Pythonで実際にArgoデータを読み込んでグラフを描いてみましょう

対象：主に（高校生～）学部生～大学院生

2022年6月2日

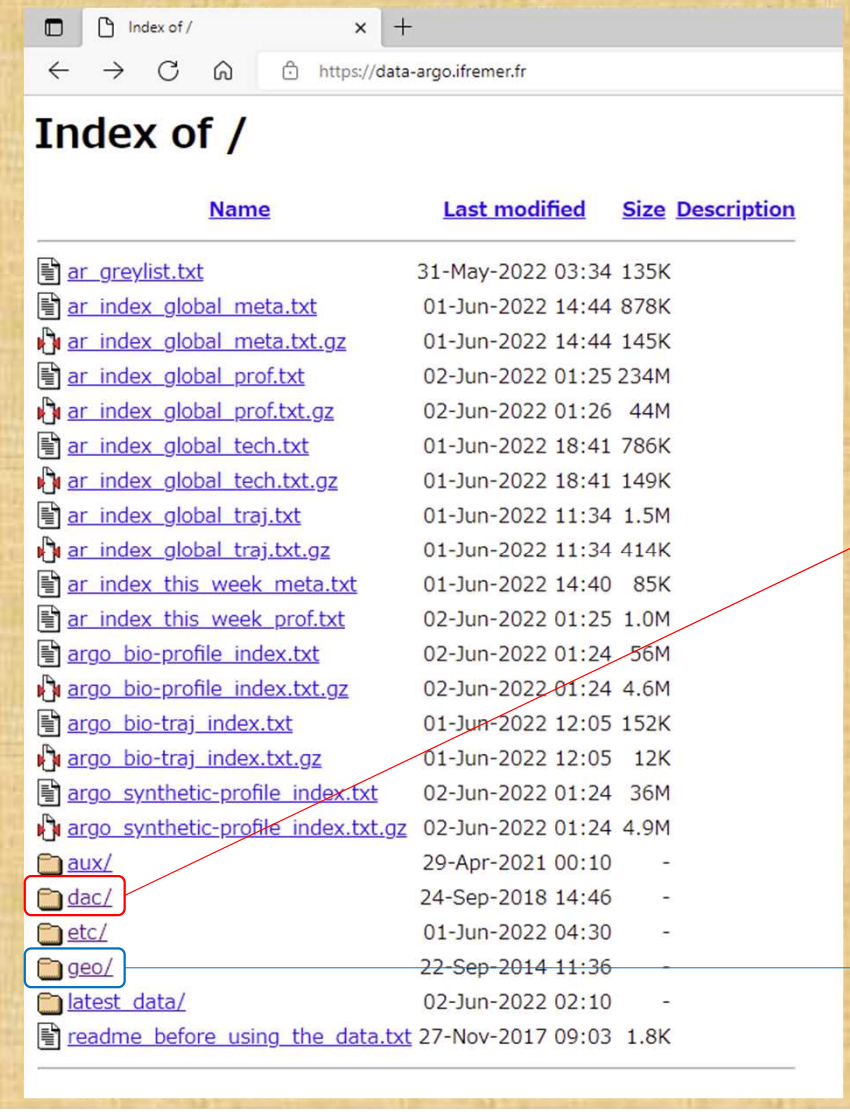
海洋研究開発機構　川合義美

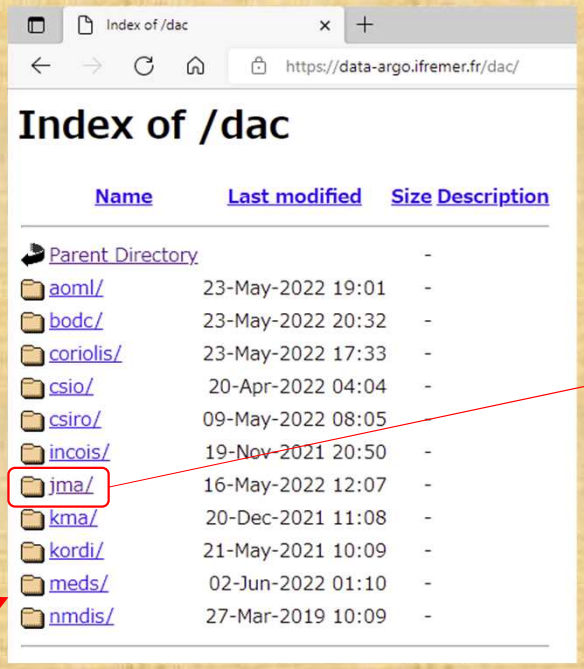
Pythonで、アルゴフロートの個々の観測データのファイルを読み込んで簡単なグラフを描いてみます

