

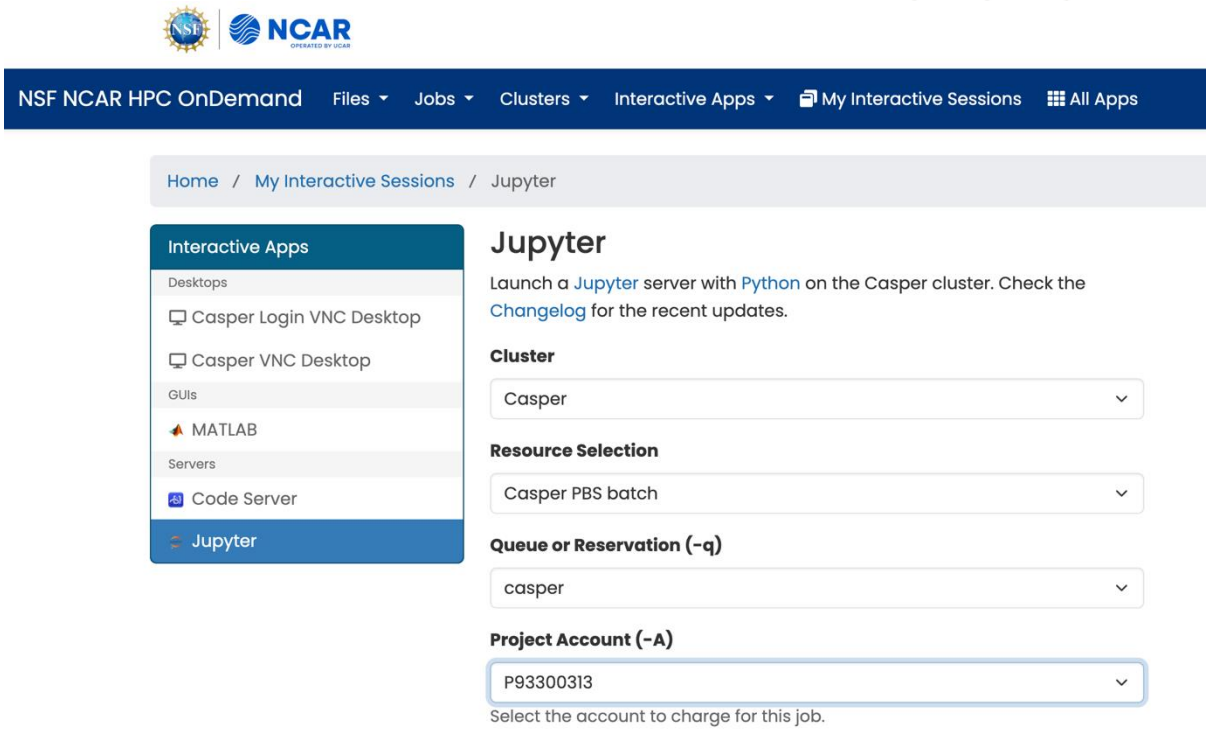


NCAR JupyterHub

- I'll step through some screenshots showing a jupyterhub login

ondemand.hpc.ucar.edu

- Select project account
- Further down change your wall time(default is 1hr)
- Click launch

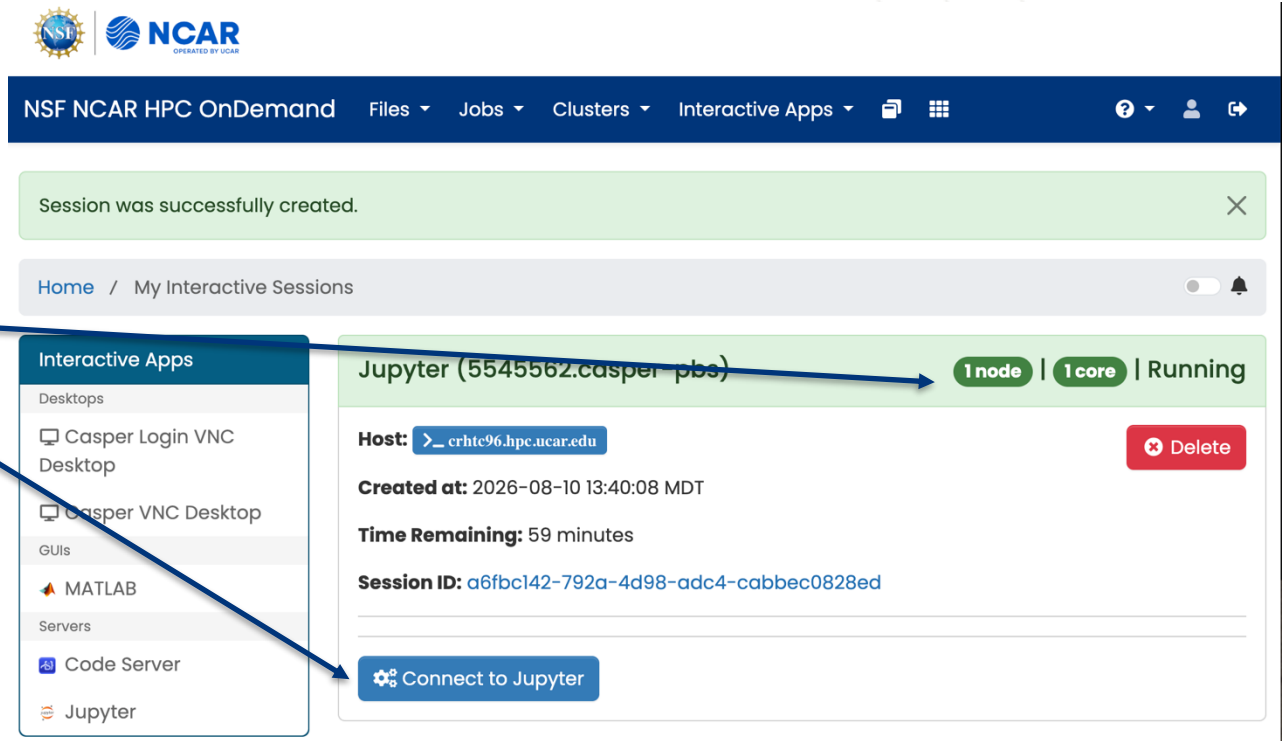


NCAR JupyterHub

- I'll step through some screenshots showing a jupyterhub login

ondemand.hpc.ucar.edu

- It might be in the queue for a bit
- When it says running and is green click 'Connect to Jupyter'
- It will take you to a jupyter launch page

