Melville_floats_plot

March 22, 2023

1 Plot Melville Bay APEX float data: 2020-2022

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from pathlib import Path
import xarray as xr
from matplotlib import cm
import matplotlib as mpl
import matplotlib.colors as mcolors
import cartopy.crs as ccrs
import cmocean
from matplotlib.colors import ListedColormap
```

Three floats were deployed in Melville Bay (2 APEX, 1 ALAMO). The Alamo float only obtained 28 dives in less than a month in 2017. Here, I only use the APEX floats since they provide > 2 years of data from 2020-2022.

APEX Float F9185 sampling period: **October 22, 2020 - December 16, 2021** APEX Float F9444 sampling period: **September 20, 2021 - November 15, 2022**

First, linearly interpolate lat/lon locations for gaps in GPS fixes

[2]:
```
# Read in APEX float datasets
apex_dir = Path('../../../data/OMG_Float_data/OMG_APEX_Float_Data/APEX')
F9185_data_ds = xr.open_dataset(apex_dir / 'F9185/APEX_F9185_profiles_binned.
 ↪nc')
F9444_data_ds = xr.open_dataset(apex_dir / 'F9444/APEX_F9444_profiles_binned.
 ↪nc')
```

[3]:
```
# Fill any gaps in profiles
# these gaps are infrequent small gaps in the profile (e.g., when there was a␣
 ↪brief satellite transmission issue)
float_data_ds_fill = F9185_data_ds.interpolate_na(dim='depth_bins')
# swap dims with dives and date
F9185_data_ds_fill = float_data_ds_fill.swap_dims({'dive':'date'})

# do the same for other float
float_data_ds_fill = F9444_data_ds.interpolate_na(dim='depth_bins')
```

```
# swap dims with dives and date
F9444_data_ds_fill = float_data_ds_fill.swap_dims({'dive':'date'})
```

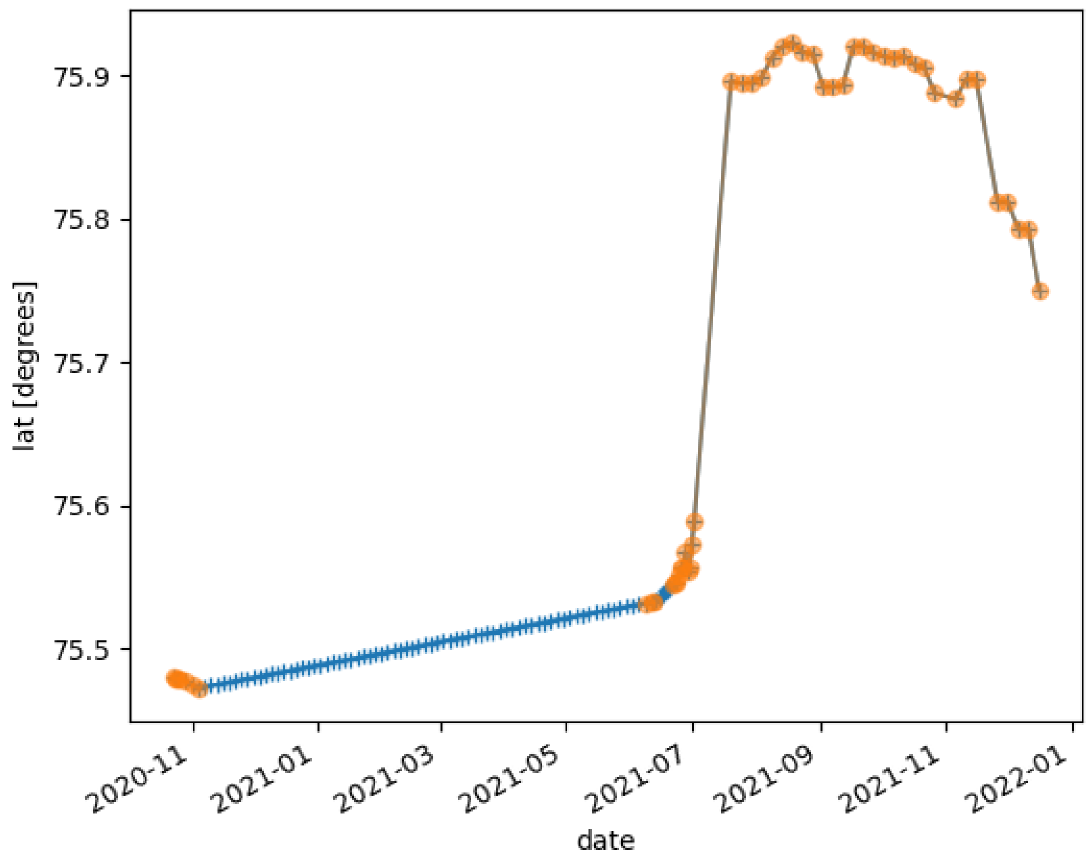
[4]:
```
# print start/end times for each float
print(F9185_data_ds_fill.date[0].values)
print(F9185_data_ds_fill.date[-1].values)

print(F9444_data_ds_fill.date[0].values)
print(F9444_data_ds_fill.date[-1].values)
```

