day7_nb_full

May 23, 2022

1 Python Bootcamp: June 7th, 2022

1.1 Maps and Spatial Data

1.1.1 Samuel Mogen and Megan TM

1.2 Scientific Goals

- Understand different map projections
- Understand how data is projected

1.3 Coding Goals

- Review reading in data with Xarray
- Learn the basics of Cartopy
- Create good looking maps with Matplotlib and Cartopy

1.4 Notebook Outline

- 5.1: Reading netCDF files
- 5.2: Plotting spatial data without projections
- 5.3: Different map projections
- 5.4: Introduction to Cartopy
- 5.5: Map your own data
- 5.6: Subplots and saving figures

As a reminder: Text in black will be instruction and guidance and will usually start with a section number. Text in blue will be tasks to do. Text in red will be optional challenges and advanced concepts.

1.5 5.1: Reading in netCDF files with Xarray

[1]:
```
import xarray as xr
import numpy as np
import matplotlib.pyplot as plt
```

As a review, please make sure you have `pH.nc` downloaded and read it in using Xarray

hint: use `xr.open_dataset`

[2]:
```
ds = xr.open_dataset('pH.nc')
```

Check out the dimensions of the dataset…

[3]:
```
ds
```

[3]:
```
<xarray.Dataset>
Dimensions:  (lon: 360, lat: 180, time: 180)
Coordinates:
  * lon      (lon) float32 0.5 1.5 2.5 3.5 4.5 … 355.5 356.5 357.5 358.5 359.5
  * lat      (lat) float32 -89.5 -88.5 -87.5 -86.5 -85.5 … 86.5 87.5 88.5 89.5
  * time     (time) datetime64[ns] 1998-01-01 1998-02-01 … 2012-12-01
Data variables:
    pH       (time, lat, lon) float32 …
```

So we have a coordinate system of latitude by longitude by time…

How do we index to select a certain variable at some lat/lon/time?

Now try to index some data of this data

Select data for the pH of water…

[4]:
```
pH = ds.pH
```

Then, index this data so you only have the time from the year 1998

[5]:
```
pH_1998 = pH.sel(time=slice('1998-01-01','1998-12-01')).mean(dim='time')
```

now get a slice of data that corresponds to latitudes -65.5:65.5 and longitudes from 109.5:275.5

[6]:
```
pH_pac = pH_1998.sel(lat=slice(-65.5,65.5),lon=slice(109.5,288.5))
```

average this newly sliced data over the year and save out as a new variable!

[7]:
```
pH_pac_1998 = pH_pac
```

1.6 5.2: plot this data

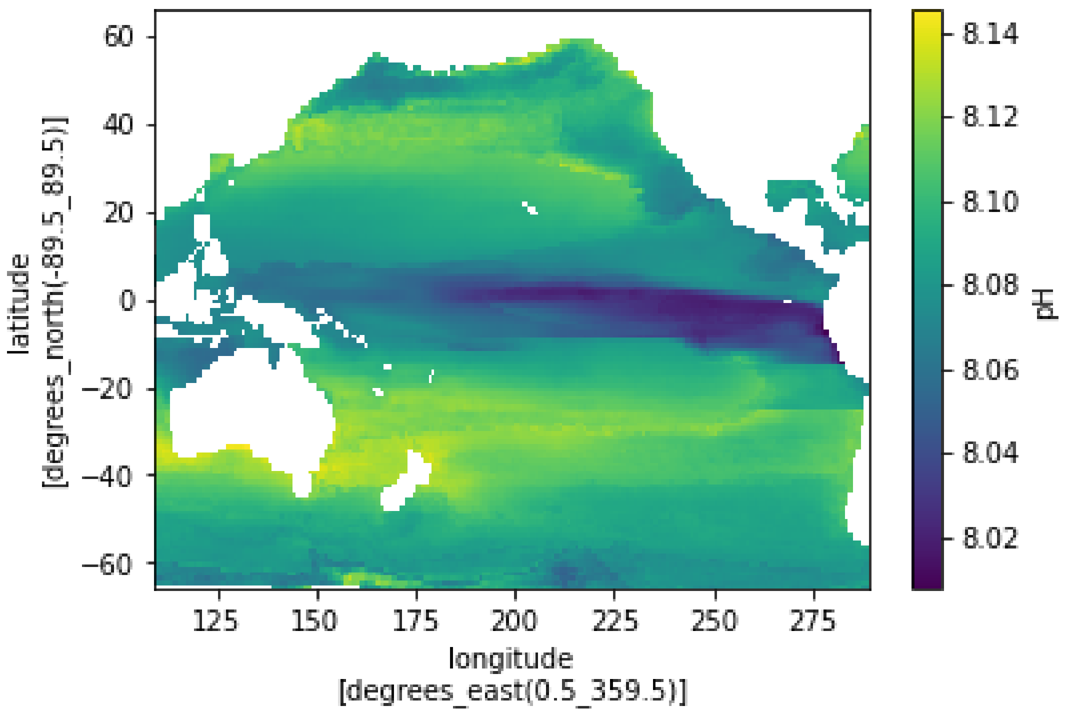
now that we know how to index this data by location and time, we want to start thinking about plotting the data!

Xarray makes it really easy to make basic plots with given data…

For example, you can simply say `ds.plot()` and Xarray will attempt to plot the data in a nice way

