part-2

December 5, 2019

1 Using Python with Earth Science Datasets: Cartopy, xarray, and pyresample

1.1 Objective: working with Earth science datasets

- You will learn to:
  - Plot data on a map
  - Read netCDF datasets
  - Perform re-gridding, averaging, filtering, and out-of-memory operations

---

2 Recap on packages

Packages give us additional functionality, saving us the trouble of writing procedures ourselves.

Primary libraries from the last part... * NumPy: Fast mathematical operations on arrays * Pandas: Operations and easy read/write of tabular data. Builds extra functionality on top of NumPy. * Matplotlib: Primary Python plotting/visualization package. You can generate line plots, histograms, scatter plots, etc., with just a few lines of code.

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
# Bring back our list of aeronet stations...
filename = 'data/aeronet_locations_v3.txt'
station_list = pd.read_csv(filename, skiprows=1)
station_list.columns = ['site', 'lon', 'lat', 'elev']
```

2.1 Cartopy

- Cartopy is not included in Anaconda, need to install yourself.

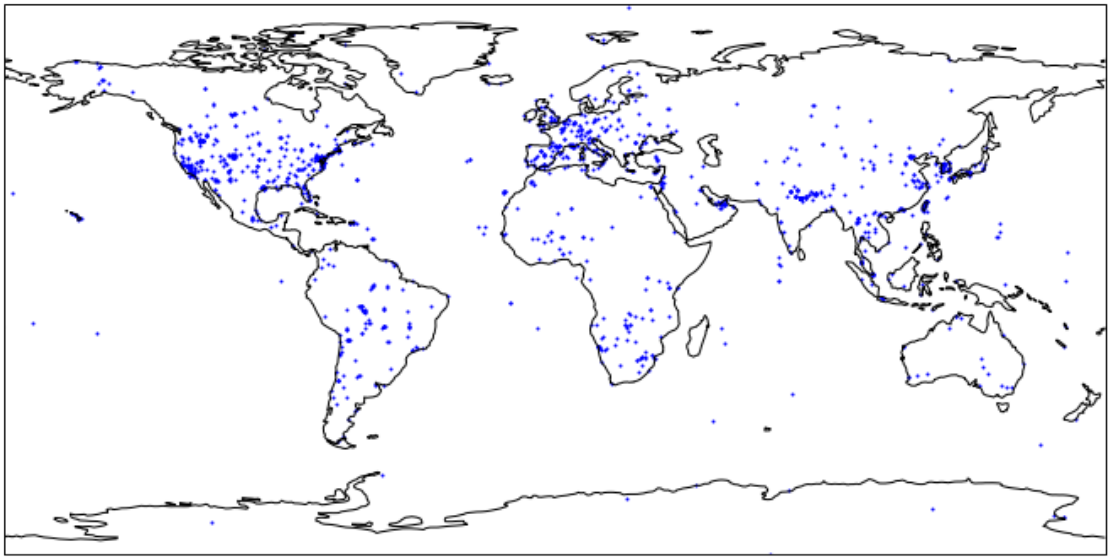
Open the terminal (Mac/Linux) or Anaconda Prompt (Windows) and type:

```
conda install -c conda-forge cartopy
```

- Rather than import all of Cartopy, we just want the projection classes to pair with matplotlib.
- More map projections.

[3]:
```
# Options to increase figure size
plt.rcParams['figure.figsize'] = [12, 6]
plt.rcParams.update({'font.size': 16})