```
2020-10-22T23:40:52.000000000
2021-12-16T00:04:46.000000000
2021-09-20T09:08:43.000000000
2022-11-15T04:46:08.000000000
```

Create new coordinate for lat/lon that includes linearly interpolated locations where nans were present

[5]:
```
# Check to make sure interpolate function is doing what we want it to:
F9185_data_ds_fill.lat.interpolate_na(dim='date',method='linear').
 ↪plot(marker='+')
F9185_data_ds_fill.lat.plot(marker='o',alpha=0.5)
```

[5]:
```
[<matplotlib.lines.Line2D at 0x16756e4ae50>]
```



Looks good - Now make a new coordinate in the datasets that include these new lat/lon arrays

F9185

[6]:
```
# add new lat coord
F9185_data_ds_fill_interp = F9185_data_ds_fill.
↪assign_coords(lat_interp=('date',F9185_data_ds_fill.lat.
↪interpolate_na(dim='date',method='linear').values))
# add new lon coord
F9185_data_ds_fill_interp = F9185_data_ds_fill_interp.
↪assign_coords(lon_interp=('date',F9185_data_ds_fill.lon.
↪interpolate_na(dim='date',method='linear').values))
```

[7]:
```
# add attributes to new coords
F9185_data_ds_fill_interp.lat_interp.attrs['units'] = 'degrees'
F9185_data_ds_fill_interp.lon_interp.attrs['units'] = 'degrees'
F9185_data_ds_fill_interp.lat_interp.attrs['comments'] = 'Latitude of profile.␣
↪Nans in "lat" coordinate have been linearly interpolated.'
F9185_data_ds_fill_interp.lon_interp.attrs['comments'] = 'Longitude of profile.␣
↪Nans in "lon" coordinate have been linearly interpolated.'
```

[8]:
```
# Save new dataset
F9185_data_ds_fill_interp.to_netcdf(apex_dir / 'F9185/
↪APEX_F9185_profiles_binned_gps-interpolation.nc')
```

F9444

[9]:
```
# add new lat coord
F9444_data_ds_fill_interp = F9444_data_ds_fill.
↪assign_coords(lat_interp=('date',F9444_data_ds_fill.lat.
↪interpolate_na(dim='date',method='linear').values))
# add new lon coord
F9444_data_ds_fill_interp = F9444_data_ds_fill_interp.
↪assign_coords(lon_interp=('date',F9444_data_ds_fill.lon.
↪interpolate_na(dim='date',method='linear').values))
```

[10]:
```
# add attributes to new coords
F9444_data_ds_fill_interp.lat_interp.attrs['units'] = 'degrees'
F9444_data_ds_fill_interp.lon_interp.attrs['units'] = 'degrees'
F9444_data_ds_fill_interp.lat_interp.attrs['comments'] = 'Latitude of profile.␣
↪Nans in "lat" coordinate have been linearly interpolated.'
F9444_data_ds_fill_interp.lon_interp.attrs['comments'] = 'Longitude of profile.␣
↪Nans in "lon" coordinate have been linearly interpolated.'
```

[142]:
```
# Save new dataset
F9444_data_ds_fill_interp.to_netcdf(apex_dir / 'F9444/
↪APEX_F9444_profiles_binned_gps-interpolation.nc')
```

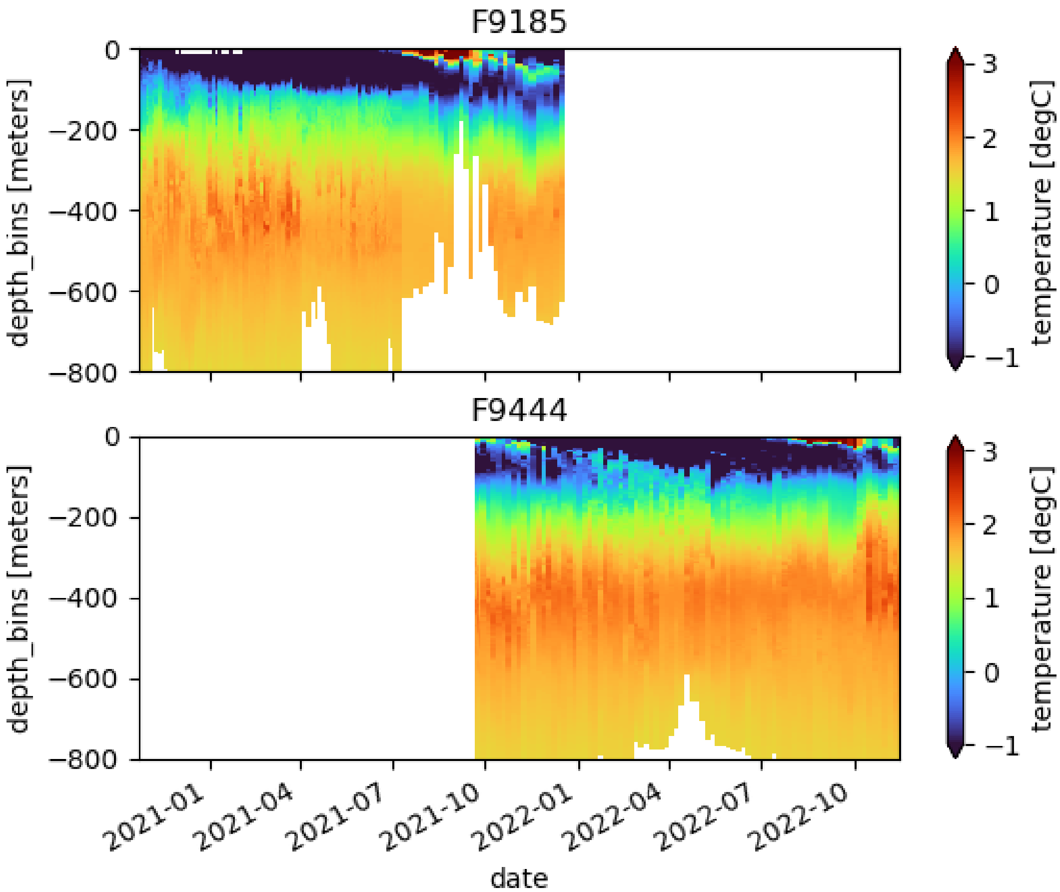
1.1 Plot float data profiles

Open datasets with interpolated lat/lon locations

[11]:
```
# Read in APEX float datasets
apex_dir = Path('../../../data/OMG_Float_data/OMG_APEX_Float_Data/APEX')
F9185_data_ds = xr.open_dataset(apex_dir / 'F9185/
↪APEX_F9185_profiles_binned_gps-interpolation.nc')
F9444_data_ds = xr.open_dataset(apex_dir / 'F9444/
↪APEX_F9444_profiles_binned_gps-interpolation.nc')
```

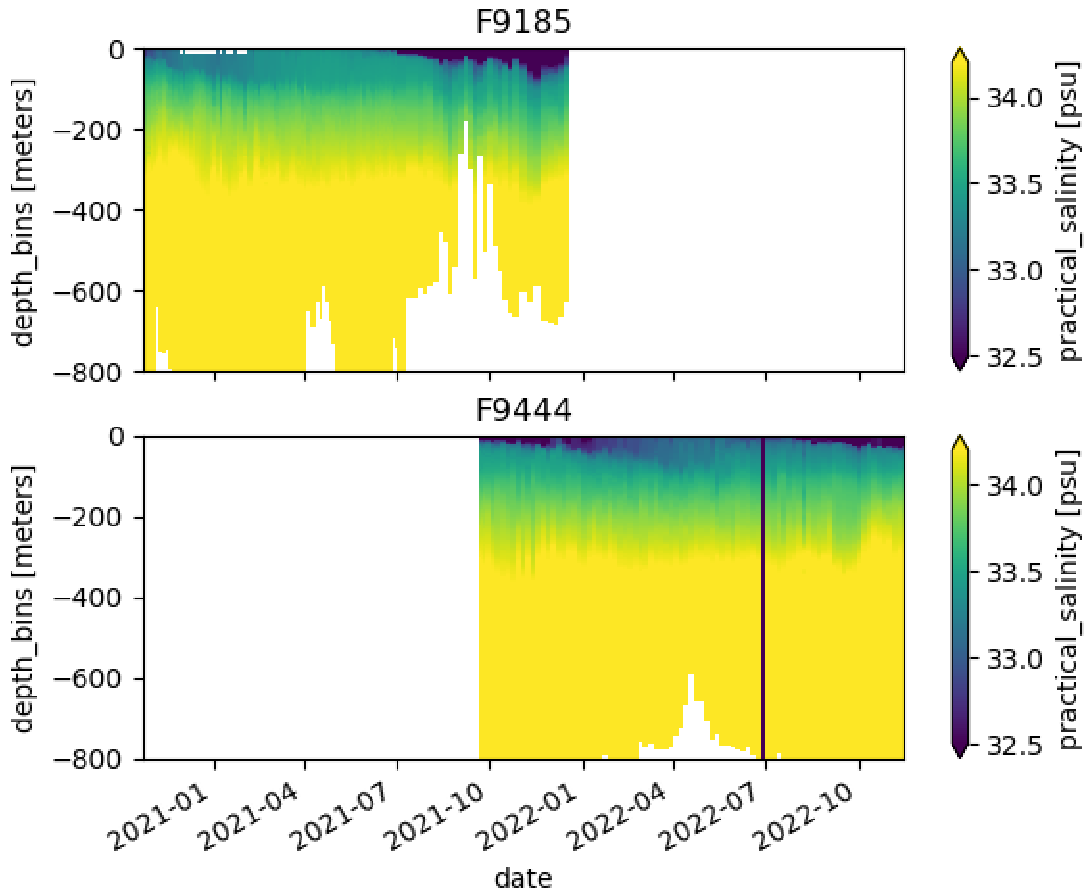
1.1.1 Plot temperature for each float

[12]:
```
fig, (ax1,ax2) = plt.subplots(2,sharex=True,sharey=True)
F9185_data_ds.temperature.plot(ax=ax1,vmin=-1,vmax=3,cmap='turbo')
F9444_data_ds.temperature.plot(ax=ax2,vmin=-1,vmax=3,cmap='turbo')
ax1.set_title("F9185")
ax2.set_title("F9444")
ax1.set_xlabel("")
ax1.set_xlim(F9185_data_ds.date[0].values, F9444_data_ds.date[-1].values);
ax1.set_ylim(-800,0);
```



1.1.2 Plot salinity for each float

[13]:
```
fig, (ax1,ax2) = plt.subplots(2,sharex=True,sharey=True)
F9185_data_ds.salinity.plot(ax=ax1,vmin=32.5,vmax=34.2,cmap='viridis')
F9444_data_ds.salinity.plot(ax=ax2,vmin=32.5,vmax=34.2,cmap='viridis')
ax1.set_title("F9185")
ax2.set_title("F9444")
ax1.set_xlabel("")
ax1.set_xlim(F9185_data_ds.date[0].values, F9444_data_ds.date[-1].values)
ax1.set_ylim(-800,0);
```



1.1.3 See where profiles were located

[14]:
```
bathy_geb = xr.open_dataset('../../../data/bathymetry/GEBCO/gebco_2021_n90.
↪0_s30.0_w-120.0_e30.0.nc')
```

[15]:
```
bathy_select = bathy_geb.elevation.isel(lat=slice(0,None,2),␣
↪lon=slice(0,None,2)).sel(lat=slice(70,80), lon=slice(-75,50))
```

[16]:
```
# identify when F9185 was pushed north
F9185_data_ds_cut = F9185_data_ds_fill.sel(date=slice('2021-7-05','2021-12-16'))
```

[17]:
```
F9185_data_ds_fill.sel(date=slice('2021-7-01','2022-1-01'))
```

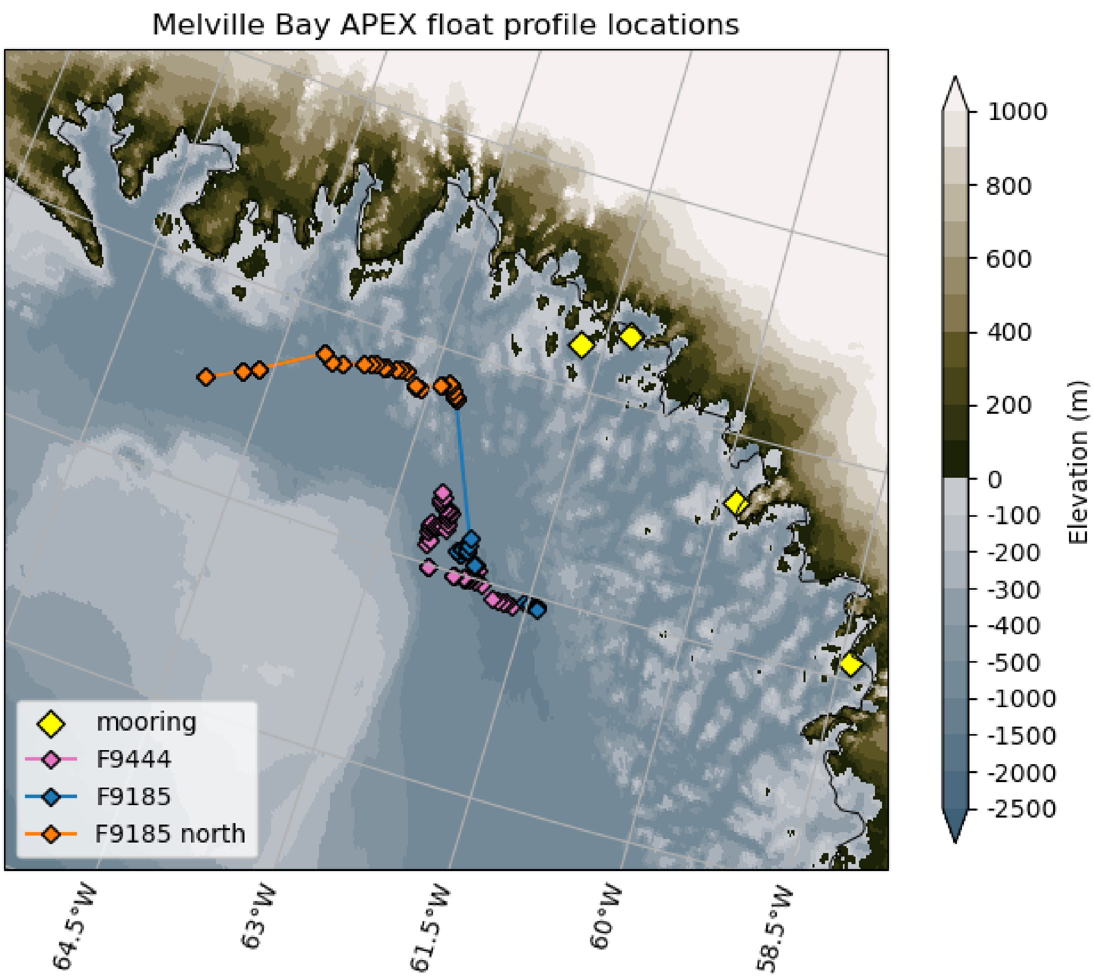
[17]:
```
<xarray.Dataset>
Dimensions:         (date: 30, depth_bins: 601)
Coordinates:
    dive            (date) int32 131 138 139 140 141 142 … 164 165 166 167 168
  * depth_bins      (depth_bins) int32 -1200 -1198 -1196 -1194 … -6 -4 -2 0
```

```
    lat           (date) float64 75.59 75.9 75.89 75.9 … 75.79 75.79 75.75
    lon           (date) float64 -62.21 -62.74 -62.74 … -64.78 -64.78 -65.09
  * date          (date) datetime64[ns] 2021-07-02T00:22:08 … 2021-12-16T00…
Data variables:
    pressure      (depth_bins, date) float64 nan nan nan … 1.022 nan 0.03936
    temperature   (depth_bins, date) float64 nan nan nan … -1.46 nan -1.782
    salinity      (depth_bins, date) float64 nan nan nan nan … 32.2 nan 32.31
    conductivity  (depth_bins, date) float64 nan nan nan nan … 25.76 nan 25.59
Attributes:
    Float_type:    APEX
    Float_number:  F9185
```

Between 2021-07-02 and 2021-07-19 the float was pushed north outside of the main trough

[28]:
```
plt.figure(figsize=(8,8))
plt.rcParams['font.size'] = '10'

ax = plt.axes(projection=ccrs.NorthPolarStereo(central_longitude = -45))
ax.set_extent([-66, -59, 75, 76.5], ccrs.PlateCarree())

# define top and bottom colormaps
top        = cm.get_cmap(cmocean.cm.diff, 11)
bottom     = cm.get_cmap(cmocean.cm.diff_r, 12)
newcolors  = np.vstack((top(np.linspace(0.15, 0.4, 11)),
                       bottom(np.linspace(0, 0.5, 12)))) # create a new␣
 ↪colormaps
ocean_land = ListedColormap(newcolors, name='ocean_land')
bounds     =␣
 ↪[-2500,-2000,-1500,-1000,-500,-400,-300,-200,-100,0,100,200,300,400,500,600,700,800,900,100
ticks      =␣
 ↪[-2500,-2000,-1500,-1000,-500,-400,-300,-200,-100,0,200,400,600,800,1000]

pc   = bathy_select.plot.pcolormesh('lon','lat',ax=ax,transform=ccrs.
 ↪PlateCarree(),shading='auto',rasterized=True,cmap=ocean_land,levels=bounds,add_colorbar=Fal
cbar = plt.colorbar(pc, label='Elevation␣
 ↪(m)',ticks=ticks,boundaries=bounds,orientation='vertical',shrink=0.
 ↪7,spacing='uniform',pad=0.05,aspect=30)
cbar.ax.set_yticklabels(ticks,rotation=0)

ax.coastlines(linewidths=0.7)

gl = ax.gridlines(crs=ccrs.PlateCarree(), draw_labels=True)
gl.top_labels = False
gl.left_labels = True
gl.right_labels = True

