Bathymetry

We've just seen that xarray can 'play nice' with cartopy. Guess what format most bathymetry files are given in? Netcdf. That is nice, so we know how to make maps with bathmetry already

ETOPO:

Etopo is a global dataset with both ocean depth and land elevations. It comes in two different resolutions, and is generally a workhorse for ocean bathymetry maps. If you are working in a special region you may need to find you own special bathymetry data, which someone you work with probably has.

2 minute data: https://www.ngdc.noaa.gov/mgg/global/etopo2.html

download it for yourself: https://www.ngdc.noaa.gov/mgg/global/relief/ETOPO2/ETOPO2v2-2006/ETOPO2v2c/netCDF/ETOPO2v2c_f4_netCDF.zip

1 minute data: https://www.ngdc.noaa.gov/mgg/global/global.html

fname = https://gamone.whoi.edu/thredds/dodsC/usgs/data0/bathy/ETOPO2v2c_f4.nc.html

In [1]:
```
# import statements first

import xarray as xr
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
import cartopy.feature as cfeature

%matplotlib inline
```

load the bathymetry data:

you can do this using the weblink I found for someone at whoi, or you can just get the file onto your computer somewhere (I recommend getting it yourself so you can use it anywhere). There's also a version in the data folder that we'll use.

option1: load from website:

```
fname =
'https://gamone.whoi.edu/thredds/dodsC/usgs/data0/bathy/ETOPO2v2c_f4.nc'

ds_etpo = xr.open_dataset( fname)
ds_etpo
```

option 2: load from local file:

In [2]:
```
# option2: load from local file
```

```
fname = '../data/ETOPO2v2c_f4.nc'

ds_etpo = xr.open_dataset(fname)

ds_etpo
```

```
Out[2]: xarray.Dataset

▶ Dimensions:          (x: 10800, y: 5400)
▼ Coordinates:
    x                  (x)     float32  -180.0 -179.9 ... 179.9 180.0
    y                  (y)     float32  -89.98 -89.95 ... 89.95 89.98
▼ Data variables:
    z                  (y, x)  float32  ...
▼ Attributes:
    Conventions :      COARDS
    title :
    source :                          -Rd -I2m -ZTLf
    node_offset :      1
```

select a subset, a region of interest, and plot the bathymetry

Lets make a simple plot of the Mid Atlantic Bight

Because the global bathymetry dataset can be really huge, we want to make sure we take a subset before we try to plot:

```
region = ds_etpo.sel(x=slice(-77, -70), y=slice(35, 43))
```

define the map (figure) projection:

```
proj = ___.Mercator()
```

and the data projection for the transformation:

```
data_crs = ___.PlateCarree()
```

set up your figure:

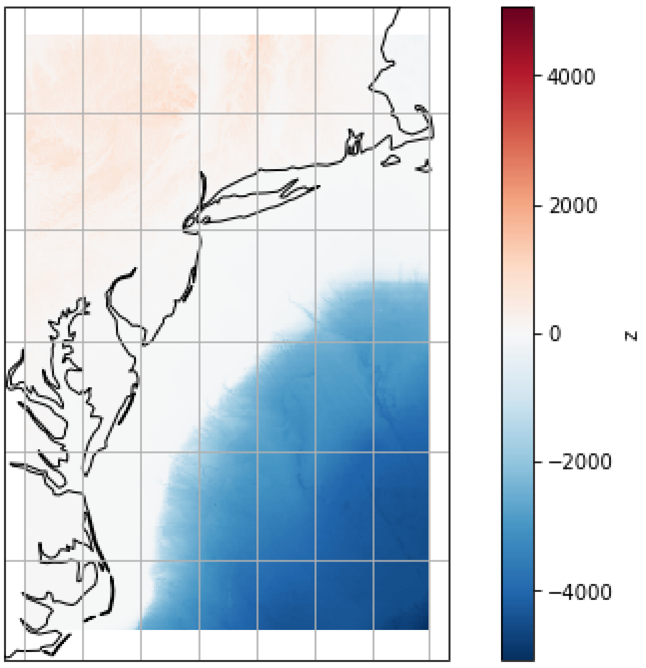
```
fig = plt.figure(figsize=(9,6))

ax = plt.axes(projection=___)

___.coastlines(resolution='50m')

ax.gridlines(draw_labels=True)
```