from cartopy import crs as ccrs
```

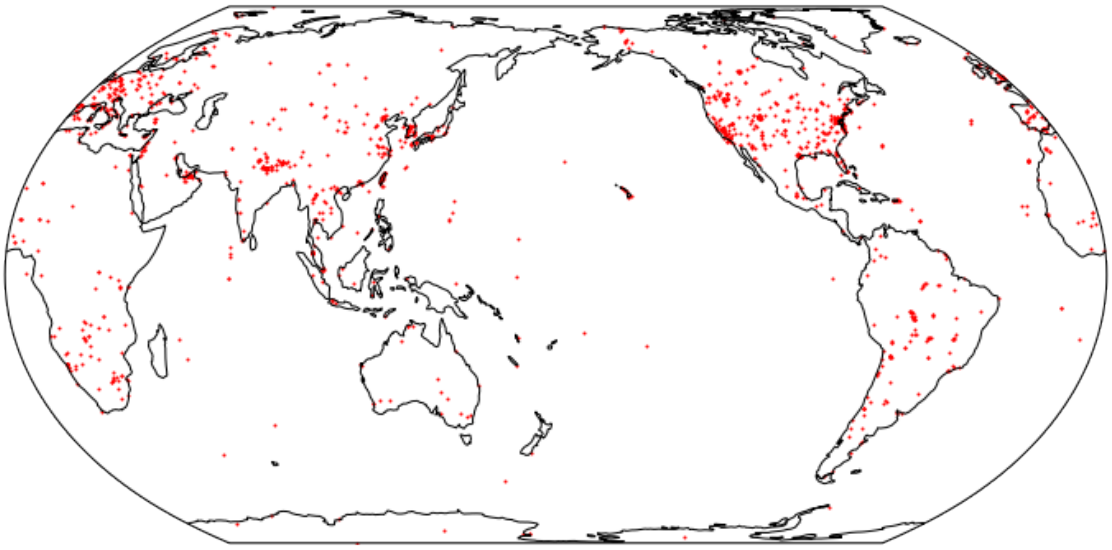
[4]:
```
from_proj = ccrs.PlateCarree()
to_proj = ccrs.PlateCarree()

# Center on the Atlantic
ax = plt.axes(projection=to_proj)
ax.coastlines()
plt.scatter(station_list['lon'], station_list['lat'], color='blue', s=1,␣
 ↪transform=from_proj)
plt.show()
```



Exercise 1

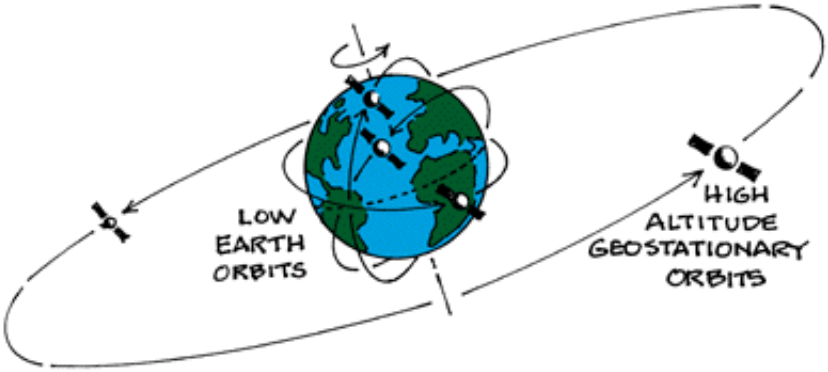
- Import aeronet_locations_v3.txt with read_csv
- Define the map axes using with option projection=ccrs.PlateCarree()
- Add coastlines
- Make a scatter plot of the station locations
- Challenge: Change the projection to Equal Earth and shift the plot to center on -180 using ccrs.EqualEarth(central_longitude=-180)

[5]:
```
# Solution: Center on the Pacific
from_proj = ccrs.PlateCarree()
to_proj = ccrs.EqualEarth(central_longitude=-180)

ax = plt.axes(projection=to_proj)
ax.coastlines()
plt.scatter(station_list['lon'], station_list['lat'], color='red', s=1,␣
 ↪transform=from_proj)
plt.show()
```



2.2 Examining the 2018 California Wildfires from Space

- 6,870 fires had burned over a 6,000 km^2 area.
- The smoke from the wildfires also had an impact on air quality both in proximity of the fires as well as across the country.
- We'll look at satellite observations from **Suomi-NPP** and **GOES-17** to show the impact of the California wildfires on AOD.



We're in the golden age of sallite datasets, which is a blessing and a curse:

- Inundated with datasets, don't know which ones to use
- No single repository for access of the data
- Inconsistent formatting and filetypes

netCDF4 and HDF5 are the dominant formats used in satellite remote sensing (but others do exist)

2.3 xarray

- xarray is like pandas, but for N-dimensional arrays.
- Can read and write NetCDF, HDF5, and GRIB (with the additional cfgrib package) files.
- Can perform efficient operations on these datasets, including out-of-memory and parallel operations.

2.3.1 LEO Example: Suomi-NPP

LEO Satellites orbit the Earth many times a day, data are organized in 1 minute swaths per file. We'll be looking at aerosol optical depth data over California.

[6]:
```
import xarray
```

[7]:
```
# Import your data...
filename = 'data/
 ↪JRR-AOD_v1r2_npp_s201811082130296_e201811082131537_c201811082228260.nc'
npp = xarray.open_dataset(filename)