# add markers for mooring loccations
```

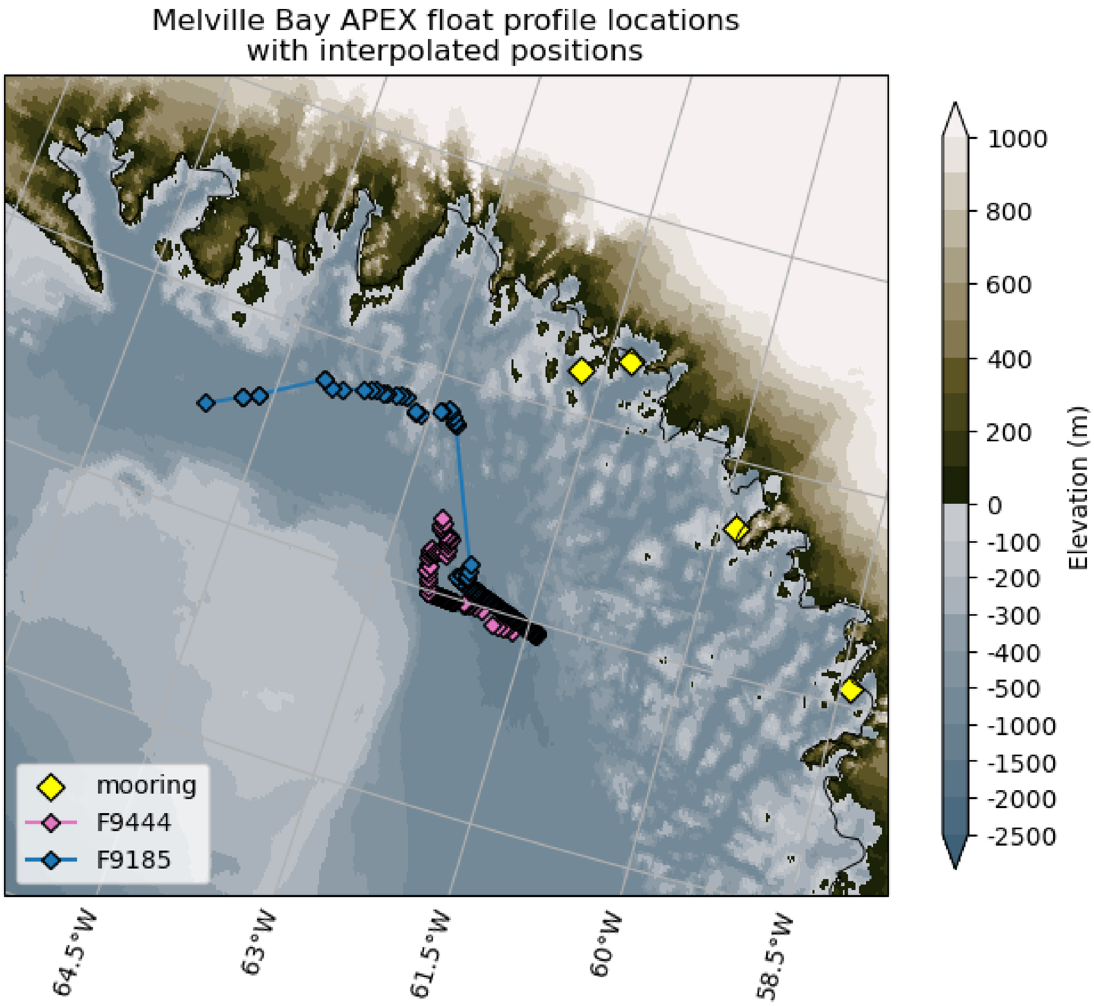
```
ax.scatter(-58.410533, 75.5413, s=60,␣
 ↪c='yellow',edgecolor='black',marker="D",transform=ccrs.PlateCarree(),␣
 ↪label='mooring')
ax.scatter(-59.8429, 75.843683, s=60,␣
 ↪c='yellow',edgecolor='black',marker="D",transform=ccrs.PlateCarree())
ax.scatter(-61.279117, 76.160533, s=60,␣
 ↪c='yellow',edgecolor='black',marker="D",transform=ccrs.PlateCarree())
ax.scatter(-61.726983, 76.103817, s=60,␣
 ↪c='yellow',edgecolor='black',marker="D",transform=ccrs.PlateCarree())

# add apex float coordinates
ax.plot(F9444_data_ds.lon,F9444_data_ds.lat,'-D', color='tab:pink',␣
 ↪markersize=5, markeredgecolor='k', transform=ccrs.PlateCarree(),␣
 ↪label="F9444")
ax.plot(F9185_data_ds.lon,F9185_data_ds.lat,'-D', color='tab:blue',␣
 ↪markersize=5, markeredgecolor='k', transform=ccrs.PlateCarree(),␣
 ↪label="F9185")

# plot different color for profiles to identify when float drifted north
ax.plot(F9185_data_ds_cut.lon,F9185_data_ds_cut.lat,'-D', color='tab:orange',␣
 ↪markersize=5, markeredgecolor='k',transform=ccrs.PlateCarree(), label="F9185␣
 ↪north")

ax.legend(loc='lower left')

plt.title("Melville Bay APEX float profile locations");
```



Plot profile locations with interpolated GPS coords

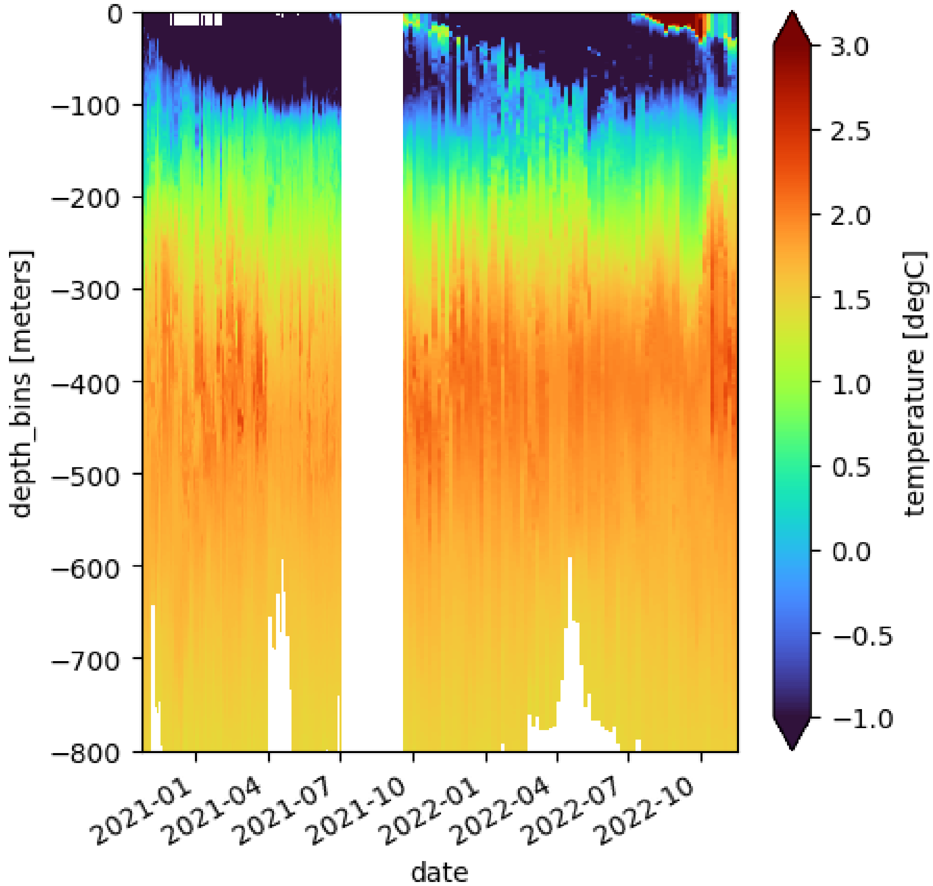
[29]:
```
plt.figure(figsize=(8,8))
plt.rcParams['font.size'] = '10'

ax = plt.axes(projection=ccrs.NorthPolarStereo(central_longitude = -45))
ax.set_extent([-66, -59, 75, 76.5], ccrs.PlateCarree())

# define top and bottom colormaps
top         = cm.get_cmap(cmocean.cm.diff, 11)
bottom      = cm.get_cmap(cmocean.cm.diff_r, 12)
newcolors   = np.vstack((top(np.linspace(0.15, 0.4, 11)),
                       bottom(np.linspace(0, 0.5, 12)))) # create a new␣
 ↪colormaps
ocean_land = ListedColormap(newcolors, name='ocean_land')
bounds      =␣
 ↪[-2500,-2000,-1500,-1000,-500,-400,-300,-200,-100,0,100,200,300,400,500,600,700,800,900,100
```

```
ticks      =␣
 ↪[-2500,-2000,-1500,-1000,-500,-400,-300,-200,-100,0,200,400,600,800,1000]

pc   = bathy_select.plot.pcolormesh('lon','lat',ax=ax,transform=ccrs.
 ↪PlateCarree(),shading='auto',rasterized=True,cmap=ocean_land,levels=bounds,add_colorbar=Fal
cbar = plt.colorbar(pc, label='Elevation␣
 ↪(m)',ticks=ticks,boundaries=bounds,orientation='vertical',shrink=0.
 ↪7,spacing='uniform',pad=0.05,aspect=30)
cbar.ax.set_yticklabels(ticks,rotation=0)

ax.coastlines(linewidths=0.7)

gl = ax.gridlines(crs=ccrs.PlateCarree(), draw_labels=True)
gl.top_labels = False
gl.left_labels = True
gl.right_labels = True

# add markers for mooring loccations
ax.scatter(-58.410533, 75.5413, s=60,␣
 ↪c='yellow',edgecolor='black',marker="D",transform=ccrs.PlateCarree(),␣
 ↪label='mooring')
ax.scatter(-59.8429, 75.843683, s=60,␣
 ↪c='yellow',edgecolor='black',marker="D",transform=ccrs.PlateCarree())
ax.scatter(-61.279117, 76.160533, s=60,␣
 ↪c='yellow',edgecolor='black',marker="D",transform=ccrs.PlateCarree())
ax.scatter(-61.726983, 76.103817, s=60,␣
 ↪c='yellow',edgecolor='black',marker="D",transform=ccrs.PlateCarree())

# add apex float coordinates
ax.plot(F9444_data_ds.lon_interp,F9444_data_ds.lat_interp,'-D', color='tab:
 ↪pink', markersize=5, markeredgecolor='k', transform=ccrs.PlateCarree(),␣
 ↪label="F9444")
ax.plot(F9185_data_ds.lon_interp,F9185_data_ds.lat_interp,'-D', color='tab:
 ↪blue', markersize=5, markeredgecolor='k', transform=ccrs.PlateCarree(),␣
 ↪label="F9185")

# plot different color for profiles to identify when float drifted north
# ax.plot(F9185_data_ds_cut.lon,F9185_data_ds_cut.lat,'-D', color='tab:orange',␣
 ↪markersize=5, markeredgecolor='k',transform=ccrs.PlateCarree(), label="F9185␣
 ↪north")

ax.legend(loc='lower left')