[8]:
```
pH_pac_1998.plot()
```

[8]:
```
<matplotlib.collections.QuadMesh at 0x7f9dc9c0ec70>
```

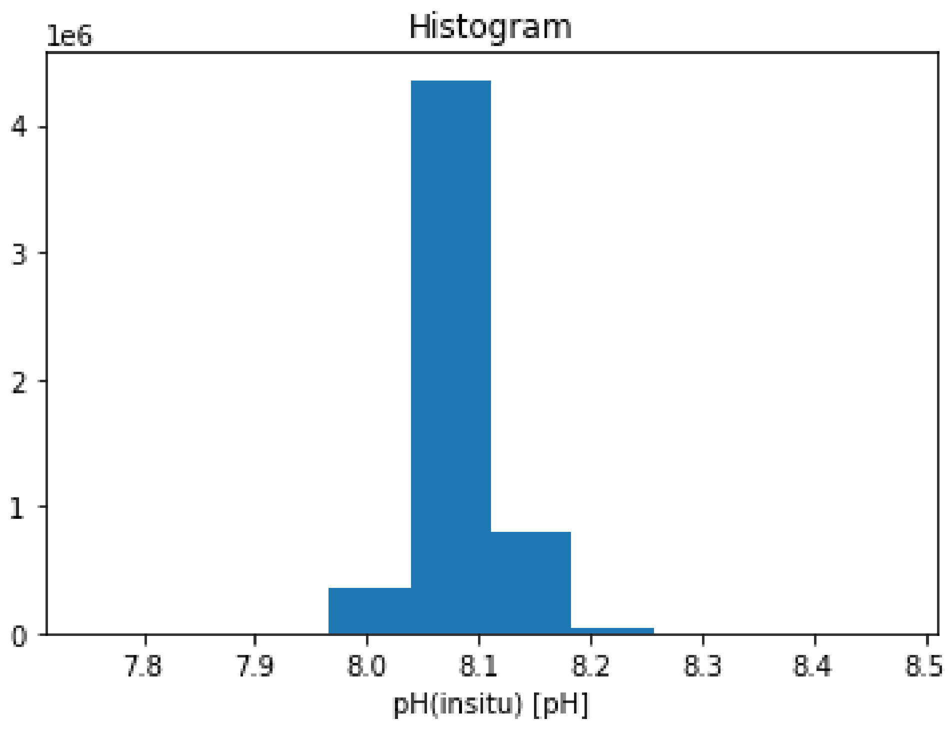


this works for data with clear dimensions… Our map here shows the portion of the Pacific that we sliced in the year 1998.

But when you give a datasets that is not as clear, you might get a weird plot…

```
[9]: ds.pH.plot()
```

```
[9]: (array([1.000000e+00, 2.000000e+00, 6.920000e+02, 3.472400e+05,
            4.355745e+06, 7.947550e+05, 3.241900e+04, 6.143000e+03,
            2.360000e+02, 2.400000e+01]),
      array([7.748    , 7.8207   , 7.8934   , 7.9661   , 8.0388   , 8.1115   ,
            8.1842   , 8.256901, 8.3296   , 8.4023   , 8.475    ], dtype=float32),
      <BarContainer object of 10 artists>)
```



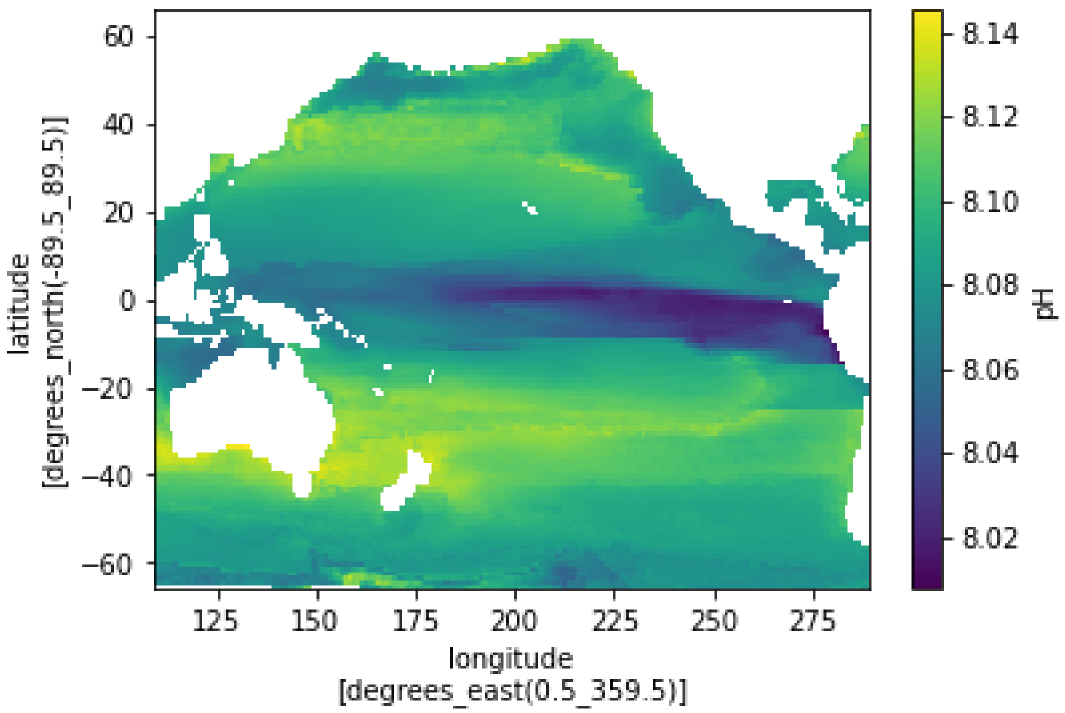
This histogram is **not** a useful way to look at information

Make sure to subset data correctly so you can use Xarray's plotting to quickly look through data

back to our previous map:

[10]:
```
pH_pac_1998.plot()
```

[10]:
```
<matplotlib.collections.QuadMesh at 0x7f9db36e9f10>
```



You can see clearly where the continent are supposed be! But they don't look very clean

We need to tell Python how we want this data to be presented.

1.7 5.3: Map projections

There are a variety of ways to present spatial data - maps can be projected in a number of ways. More information on map projections can be found here.

When is each projection useful?

1.8 5.4: Cartopy

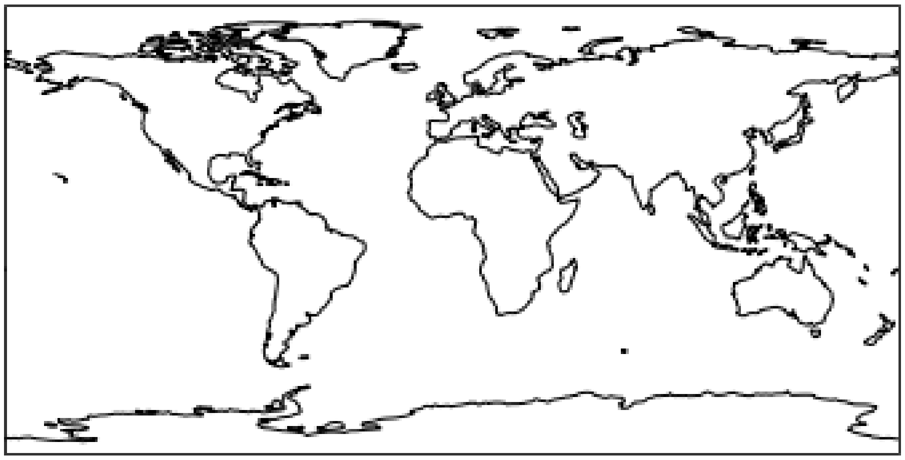
To make a base map look more professional, we are going to import a new package...