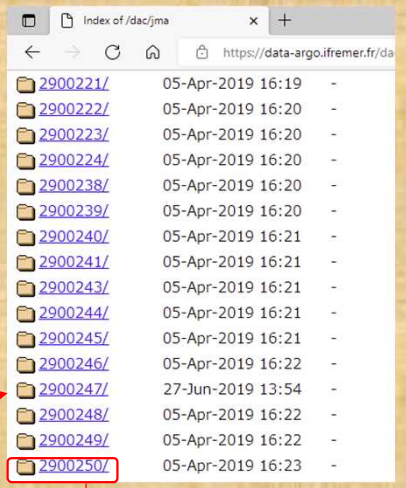
Pythonのインストール方法、開発環境などについては解説しませんのでご了承ください

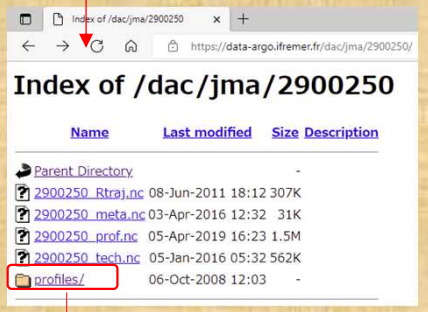
海洋観測データの解析をしたい方は、バニラPythonよりminicondaを使う方がいいかもしれません

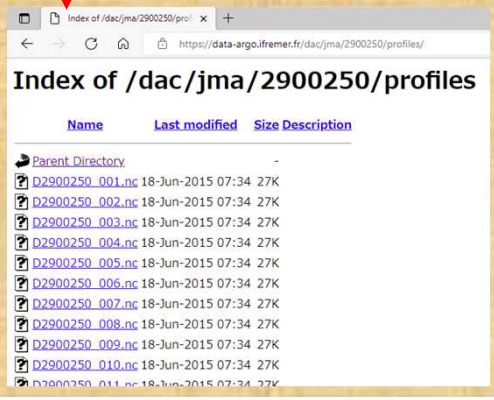
https://data-argo.ifremer.fr











NetCDFとは？

- 気象・海洋分野で広く使われているファイルフォーマットの1つ
- データに関する説明（メタデータ）を格納できる
- データを配列として読み出すことができる

- 拡張子は .nc
- Python, C, Matlab, Octave等様々なプログラミング言語で読み書きするためのパッケージが無料で利用できる（そのままでは読めない）

PythonでnetCDF形式のArgoデータを読む

準備

• ライブラリ　netCDF4　をインストールする

* 素のPython（バニラPython）の場合

> pip install netcdf4

CDFは大文字でも小文字でもよい

* Anaconda またはMinicondaの場合

> conda install netcdf4

※　xarrayを使う方法もある

注）condaで最新バージョンをインストールするとインポート時にエラーが出ることがある。
そんな時は古いバージョンを指定してインストールする

（例）> conda install netcdf4=1.5.6

今回使うライブラリ（インストール必要）

- netCDF4：　NetCDFファイルの読み書き
- numpy：　数値計算に必須
- scipy：　より高度な数値計算ができる
- matplotlib：グラフを描くために必須
- gsw：　海水の物理量を計算
- oct2py：　Pythonからmファイルを動かす

PythonでnetCDF形式のArgoデータを読む

```
import numpy as np
import netCDF4
import inspect

# ファイルをオープン
nc = netCDF4.Dataset('ファイル名', 'r')

# 次元及び変数の情報を表示する
print (nc.dimensions)
print (nc.variable)

# 全ての変数の情報を表示する
for x in inspect.getmembers(nc):
        print (x)
```

ファイル名のパターン

例） D2902474_015.nc

- R: リアルタイム / D: 遅延モード
- フロート固有の番号（WMO ID）
- 何回目の観測か

大量の情報が表示されるので
初めての人にはわかりにくい

PythonでnetCDF形式のArgoデータを読む

特に大事な変数（Delayed-modeの場合）

PRES_ADJUSTED　:補正済圧力
TEMP_ADJUSTED　:補正済水温
PSAL_ADJUSTED　:補正済実用塩分
LATITUDE　:緯度
LONGITUDE　:経度
JULD　:1950年1月1日からの日数

float型

PRES_ADJUSTED_QC　:補正済圧力の品質
TEMP_ADJUSTED_QC　:補正済水温の品質
PSAL_ADJUSTED_QC　:補正済塩分の品質
POSITION_QC　:位置の品質
JULD_QC　:日時の品質

数値ではなく文字列であることに注意

（例）

```
> print(nc.variable['PRES_ADJUSTED'].dimensions)

('N_PROF', 'N_LEVELS')
```