use xarray to make a simple plot

```
region._.plot( ___ =data_crs )
```

In [22]:
```
region = ds_etpo.sel(x=slice(-77, -70),y=slice(35,43))
proj = ccrs.Mercator()
data_crs = ccrs.PlateCarree()

fig = plt.figure(figsize=(9,6))
ax = plt.axes(projection=proj)
ax.coastlines(resolution='50m')
ax.gridlines()

region.z.plot(transform=data_crs)
# region.z.plot(transform=data_crs, vmin=-3000, vmax =0)
```

Out[22]:
```
<matplotlib.collections.QuadMesh at 0x7f90505a09d0>
```



change the plot to be the southern coast of greenland:

```
region = ds_etpo.sel(x=slice(-48.75, -40), y=slice(59, 61.5))
```

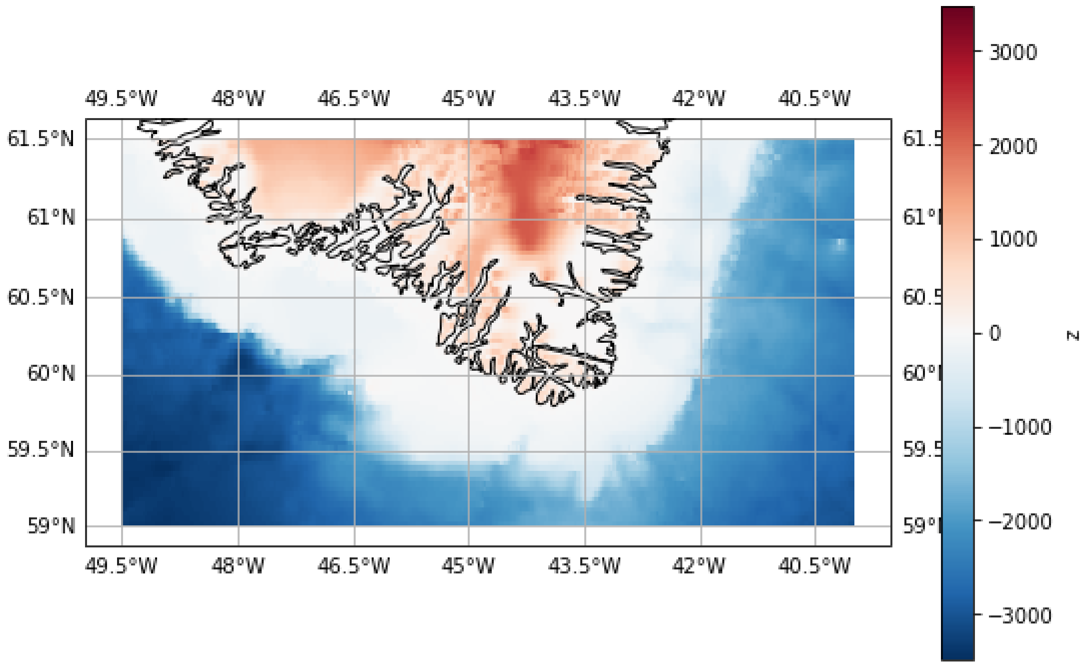
make the same map:

In [24]:
```
region = ds_etpo.sel(x=slice(-49.5, -40), y=slice(59, 61.5))

proj = ccrs.Mercator()
data_crs = ccrs.PlateCarree()

fig = plt.figure(figsize=(9,6))
ax = plt.axes(projection=proj)
ax.coastlines(resolution='10m')
ax.gridlines(draw_labels=True)
```

```python
region.z.plot( transform =data_crs )
```

Out[24]:

```
<matplotlib.collections.QuadMesh at 0x7f906188abe0>
```



use matplotlib to make a map, not xarray default.