[11]:
```
import cartopy.crs as ccrs
import cartopy.feature as cfeature
```

Cartopy allows you plot a given spatial dataset on an actual map of the world (or specific region) so that it looks nice. It will work with all sorts of datasets and relies on the plotting power of matplotlib

[12]:
```
# To start, we will plot a blank map with the PlateCarree projection...
ax = plt.axes(projection=ccrs.PlateCarree())
ax.coastlines()
# at it's simplest, cartopy allows you to plot a map of the globe with no data
```

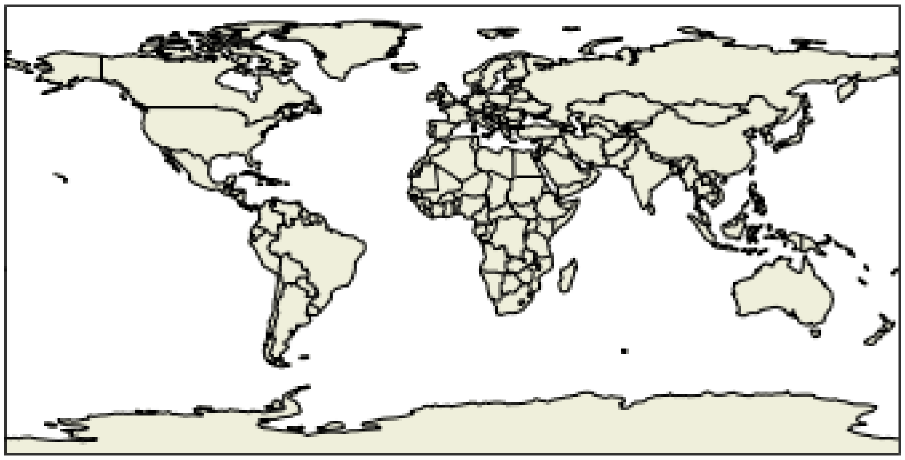
```
[12]: <cartopy.mpl.feature_artist.FeatureArtist at 0x7f9dae2d3400>
```



```
[13]: # aside from coastlines, there are various other features you can add to maps!
      ax = plt.axes(projection=ccrs.PlateCarree())

      ax.coastlines() # Coastlines
      ax.add_feature(cfeature.LAND) # Land
      ax.add_feature(cfeature.BORDERS) # National Border
```

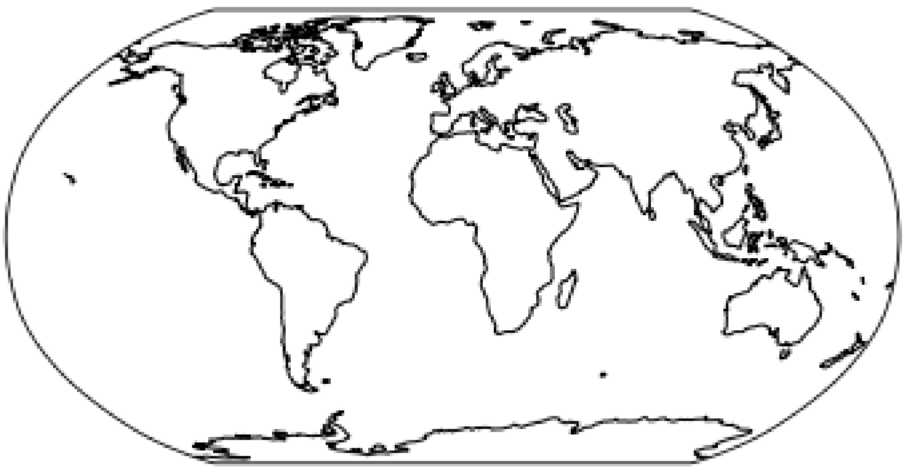
```
[13]: <cartopy.mpl.feature_artist.FeatureArtist at 0x7f9dae5e67f0>
```



As mentioned earlier, Cartopy has a number of different projections you can use.

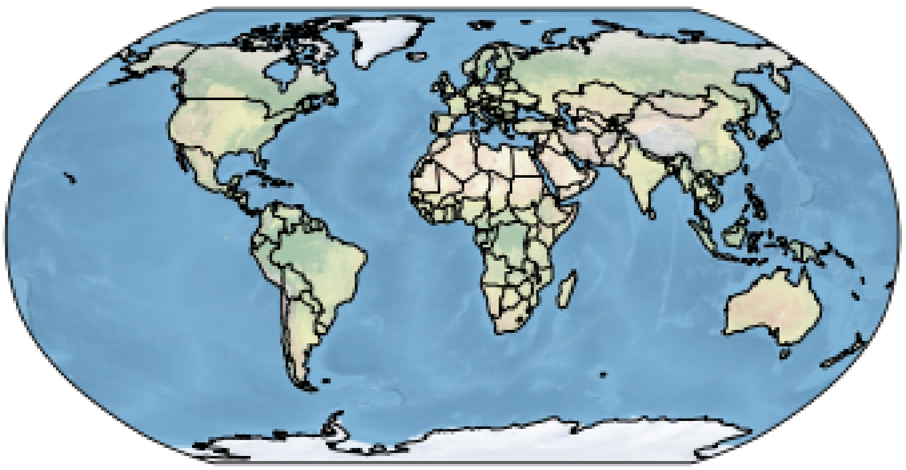
```
[14]: # plot a blank map with the Robinson projection
      ax = plt.axes(projection=ccrs.Robinson())
      ax.coastlines()
```

```
[14]: <cartopy.mpl.feature_artist.FeatureArtist at 0x7f9dae6f7a60>
```



```
[15]: # Cartopy also has built in base maps, that allow you plot what looks like the␣
       ↪Earth from space
      ax = plt.axes(projection=ccrs.Robinson())
      ax.stock_img() # satellite stock image of earth
      ax.coastlines() # Coastlines
      ax.add_feature(cfeature.BORDERS) # National borders
```

```
[15]: <cartopy.mpl.feature_artist.FeatureArtist at 0x7f9dae7bcfd0>
```



1.8.1 Breakout

Look at the projections list available in Cartopy and create some blank plots from 3 new projections

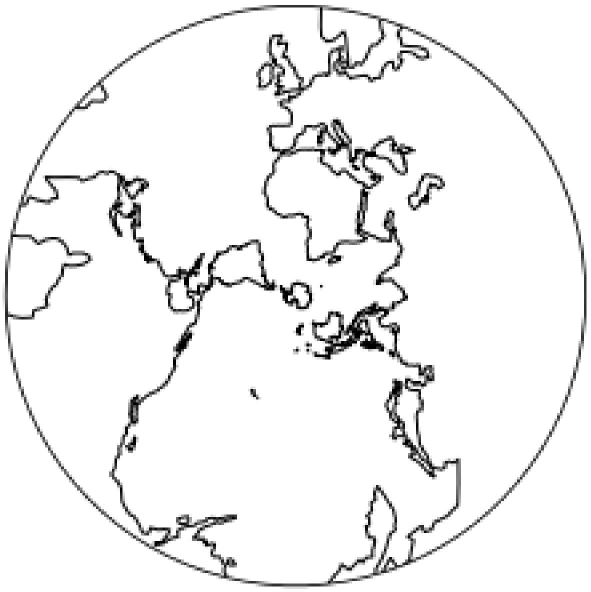
Use different projections and center the map at different points!