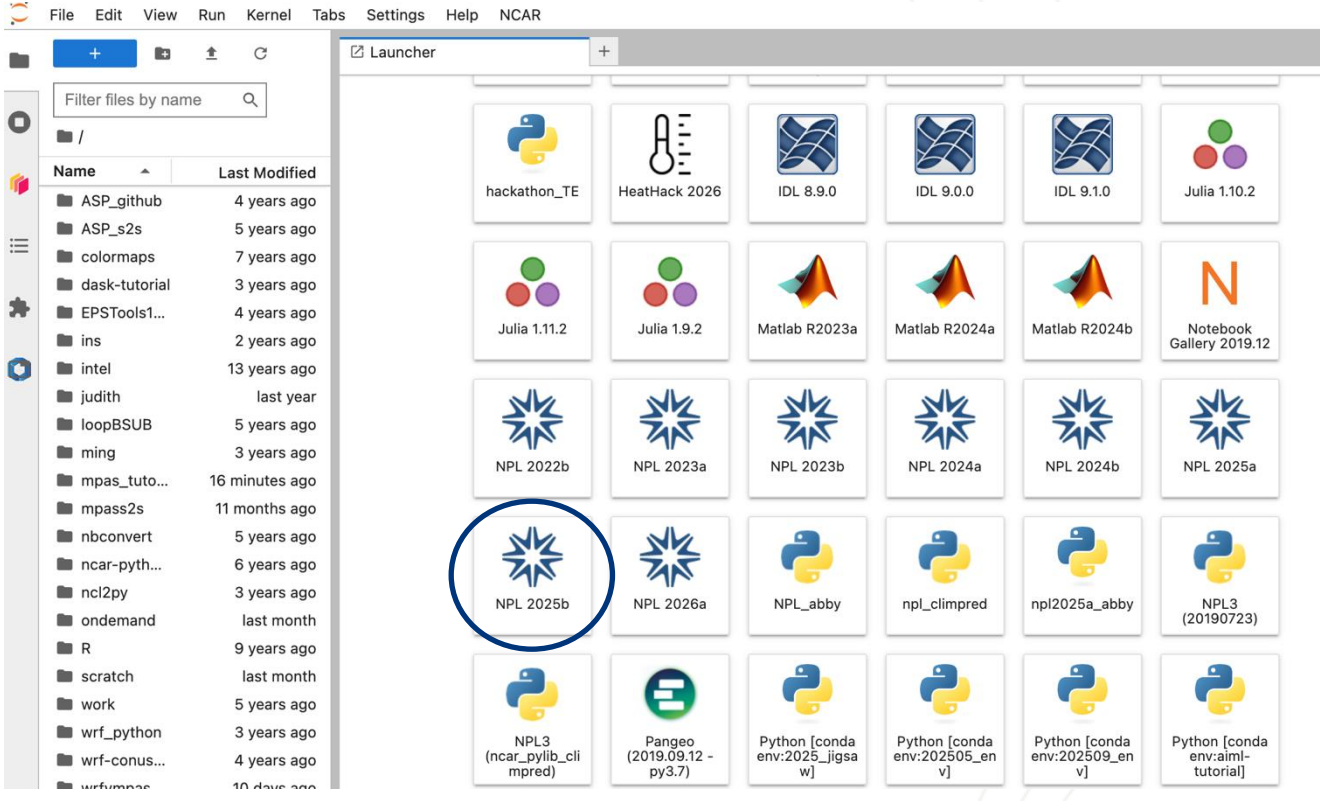


NCAR JupyterHub

- I'll step through some screenshots showing a jupyterhub login

ondemand.hpc.ucar.edu

- Click on **NPL 2025b**
- It will launch a new jupyter notebook



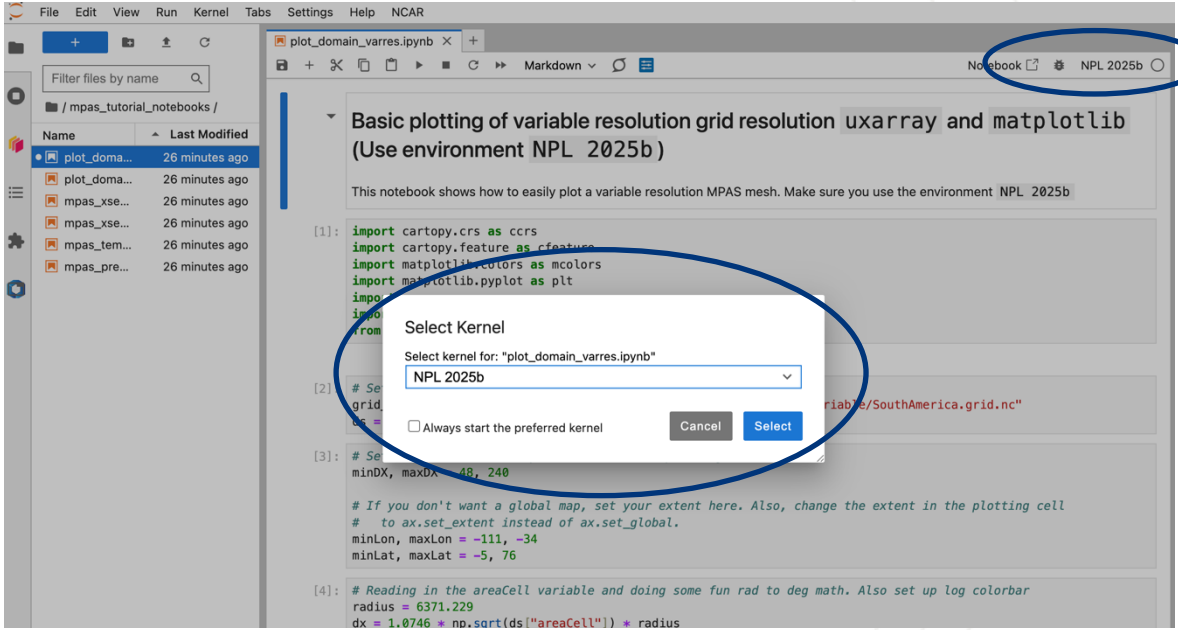
NCAR JupyterHub

- In your home directory in a terminal window you will want to link your scratch directory:

  cd /glade/u/home/${user}

  ln –sf /glade/derecho/scratch/${user} scratch

- Use the column on the left to navigate to a notebook
- When it opens click in the upper right to pick an environment
- You can select your kernel from a drop down menu.
- **NPL 2025b**



Let's plot some data!

[1]:
```
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.colors as colors
import numpy as np
import xarray as xr
import uxarray as ux
import cartopy.feature as cfeature
import cartopy.crs as ccrs
from cartopy.mpl.ticker import LongitudeFormatter, LatitudeFormatter
import glob
```

[3]:
```
base_path = "../240-48km_variable/"
files = f"{base_path}diag*nc"
grid_path = f"{base_path}SouthAmerica.grid.nc"
uxds = ux.open_mfdataset(grid_path,files,combine="nested",concat_dim="Time")
plot_time = "2014-09-13 00"
```

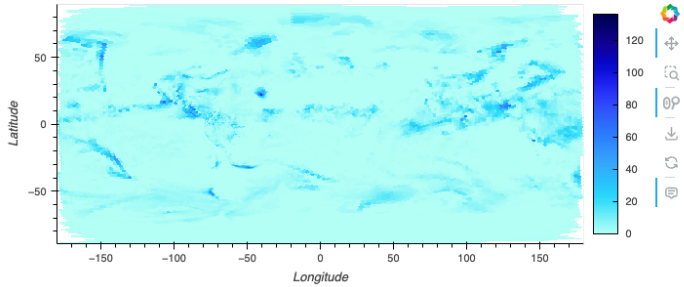
Calculate 3hourly, daily and read in accumulated precipitation.

[4]:
```
prec = uxds["rainc"] + uxds["rainnc"]
prec_3hr = prec.diff(dim="Time") # 3hourly precip
prec_daily = prec.resample(Time="1D",closed="right",label="left").max().diff(dim="Time") # Daily precip
prec_accum = prec.isel(Time=-1).squeeze() # Precip for length of entire model run. Just grabbed last timeslice
```

Let's just take a look at daily precip in a very simple way. This plot is also interactive. You can hover over it to see values and zoom in/out.