plt.title("Melville Bay APEX float profile locations\nwith interpolated␣
 ↪positions");
```



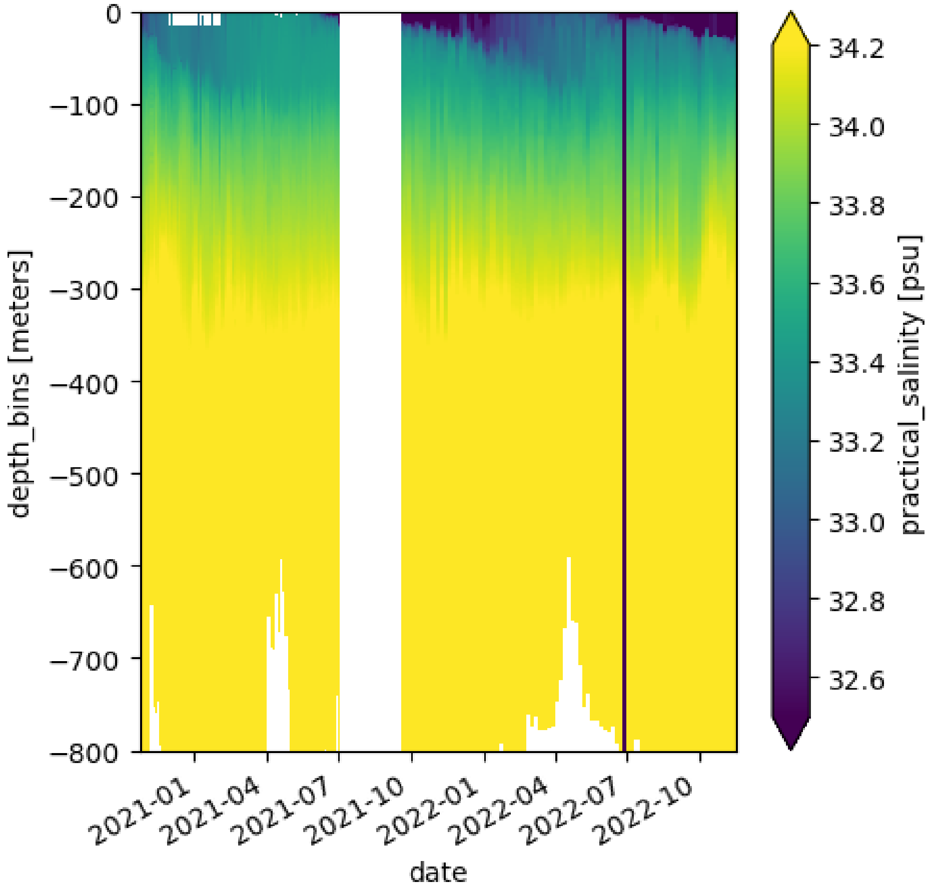
1.1.4 Only plot profiles that were in the main trough

[18]:
```
# fig, (ax1) = plt.subplots(1,sharex=True,sharey=True)
plt.figure(figsize=(5,5))
F9185_data_ds.sel(date=slice(F9185_data_ds.date[0].values,'2021-07-02')).
  ↪temperature.plot(vmin=-1,vmax=3,cmap='turbo',add_colorbar=False)
F9444_data_ds.temperature.plot(vmin=-1,vmax=3,cmap='turbo')
plt.xlim(F9185_data_ds.date[0].values, F9444_data_ds.date[-1].values)
plt.ylim(-800,0);
```



[19]:

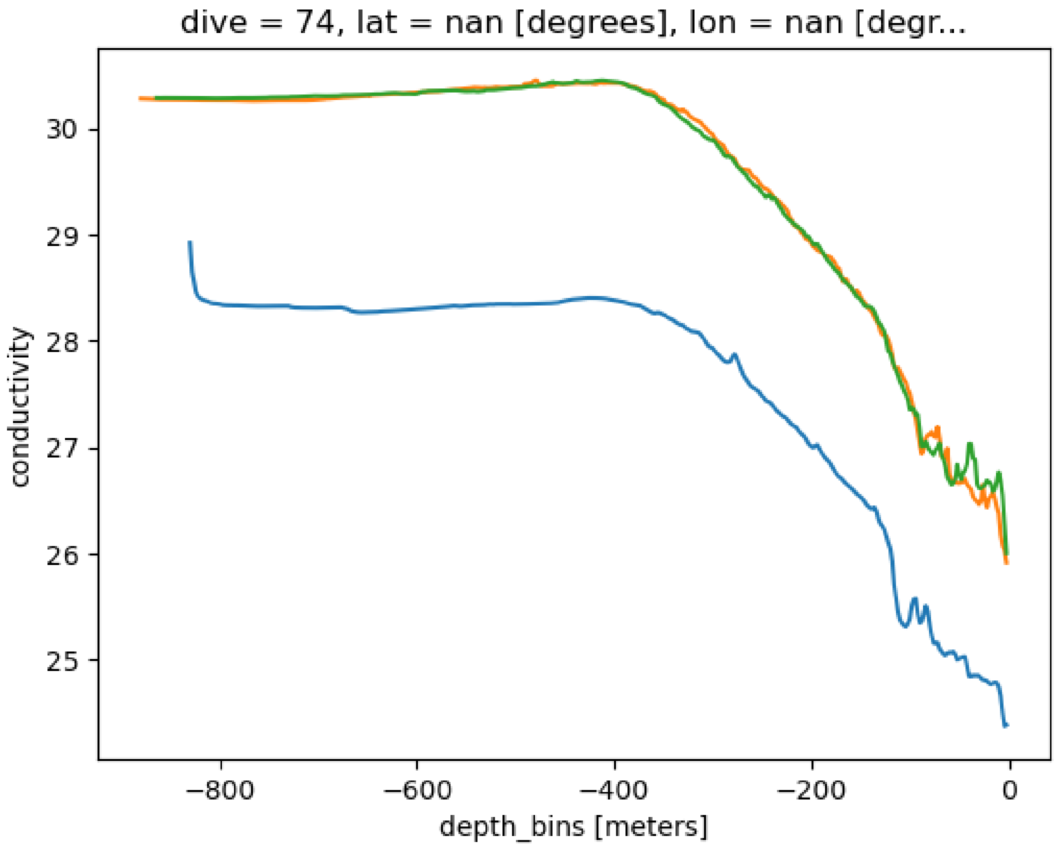
```
# fig, (ax1) = plt.subplots(1,sharex=True,sharey=True)
plt.figure(figsize=(5,5))
F9185_data_ds.sel(date=slice(F9185_data_ds.date[0].values,'2021-07-02')).
 ↪salinity.plot(vmin=32.5,vmax=34.2,cmap='viridis',add_colorbar=False)
F9444_data_ds.salinity.plot(vmin=32.5,vmax=34.2,cmap='viridis')
plt.xlim(F9185_data_ds.date[0].values, F9444_data_ds.date[-1].values)
plt.ylim(-800,0);
```



[32]:
```
# identify weird salinity profile
F9444_data_ds.sel(depth_bins=-400).where(F9444_data_ds.sel(depth_bins=-400).
↪salinity<34, drop=True).dive.values
```

[32]: array([73])

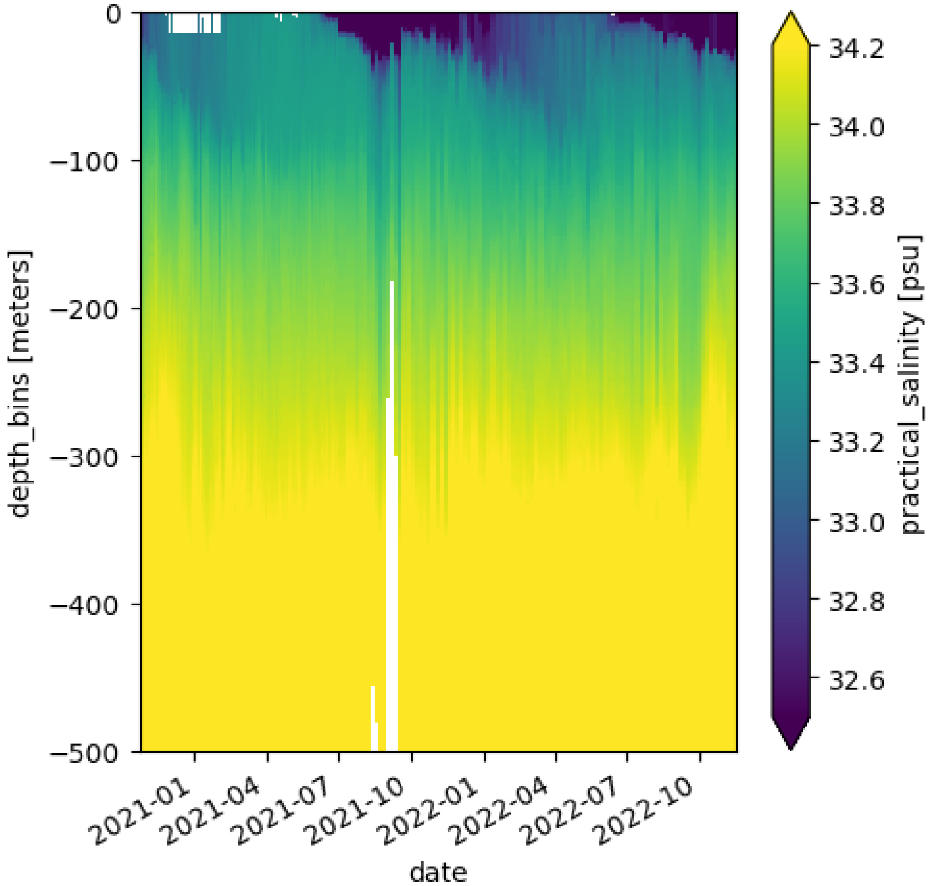
[35]:
```
# look at full profile compared to neighboring ones
F9444_data_ds.swap_dims({'date':'dive'}).sel(dive=73).conductivity.plot()
F9444_data_ds.swap_dims({'date':'dive'}).sel(dive=72).conductivity.plot()
F9444_data_ds.swap_dims({'date':'dive'}).sel(dive=74).conductivity.plot();
```



Conductivity and salinity are lower than they should be - maybe mud or something got stuck on float and obstructed CTD sensor?

[22]:
```
# remove profile 73 from dataset
F9444_data_ds_filter = F9444_data_ds.where(F9444_data_ds.date!=F9444_data_ds.
↪swap_dims({'date':'dive'}).sel(dive=73).date.values, drop=True)
```

[37]:
```
# now plot again
plt.figure(figsize=(5,5))
F9185_data_ds.salinity.plot(vmin=32.5,vmax=34.
↪2,cmap='viridis',add_colorbar=False)
F9444_data_ds_filter.salinity.plot(vmin=32.5,vmax=34.2,cmap='viridis')
plt.xlim(F9185_data_ds.date[0].values, F9444_data_ds.date[-1].values)
plt.ylim(-500,0);
```



1.1.5 Combine float datasets

Since F9444 stayed in the trough, we will use all profiles from this float, and truncate F9185

[20]:
```
# trim F9185 to end on the start date of F9444
F9185_data_ds_cut = F9185_data_ds.sel(date=slice(F9185_data_ds.date.
 ↪values[0],F9444_data_ds.date.values[0]))
```

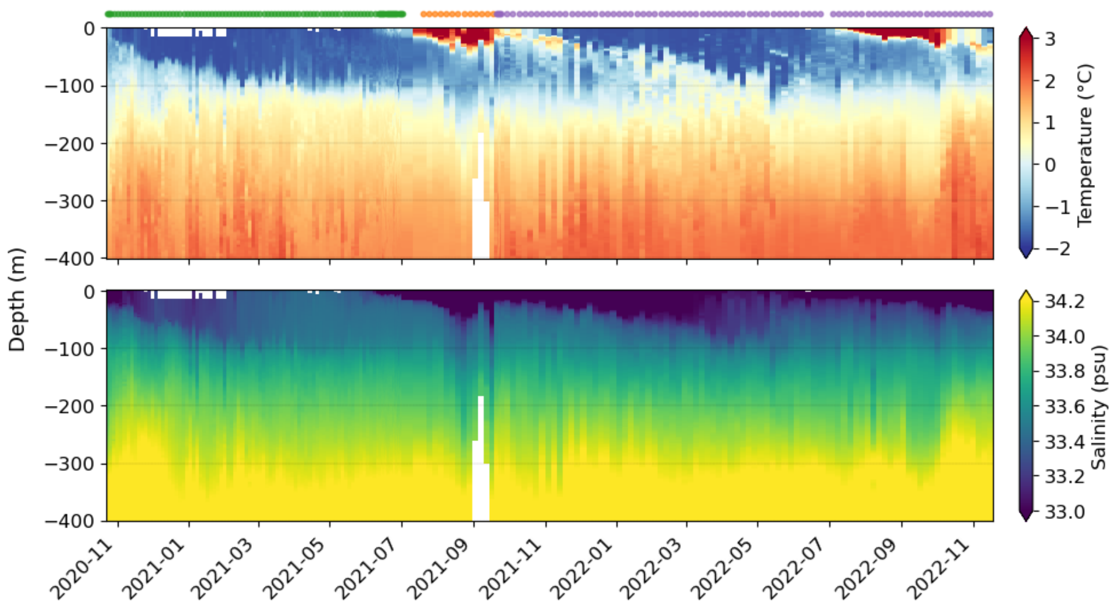
[23]:
```
# combine datasets
melville_float_ds = xr.concat((F9185_data_ds_cut, F9444_data_ds_filter),␣
 ↪dim='date')
```

[24]:
```
# separate profiles that were in the trough and north of the trough for F9185
F9185_data_trough_ds = F9185_data_ds.sel(date=slice(F9185_data_ds.date[0].
 ↪values,'2021-07-03'))
F9185_data_north_ds  = F9185_data_ds.sel(date=slice('2021-07-03',F9444_data_ds.
 ↪date.values[0]))
```

Now plot temperature and salinity profiles for combined dataset but have a color-coded label that corresponds to a map so we know which profiles were outside of the trough

[237]:
```python
from mpl_toolkits.axes_grid1.inset_locator import inset_axes

plt.rcParams["figure.figsize"] = (10,5)
plt.rcParams['font.size'] = 12
fig, (ax1,ax2,ax3,ax4) = plt.subplots(4, sharex=True, sharey=False, gridspec_kw={'height_ratios': [0.1,6,0.2,6]})
plt.subplots_adjust(top=1, hspace=0.1)

# plot profile dots with color corresponding to locations on map
ax1.axis('off')
ax3.axis('off')
ax1.scatter(F9185_data_trough_ds.date.values, [0]*len(F9185_data_trough_ds.date.values), color='tab:green', s=10, alpha=0.7, clip_on=False)
ax1.scatter(F9185_data_north_ds.date.values, [0]*len(F9185_data_north_ds.date.values), color='tab:orange', s=10, alpha=0.7, clip_on=False)
ax1.scatter(F9444_data_ds_filter.date.values, [0]*len(F9444_data_ds_filter.date.values), color='tab:purple', s=10, alpha=0.7, clip_on=False)