Our bathymetry maps could use a little help. Lets make a map that has depth contours, which is the way we typically see maps. To do this we will use a standard matplotlib function called `plt.contourf()` that makes filled contours. We just need to give it the x, y, z data, and a list of contours we want to draw (i.e. isobaths). Then we need to pass it the `transform=` command since we are working with a cartopy map

Define the map and data projections, get the same region of the etopo bathymetry and create the figure.

Now through we are going to use matplotlib:

```python
# define the isobaths I want to plot:
lvls = [-4000, -3000, -2500, -2000, -1500, -1000, -700, -400, -145, -10,
0]

plt.contourf( ___.x, ___.y, ___.z, levels=lvls, ___=data_crs ,
cmap='Blues_r')
```

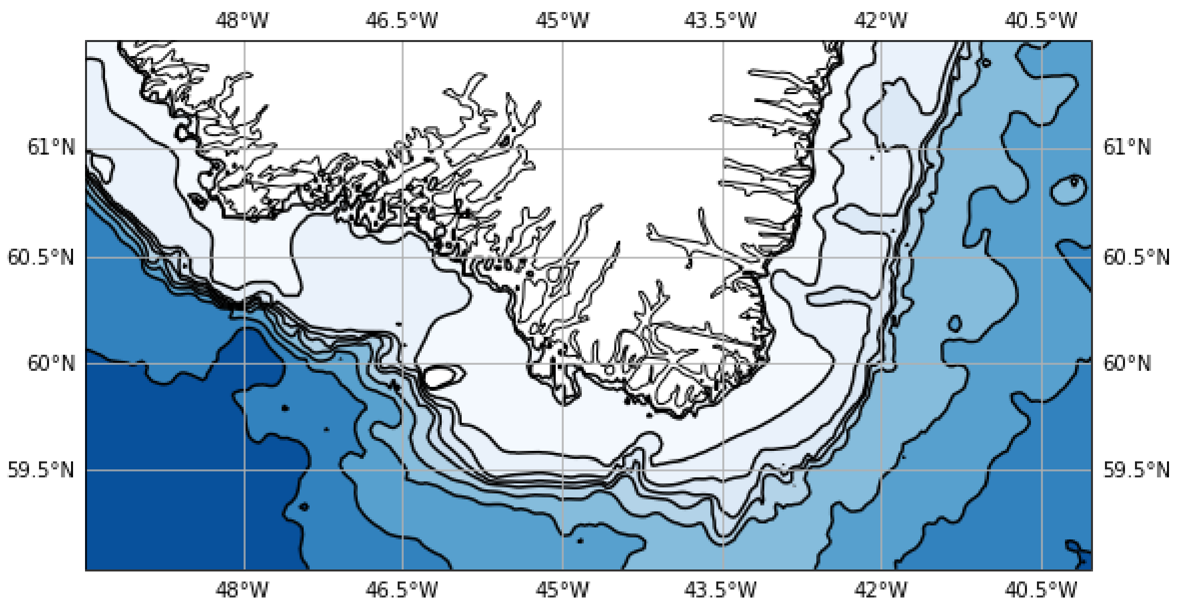
In [31]:

```python
proj = ccrs.Mercator()
data_crs = ccrs.PlateCarree()

# define the isobaths I want to plot:
lvls = [-4000, -3000, -2500, -2000, -1500, -1000, -700, -400, -145, -10, 0]

fig = plt.figure(figsize=(9,6))
ax = plt.axes(projection=proj)
```

```
ax.coastlines(resolution='10m')
ax.gridlines(draw_labels=True)

plt.contourf( region.x, region.y, region.z, levels=lvls, transform=data_crs , cm
plt.contour( region.x, region.y, region.z, levels=lvls, transform=data_crs , col
```

Out[31]:

```
<cartopy.mpl.contour.GeoContourSet at 0x7f9072154d90>
```



finally, plot some data on a map:

Note that we are still just using matplotlib to plot things. The only extra thing we are doing is passing the `transform=` argument that tells the figure how to place our data onto a map. So we should just be able to make any normal plots we want and add data to the map.