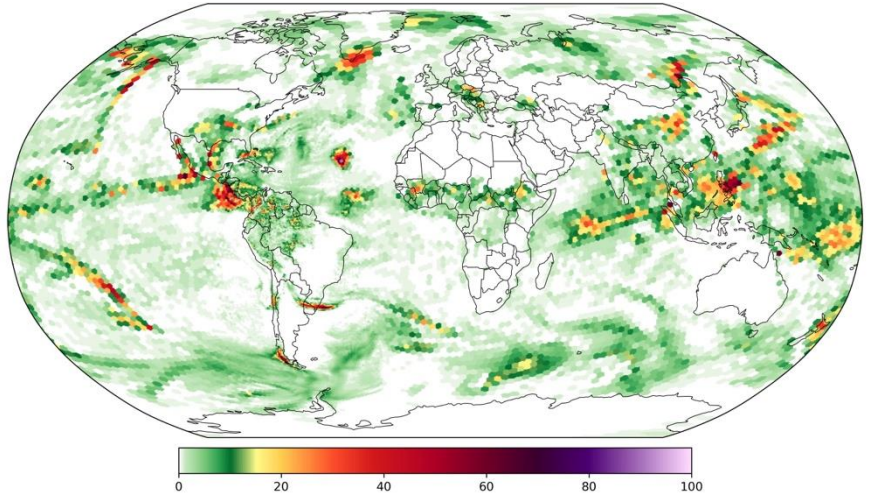
[5]:
```
prec_daily.sel(Time=plot_time).plot()
```

[5]:

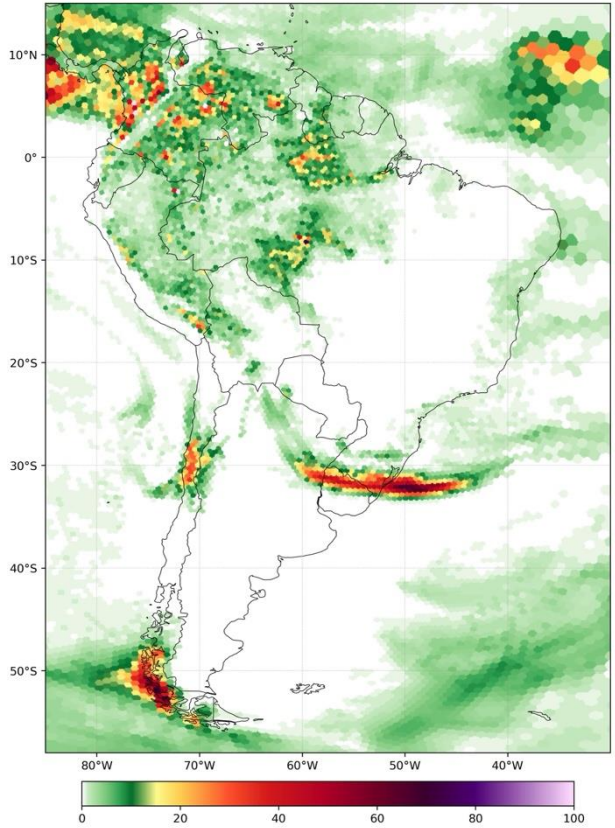


We want to plot daily precipitation for a variable resolution mesh, then zoom in

[6]:
```
projection = ccrs.Robinson()
fig, ax = plt.subplots(figsize=(10,10), subplot_kw={"projection": projection},facecolor="w",layout="constrained")
ax.set_global()

raster = prec_daily.sel(Time=plot_time).to_raster(ax=ax)
im = ax.imshow(raster,origin="lower",extent=ax.get_xlim() + ax.get_ylim(),cmap=cmap,clim=(0,100),zorder=1)
ax.add_feature(cfeature.COASTLINE.with_scale("110m"), linewidth=0.5, edgecolor="black")
ax.add_feature(cfeature.BORDERS.with_scale("110m"), linewidth=0.5, edgecolor="black")

cbar = fig.colorbar(im,ax=ax,orientation="horizontal",pad=0.01,shrink=0.6)

#plt.savefig("precip_global.jpg", bbox_inches="tight", dpi=300)
plt.show()
```



Let's plot some data!



We want to plot daily precipitation for a variable resolution mesh, then zoom in

[7]:
```
minLon, maxLon = -85, -30
minLat, maxLat = -58, 15
projection = ccrs.PlateCarree()
fig, ax = plt.subplots(figsize=(10,10), subplot_kw={"projection": projection},facecolor="w",layout="constrained")
ax.set_extent([minLon,maxLon,minLat,maxLat],crs=ccrs.PlateCarree())

raster = prec_daily.sel(Time=plot_time).to_raster(ax=ax)
im = ax.imshow(raster,origin="lower",extent=ax.get_xlim() + ax.get_ylim(),cmap=cmap,clim=(0,100),zorder=1)
ax.add_feature(cfeature.COASTLINE.with_scale("50m"),linewidth=0.5,edgecolor="black")
ax.add_feature(cfeature.BORDERS.with_scale("50m"),linewidth=0.5,edgecolor="black")

# Add lats and lons
step = 10  # Change to 5 if you want ticks every 5 degrees
start_lat = np.ceil(minLat / step) * step
start_lon = np.ceil(minLon / step) * step
ax.set_yticks(np.arange(start_lat,maxLat,step),crs=projection)
ax.yaxis.set_major_formatter(LatitudeFormatter()); ax.tick_params(axis="y",labelsize=10)
ax.set_xticks(np.arange(start_lon,maxLon,step),crs=projection)
ax.xaxis.set_major_formatter(LongitudeFormatter(zero_direction_label=True)); ax.tick_params(axis="x",labelsize=10)
ax.grid(True, linestyle=":", linewidth=0.5, color="gray", alpha=0.6)

cbar = fig.colorbar(im,ax=ax,orientation="horizontal",pad=0.01,shrink=0.6)

#plt.savefig("precip_samerica.jpg", bbox_inches="tight", dpi=300)
plt.show()
```

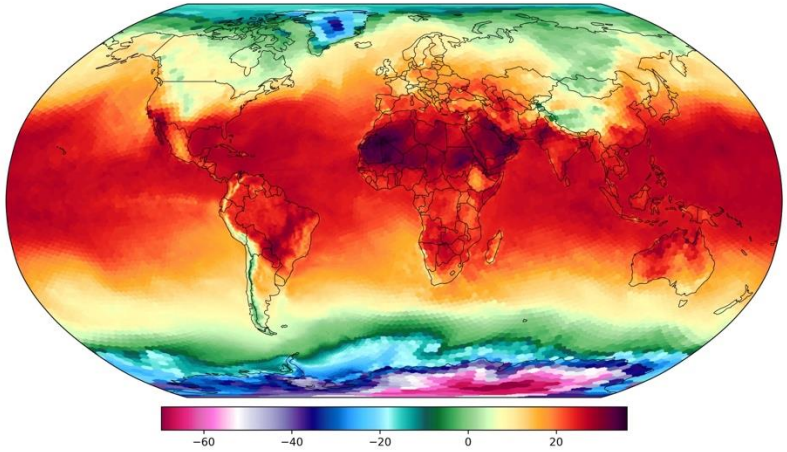
Let's plot some data!