# temperature
cb_temp = melville_float_ds.temperature.sel(depth_bins=slice(-400,0)).plot(ax=ax2,vmin=-2,vmax=3,cmap='RdYlBu_r',add_colorbar=False,extend=True)
# salinity
cb_salt = melville_float_ds.salinity.sel(depth_bins=slice(-400,0)).plot(ax=ax4,vmin=33,vmax=34.2,cmap='viridis',add_colorbar=False)

# add colorbars
axins_temp = inset_axes(ax2,width="1.5%",height="100%",loc="upper left",bbox_to_anchor=(1.03, 0., 1, 1),bbox_transform=ax2.transAxes,borderpad=0)
cb = fig.colorbar(cb_temp, cax=axins_temp, ticks=np.arange(-2,4),extend='both',label='Temperature (°C)')

axins_salt = inset_axes(ax4,width="1.5%",height="100%",loc="upper left",bbox_to_anchor=(1.03, 0., 1, 1),bbox_transform=ax4.transAxes,borderpad=0)
cb = fig.colorbar(cb_salt, cax=axins_salt, ticks=np.arange(33,34.2,0.2),extend='both',label='Salinity (psu)')

# axes options
for ax in (ax2,ax4):
    ax.set_xlabel("");
    ax.grid(axis='y',linewidth=0.1,color='k',linestyle='dashed')
    ax.set_ylabel("")
```

```
months   = pd.period_range(np.datetime64('2020-11'), freq='2M', periods=13).
↪strftime('%Y-%m').tolist()
ax4.set_xticks(months, months, rotation=45, horizontalalignment = 'right');

fig.text(0.04, 0.5, 'Depth (m)', size=13, va='center', rotation='vertical');

# plt.savefig("C:/Users/marie/Documents/PhD/Chapter_3/OMG_manuscript_Github/
↪OMG_Narwhals_hydrography-manuscript/Analyses-and-plots/figures/
↪APEX_float_profiles.png", bbox_inches='tight', dpi=300, facecolor='white');
```



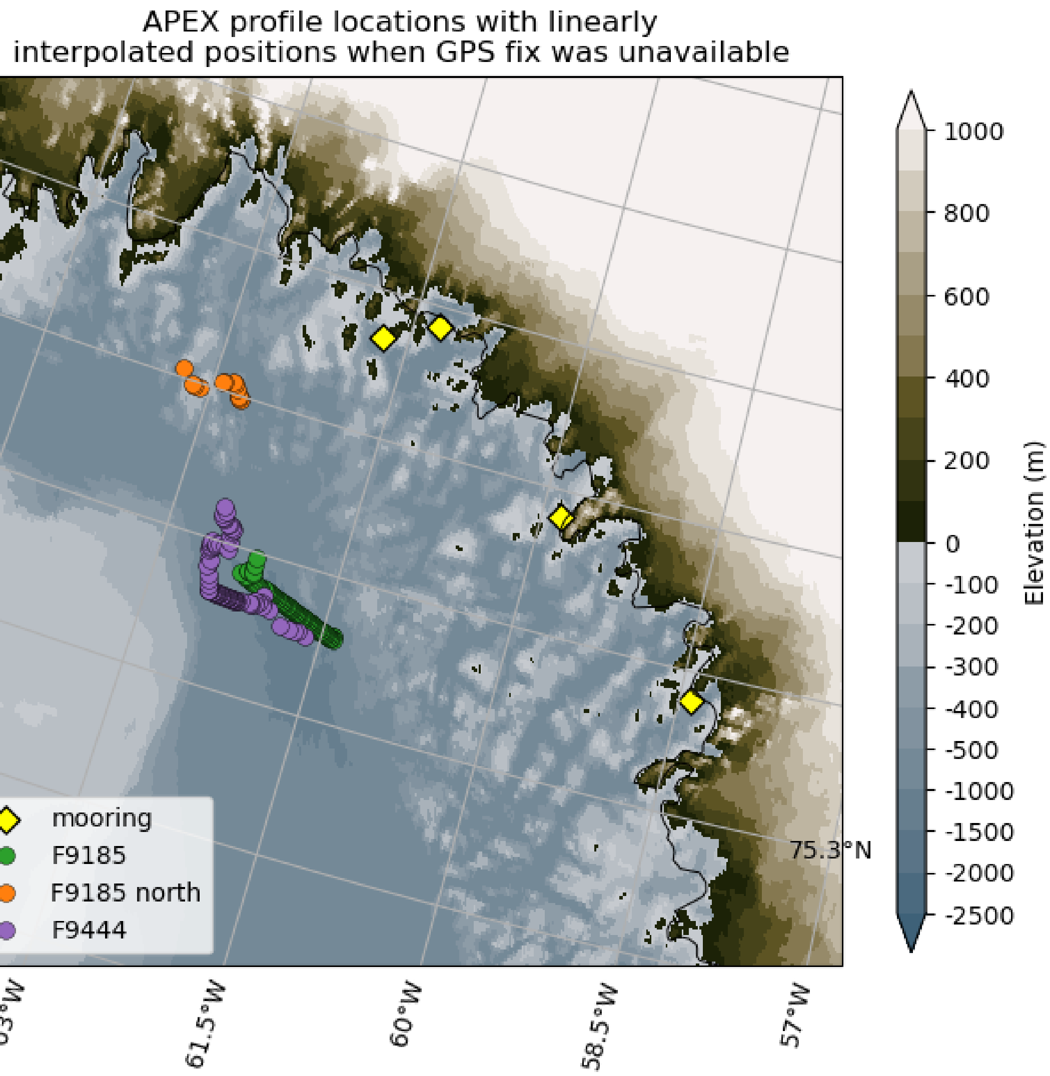
Plot map that links profile locations to the profiles plotted above (i.e., so we know where these profiles were located)

[25]:
```
bathy_select = bathy_geb.elevation.isel(lat=slice(0,None,2),␣
↪lon=slice(0,None,2)).sel(lat=slice(70,80), lon=slice(-75,50))
```

[233]:
```
plt.figure(figsize=(8,7))
plt.rcParams['font.size'] = '10'

ax = plt.axes(projection=ccrs.NorthPolarStereo(central_longitude = -45))
ax.set_extent([-64, -58, 75, 76.5], ccrs.PlateCarree())

# define top and bottom colormaps
top         = cm.get_cmap(cmocean.cm.diff, 11)
bottom      = cm.get_cmap(cmocean.cm.diff_r, 12)
newcolors   = np.vstack((top(np.linspace(0.15, 0.4, 11)),
```

```
                        bottom(np.linspace(0, 0.5, 12)))) # create a new␣
  ↪colormaps
ocean_land = ListedColormap(newcolors, name='ocean_land')
bounds     =␣
  ↪[-2500,-2000,-1500,-1000,-500,-400,-300,-200,-100,0,100,200,300,400,500,600,700,800,900,100
ticks      =␣
  ↪[-2500,-2000,-1500,-1000,-500,-400,-300,-200,-100,0,200,400,600,800,1000]

pc   = bathy_select.plot.pcolormesh('lon','lat',ax=ax,transform=ccrs.
  ↪PlateCarree(),shading='auto',rasterized=True,cmap=ocean_land,levels=bounds,add_colorbar=Fal
cbar = plt.colorbar(pc, label='Elevation␣
  ↪(m)',ticks=ticks,boundaries=bounds,orientation='vertical',shrink=0.
  ↪9,spacing='uniform',pad=0.05,aspect=30)
cbar.ax.set_yticklabels(ticks,rotation=0)

ax.coastlines(linewidths=0.7)

gl = ax.gridlines(crs=ccrs.PlateCarree(), draw_labels=True)
gl.top_labels = False
gl.left_labels = True
gl.right_labels = True

# add markers for mooring loccations
ax.scatter(-58.410533, 75.5413, s=60,␣
  ↪c='yellow',edgecolor='black',marker="D",transform=ccrs.PlateCarree(),␣
  ↪label='mooring')
ax.scatter(-59.8429, 75.843683, s=60,␣
  ↪c='yellow',edgecolor='black',marker="D",transform=ccrs.PlateCarree())
ax.scatter(-61.279117, 76.160533, s=60,␣
  ↪c='yellow',edgecolor='black',marker="D",transform=ccrs.PlateCarree())
ax.scatter(-61.726983, 76.103817, s=60,␣
  ↪c='yellow',edgecolor='black',marker="D",transform=ccrs.PlateCarree())

# add apex float coordinates
ax.plot(F9185_data_trough_ds.lon_interp,F9185_data_trough_ds.lat_interp,'o',␣
  ↪color='tab:green', markersize=7, markeredgecolor='k', markeredgewidth=0.3,␣
  ↪transform=ccrs.PlateCarree(), label="F9185")
# plot different color for profiles to identify when float drifted north
ax.plot(F9185_data_north_ds.lon,F9185_data_north_ds.lat,'o', color='tab:
  ↪orange', markersize=7, markeredgecolor='k',markeredgewidth=0.
  ↪3,transform=ccrs.PlateCarree(), label="F9185 north")
ax.plot(F9444_data_ds_filter.lon_interp,F9444_data_ds_filter.lat_interp,'o',␣
  ↪color='tab:purple', markersize=7, markeredgecolor='k', markeredgewidth=0.3,␣
  ↪transform=ccrs.PlateCarree(), label="F9444")

ax.legend(loc='lower left')
```

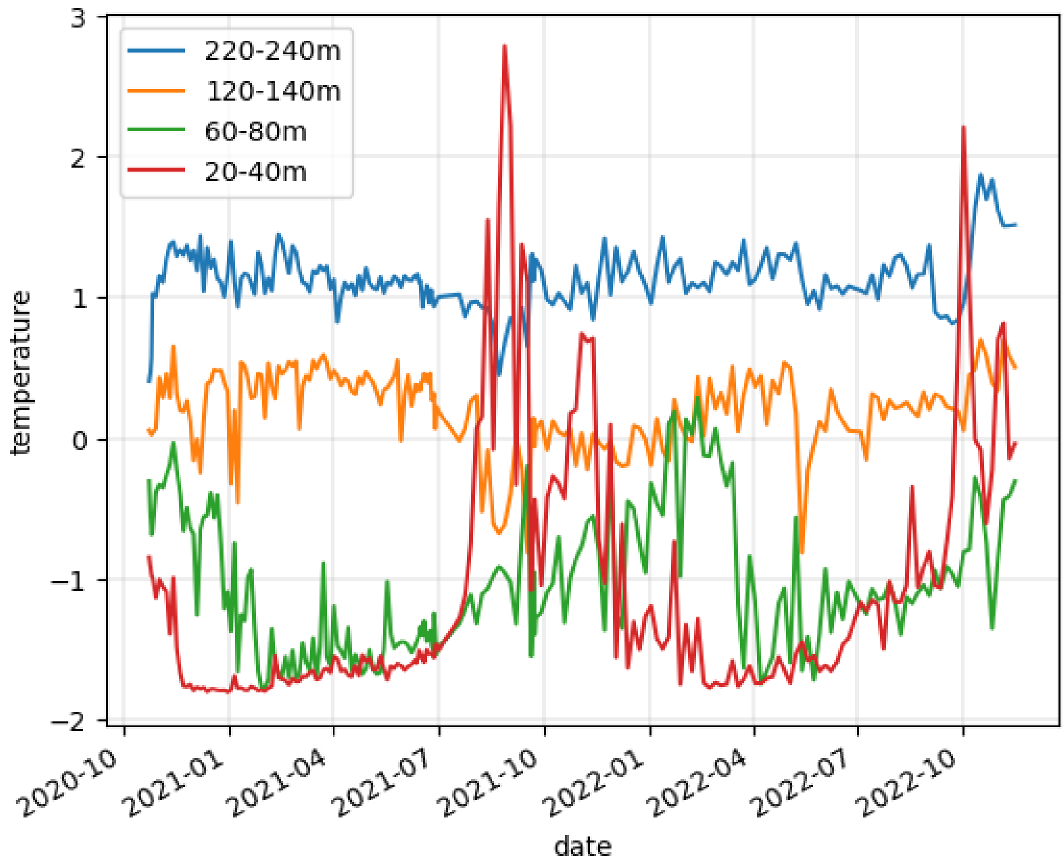
```python
plt.title("APEX profile locations with linearly\ninterpolated positions when GPS fix was unavailable");

# plt.savefig("C:/Users/marie/Documents/PhD/Chapter_3/OMG_manuscript_Github/OMG_Narwhals_hydrography-manuscript/Analyses-and-plots/figures/APEX_float_map.png", bbox_inches='tight', dpi=300, facecolor='white');
```



1.2 Plot mean temperature and salinity for specified depths

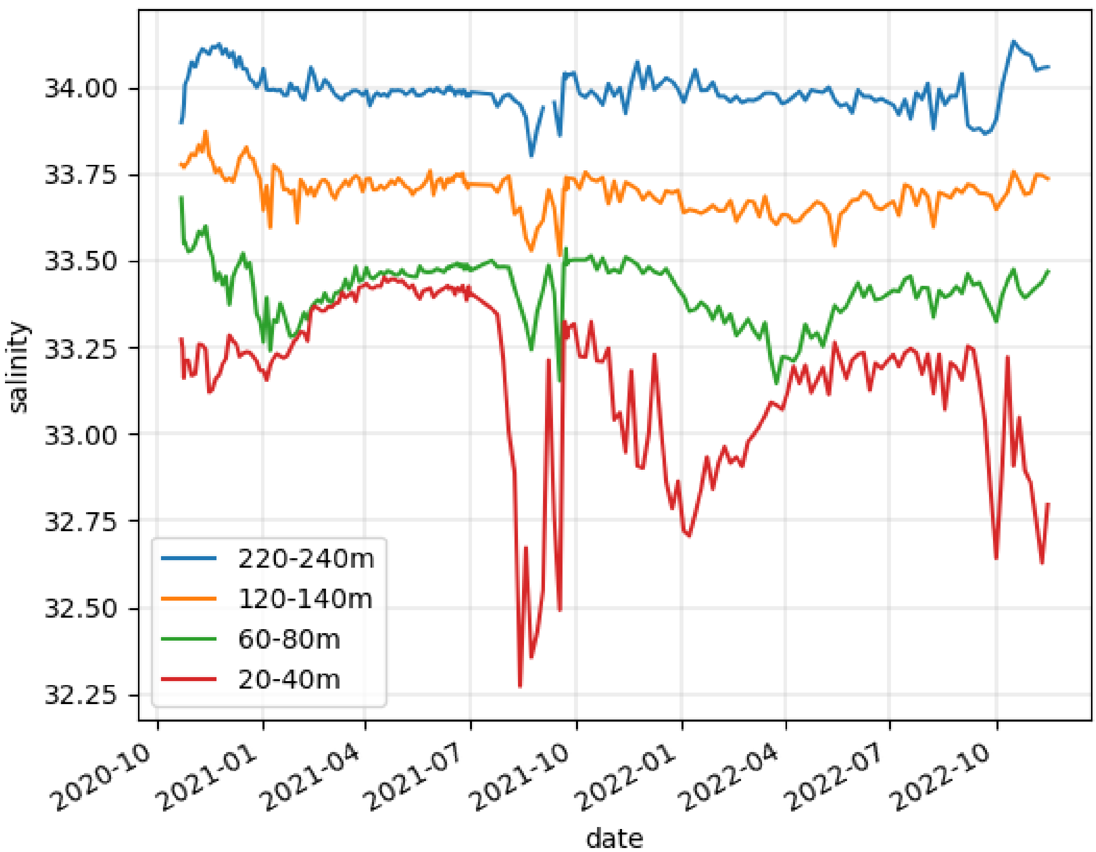
Initial look at several depths:

[26]:
```
melville_float_ds.sel(depth_bins=slice(-240,-220)).mean(dim='depth_bins',␣
 ↪skipna=True).temperature.plot(label='220-240m')
melville_float_ds.sel(depth_bins=slice(-140,-120)).mean(dim='depth_bins',␣
 ↪skipna=True).temperature.plot(label='120-140m')
melville_float_ds.sel(depth_bins=slice(-80,-60)).mean(dim='depth_bins',  ␣
 ↪skipna=True).temperature.plot(label='60-80m')
melville_float_ds.sel(depth_bins=slice(-40,-20)).mean(dim='depth_bins',  ␣
 ↪skipna=True).temperature.plot(label='20-40m')
plt.grid(linewidth=0.3)
plt.legend();
```



[27]:
```
melville_float_ds.sel(depth_bins=slice(-240,-220)).mean(dim='depth_bins',␣
 ↪skipna=True).salinity.plot(label='220-240m')
melville_float_ds.sel(depth_bins=slice(-140,-120)).mean(dim='depth_bins',␣
 ↪skipna=True).salinity.plot(label='120-140m')
melville_float_ds.sel(depth_bins=slice(-80,-60)).mean(dim='depth_bins',  ␣
 ↪skipna=True).salinity.plot(label='60-80m')
```

```
melville_float_ds.sel(depth_bins=slice(-40,-20)).mean(dim='depth_bins', ␣
↪skipna=True).salinity.plot(label='20-40m')
plt.grid(linewidth=0.3)
plt.legend();
```



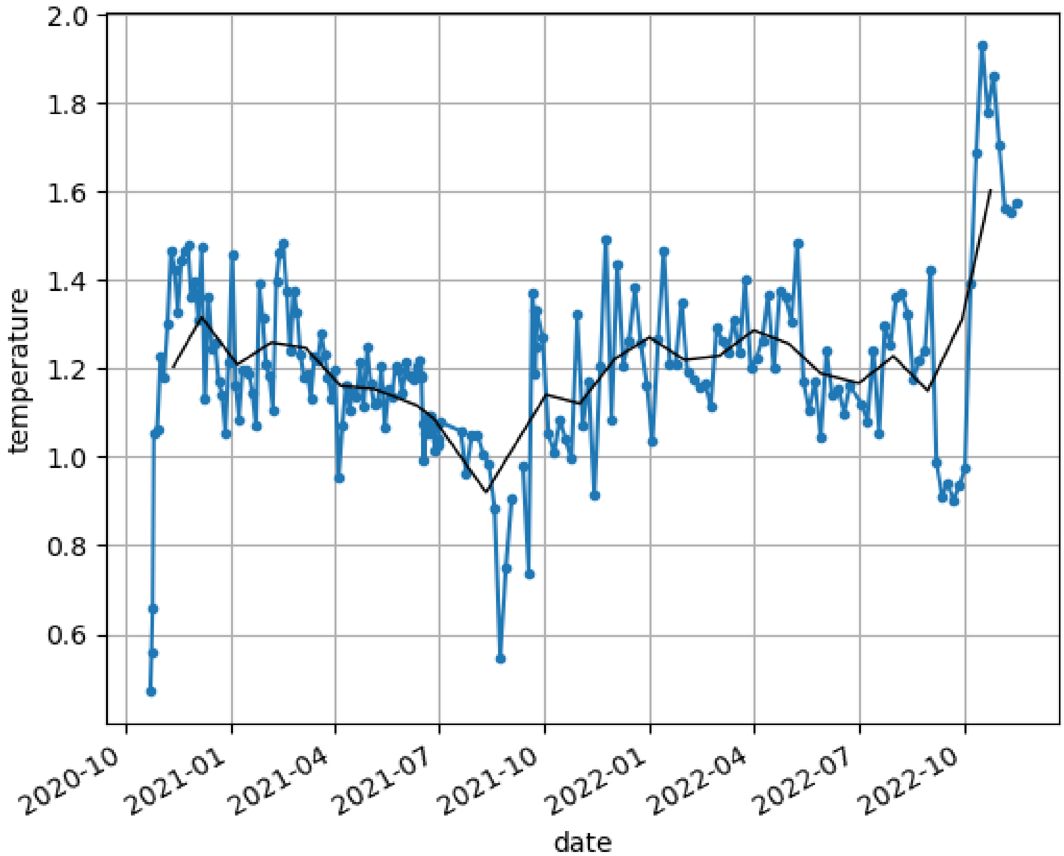
Now remove the surface depth and also add a running mean

[28]:
```
# running mean function from ITS_LIVE github repo
def runningMean(
    mid_dates,
    variable,
    minpts,
    tFreq
):
    """
    mid_dates: center dates of `variable` data [datetime64]
    variable: data to be averaged
    minpts: minimum number of points needed for a valid value, else filled with␣
↪nan
```

```
    tFreq: the spacing between centered averages in Days, default window size =␣
↪tFreq*2
    """
    tsmin = pd.Timestamp(np.min(mid_dates))
    tsmax = pd.Timestamp(np.max(mid_dates))
    ts = pd.date_range(start=tsmin, end=tsmax, freq=f"{tFreq}D")
    ts = pd.to_datetime(ts).values
    idx0 = ~np.isnan(variable)
    runmean = np.empty([len(ts) - 1, 1])
    runmean[:] = np.nan
    tsmean = ts[0:-1] # times for final running mean data

    t_np = mid_dates.astype(np.int64)

    for i in range(len(ts) - 1):
        idx = (
            (mid_dates >= (ts[i] - np.timedelta64(int(tFreq / 2), "D")))
            & (mid_dates < (ts[i + 1] + np.timedelta64(int(tFreq / 2), "D")))
            & idx0
        )
        if sum(idx) >= minpts:
            runmean[i] = np.mean(variable[idx])
            tsmean[i] = np.mean(t_np[idx])

    tsmean = pd.to_datetime(tsmean).values
    return (runmean, tsmean)
```

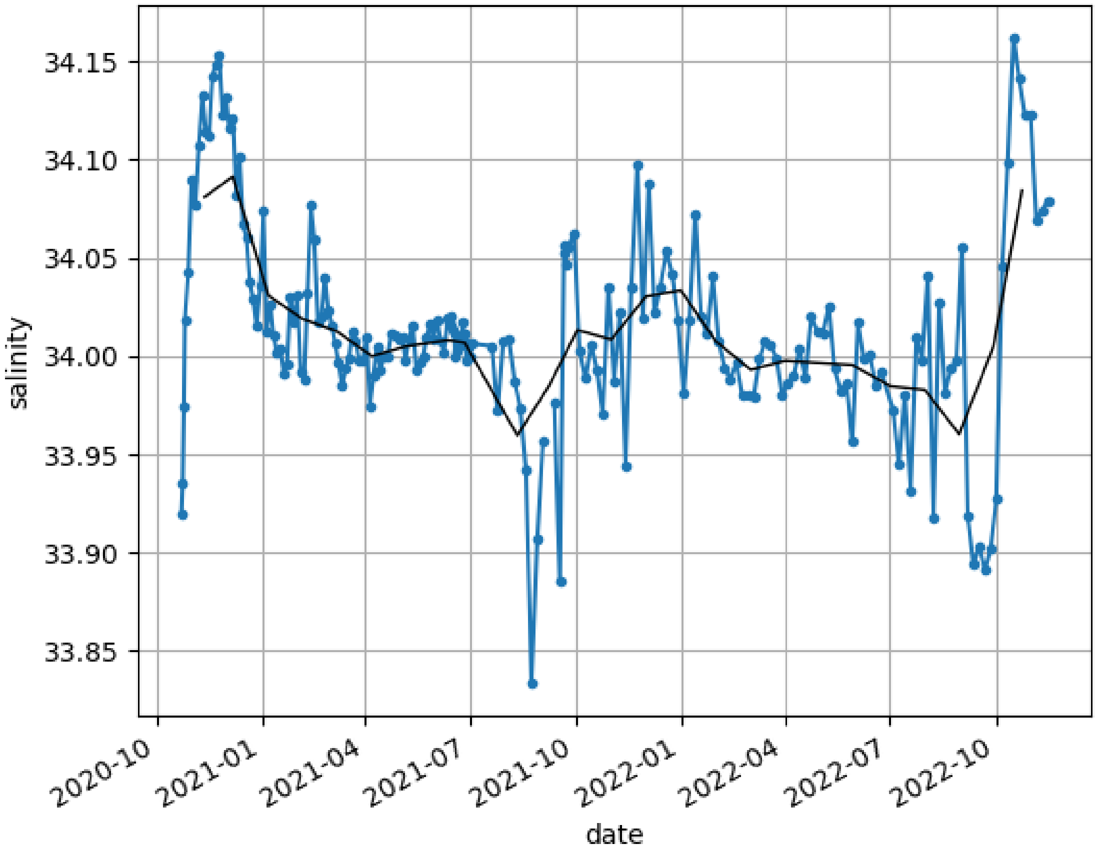
Plot mean temperature for three depth ranges and 30-day running mean

[38]:
```
depth = -220
melville_float_ds_select = melville_float_ds.
 ↪sel(depth_bins=slice(depth-40,depth)).mean(dim='depth_bins', skipna=True)
runmean_float, ts_float = runningMean(melville_float_ds_select.date.values,\
                                      melville_float_ds_select.temperature.
 ↪values,5,30)
melville_float_ds.sel(depth_bins=slice(depth-40,depth)).mean(dim='depth_bins',␣
 ↪skipna=True).temperature.plot(marker='.',\
                                                                                ␣
 ↪                              label=str(depth*-1)+"-"+str(depth*-1+20)+"m");
plt.plot(ts_float,runmean_float, color='k', linewidth=1)
plt.grid()
# plt.legend(loc='upper left');
```

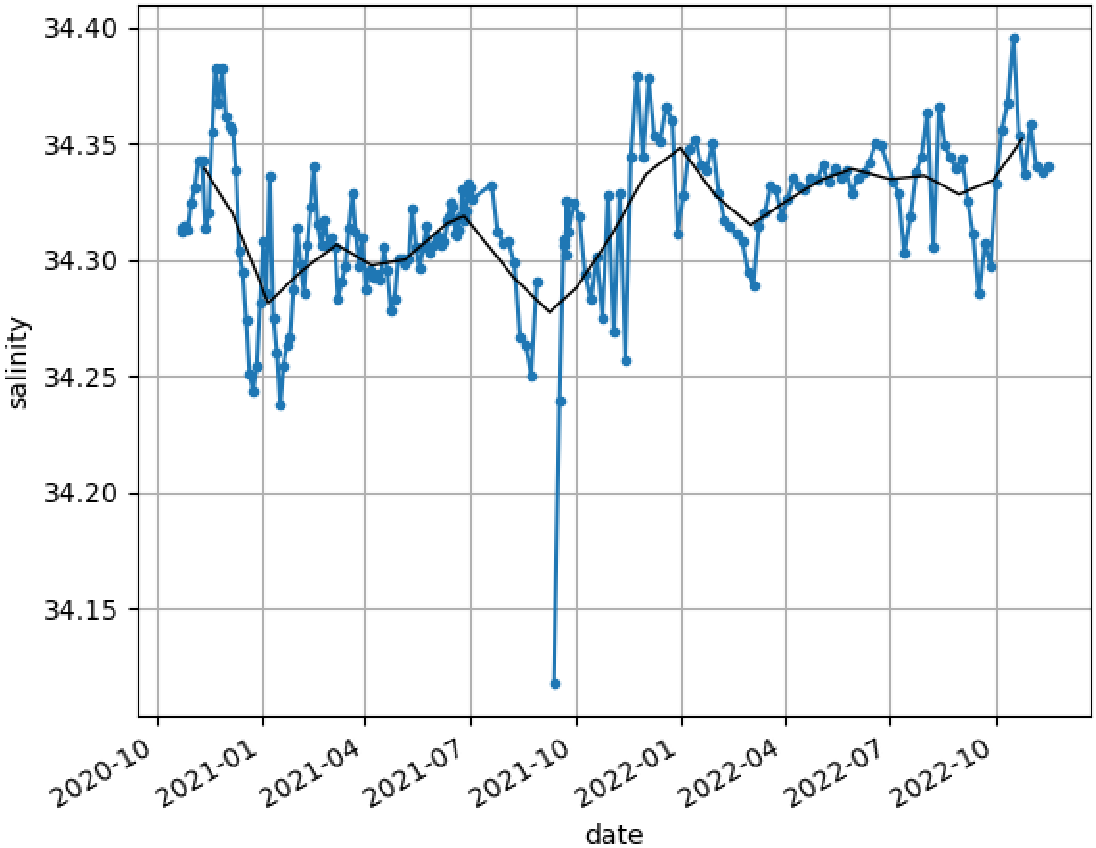


[39]:

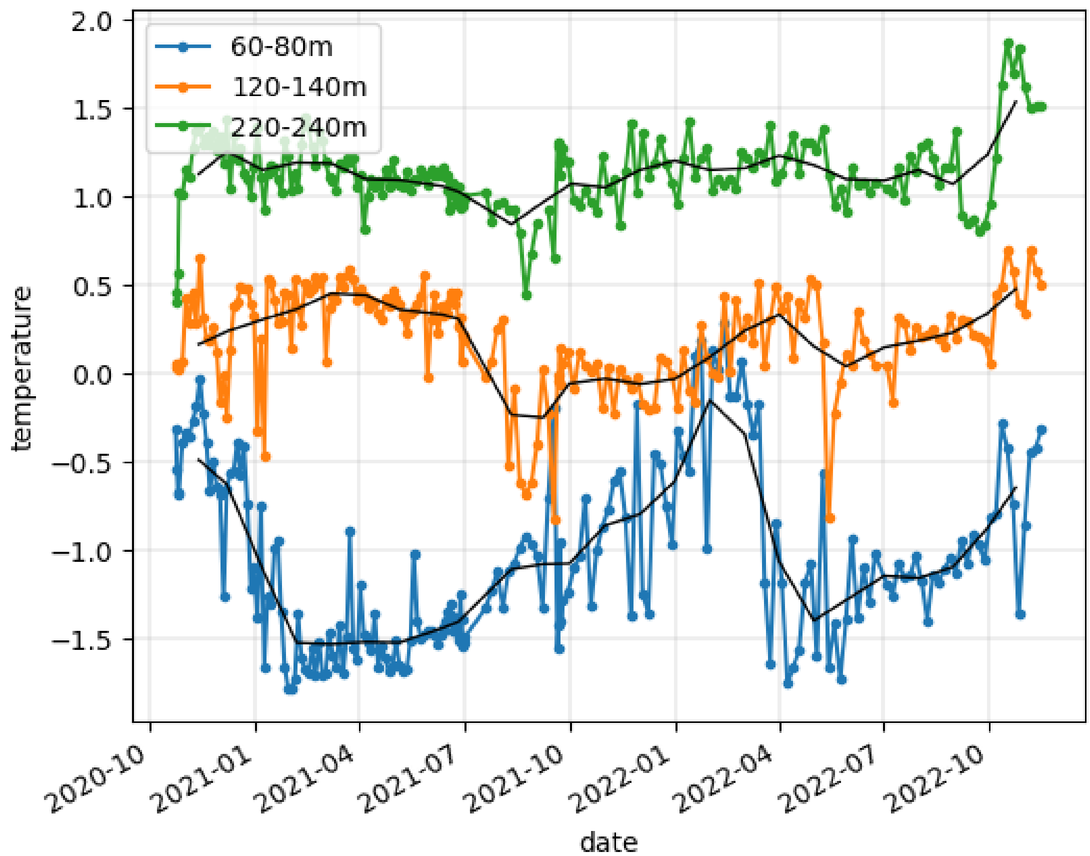
```
depth = -220
melville_float_ds_select = melville_float_ds.
 ↪sel(depth_bins=slice(depth-40,depth)).mean(dim='depth_bins', skipna=True)
runmean_float, ts_float = runningMean(melville_float_ds_select.date.values,\
                                      melville_float_ds_select.salinity.
 ↪values,5,30)
melville_float_ds.sel(depth_bins=slice(depth-40,depth)).mean(dim='depth_bins',␣
 ↪skipna=True).salinity.plot(marker='.',\
                                                                               ␣
 ↪                                label=str(depth*-1)+"-"+str(depth*-1+20)+"m");
plt.plot(ts_float,runmean_float, color='k', linewidth=1)
plt.grid()
# plt.legend(loc='upper left');
```



[37]:
```
depth = -300
melville_float_ds_select = melville_float_ds.
  ↪sel(depth_bins=slice(depth-200,depth)).mean(dim='depth_bins', skipna=True)
runmean_float, ts_float = runningMean(melville_float_ds_select.date.values,\
                                      melville_float_ds_select.salinity.
  ↪values,5,30)
melville_float_ds.sel(depth_bins=slice(depth-200,depth)).mean(dim='depth_bins',␣
  ↪skipna=True).salinity.plot(marker='.',\
                                                                              ␣
  ↪                                 label=str(depth*-1)+"-"+str(depth*-1+20)+"m");
plt.plot(ts_float,runmean_float, color='k', linewidth=1)
plt.grid()
# plt.legend(loc='upper left');
```

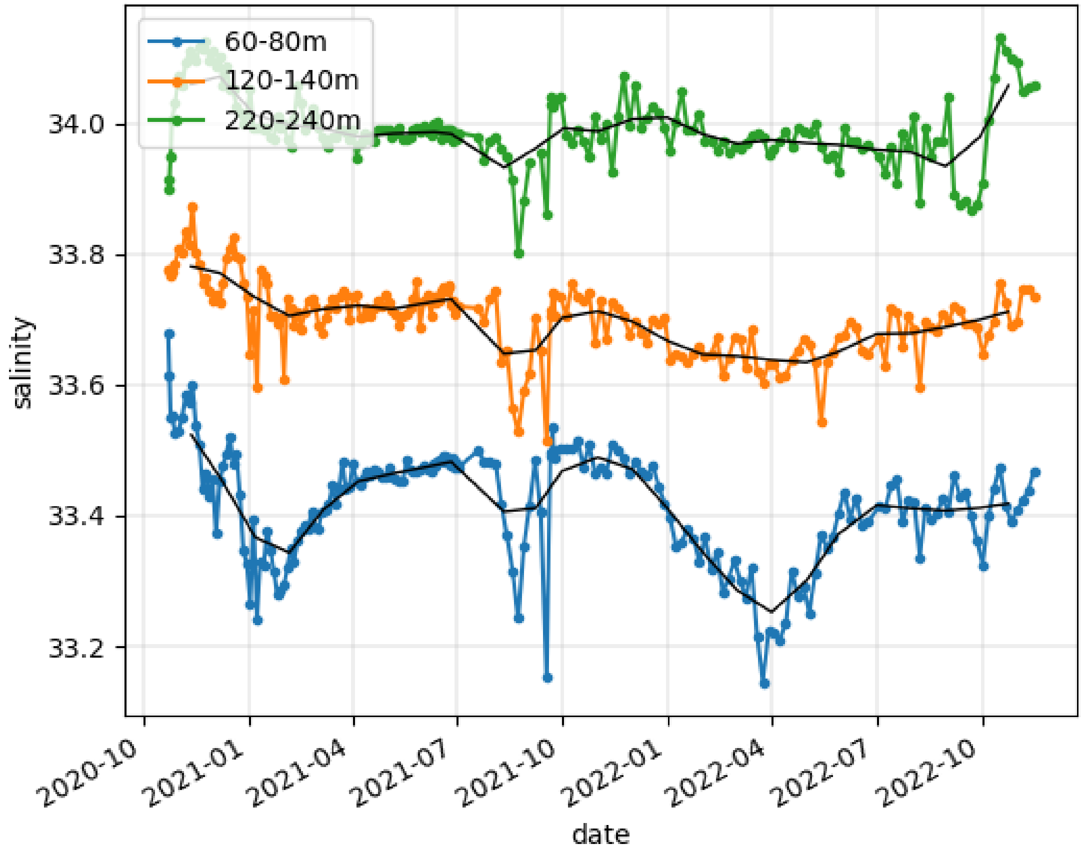


[29]:
```
depths = [-60,-120,-220]
for depth in depths:
    melville_float_ds_select = melville_float_ds.
 ↪sel(depth_bins=slice(depth-20,depth)).mean(dim='depth_bins', skipna=True)

    runmean_float, ts_float = runningMean(melville_float_ds_select.date.values,\
                                          melville_float_ds_select.temperature.
 ↪values,5,30)
    melville_float_ds.sel(depth_bins=slice(depth-20,depth)).
 ↪mean(dim='depth_bins', skipna=True).temperature.plot(marker='.',\
                                                                                ␣
 ↪                                     ␣
 ↪label=str(depth*-1)+"-"+str(depth*-1+20)+"m")
    plt.plot(ts_float,runmean_float, color='k', linewidth=1)
    plt.legend(loc='upper left')
plt.grid(linewidth=0.3)
```



Plot mean salinity for three depth ranges and 30-day running mean

[78]:
```
depths = [-60,-120,-220]
for depth in depths:
    melville_float_ds_select = melville_float_ds.
 ↪sel(depth_bins=slice(depth-20,depth)).mean(dim='depth_bins', skipna=True)