⇒　変数PRES_ADJUSTEDの配列の次元
　N_PROF = プロファイル数（通常は1）
　N_LEVELS = 深さ方向の層の数

```
> print(nc.dimensions['N_LEVELS'])

<class 'netCFD4._netCDF4.Dimension'>: name =
'N_LEVELS', size = 101
```

このファイルでは層の数は101

（例）

```
> print(nc.variable['TEMP_ADJUSTED'])

<class 'netCFD4._netCDF4.Variables'>
float32 TEMP_ADJUSTED(N_PROF, N_LEVELS)
        long_name: Sea temperature in-situ ITS-90 scale
        standard_name: sea_water_temperature
        units: degree_Celsius
        _FillValue: 99999.0
        valid_min: -2.5
        valid_max: 40.0
        C_format: %9.3f
        FORTRAN_format: F9.3
        resolution: 0.001
unlimited dimensions:
current shape = (1, 101)
filling on
```

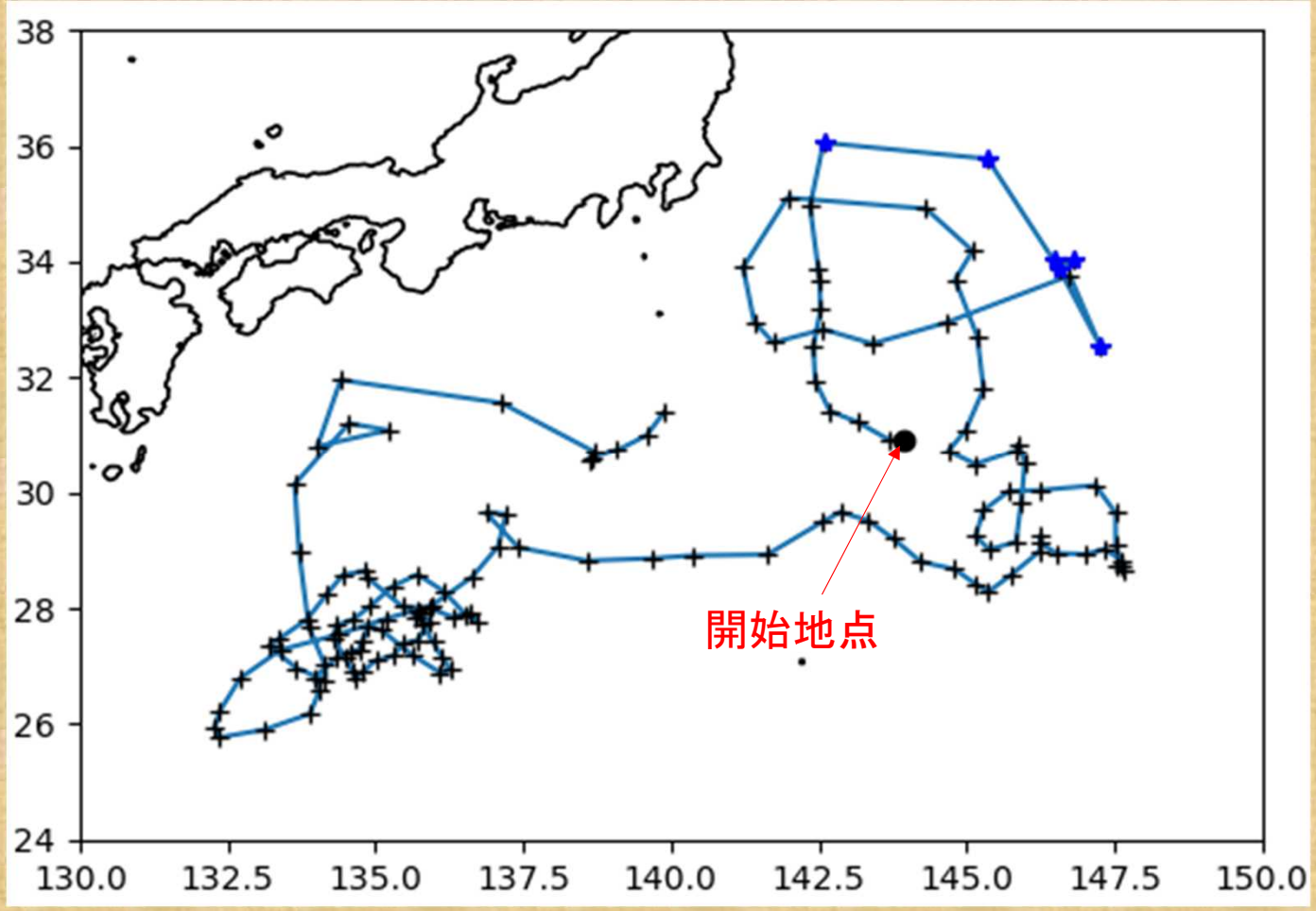
配列（行列）の行数と列数

※　まれにN_PROFが1でないファイルがある（1つのファイルに2つ以上のプロファイルがはいっている）ので注意

このファイルの場合はプロファイル数が1、層の数が101
実質的に1次元のベクトル

あるフロートの例

WMOID 2902474のフロート



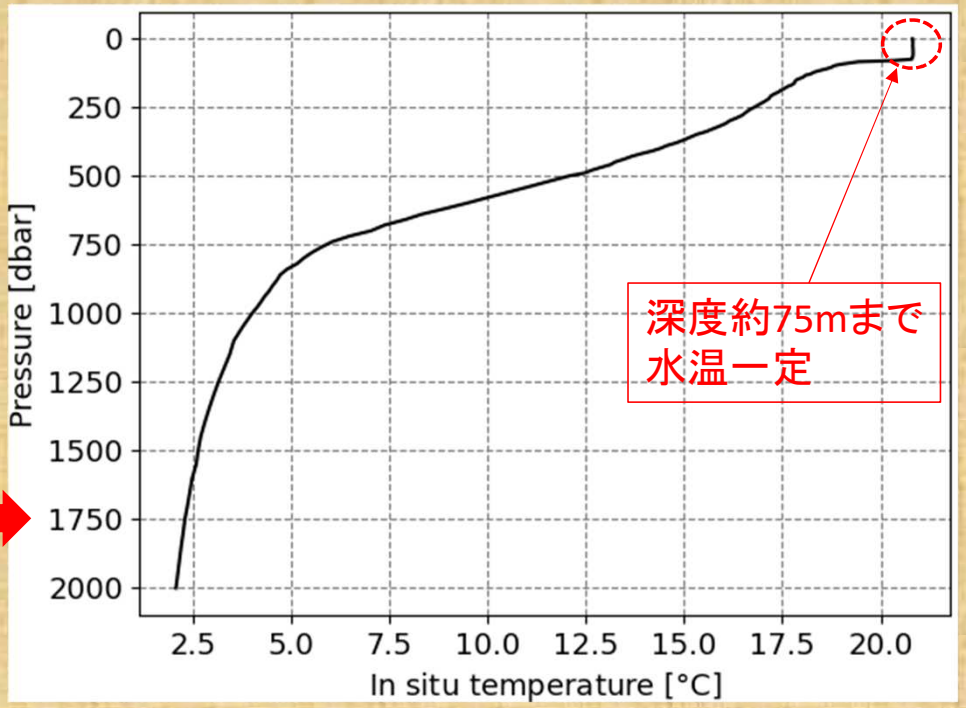
※　地図の描き方は今回は触れません

PythonでnetCDF形式のArgoデータを読む

```
import numpy as np
import matplotlib.pyplot as plt
import netCDF4

nc = netCDF4.Dataset('D2902474_000.nc', 'r')
# 水温と圧力（深度）を読み込む
temp = np.squeeze ( nc.variables ['TEMP_ADJUSTED'] [:] )
pres = np.squeeze ( nc.variables ['PRES_ADJUSTED'] [:] )

# グラフを描く
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.plot(temp, pres, color = 'k')
ax.set_xlabel('In situ temperature [$¥u00B0$C]', fontsize=13)
ax.set_ylabel('Pressure [dbar]', fontsize=13)
ax.tick_params(axis='both', labelsize=13)
ax.grid(which='major', color='gray', linestyle='dashed')
ax.invert_yaxis()
plt.show()
```