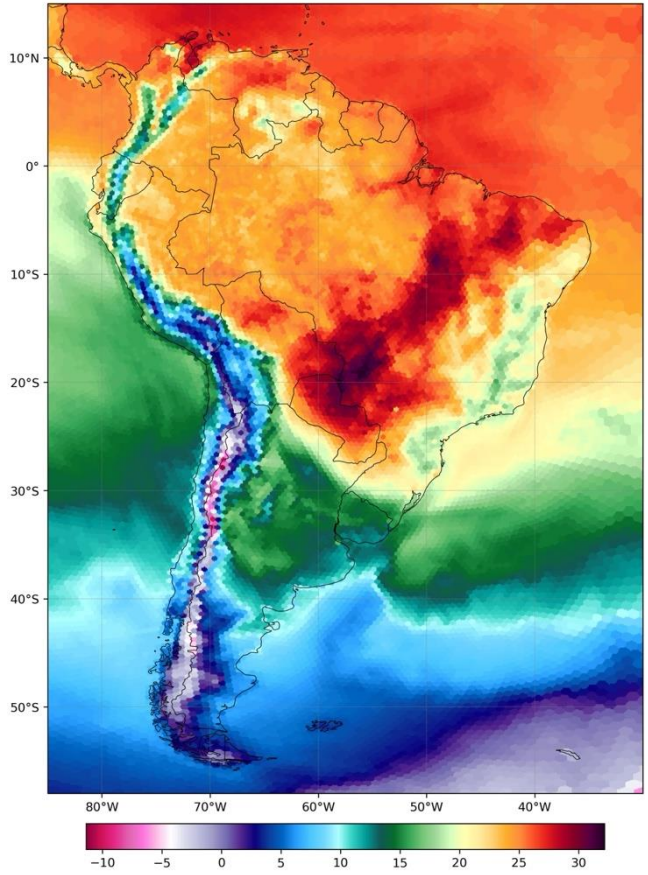
Calculate daily max, min and average 2m temperature. Plotting is essentially the same as it was for precip

```
[3]: base_path = "/glade/derecho/scratch/jaye/mpas_tutorial/240-48km_variable/"
     files = f"{base_path}diag*nc"
     grid_path = f"{base_path}SouthAmerica.grid.nc"
     uxds = ux.open_mfdataset(grid_path,files,combine="nested",concat_dim="Time")
     plot_time = "2014-09-13 00"
     ms2kts = 1.94384
```

Some really basic analysis showing daily mean/max/min calculations, selecting our time slice a

```
[4]: t2max = uxds["t2m"].resample(Time="D").max().sel(Time=plot_time)-273.15
     t2min = uxds["t2m"].resample(Time="D").min().sel(Time=plot_time)-273.15
     t2avg = uxds["t2m"].resample(Time="D").mean().sel(Time=plot_time)-273.15
```





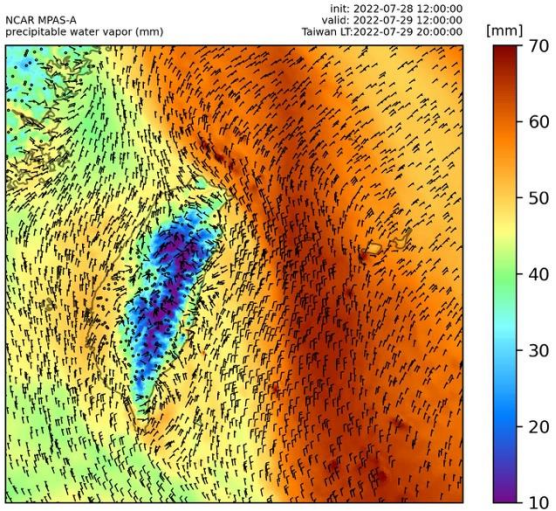
Overlaying wind barbs

- Now we want to overlay wind barbs over temperature and plot the global mesh on a different projection
- Wind barbs can be tricky with an unstructured mesh
  - We need to interpolate to a 2d grid using **griddata**

[7]:
```
t2_time = uxds["t2m"].sel(Time=plot_time)-273.15
u10_time = uxds["u10"].sel(Time=plot_time).values.squeeze()*ms2kts
v10_time = uxds["v10"].sel(Time=plot_time).values.squeeze()*ms2kts
```

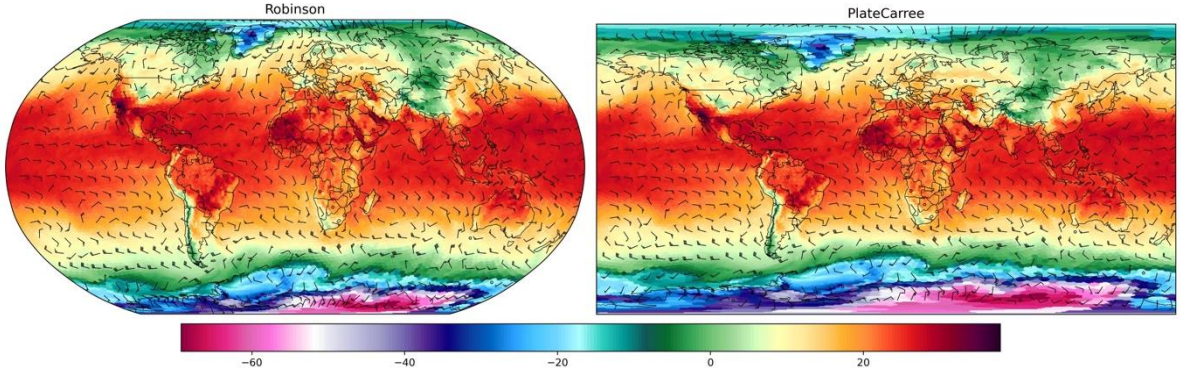
To make the wind barbs look evenly spaced on the map, we need to interpolate to a 2d grid using `griddata`. This is very easy and fast!

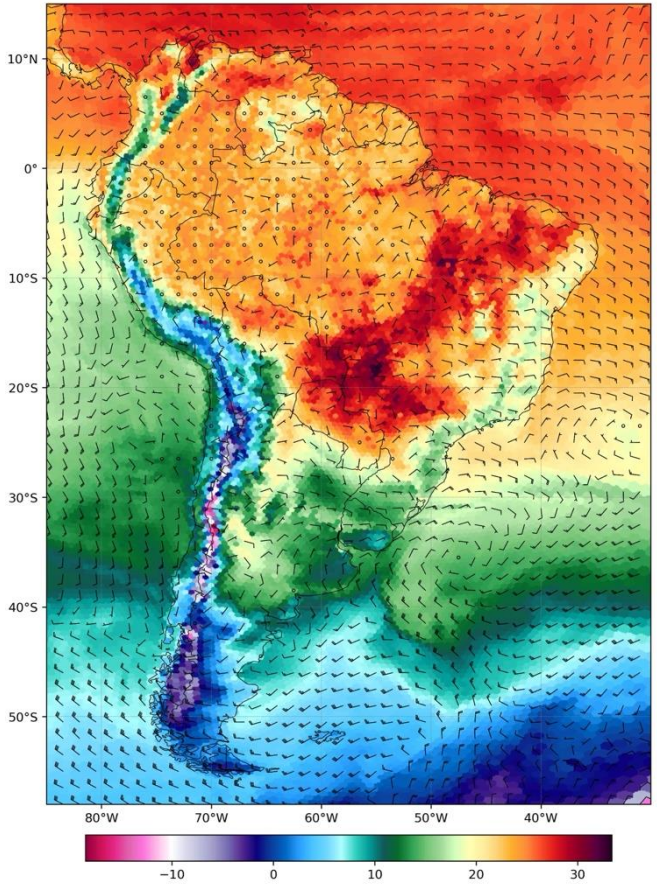
[8]:
```
# Interpolation for wind barbs
minLat, maxLat = -90, 90
minLon, maxLon = -180, 180
barb_res = 8.5 # Degree spacing of barbs
lon2d, lat2d = np.meshgrid(np.arange(minLon,maxLon,barb_res),np.arange(minLat,maxLat,barb_res))

lons = np.where(uxds.uxgrid.face_lon.values > 180,uxds.uxgrid.face_lon.values - 360,uxds.uxgrid.face_lon.values)
lats = uxds.uxgrid.face_lat.values

u_interp = griddata((lons,lats),u10_time.flatten(),(lon2d,lat2d),method='linear')
v_interp = griddata((lons,lats),v10_time.flatten(),(lon2d,lat2d),method='linear')
```