    runmean_float, ts_float = runningMean(melville_float_ds_select.date.values,\
                                          melville_float_ds_select.salinity.
 ↪values,5,30)
    melville_float_ds.sel(depth_bins=slice(depth-20,depth)).
 ↪mean(dim='depth_bins', skipna=True).salinity.plot(marker='.',\
                                                                              ␣
 ↪                                   label=str(depth*-1)+"-"+str(depth*-1+20)+"m")
    plt.plot(ts_float,runmean_float, color='k', linewidth=1)
    plt.legend(loc='upper left')
plt.grid(linewidth=0.3)
```



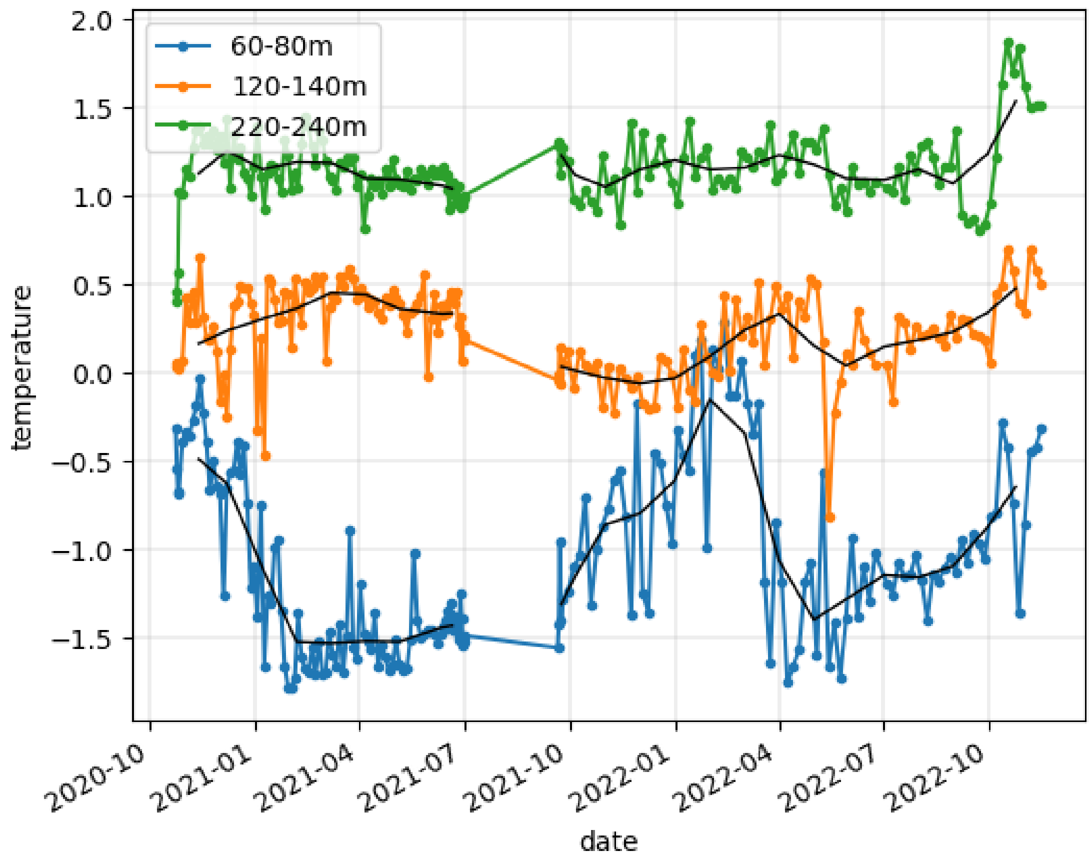
1.2.1 Concatenate and plot data that is only within the deep Melville Bay trough

I.e., remove the northern most profiles

[79]:
```
melville_trough_float_ds = xr.concat((F9185_data_ds.
    ↪sel(date=slice(F9185_data_ds.date[0].values,'2021-07-02')),␣
    ↪F9444_data_ds_filter), dim='date')
```

[87]:
```
depths = [-60,-120,-220]
for depth in depths:
    melville_float_ds_select = melville_trough_float_ds.
    ↪sel(depth_bins=slice(depth-20,depth)).mean(dim='depth_bins', skipna=True)

    runmean_float, ts_float = runningMean(melville_float_ds_select.date.values,\
                                          melville_float_ds_select.temperature.
    ↪values,5,30)
    melville_trough_float_ds.sel(depth_bins=slice(depth-20,depth)).
    ↪mean(dim='depth_bins', skipna=True).temperature.plot(marker='.',\
```

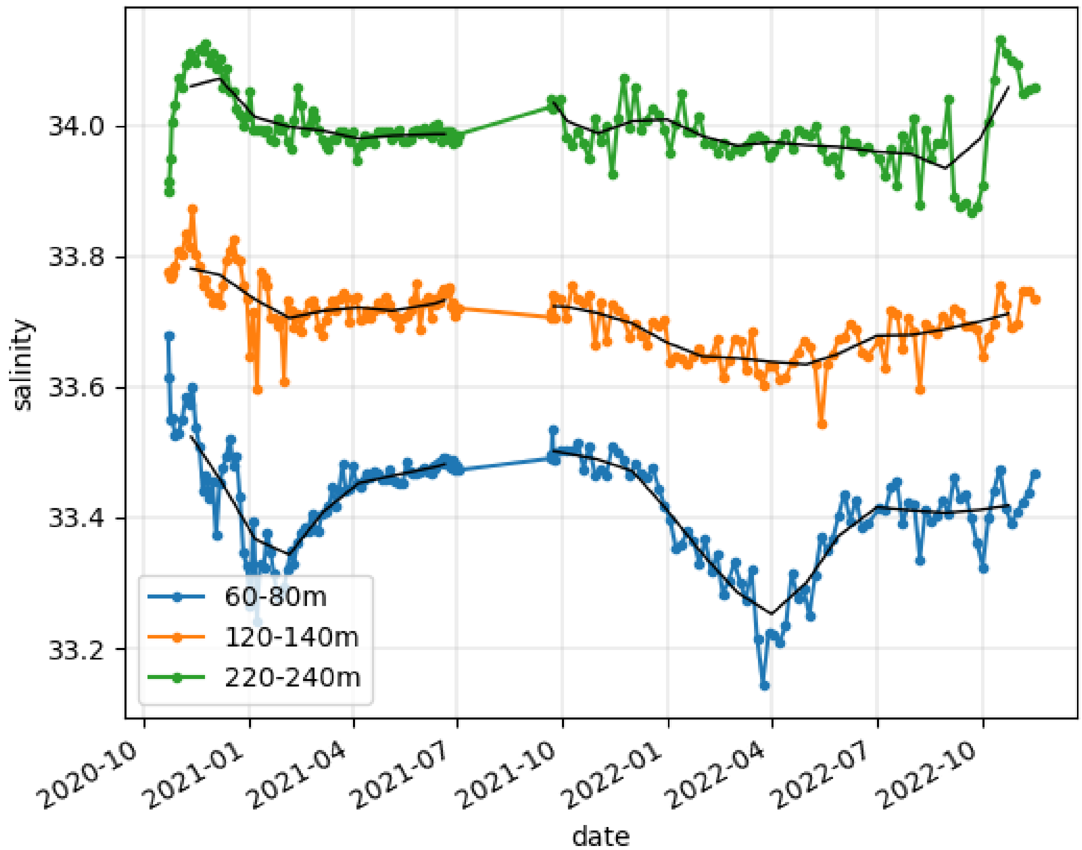
```
↪                                                                              ␣
↪label=str(depth*-1)+"-"+str(depth*-1+20)+"m")
    plt.plot(ts_float,runmean_float, color='k', linewidth=1)
    plt.legend(loc='upper left')
plt.grid(linewidth=0.3)
```



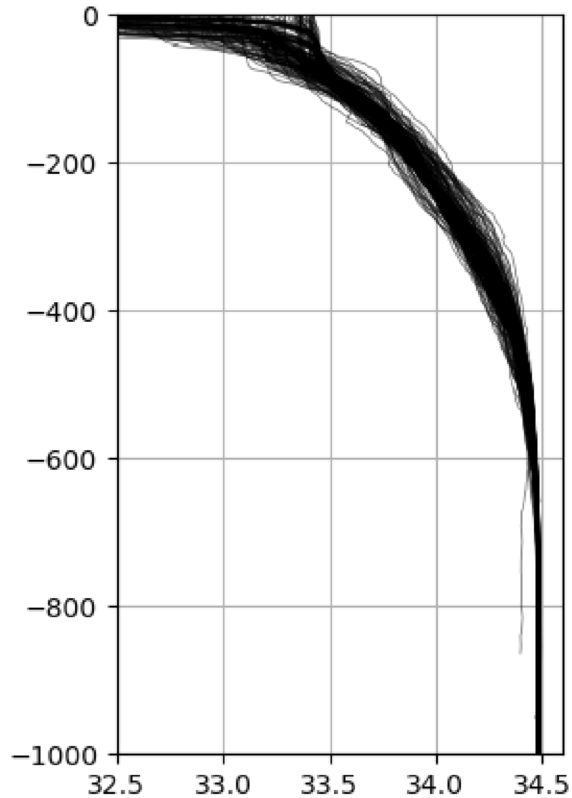
Plot mean salinity for three depth ranges and 30-day running mean

[89]:
```
depths = [-60,-120,-220]
for depth in depths:
    melville_float_ds_select = melville_trough_float_ds.
↪sel(depth_bins=slice(depth-20,depth)).mean(dim='depth_bins', skipna=True)

    runmean_float, ts_float = runningMean(melville_float_ds_select.date.values,\
                                          melville_float_ds_select.salinity.
↪values,5,30)
    melville_trough_float_ds.sel(depth_bins=slice(depth-20,depth)).
↪mean(dim='depth_bins', skipna=True).salinity.plot(marker='.',\
```

```
                                                  label=str(depth*-1)+"-"+str(depth*-1+20)+"m")
    plt.plot(ts_float,runmean_float, color='k', linewidth=1)
    plt.legend(loc='lower left')
plt.grid(linewidth=0.3)
```

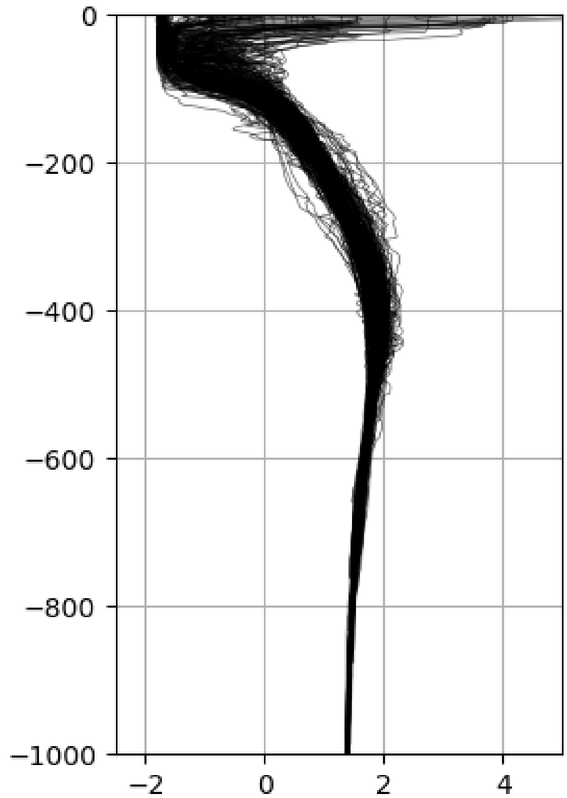


[137]:
```
plt.figure(figsize=(3,5))
plt.rcParams['font.size'] = '10'
for i in range(len(melville_trough_float_ds.date)):
    tmp = melville_trough_float_ds.isel(date=i)
    plt.plot(tmp.salinity.values, tmp.depth_bins.values, c='k', linewidth=0.5,alpha=0.6)
    plt.xlim(32.5,34.6)
    plt.ylim(-1000,0)
plt.grid()
plt.margins(y=0, x=0)
```



[136]:

```
plt.figure(figsize=(3,5))
plt.rcParams['font.size'] = '10'
for i in range(len(melville_trough_float_ds.date)):
    tmp = melville_trough_float_ds.isel(date=i)
    plt.plot(tmp.temperature.values, tmp.depth_bins.values, c='k', linewidth=0.
 ↪5,alpha=0.6)
    plt.xlim(-2.5,5)
    plt.ylim(-1000,0)
plt.margins(y=0, x=0)
plt.grid()
```


0
−200
−400
−600
−800
−1000
−2
0
2
4

```
[ ]:
```