# Print a list of variables
list(npp.variables)
```

[7]:
```
['Latitude', 'Longitude', 'QCAll', 'AOD550']
```

[8]:
```
AOD_NPP = npp['AOD550']
lat_NPP = npp['Latitude']
lon_NPP = npp['Longitude']
```

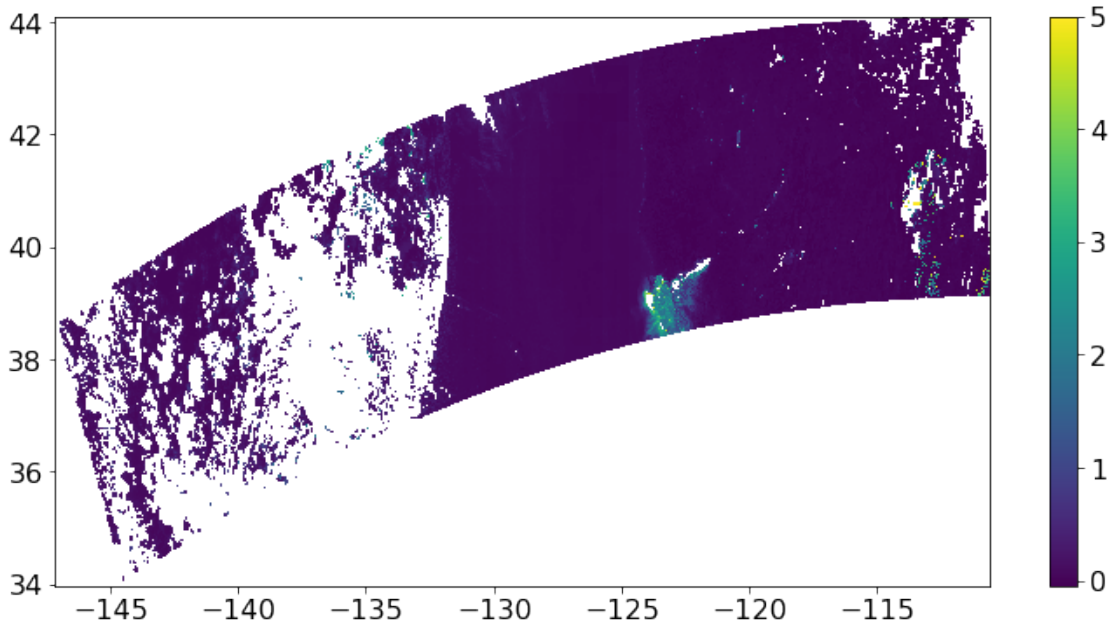
[9]:
```
# xarray has automatically read in the missing values as nan
AOD_NPP
```

[9]:
```
<xarray.DataArray 'AOD550' (Rows: 768, Columns: 3200)>
[2457600 values with dtype=float32]
Coordinates:
    Latitude   (Rows, Columns) float32 ...
    Longitude  (Rows, Columns) float32 ...
Dimensions without coordinates: Rows, Columns
Attributes:
    long_name:    Aerosol optical depth at 550 nm
    units:        1
    valid_range:  [-0.05  5.  ]
```

[10]:
```
AOD_NPP.mean()
```

[10]:
```
<xarray.DataArray 'AOD550' ()>
array(0.082712, dtype=float32)
```

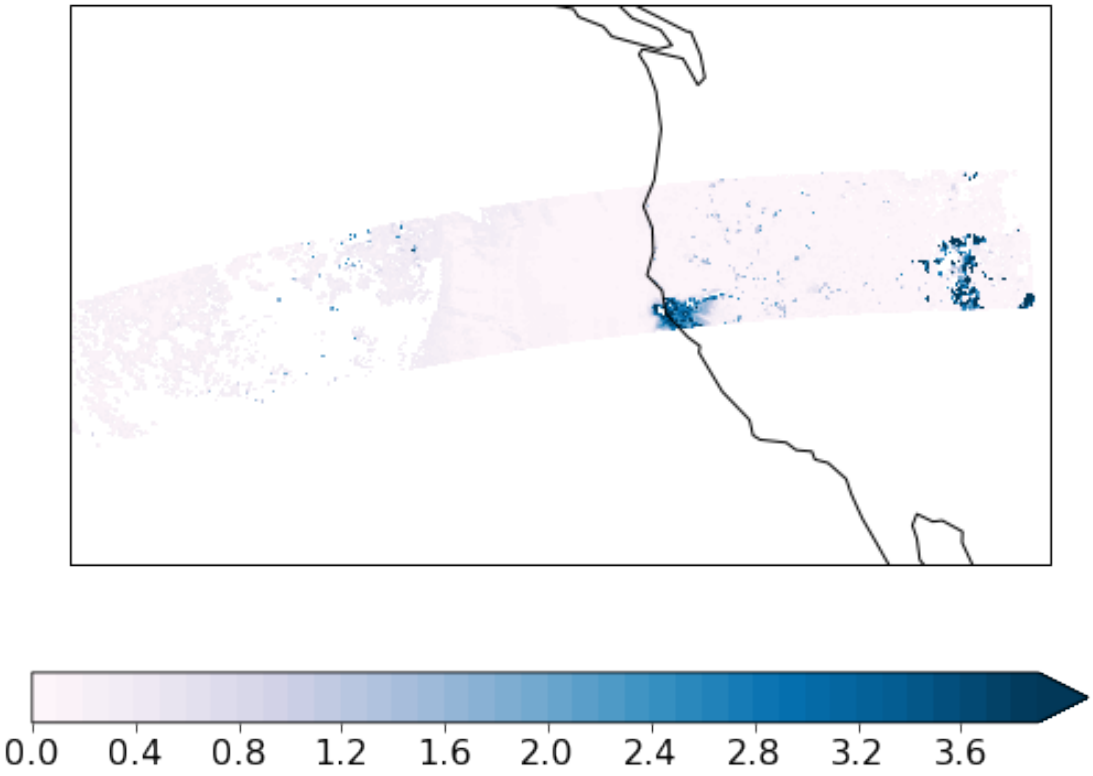
[11]:
```
plt.pcolormesh(lon_NPP, lat_NPP, AOD_NPP)
plt.colorbar()
plt.show()
```



In the next plot, we'll * Add in the cartopy map * Change the color scheme * See the matplotlib color schemes and ColorBrewer schemes. * Recommend cmocean, a collection of colormaps for oceanography but great for other purposes too. * There is an option to zoom in on the data using plt.xlim and plt.ylim

[12]:
```
from_proj = ccrs.PlateCarree()
to_proj = ccrs.PlateCarree()

levs = np.arange(0, 4, 0.1)

# Using cartopy, create the map projection and plot the data
ax = plt.axes(projection=to_proj)
ax.coastlines()

# Can change color scheme using the cmap argument
plt.contourf(lon_NPP, lat_NPP, AOD_NPP, levs, transform=from_proj, extend='max',
             cmap="PuBu")

plt.colorbar(orientation="horizontal", fraction=0.07)

plt.xlim(-145, -110)
```

```
plt.ylim(30, 50)

# To zoom in on data:
#plt.xlim(-125, -120)
#plt.ylim(38, 44)
plt.show()
```



Exercise 2
??? = Fill in the blanks!
Import from netCDF files

- Import JRR-AOD_v1r1_npp_s201808091957192_e201808091958434_c201808092051240.nc using the xarray.open_dataset command.
- Inpect the available variables.
- Save the latitude, longitude, and AOD to arrays.

Create a cartopy plot

- Define the axes, including the projection.
  - Challenge: Make an orthographic plot using: ccrs.Orthographic(central_longitude=-75.0, central_latitude=0.0)
- Create a plot using plt.contourf(???, ???, ???, transform=???)

  - Challenge: Change the data scale from the default to 0.0-1.6

- Remember to plt.show() at the end to display!

2.4 Common tasks

1. Regridding
2. Masking datasets
3. Filtering with Quality Flags

2.4.1 Regridding

Regridding is the process of interpolating from one grid resolution to a different grid resolution and UCAR's website has a good discussion.

There are a few options:

- Interpolate using griddata from SciPy package (2D interpolate too slow/limited number of points)
- Regridding in iris package (autodetection of GRIB and NC fileformats… if they follow the conventions!)
- xESMF, a Python package for regridding for gridded datasets (not designed for satellite data)
- **Pyresample**: the syntax can be fairly complex, but very fast

Pyresample also is not included in Anaconda, so install using:

```
conda install -c conda-forge pyresample
```

Steps: 1. Define the new grid (either in Python or import an irregular array) 2. Define the swath/satellite grid 3. Resample to get the new values for the updated grid

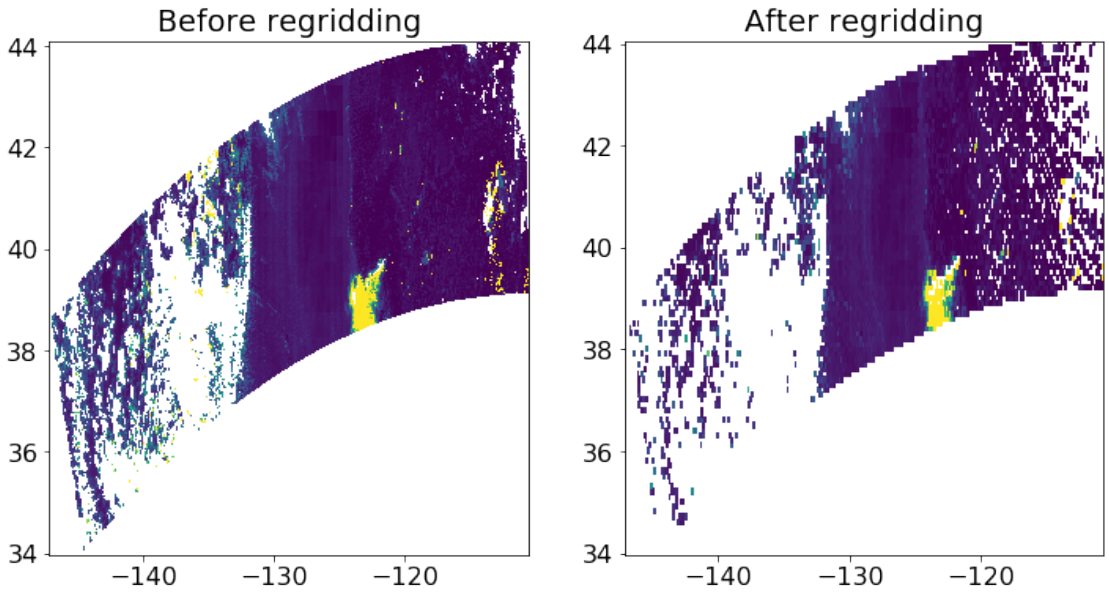
[13]:
```
from pyresample import geometry
from pyresample.kd_tree import resample_nearest
```

[14]:
```
# Create a new grid at 0.1 degree resolution
x = np.arange(lon_NPP.min(), lon_NPP.max(), 0.1)
y = np.arange(lat_NPP.min(), lat_NPP.max(), 0.1)
new_lon, new_lat = np.meshgrid(x,y)

# define the new grid using
new_lonlat = geometry.GridDefinition(lons=new_lon, lats=new_lat)
```

[15]:
```
# Input list of swath points
old_lonlat = geometry.SwathDefinition(lons=lon_NPP, lats=lat_NPP)

# Resample the data
new_AOD = resample_nearest(old_lonlat, AOD_NPP.values, new_lonlat,␣
 ↪radius_of_influence=5000, fill_value=None)
```

[16]:
```
# To compare: Putting two plots together using plt.subplot
plt.subplot(1, 2, 1)
plt.title("Before regridding")
```

```
plt.pcolormesh(lon_NPP, lat_NPP, AOD_NPP, vmin=0, vmax=1)

plt.subplot(1, 2, 2)
plt.pcolormesh(new_lon, new_lat, new_AOD, vmin=0, vmax=1)
plt.title("After regridding")
plt.show()
```

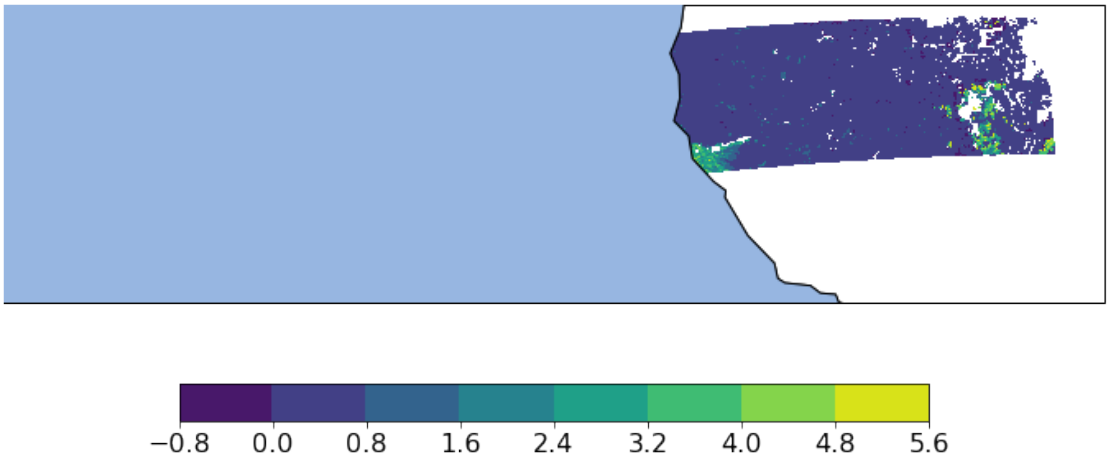


2.4.2 Masking datasets

May want to add a land/ocean mask to our datasets

[17]:
```
import cartopy.feature as feature
```

[18]:
```
from_proj = ccrs.PlateCarree()
to_proj = ccrs.PlateCarree()

ax = plt.axes(projection=to_proj)
ax.coastlines()
plot = ax.contourf(lon_NPP, lat_NPP, AOD_NPP, transform=from_proj)

# Mask out the ocean: zorder places it on top of the existing data
ax.add_feature(feature.OCEAN, zorder=100, edgecolor='black')

cb = plt.colorbar(plot, orientation="horizontal", fraction=0.07)
plt.show()
```



Exercise 3

- Filter your plot in Exercise 2 (using NPP data) by data quality using the variable 'QCAll'
- Import QCAll into a variable
- Create a mask for medium/low quality data or pixels with no retrieval
- Make a new map with the filtered data

[19]:
```
qc_flag = npp['QCAll']

mask_HQ = (qc_flag != 0)
AOD_NPP_HQ = np.ma.masked_where(mask_HQ, qc_flag)
```

2.5 Saving images

- Create the plot, then use plt.savefig

[20]:
```
from_proj = ccrs.PlateCarree()
to_proj = ccrs.PlateCarree()

levs = np.arange(0, 1.8, 0.1)

ax = plt.axes(projection=to_proj)
plt.contourf(lon_NPP, lat_NPP, AOD_NPP_HQ, levs, transform=from_proj)
ax.coastlines()
plt.colorbar(orientation="horizontal", fraction=0.07)

# Add DPI option to change resolution of the plot (maintains the same aspect␣
 ↪ratio)
plt.savefig('Wildfires_AOD_2018221_1612.png', dpi=500)

# TIP: If you want to suppress the output (runs faster!), use the following:
```

```
plt.close()
```

2.6 Out-of-memory processing

- Some datasets are so big that it is not a good idea to load them all into memory.
- xarray operations can be run on successive pieces of datasets so the entire data is never in memory. Operations will be run in parallel! See more details.
- We'll use the MERRA-2 reanalysis of AOD.

[21]:
```
filename = 'data/MERRA2_400.inst3_2d_gas_Nx.20181109.nc4.nc'
merra = xarray.open_dataset(filename, chunks={'time': 1})
```

[22]:
```
merra['AODANA']
```

[22]:
```
<xarray.DataArray 'AODANA' (time: 8, lat: 361, lon: 576)>
dask.array<shape=(8, 361, 576), dtype=float32, chunksize=(1, 361, 576)>
Coordinates:
  * lat      (lat) float64 -90.0 -89.5 -89.0 -88.5 -88.0 ... 88.5 89.0 89.5 90.0
  * lon      (lon) float64 -180.0 -179.4 -178.8 -178.1 ... 178.1 178.8 179.4
  * time     (time) datetime64[ns] 2018-11-09 ... 2018-11-09T21:00:00
Attributes:
    long_name:       Aerosol Optical Depth Analysis
    units:           1
    fmissing_value:  1000000000000000.0
    standard_name:   Aerosol Optical Depth Analysis
    vmax:            1000000000000000.0
    vmin:            -1000000000000000.0
    valid_range:     [-1.e+15  1.e+15]
    origname:        AODANA
    fullnamepath:    /AODANA
```

[23]:
```
# Let's do a time-mean

merra['AODANA'].mean(dim='time')
```

[23]:
```
<xarray.DataArray 'AODANA' (lat: 361, lon: 576)>
dask.array<shape=(361, 576), dtype=float32, chunksize=(361, 576)>
Coordinates:
  * lat      (lat) float64 -90.0 -89.5 -89.0 -88.5 -88.0 ... 88.5 89.0 89.5 90.0
  * lon      (lon) float64 -180.0 -179.4 -178.8 -178.1 ... 178.1 178.8 179.4
```

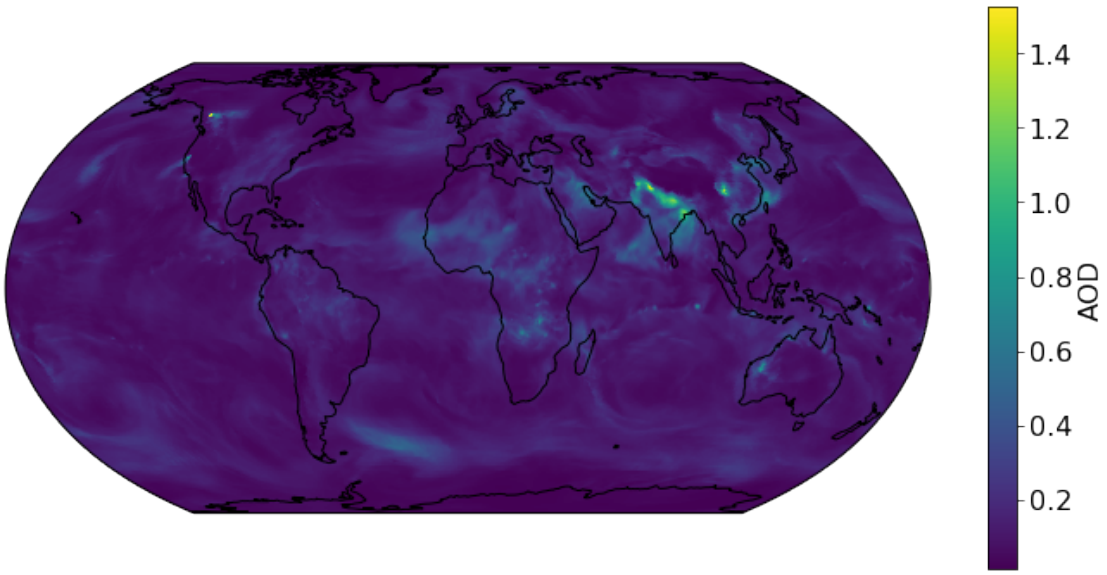
The mean does not compute! xarray will avoid doing the computation with out-of-memory datasets until you tell it to.

[24]:
```
mean_AOD = merra['AODANA'].mean(dim='time').compute()
mean_AOD
```

[24]:
```
<xarray.DataArray 'AODANA' (lat: 361, lon: 576)>
array([[0.045397, 0.045397, 0.045397, ..., 0.045397, 0.045397, 0.045397],
       [0.047764, 0.047781, 0.047797, ..., 0.047711, 0.047729, 0.047746],
       [0.049648, 0.049689, 0.049729, ..., 0.049524, 0.049565, 0.049607],
```

```
       ...,
       [0.050253, 0.05023 , 0.050207, ..., 0.050324, 0.0503  , 0.050277],
       [0.052478, 0.052453, 0.052429, ..., 0.052554, 0.052529, 0.052503],
       [0.054713, 0.054713, 0.054713, ..., 0.054713, 0.054713, 0.054713]],
      dtype=float32)
Coordinates:
  * lat      (lat) float64 -90.0 -89.5 -89.0 -88.5 -88.0 ... 88.5 89.0 89.5 90.0
  * lon      (lon) float64 -180.0 -179.4 -178.8 -178.1 ... 178.1 178.8 179.4
```

[25]:
```
from_proj = ccrs.PlateCarree()
to_proj = ccrs.EqualEarth()

# Using cartopy, create the map projection and plot the data
ax = plt.axes(projection=to_proj)
ax.coastlines()

# Can change color scheme using the cmap argument
plt.pcolormesh(merra['lon'], merra['lat'], mean_AOD, transform=from_proj)
plt.colorbar(label='AOD')

plt.show()
```



2.7 Scripting

- Notebooks are nice for sharing results with others, but scripts are useful for automating tasks.
- You can export a script from Jupyter Notebook directly: File Download As Python (.py)