Overlaying wind barbs

```
[9]: fig = plt.figure(figsize=(16, 6), facecolor="w", layout="constrained")

ax1 = fig.add_subplot(1,2,1,projection=ccrs.Robinson())
ax1.set_global()
raster1 = t2_time.to_raster(ax=ax1)
im1 = ax1.imshow(raster1,origin="lower",extent=ax1.get_xlim() + ax1.get_ylim(),cmap=cmap,zorder=1)
ax1.set_title("Robinson",fontsize=13)

ax2 = fig.add_subplot(1,2,2,projection=ccrs.PlateCarree())
ax2.set_global()
raster2 = t2_time.to_raster(ax=ax2)
im2 = ax2.imshow(raster2,origin="lower",extent=ax2.get_xlim() + ax2.get_ylim(),cmap=cmap,zorder=1)
ax2.set_title("PlateCarree",fontsize=13)

for ax in (ax1, ax2):
    ax.barbs(lon2d,lat2d,u_interp,v_interp,length=4,linewidth=0.5,color="black",transform=ccrs.PlateCarree())
    ax.add_feature(cfeature.COASTLINE.with_scale("110m"),linewidth=0.5,edgecolor="black")
    ax.add_feature(cfeature.BORDERS.with_scale("110m"),linewidth=0.5,edgecolor="black")

cbar = fig.colorbar(im1,ax=[ax1,ax2],orientation="horizontal",pad=0.02,shrink=0.7,aspect=30)

#plt.savefig("temp_barbs_global.jpg", bbox_inches="tight", dpi=300)
plt.show()
```





Vertical cross-sections

- Here we will make a cross-section of zonal (uwind) water vapor mixing ratio (qv) flux and plot with either quivers or wind barbs
- flux = qv*uwind

[3]:
```
base_path = "../240-48km_variable/"
static_path = f"{base_path}SouthAmerica.static.nc"
data_path = f"{base_path}history.2014-09-13_00.00.00.nc"

ds_stat = xr.open_dataset(static_path,decode_times=False).squeeze()
ds_data = xr.open_dataset(data_path).squeeze()
```

Here we define our zonal cross-section line cutting across S America at 31 S. We can also choose to plot U and W with windbarbs or quivers

[4]:
```
# Define cross-section line
lat_start, lon_start = -31.0, 285.0
lat_end, lon_end = -31.0, 315.0
wind = "quiver" # Can be barb or quiver
n_pt = 500  # Number of points along the line

# Pressure levels to interpolate to
p_1d = np.arange(1000,50,-25)
n_p = len(p_1d)
```

[2]:
```
def interp_vertical(var_line,pres_line,target_p):
    """Interpolates 2D line data vertically onto standard pressure levels."""
    n_pts = var_line.shape[0]
    n_p = len(target_p)
    result = np.zeros((n_p,n_pts))

    for i in range(n_pts):
        # Reverse profiles so pressure increases
        p_prof = pres_line[i,::-1]
        v_prof = var_line[i,::-1]
        # Log-linear pressure interpolation
        result[:,i] = np.interp(np.log(target_p[::-1]),np.log(p_prof),v_prof)[::-1]

    return result
```

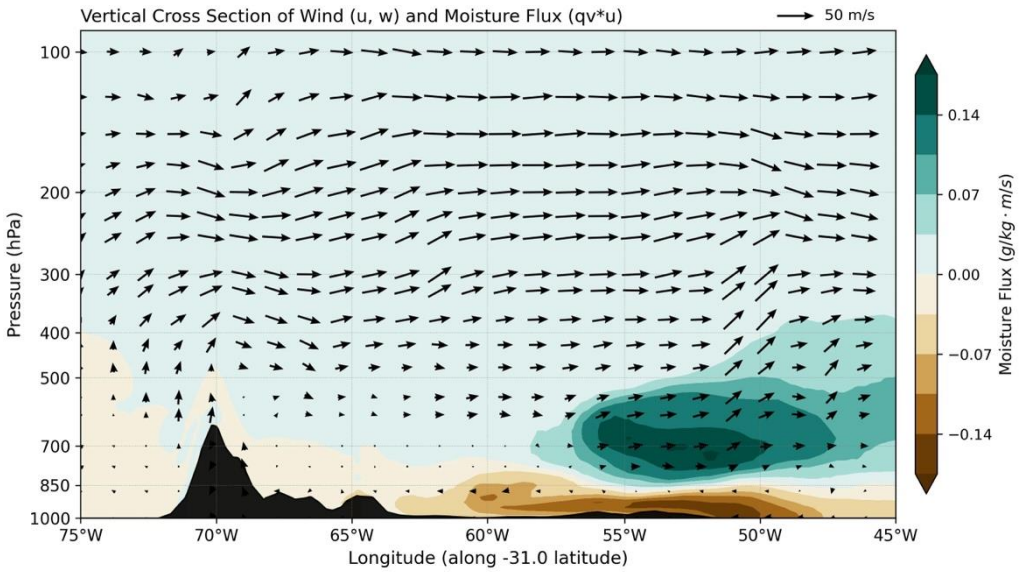
[6]:
```
lon_1d = np.linspace(lon_start, lon_end, n_pt)
lat_1d = np.linspace(lat_start, lat_end, n_pt)
line_points = np.column_stack((lon_1d, lat_1d))  # (500, 2)

# MPAS grid triangulation
mpas_lons = np.degrees(ds_stat["lonCell"].values)
mpas_lats = np.degrees(ds_stat["latCell"].values)
tri = Delaunay(np.column_stack((mpas_lons, mpas_lats)))
```

[7]:
```
# Interpolate winds and mflux
n_levels = pres.shape[1]

pres_line  = np.zeros((n_pt,n_levels))
mflux_line = np.zeros((n_pt,n_levels))
uwind_line = np.zeros((n_pt,n_levels))
vwind_line = np.zeros((n_pt,n_levels))
wwind_line = np.zeros((n_pt,n_levels))

for k in range(n_levels):
    pres_line[:,k]  = LinearNDInterpolator(tri,pres[:,k])(line_points)
    mflux_line[:,k] = LinearNDInterpolator(tri,mflux[:,k])(line_points)
    uwind_line[:,k] = LinearNDInterpolator(tri,uwind[:,k])(line_points)
    vwind_line[:,k] = LinearNDInterpolator(tri,vwind[:,k])(line_points)
    wwind_line[:,k] = LinearNDInterpolator(tri,wwind[:,k])(line_points)

hh_cross_section = interp_vertical(mflux_line,pres_line,p_1d)
uu_cross_section = interp_vertical(uwind_line,pres_line,p_1d)
vv_cross_section = interp_vertical(vwind_line,pres_line,p_1d)
ww_cross_section = interp_vertical(wwind_line,pres_line,p_1d)
```

Now, interpolate terrain along that line and interpolate to pressure level so we can see the mountains.

[8]:
```
# Terrain/pressure interpolation needed to plot terrain
ter = ds_stat["ter"].squeeze().values
terrain_line = LinearNDInterpolator(tri,ter)(line_points)
terrain_pressure = 1000*np.exp(-9.81*terrain_line/(287.0*280.0))
```

Vertical cross-sections



[9]:
```
fig, ax = plt.subplots(figsize=(12,6))

# Make mflux contour plot
vmax = np.nanmax(np.abs(hh_cross_section))
levels = np.round(np.linspace(-vmax,vmax,11),2)
t_cs = ax.contourf(np.arange(n_pt),p_1d,hh_cross_section,cmap="BrBG",levels=levels,extend="both")
cbar = plt.colorbar(t_cs,ax=ax,shrink=0.9,pad=0.02)
cbar.set_label("Moisture Flux ($g/kg \cdot m/s$)",fontsize=11)

# Make wind plot
# Subsample vertical levels on a log-scale so arrows/barbs don't crowd
log_p_levels = np.linspace(np.log10(p_1d.min()),np.log10(p_1d.max()),22)
selected_indices = sorted(set([np.abs(p_1d-10**p).argmin() for p in log_p_levels]))
X, Y = np.meshgrid(np.arange(n_pt),p_1d)
if wind=="quiver":
    sub_y = selected_indices
    sub_x = slice(None,None,20)
    Q = ax.quiver(X[sub_y,sub_x],Y[sub_y,sub_x],uu_cross_section[sub_y,sub_x],ww_cross_section[sub_y,sub_x],
                  scale=1100,width=0.0025,headwidth=4,color="black",pivot="mid")
    ax.quiverkey(Q,X=0.90,Y=1.03,U=50,label="50 m/s",labelpos="E")
elif wind=="barb":
    ax.barbs(X[selected_indices,::30],Y[selected_indices,::30],uu_cross_section[selected_indices,::30],
             ww_cross_section[selected_indices,::30],length=5,linewidth=0.5,barbcolor="black",pivot="middle")

# Plot terrain as a fill
ax.fill_between(np.arange(n_pt),p_1d.max(),terrain_pressure,color="black",alpha=0.9)

# Label lons
lon_1d_180 = np.where(lon_1d>180,lon_1d-360,lon_1d)
target_lons = np.arange(np.ceil(lon_1d_180.min()/5)*5,lon_1d_180.max()+0.01,5)
ax.set_xticks([np.abs(lon_1d_180 - l).argmin() for l in target_lons])
ax.set_xticklabels([f"{abs(int(l))}°W" for l in target_lons], fontsize=11)
ax.set_xlabel(f"Longitude (along {lat_start} latitude)",fontsize=12)

# Format Y-axis and add labels
ax.set_yscale("log")
ax.set_ylim(1000,90)
ax.set_yticks(ticks := [1000, 850, 700, 500, 400, 300, 200, 100],labels=ticks,fontsize=11)
ax.tick_params(axis="y", which="minor", left=False, right=False)
ax.set_ylabel("Pressure (hPa)",fontsize=12)
ax.grid(True, linestyle=":", linewidth=0.5, color="gray", alpha=0.6)

# Title and save fig
plt.title("Vertical Cross Section of Wind (u, w) and Moisture Flux (qv*u)",loc="left",fontsize=12)
#plt.savefig("xsec_mflux.jpg", bbox_inches="tight", dpi=300)
plt.show()
```

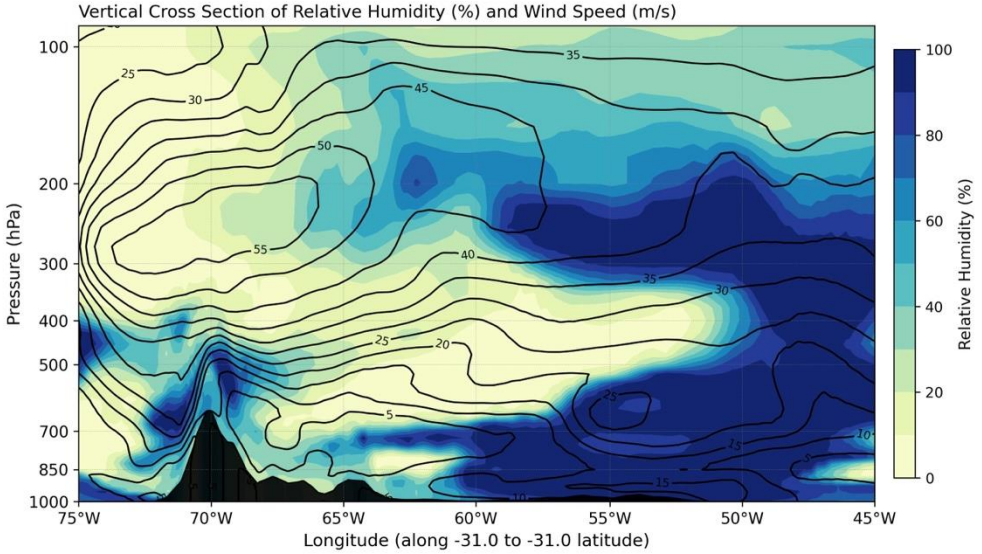
Vertical cross-sections

- Now we will plot relative humidity with contours of wind overlayed
- Also, it would make sense here if we did non-zonal cross-sections!

[4]:
```
# Define cross-section line
lat_start, lon_start = -31.0, 285.0 # Across South America along 31 S
lat_end, lon_end = -31.0, 315.0
#lat_start, lon_start = 45.0, 236.0 # NW to SE CONUS
#lat_end, lon_end = 28.0, 278.0
#lat_start, lon_start = 32.0, 242.0 # SW to NE CONUS
#lat_end, lon_end = 45.0, 291.0
n_pt = 500  # Number of points along the line

# Pressure levels
p_1d = np.arange(1000,50,-25)
n_p = len(p_1d)
```

Reading in the 3D variables we care about. rh, and winds.

[5]:
```
relhum = ds_data["relhum"].values
uwind = ds_data["uReconstructZonal"].values
vwind = ds_data["uReconstructMeridional"].values
pres = ds_data["pressure"].values * 0.01
wind = np.sqrt(uwind**2+vwind**2)
```



Variable mesh resolution plot

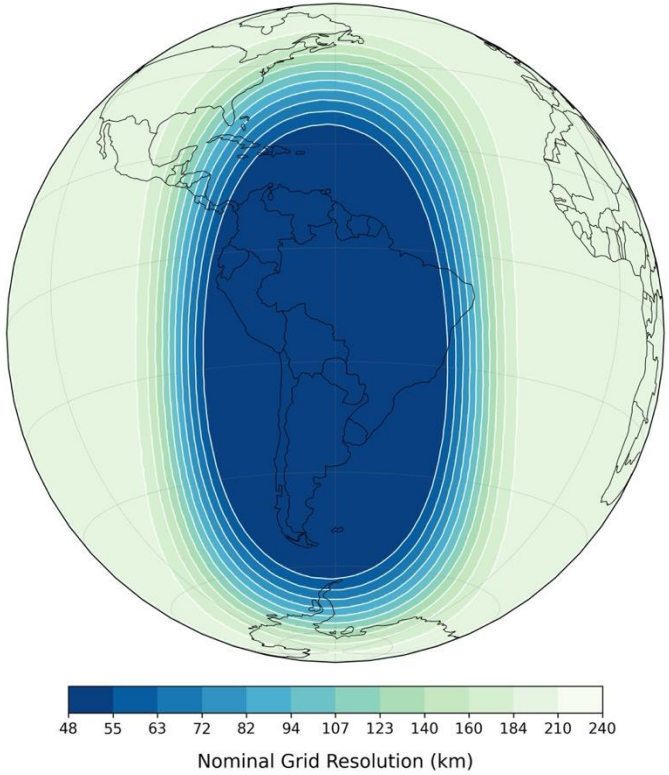
[2]:
```
# Set the location of your *grid.nc file
grid_file = "../240-48km_variable/SouthAmerica.grid.nc"
ds = ux.open_dataset(grid_file,grid_file,decode_times=False)
```

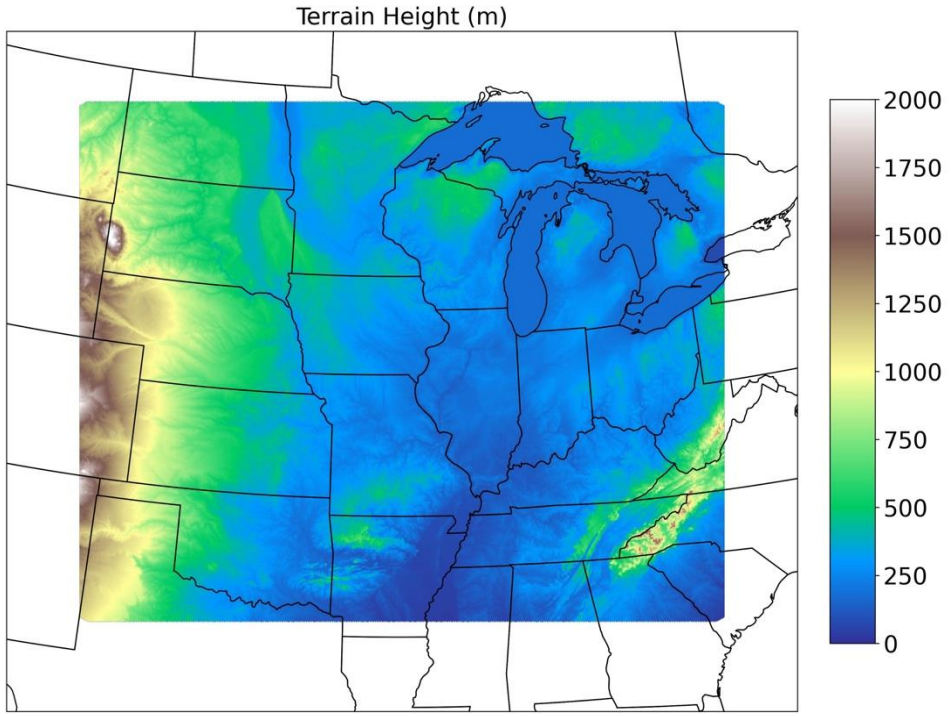
[3]:
```
# Set minDX and maxDX of your mesh for nice plotting
minDX, maxDX = 48, 240

# If you don't want a global map, set your extent here. Also, change
#   to ax.set_extent instead of ax.set_global.
minLon, maxLon = -111, -34
minLat, maxLat = -5, 76
```

[4]:
```
# Reading in the areaCell variable and doing some fun rad to deg math and set up log colorbar.
# Math comes from Michael Duda's mesh_resolution.py script
uxda_area = ds["areaCell"].values
uxda_den = ds["meshDensity"].values

# Scale area if on a unit sphere
if np.sum(uxda_area) < 4.0 * np.pi * 2.0:
    uxda_area = uxda_area * (6371229.0**2)

# Calculate nominal cell spacing field from mesh density
minSpacingKm = np.sqrt(np.min(uxda_area) * 2.0 / np.sqrt(3.0)) * 0.001
fld_km = minSpacingKm / np.power(uxda_den,0.25)

# Assign log10 values back into UXarray dataset
ds["log_dx"] = (ds["areaCell"].dims[0],np.log10(np.clip(fld_km,minDX,maxDX)))

log_space_ticks = np.linspace(np.log10(minDX),np.log10(maxDX), 13)
discrete_cmap = plt.colormaps["GnBu_r"].resampled(12)
norm = mcolors.BoundaryNorm(boundaries=log_space_ticks,ncolors=discrete_cmap.N)
```

[5]:
```
projection = ccrs.Orthographic(central_longitude=-60,central_latitude=-15)
fig, ax = plt.subplots(figsize=(8,8),dpi=50,subplot_kw={"projection": projection},facecolor="w",layout="constrained")
#ax.set_extent([minLon,maxLon,minLat,maxLat],crs=ccrs.PlateCarree())
ax.set_global()

raster = ds["log_dx"].to_raster(ax=ax)
raster_smooth = gaussian_filter(np.nan_to_num(raster,nan=np.nanmean(raster)),sigma=1.5)+(raster*0)
extent = ax.get_xlim() + ax.get_ylim()

# Draw filled contours
im = ax.imshow(raster_smooth,origin="lower",extent=extent,transform=projection,cmap=discrete_cmap,norm=norm,zorder=1)

# Draw white contours on top of filled contour borders so they pop
ny, nx = raster.shape
x, y = np.linspace(extent[0],extent[1],nx),np.linspace(extent[2],extent[3],ny)
ax.contour(x,y,raster_smooth,levels=log_space_ticks[1:-1],
           colors="white",linewidths=0.8,transform=projection,zorder=2)

# Add borders and gridlines
ax.add_feature(cfeature.COASTLINE.with_scale("110m"),linewidth=0.5,edgecolor="black",zorder=3)
ax.add_feature(cfeature.BORDERS.with_scale("110m"),linewidth=0.5,edgecolor="black",zorder=3)
ax.gridlines(color="gray",linestyle=":",linewidth=0.5,alpha=0.7,zorder=4)

cbar = fig.colorbar(im,ax=ax,orientation="horizontal",pad=0.03,shrink=0.7,ticks=log_space_ticks)
cbar.ax.set_xticklabels([f"{10**t:.0f}" for t in log_space_ticks],fontsize=11)
cbar.set_label("Nominal Grid Resolution (km)",fontsize=14,labelpad=12)

fig.set_dpi(100)
#plt.savefig("domain_varres.jpg",bbox_inches="tight",dpi=300)
plt.show()
```

Variable mesh resolution plot (and regional!)