Plot using: - SouthPolarStereo - Orthographic - Projection of your choice

For each projection, **plot a map** centered on a longitude of your choosing and **explain when this projection might be useful**

[16]:
```
ax = plt.axes(projection=ccrs.SouthPolarStereo())
ax.coastlines()
```

[16]:
```
<cartopy.mpl.feature_artist.FeatureArtist at 0x7f9dae866cd0>
```

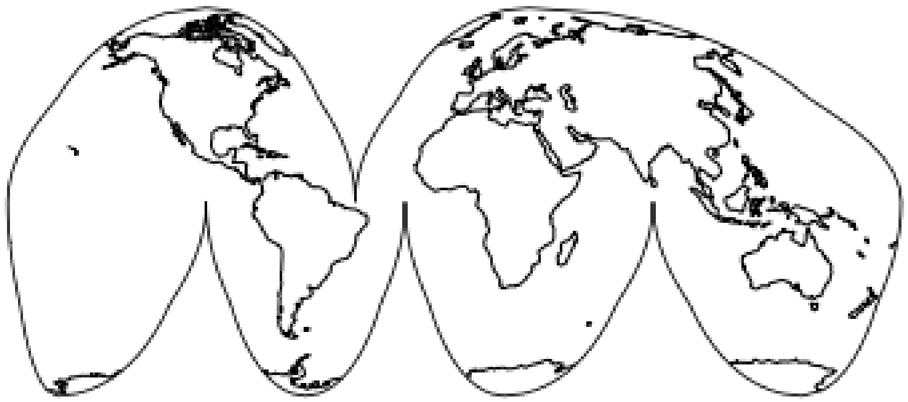


[17]:
```
ax = plt.axes(projection=ccrs.Orthographic(central_longitude=300))
ax.coastlines()
```

[17]:
```
<cartopy.mpl.feature_artist.FeatureArtist at 0x7f9dae9cdf70>
```

```
[18]: ax = plt.axes(projection=ccrs.InterruptedGoodeHomolosine())
      ax.coastlines()
```

```
[18]: <cartopy.mpl.feature_artist.FeatureArtist at 0x7f9dae877d60>
```



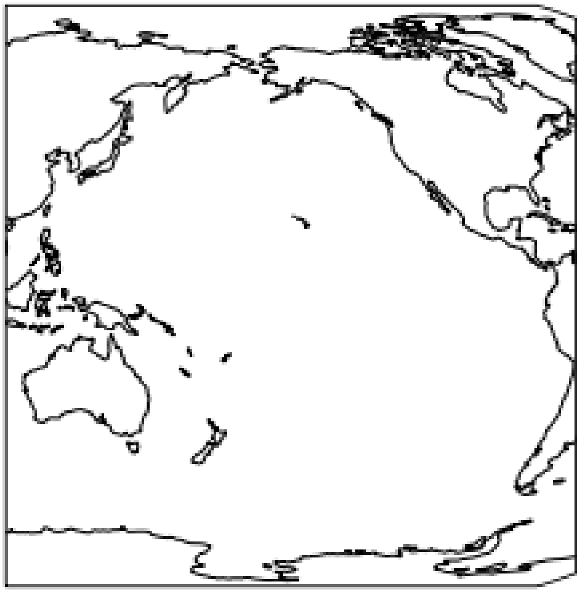
1.8.2 cartopy also allows you subset data to specific regions on maps themselves, meaning you don't necessarily need to subset the data itself by latitude and longitude...

[19]:
```
ax = plt.axes(projection=ccrs.Robinson(central_longitude=180))
# you can change the central longitude of the map

ax.coastlines()

# we earlier subset to latitudes -65.5:65.5 and longitudes from 109.5:275.5
# We can use ax.set_extent([x0,x1,y0,y1]) to replicate this without limiting␣
 ↪our data!

ax.set_extent([109.5,288.5,-65.5, 65.5])
```



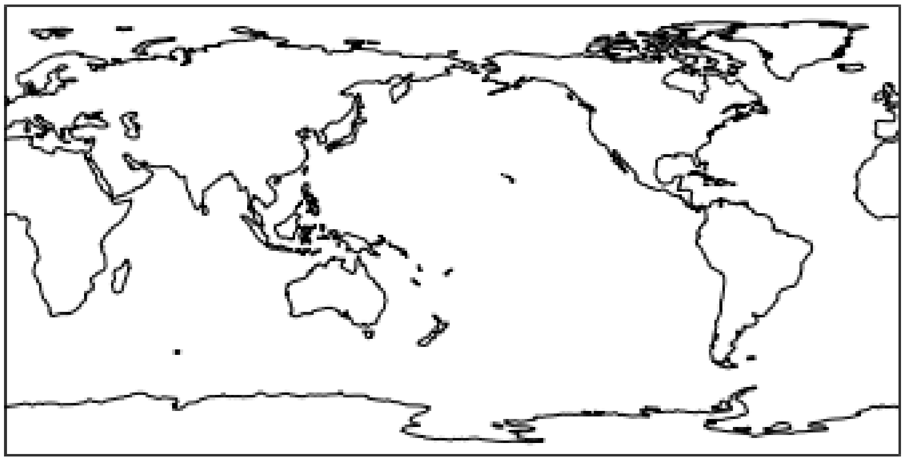
1.9 5.5: Add some of your own data to these

Now that we know how to make blank maps using Cartopy, how can we actually add our own data?

[20]:
```
ax = plt.axes(projection=ccrs.PlateCarree(central_longitude=180)) # you can␣
 ↪change the central longitude of the map
ax.coastlines()

# data???
```

[20]:
```
<cartopy.mpl.feature_artist.FeatureArtist at 0x7f9daee07f70>
```



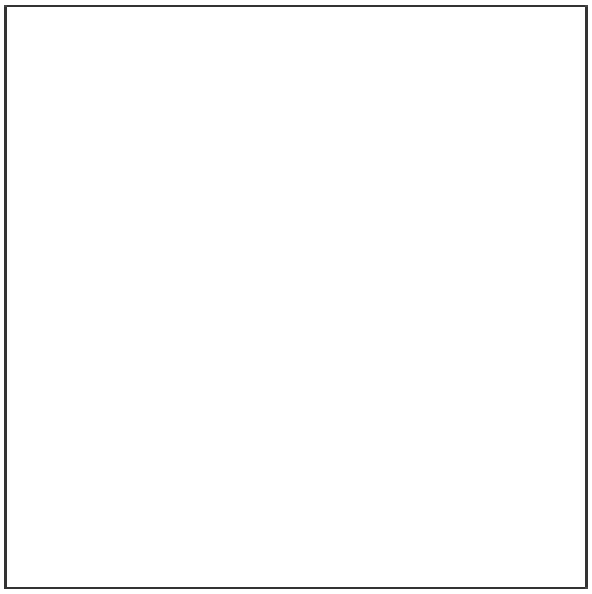
Try by plotting using the contourf feature

For example, you can call it using `ax.contourf()`

```
[21]: ax = plt.axes(projection=ccrs.PlateCarree(central_longitude=180)) # you can␣
       ↪change the central longitude of the map
      ax.coastlines()


      ax.contourf(pH_1998)
```

```
[21]: <cartopy.mpl.contour.GeoContourSet at 0x7f9daf0cfdf0>
```



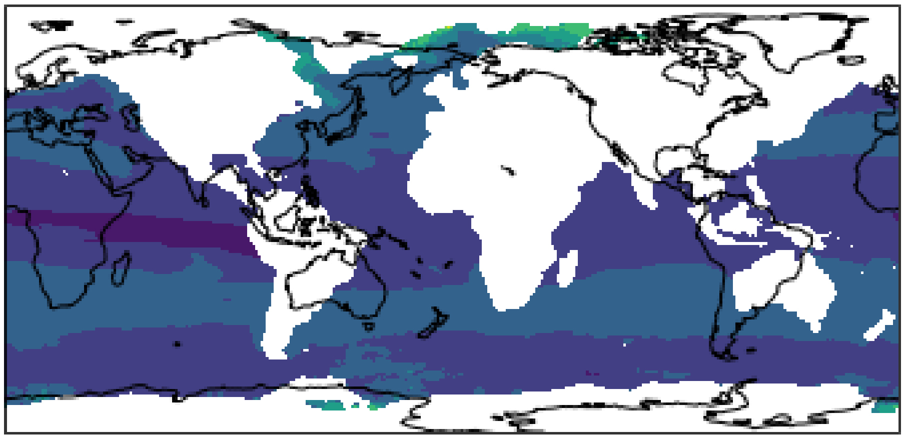
Without information on the latitude and longitude, Cartopy and the data have no idea how to interact

To plot effectively, you need to supply the function with: the longitude, latitude and data

[22]:
```
ax = plt.axes(projection=ccrs.PlateCarree(central_longitude=180)) # you can␣
↪change the central longitude of the map
ax.coastlines()

ax.contourf(pH_1998.lon,pH_1998.lat,pH_1998)
```

[22]:
```
<cartopy.mpl.contour.GeoContourSet at 0x7f9dafd1a490>
```



but this looks bad…

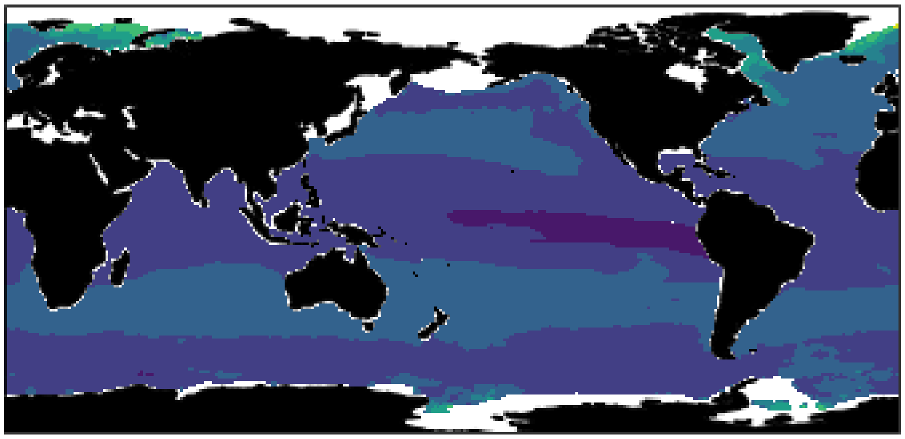
The data and cartopy have no clue how to interact!

You need to tell the contourf data how… use the transform = ccrs.Projection() command when plotting

[23]:
```
ax = plt.axes(projection=ccrs.PlateCarree(central_longitude=180)) # you can␣
↪change the central longitude of the map
ax.add_feature(cfeature.LAND,color='k')

ax.contourf(pH_1998.lon,pH_1998.lat,pH_1998,
            transform=ccrs.PlateCarree())
```

[23]:
```
<cartopy.mpl.contour.GeoContourSet at 0x7f9daf6b1490>
```



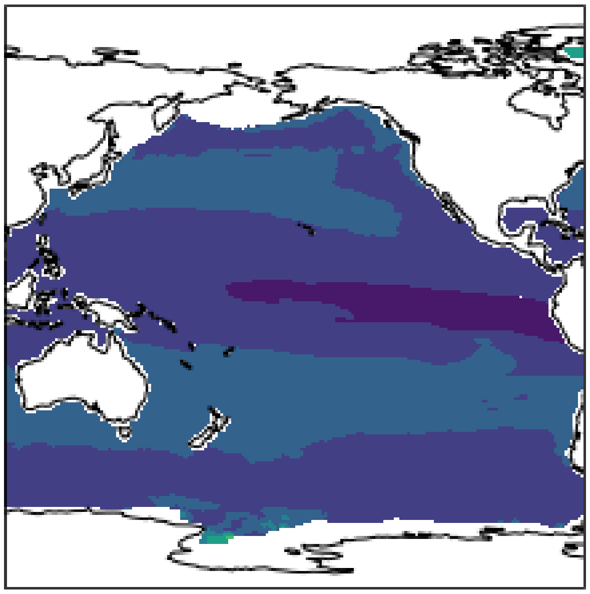
again, this data can be subset…

1.9.1 Breakout

Try subsetting data for yourself using `ax.set_extent()` - and compare to the plot of the subset data we made earlier!

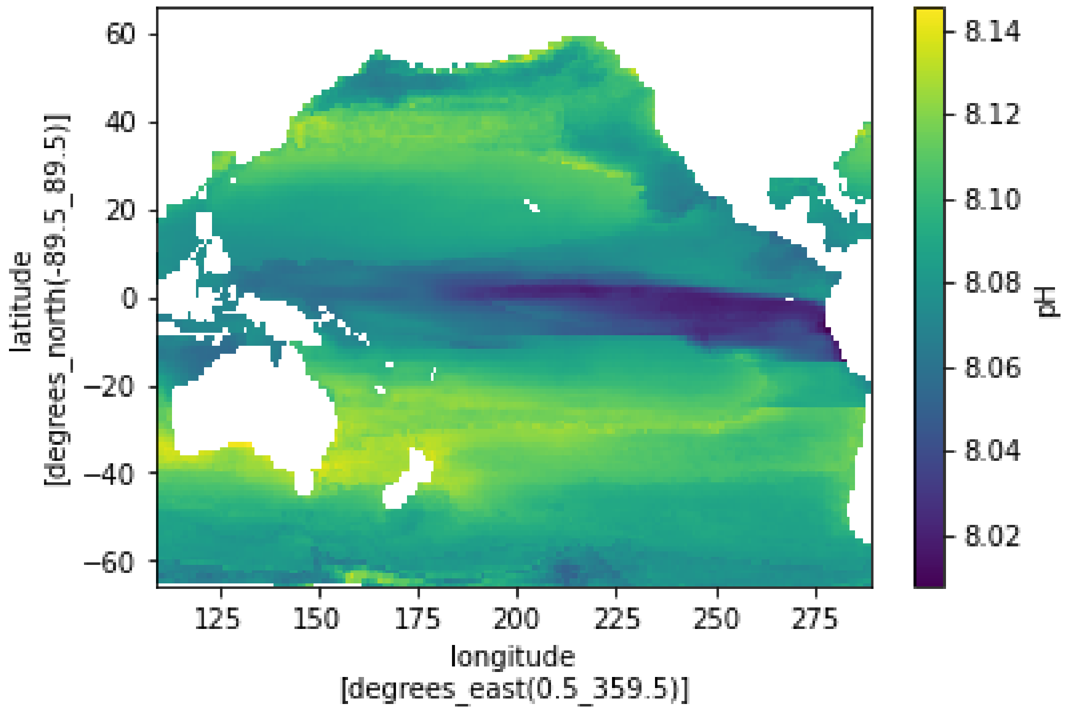
```
[24]: ax = plt.axes(projection=ccrs.PlateCarree(central_longitude=180)) # you can␣
       ↪change the central longitude of the map
      ax.coastlines()
      ax.contourf(pH_1998.lon,pH_1998.lat,pH_1998,transform=ccrs.PlateCarree())

      ax.set_extent([109.5,288.5,-65.5, 65.5]) # solution
```



```
[25]: # compared to the following plot, this looks so clean
      pH_pac_1998.plot()
```

```
[25]: <matplotlib.collections.QuadMesh at 0x7f9db0694130>
```



1.9.2 5.5.1: How can we locate a specific spot?

What happens if we want to mark the general location of, say, Hawaii on a map?

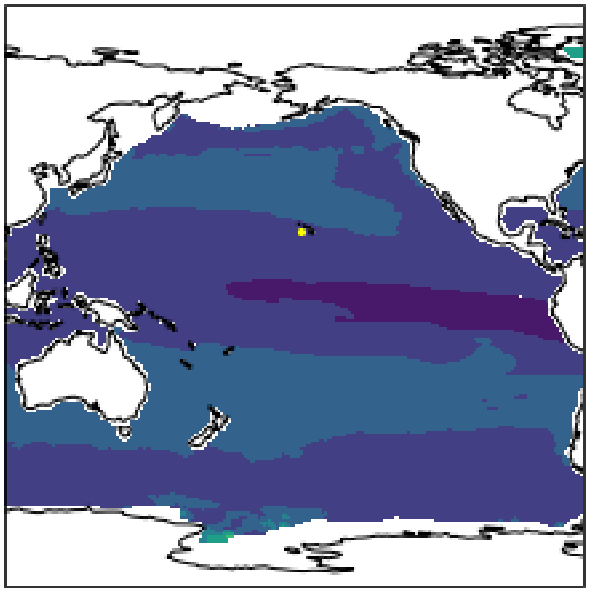
Hawaii (the Big Island) is located near latitude 19.5N and longitude 201E.

```
[26]: # set the latitude and longitude we want
      # (remember to convert the longitude) to degrees east!
      lat_hawaii = 19.5
      lon_hawaii = 201
```

```
[27]: ax = plt.axes(projection=ccrs.PlateCarree(central_longitude=180)) # you can␣
       ↪change the central longitude of the map
      ax.coastlines()
      ax.contourf(pH_1998.lon,pH_1998.lat,pH_1998,transform=ccrs.PlateCarree())

      ax.set_extent([109.5,288.5,-65.5, 65.5]) # solution
      ax.scatter(lon_hawaii, lat_hawaii,
                  s=4, # sets the markersize
                  color='Yellow', # sets the color
                  transform=ccrs.PlateCarree(), # sets the projection!
                  zorder = 10 # makes sure that this is plotted above other data!
                )
```

[27]:
```
<matplotlib.collections.PathCollection at 0x7f9db0574910>
```



1.10 5.6: Colormaps, Subplots, and Saving figures

5.6.1: Colormaps You may have noticed that plotting will choose its own color scheme for data. Oftentimes, we want to control the colors that are used and the range of values that are displayed - to do this, we will use the `cmap =` and `levels =` commands…

A list of available base colormaps can be seen in the Matplotlib documentation here

also include the `figsize = (width,height)`. `figize = (1,1)` would created a 1 inch by 1 inch plot

The data we are working with today seems most suited for sequential colormaps. Choose the colormap `Reds` from below and use that in plotting…

- Plot `pH_1998` with the `cmap = 'Reds'`. Also include a specific `figsize`.
- Include a colorbar in this plot (hint: the colorbar needs to know what plot it is referencing)

[28]:
```
f, ax = plt.
↪subplots(nrows=1,ncols=1,figsize=(6,6),subplot_kw=dict(projection=ccrs.
↪PlateCarree(central_longitude=180)))
```

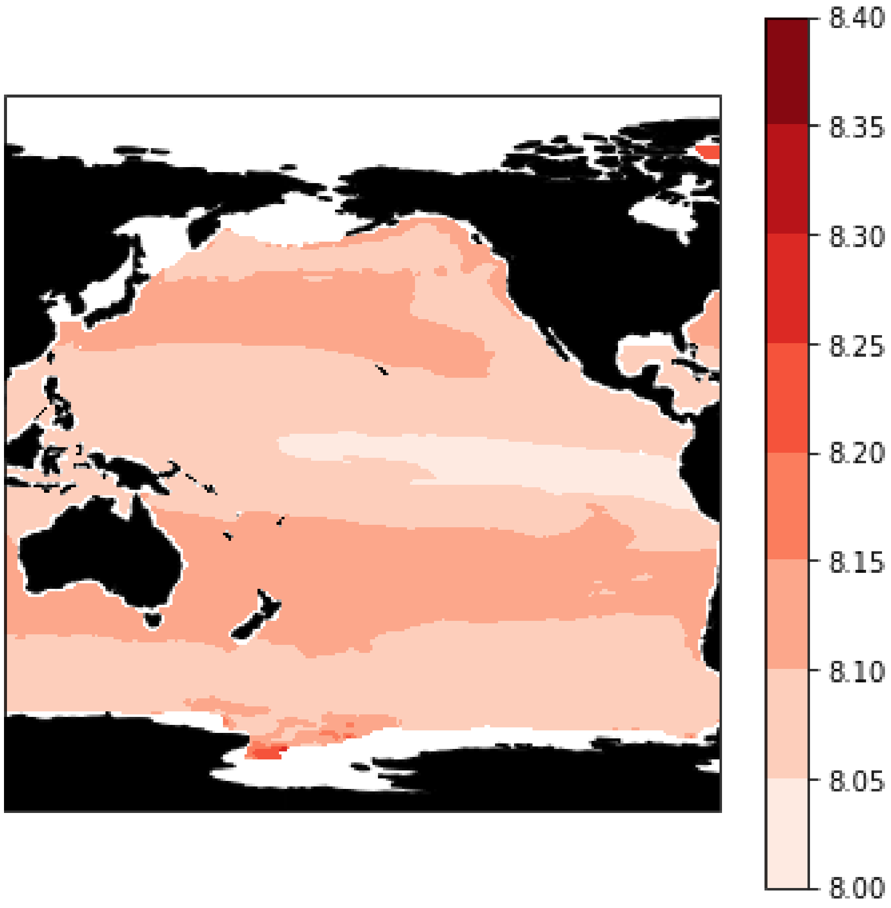
```
ax.add_feature(cfeature.LAND,color='k')

im = ax.contourf(pH_1998.lon,pH_1998.lat,pH_1998,
                 transform=ccrs.PlateCarree(),
                 cmap = 'Reds')

ax.set_extent([109.5,288.5,-65.5, 65.5])

f.colorbar(im)
```

```
[28]: <matplotlib.colorbar.Colorbar at 0x7f9db0a8ce50>
```



Because we want lower values of pH to pop out, lets reverse the Reds (so lower values are **more** red…

Change `Reds` to `Reds_r`

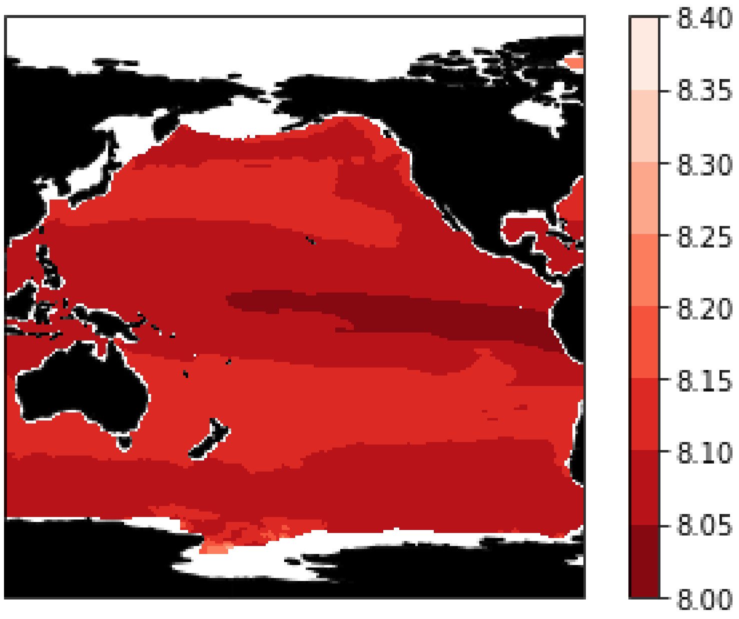
```
[29]: f, ax = plt.subplots(nrows=1,ncols=1,subplot_kw=dict(projection=ccrs.
       ↪PlateCarree(central_longitude=180)))
```

```
ax.add_feature(cfeature.LAND,color='k')

im = ax.contourf(pH_1998.lon,pH_1998.lat,pH_1998,
                 transform=ccrs.PlateCarree(),
                 cmap = 'Reds_r')
ax.set_extent([109.5,288.5,-65.5, 65.5])

f.colorbar(im)
```

[29]:
```
<matplotlib.colorbar.Colorbar at 0x7f9db0e3b7f0>
```



We also want to control exactly where the cutoffs are for the data using the levels command and extend the colorbar so values outside of the range are plotted correctly!

Use `levels = np.arange(lower_bound, upper_bound, step)` and `extend = 'both'`

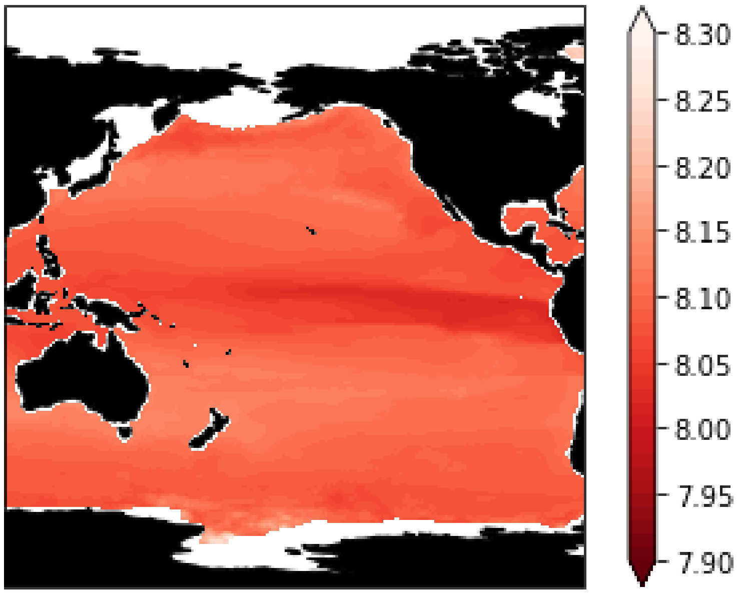
[30]:
```
f, ax = plt.subplots(nrows=1,ncols=1,subplot_kw=dict(projection=ccrs.
 ↪PlateCarree(central_longitude=180)))
ax.add_feature(cfeature.LAND,color='k')

im = ax.contourf(pH_1998.lon,pH_1998.lat,pH_1998,
                 transform=ccrs.PlateCarree(),
                 cmap = 'Reds_r',
                 levels = np.arange(7.9,8.3,0.01),
                 extend = 'both')

ax.set_extent([109.5,288.5,-65.5, 65.5])
```

```
f.colorbar(im)
```

[30]:
```
<matplotlib.colorbar.Colorbar at 0x7f9db12c67c0>
```



5.6.2: The magic of `.groupby()` and subplots

1.10.1 Breakout

how would you make group data different seasons and make subplots?

You can use the following command to group data by seasons (Xarray is great!)

`pH.groupby('time.season').mean()`

You can then index this data, to get global values for each season

Helpful hint, subplots can be created with:

`f, axs = plt.subplots(nrows= ? ,ncols= ?,subplot_kw=dict(projection= ?? ))`

Where each sublot from 0 to n can be called with:

`axs[n].contourf(...)`

Using this grouping, make a plot with four subplots showing pH in the Pacific for each season (hint: use a for loop in calling the plotting functions to loop through the seasons and subplots).

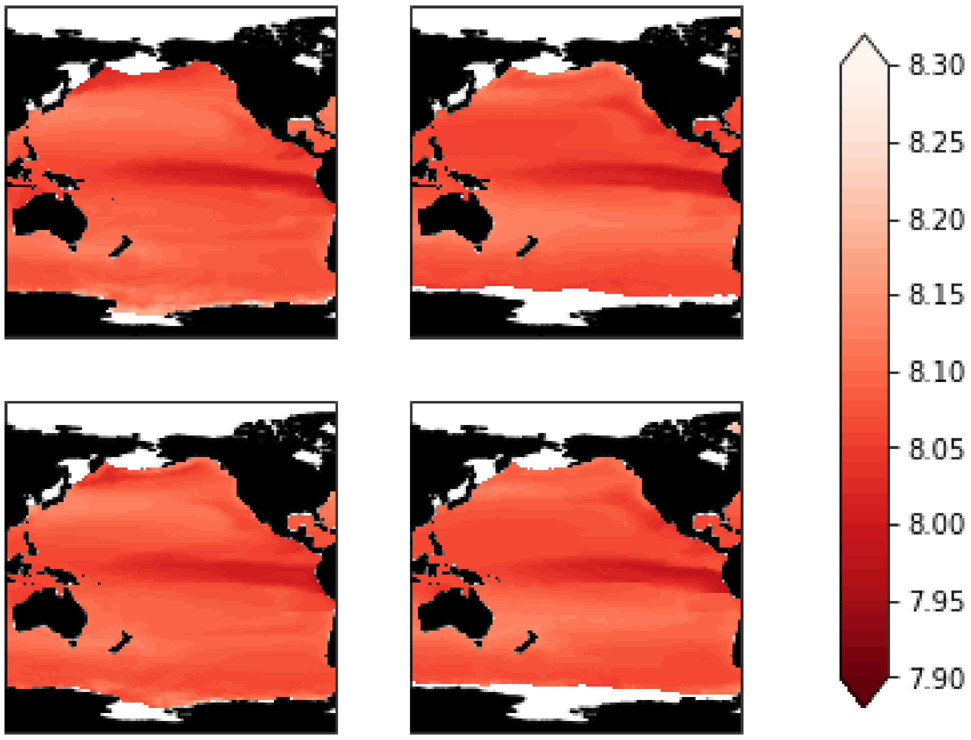
[31]:
```
seasons = pH.groupby('time.season').mean()
```

```
[32]: f, axs = plt.
       ↪subplots(nrows=2,ncols=2,figsize=(5,5),subplot_kw=dict(projection=ccrs.
       ↪PlateCarree(central_longitude=180)))
      axs = axs.flatten()

      for i in range(0,4):
          im = axs[i].contourf(seasons.lon,seasons.lat,seasons.isel(season=i),
                               transform=ccrs.PlateCarree(),
                               cmap = "Reds_r",
                               levels=np.arange(7.9, 8.3, 0.01),
                               extend = 'both')
          axs[i].set_extent([109.5,288.5,-65.5, 65.5])
          axs[i].add_feature(cfeature.LAND,color='k')

      cbar_ax = f.add_axes([1, 0.15, 0.05, 0.7])
      f.colorbar(im,cax=cbar_ax,fraction=0.046,pad=0.04)
```

```
[32]: <matplotlib.colorbar.Colorbar at 0x7f9db16a29d0>
```



5.6.3: Saving figures Use the function `f.savefig(dpi = )` to save a plot as an image. DPI refers to 'dots per inch', which basically controls how the file being saved is.

```
[ ]: f, axs = plt.
     ↪subplots(nrows=2,ncols=2,figsize=(5,5),subplot_kw=dict(projection=ccrs.
     ↪PlateCarree(central_longitude=180)))
     axs = axs.flatten()

     for i in range(0,4):
         im = axs[i].contourf(seasons.lon,seasons.lat,seasons.isel(season=i),
                              transform=ccrs.PlateCarree(),
                              cmap = "Reds_r",
                              levels=np.arange(7.9, 8.3, 0.01),
                              extend = 'both')
         axs[i].set_extent([109.5,288.5,-65.5, 65.5])
         axs[i].add_feature(cfeature.LAND,color='k')

     cbar_ax = f.add_axes([1, 0.15, 0.05, 0.7])
     f.colorbar(im,cax=cbar_ax,fraction=0.046,pad=0.04)

     f.savefig("test_file_save.png",facecolor='white',dpi = 100)
```