北緯30度56分、東経143度56分の
2013年12月20日の鉛直水温分布

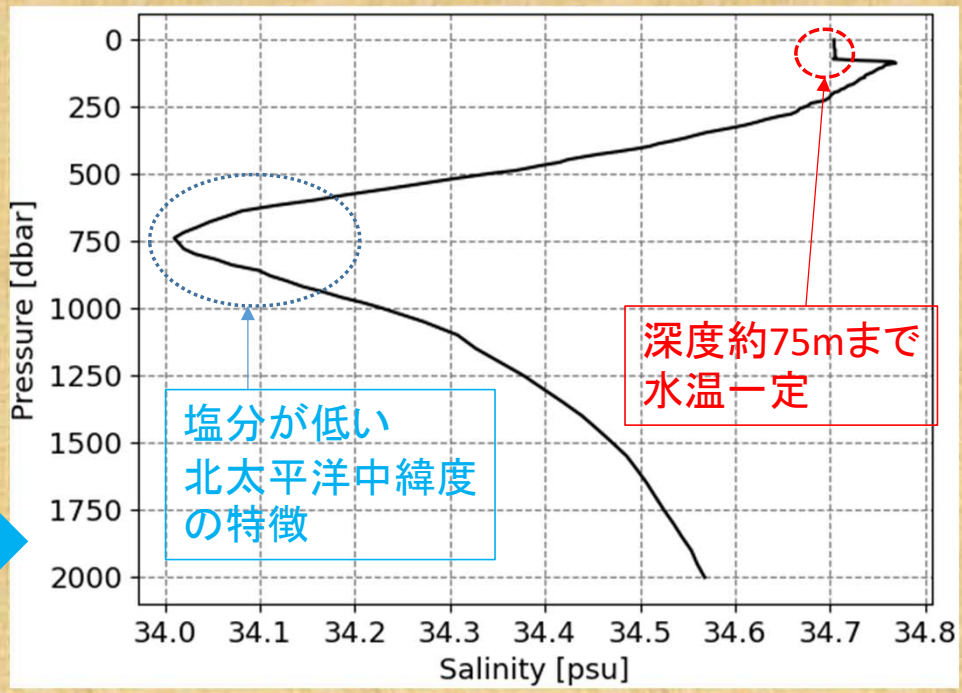
注）netCDF4はディレクトリ名・ファイル名に日本語を入れてはダメ

PythonでnetCDF形式のArgoデータを読む

```
import numpy as np
import matplotlib.pyplot as plt
import netCDF4

nc = netCDF4.Dataset('D2902474_000.nc', 'r')
# 塩分と圧力（深度）を読み込む
psal = np.squeeze ( nc.variables [‘PSAL_ADJUSTED’] [:] )
pres = np.squeeze ( nc.variables ['PRES_ADJUSTED’] [:] )

# グラフを描く
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.plot(psal, pres, color = 'k')
ax.set_xlabel(‘Salinity [psu]', fontsize=13)
ax.set_ylabel('Pressure [dbar]', fontsize=13)
ax.tick_params(axis='both', labelsize=13)
ax.grid(which='major', color='gray', linestyle='dashed')
ax.invert_yaxis()
plt.show()
```



北緯30度56分、東経143度56分の
2013年12月20日の鉛直塩分分布

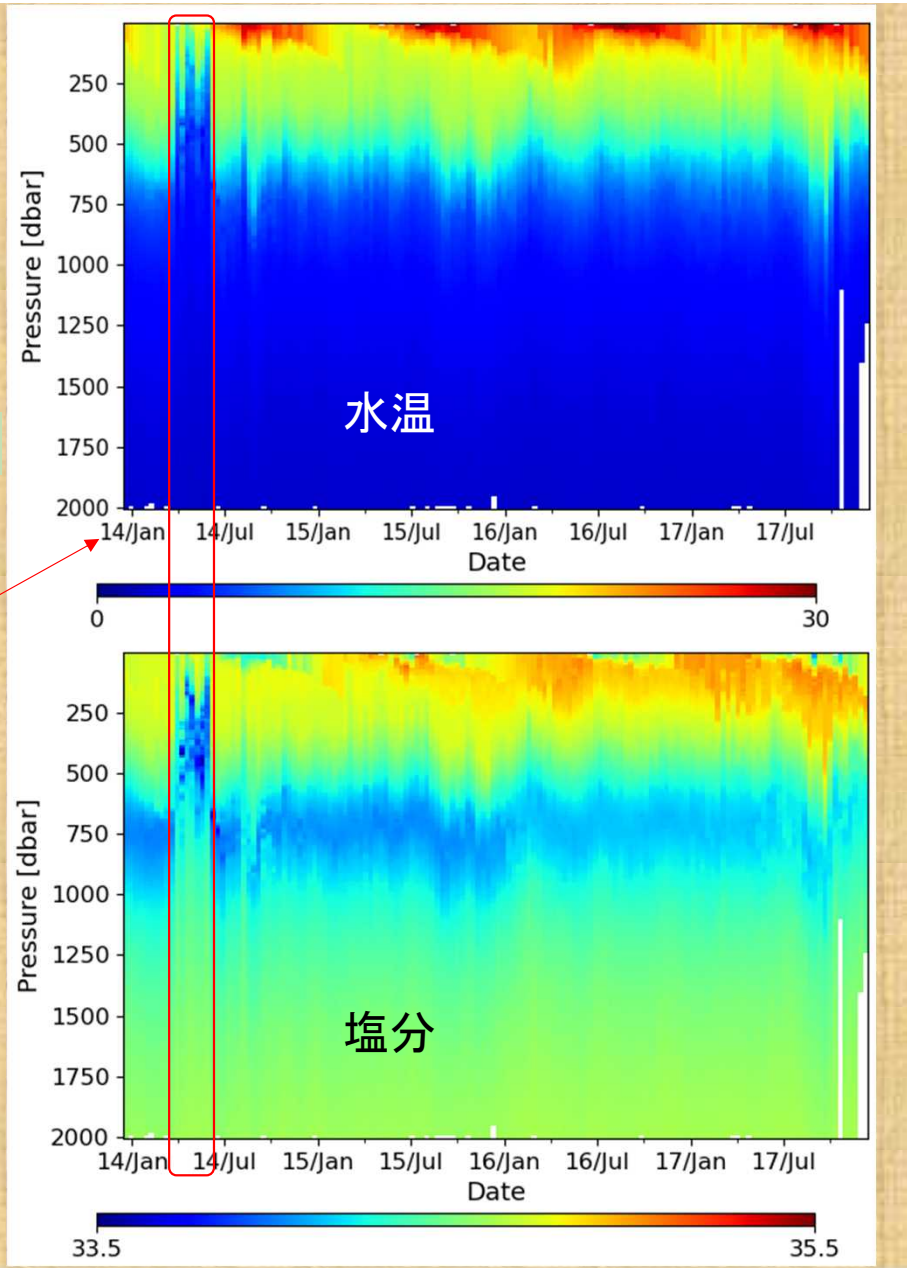
```
juld = float(nc.variables['JULD'][:])
d = datetime.datetime(1950, 1, 1, 0, 0, 0, 0)
                + datetime.timedelta(days=juld)
day.append(d)
```

```
clrm = (0, 30)
fig = plt.figure()
ax = fig.add_axes([0.13, 0.19, 0.83, 0.72])
sc = ax.pcolormesh(day, pres, temp.T, cmap=plt.cm.jet, ¥
      vmin=clrm[0], vmax=clrm[1], shading='nearest')
ax.set_xlabel('Date', fontsize=13)
ax.set_ylabel('Pressure [dbar]', fontsize=13)
ax.tick_params(axis='both', labelsize=11)
ax.invert_yaxis()
```

行列：観測回数×層の数

```
import matplotlib.dates as mdates
ax.xaxis.set_major_formatter(mdates.DateFormatter('%y/%b'))
ax.xaxis.set_minor_locator(mdates.MonthLocator([1, 4, 7, 10]))
```

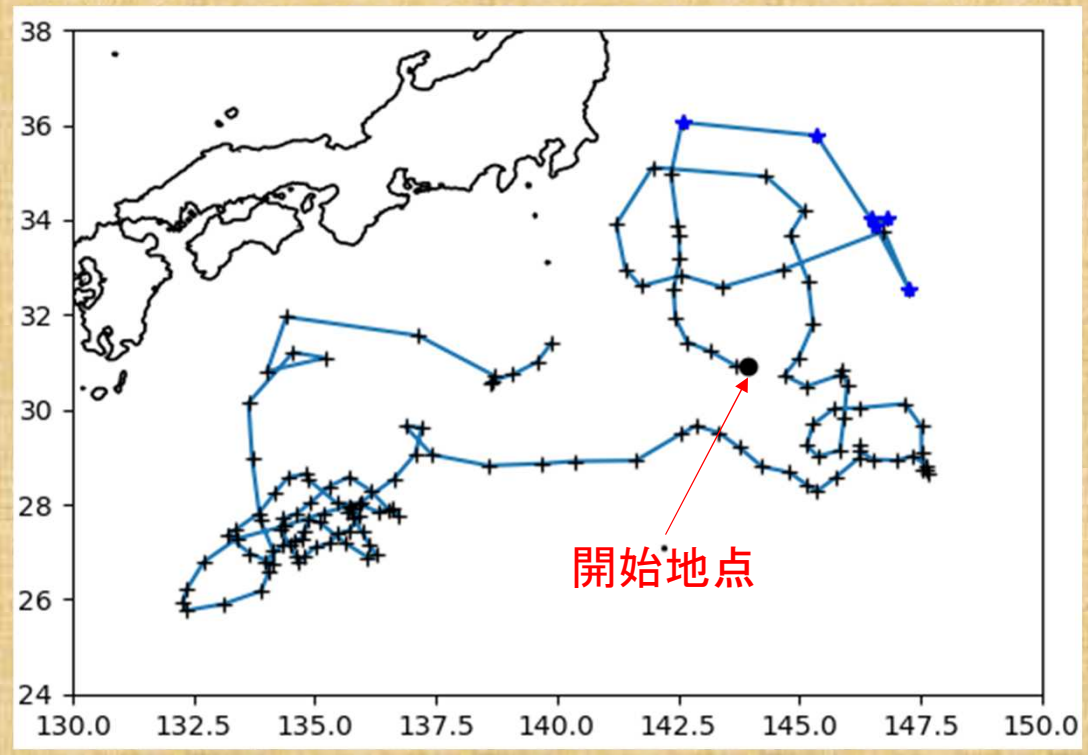
```
#  カラーバー
cax = fig.add_axes([0.1, 0.06, 0.8, 0.02])
cbar = fig.colorbar(sc, cax=cax, orientation="horizontal")
cbar.set_ticks([clrm[0], clrm[1]])
cbar.ax.tick_params(labelsize=12)
plt.show()
```



※ 年、月、日、時刻を得る

```
import netCDF4
import datetime

juld = float(nc.variables['JULD'][:])
ut = datetime.datetime(1950, 1, 1, 0, 0, 0, 0)  + datetime.timedelta(days=juld)
year          = int( ut.strftime('%Y') )   # 西暦年
month         = int( ut.strftime('%m') )  # 月
day           = int( ut.strftime('%d') )   # 日
jday          = int( ut.strftime('%j') )   # 年初からの通算日
hour          = int( ut.strftime('%H') )  # 時間
minute        = int( ut.strftime('%M') ) # 分
```



WMOID 2902474のフロート

```
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.plot(lon, lat)
ax.plot(lon, lat, '+k')
ax.plot(lon[0], lat[0], 'ok')
ax.plot(lon[10:16], lat[10:16], '*b')
ax.contour(lon1, lat1, map, 0, linewidths=1.2, colors='k')
ax.set_xlim([130, 150])
ax.set_ylim([24, 38])
ax.set_aspect('equal')
plt.show()
```

※　地図の描き方は今回は触れません

断面図を描く時のtips

- 鉛直層の数・深度は、同じフロートでも全ての観測（プロファイル）で同じとは限らない
- 不良データが混じることもある

- 不良データを取り除いたのち、統一された深度面に内挿するとよい

（例）不良データの除去

```
temp = np.squeeze( nc.variables['TEMP_ADJUSTED'][:] ).tolist(np.nan)
qc = np.squeeze( nc.variables['TEMP_ADJUSTED_QC'][:] )
for idx, n in enumerate(qc):
        if int(n) != 1: temp[idx] = np.nan     # QCフラグ1以外は欠損値に
dat0 = np.vstack( [pres, temp] )                 # 1つの行列にまとめる
dat = dat0[:, ~np.isnan(dat0).any(axis=0)]     # 欠損値の要素を除去
```

（例）

```
from scipy import interpolate
import os

# 深度10dbarから2000dbarまで10dbar毎の深度
prs_ref = np.arange(10, 2010, 10)
num = 0
temp_itp = []

filename = 'D2902474_{:0>3d}.nc'
while num < 1000:
   fname = filename.format(num)

   if os.path.isfile(fname):     # ファイルが存在する場合
      nc = netCDF4.Dataset(fname, 'r')
      #   途中省略

      f = interpolate.Akima1DInterpolator(dat[0,:], dat[1,:])
      temp_itp.append( f(prs_ref) )              # 補間

   num += 1
```

※　まれにN_PROFが1でないファイルがある（1つのファイルに2つ以上のプロファイルがはいっている）ので注意

Pythonで GSW(Gibbs SeaWater) Oceanographic Toolbox を使う

- 海水密度、力学高度、等々の海洋学で使う様々な量を計算できる非常に便利なtoolbox https://www.teos-10.org/software.htm
- Pythonを含め計8つのプログラミング言語対応
- Pythonの場合、conda-basedディストリビューションを「強く」推奨

  > conda install -c conda-forge gsw

Pythonで中立密度を求める（やや中級者編）

- 中立密度（γ_n: Jacket and McDougall 1997, JPO）を求める関数はGSW Toolboxに入っていない

- 公開されているプログラムはMATLAB用とFortran用だけ（2022年5月現在）

- Pythonのパッケージoct2pyを使って、pythonからGNU octave（MATLABと互換性のある無料ソフト）を呼び出してmファイルを動かす

  （※ fortranを呼び出す方法もあり）

- まずはGNU octaveをインストール（～-installer.exeでよい）
  https://www.gnu.org/software/octave/download.html

oct2pyを使うための準備:

Pythonにoct2pyをインストール

> conda install -c conda-forge oct2py　または　pip install oct2py

PATH設定（Windows 10の場合）

- コントロールパネル　→　システム　→　詳細情報　→　システムの詳細設定（あるいはウィンドウズキー　＋　Pause/Breakキー）　で「システムのプロパティ」パネル
- 詳細設定　→　環境変数　→　変数 Pathに　C:¥Program Files¥GNU Octave¥Octave-6.4.0¥mingw64¥bin　を追加する

関数eos80_legacy_gamma_nを使うための準備:

- http://www.teos-10.org/preteos10_software/neutral_density.html　からMATLAB用のToolbox（eos80_legacy_gamma_n.zip）をダウンロードして解凍
- ディレクトリ eos80_legacy_gamma_n を C:¥Program Files¥GNU Octave¥Octave-6.4.0¥mingw64¥share¥octave¥6.4.0¥m に置く

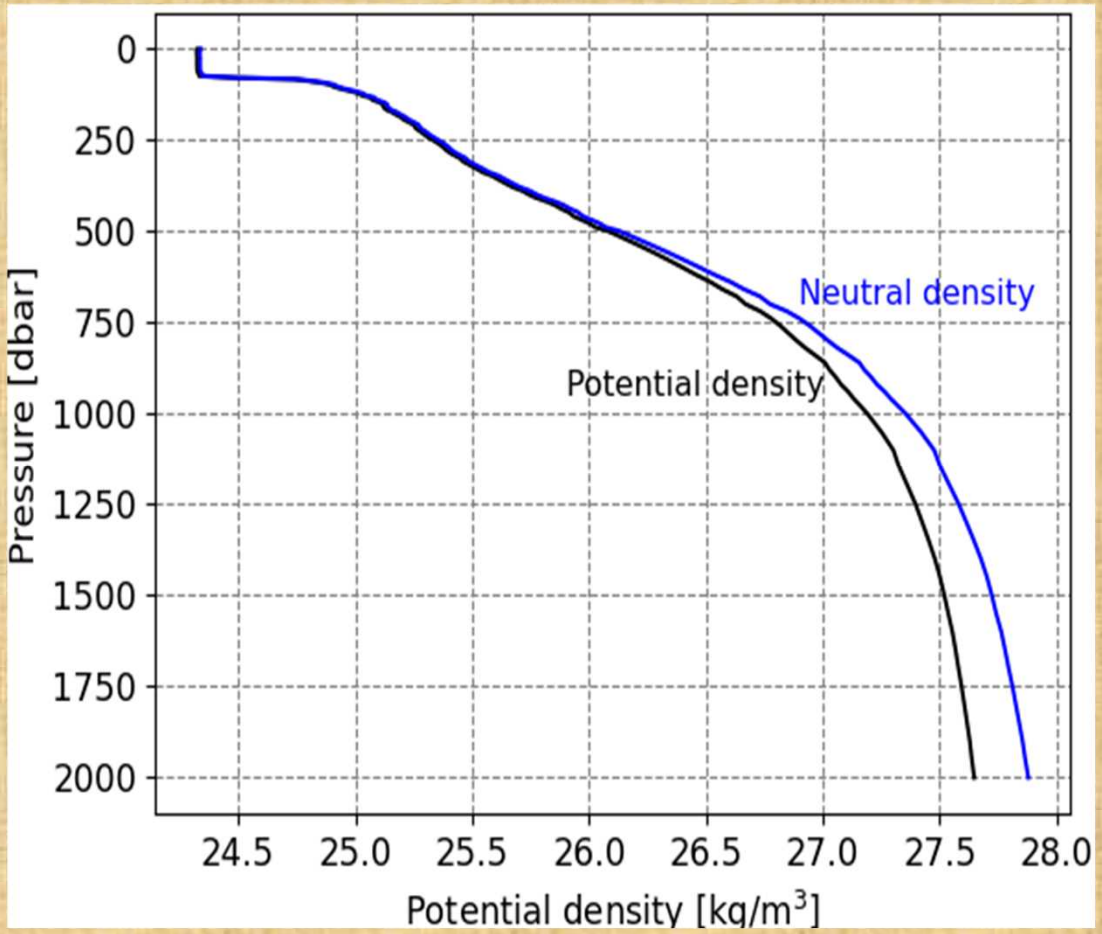
```
import gsw
from oct2py import octave

# データ読み込み部分は省略

# 中立密度
nd = np.squeeze( octave.eos80_legacy_gamma_n(sal,
temp, pres, lon, lat) )

# ポテンシャル密度
SA = gsw.SA_from_SP(sal, pres, lon, lat)
pd = gsw.pot_rho_t_exact(SA, temp, pres, 0.0) - 1000.0

# グラフを描く
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.plot(pd, pres, color = 'k')
ax.plot(nd, pres, color = 'b')
ax.grid(which='major', color='gray', linestyle='dashed')
ax.invert_yaxis()
plt.show()
```



北緯30度56分、東経143度56分の
2013年12月20日の鉛直密度分布

Argoデータも各種Toolboxもタダで手に入ります

グラフを描いて、海の中がどうなっているのか
自分で実際に見てみましょう