I've put a csv file in the data folder that has lat, lon, and average 0-500m helium observations from a cruise a few years ago. You can think of this like the meltwater content in the water column. Let's plot these data on the map. We will use pandas to read the csv file, then use `plt.scatter()` to plot them

```
df = pd.read_csv('../data/OSNAP_helium.csv')
df.head()
```

In [32]:

```
# import pandas and use pd.read_csv()
import pandas as pd

df = pd.read_csv('../data/OSNAP_helium.csv')
df.head()
```

Out[32]:

	Unnamed: 0	lon	lat	meanHE
0	0	-41.420667	61.156500	1.810445e-09
1	1	-41.481667	61.165167	1.818041e-09
2	2	-41.545333	61.173500	1.819294e-09

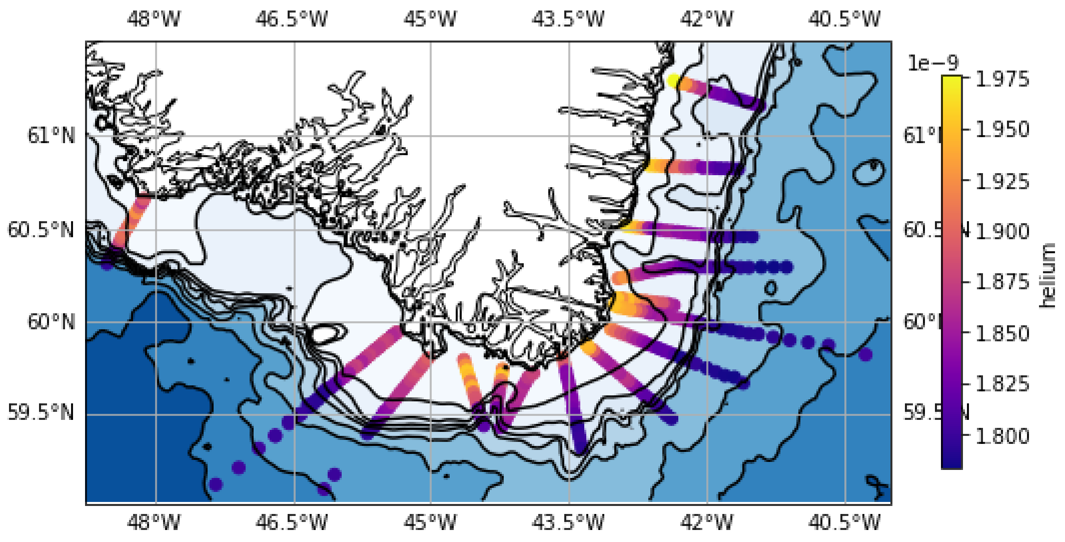
	Unnamed: 0	lon	lat	meanHE
3	3	-41.606667	61.183500	1.815365e-09
4	4	-41.670667	61.192500	1.808523e-09

now set up the same figure again, and fill in the blanks below to plot the meltwater data

```
plt.scatter(___, ___,  c=___, ___=data_crs, cmap='plasma')
ax.set_extent([-48.75, -40, 59, 61.5])
```

In [37]:
```
proj = ccrs.Mercator()
data_crs = ccrs.PlateCarree()

# define the isobaths I want to plot:
lvls = [-4000, -3000, -2500, -2000, -1500, -1000, -700, -400, -145, -10, 0]

fig = plt.figure(figsize=(9,6))
ax = plt.axes(projection=proj)
ax.coastlines(resolution='10m')
ax.gridlines(draw_labels=True)

plt.contourf( region.x, region.y, region.z, levels=lvls, transform=data_crs , cm
plt.contour( region.x, region.y, region.z, levels=lvls, transform=data_crs , col

plt.scatter(df.lon, df.lat,  c=df.meanHE, transform=data_crs, cmap='plasma')
ax.set_extent([-48.75, -40, 59, 61.5])

plt.colorbar(shrink=0.6, label='helium')
```