Regridding options

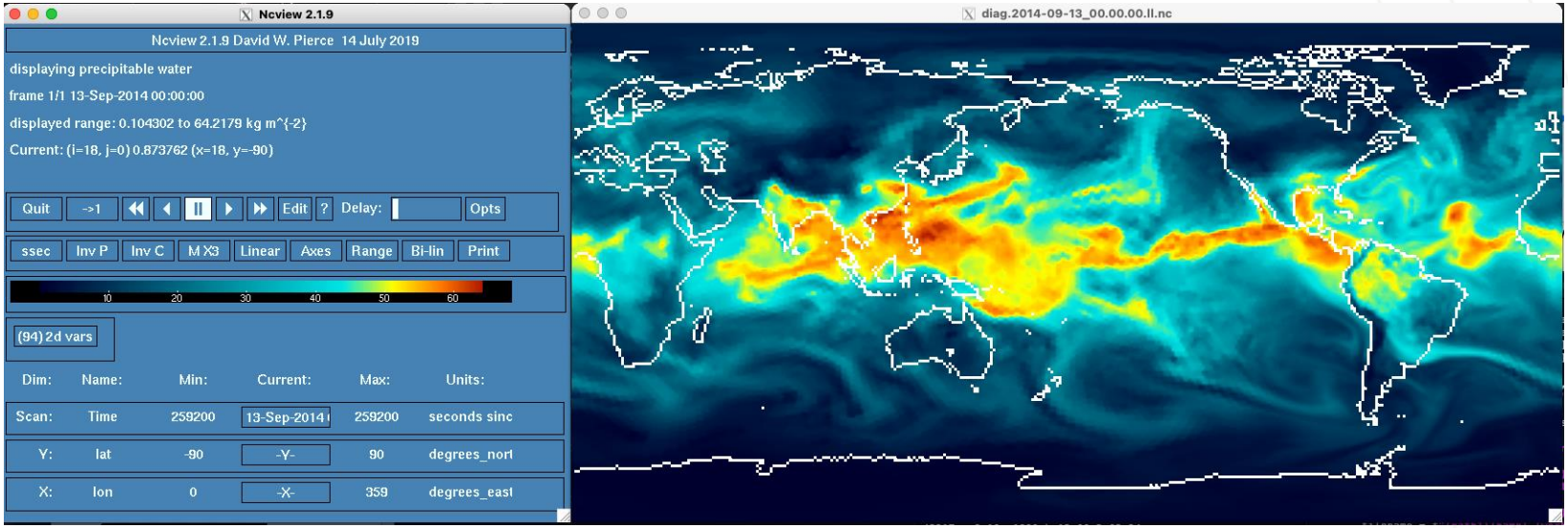
What if we need to regrid?

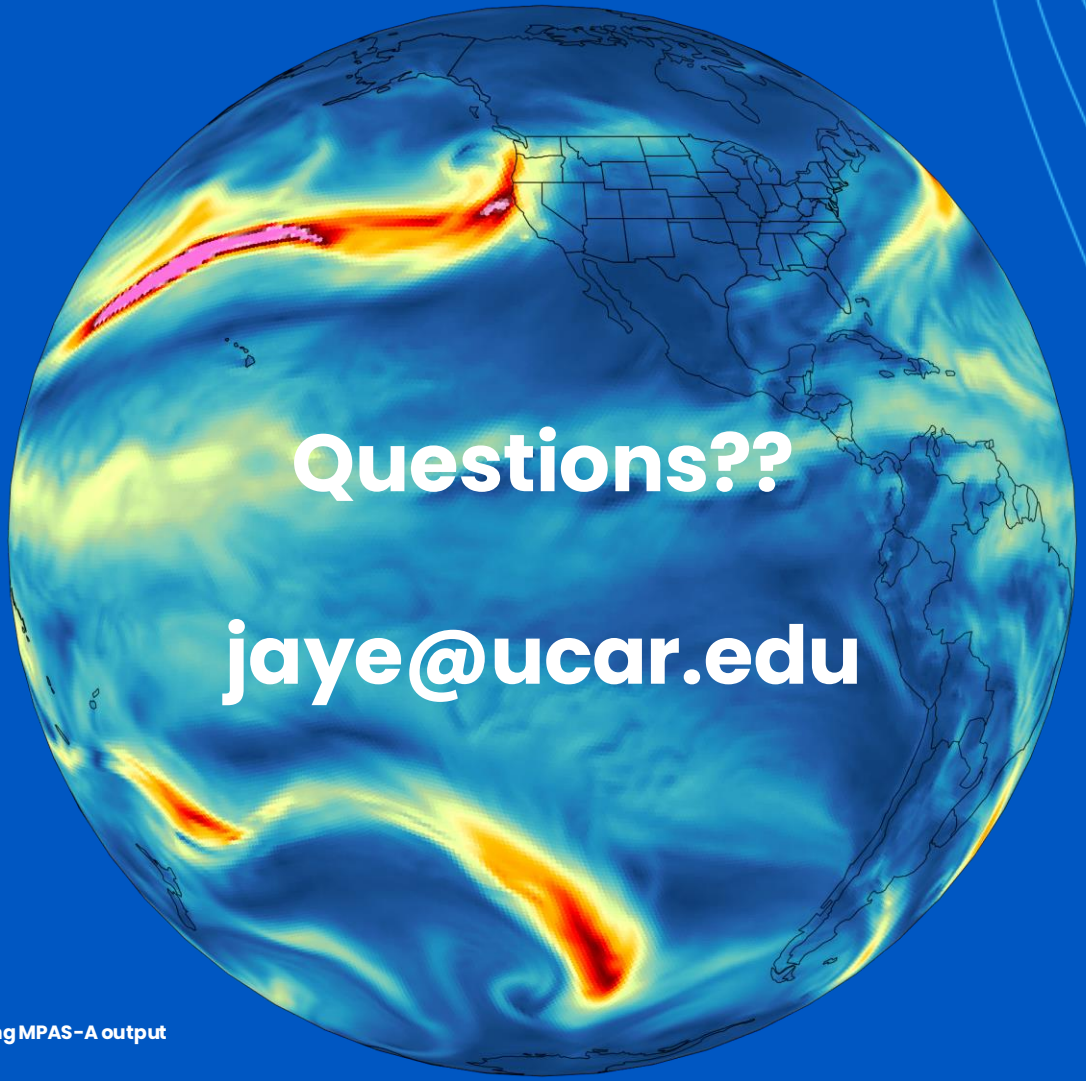
- **CDO** (climate data operators) work really well to quickly regrid files from the MPAS-A native grid to a lat-lon domain
  - Best to regrid to the finest resolution of your mesh
  - For flux fields such as precipitation you should use conservative remapping. For others, you can use bilinear or nearest neighbor as well
  - Important that you output the CF-compliant 'Time'. CDO doesn't know what to do with xtime
  - This will struggle with 3D variables. You will need to use '–selgrid,1'

```
cdo -P 1 -f nc5 sellonlatbox,270,310,25,55 -remapcon,r10018x5009 -setgrid,mpas:$mpas_grid
-selname,mslp,t2m,u10,v10 $ifile $ofile

cdo -P 1 -f nc5 -remapcon,r360x181 -setgrid,mpas:$mpas_grid –selgrid,1 $ifile $ofile
```

Regridding options

```
cdo -P 1 -f nc5 -remapcon,r360x181 -setgrid,mpas:SouthAmerica.grid.nc -selgrid,1
diag.2014-09-13_00.00.00.nc diag.2014-09-13_00.00.00.ll.nc
```





Questions??

jaye@ucar.edu