Out[37]:
```
<matplotlib.colorbar.Colorbar at 0x7f90318da850>
```



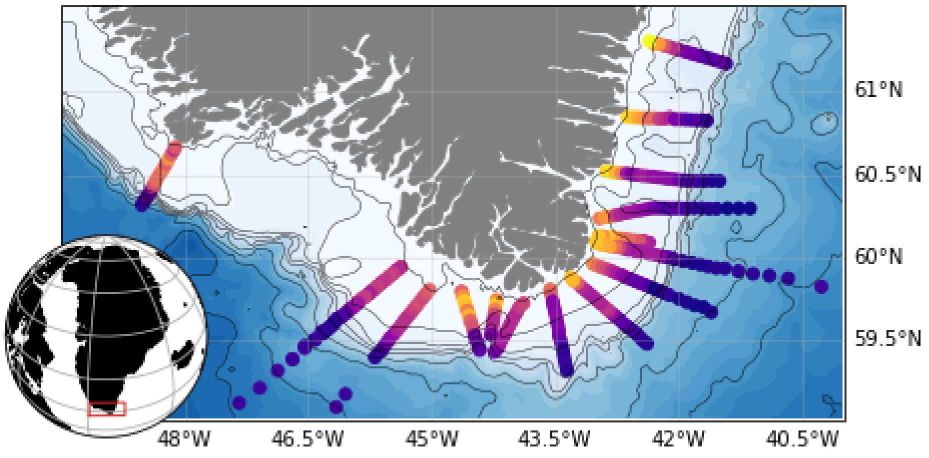
slightly nicer version

Now you should have the tools to make a simple map of your data anywhere in the world. You can make these maps infinitly complex and nice looking, but by analogy with what we have done so far you should be able to get started.

Just for fun the code below is almost like what we have done, but is more how I would make the map above for a paper or a talk, it's just a little bit nicer:

In [46]:
```
fig, ax = plt.subplots( figsize=(7, 7), subplot_kw=dict(projection=ccrs.Mercator

ax.set_extent([-49.5, -40, 59, 61.5],crs=ccrs.PlateCarree())

lvl = [-4000, -3000, -2500, -2000, -1500, -1000, -700, -400, -145, -15]

feature = ax.add_feature(cfeature.NaturalEarthFeature('physical', 'land', '10m',

ax.contourf(region.x, region.y, region.z,levels=50,  cmap='Blues_r' ,  transform
ax.contour(region.x, region.y, region.z, levels= lvl, colors='k', linestyles='so

glb = ax.gridlines(draw_labels=True, alpha=0.5, linewidth=.5)
glb.xlabels_top = glb.ylabels_left = False

plt.scatter(df.lon, df.lat,  c=df.meanHE, transform=data_crs, cmap='plasma')

# add an inset figure for context
sub_ax = fig.add_axes([0.07, 0.28, 0.2, 0.2],
                      projection=ccrs.NearsidePerspective(central_longitude=-45,

sub_ax.plot([-49.5, -49.5,  -40, -40, -49.5] ,
                        [ 59, 61.5, 61.5, 59, 59 ], color='tab:red',linewidth=1,
sub_ax.set_global()
LAND = cfeature.NaturalEarthFeature(
    'physical', 'land', '50m',
    edgecolor='face',
    facecolor='black')
sub_ax.add_feature(LAND, zorder=0)
sub_ax.gridlines(linewidths=.5)
```

Out[46]:
```
<cartopy.mpl.gridliner.Gridliner at 0x7f90630e68e0>
```



In [ ]: