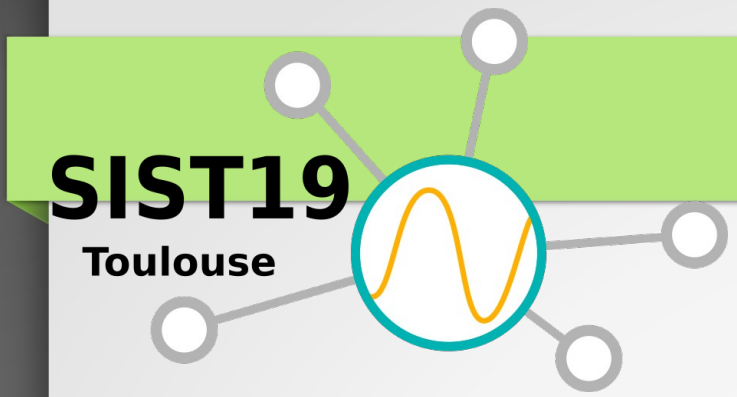

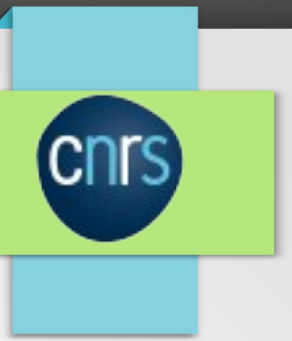

SIST 2019

Générer des fichiers NetCDF avec Python

M. Libes, D. Mallarino

Jeudi 7 novembre 2019 Observatoire Midi-Pyrénées deToulouse

sommaire

- Installation de l'API NetCDF4 - Python
- Utilisation
  - Creation des dimensions
  - Creation des variables et leurs attributs
  - Écriture de l'entete
  - Ecriture des données

- Fonctions de base pour générer des fichiers NetCDF avec le langage Python.

La problématique

- Transformer des fichiers issus de capteurs ou d'appareil de mesure posé sur un point fixe : latitude/longitude constantes dansun format NetCDF interopérable

  - Exemples sur :
  - une série temporelle
    - Les données évoluent au cours du temps :
      - *FeatureType = TimeSeries*
  - Un profile vertical
      - Point fixe, temps fixe, profondeur variable
        - *FeatureType = Profile*

Installation de la librairie netCDF4

 # sudo apt-get install python3-netcdf4

- Depends: python3 (<< 3.6), python3 (>= 3.5~), python3-numpy (>= 1:1.10.0~b1), python3-numpy-abi9, python3:any (>= 3.3.2-2~), libc6 (>= 2.14), libcurl3-gnutls (>= 7.16.2), libhdf5-100, libnetcdf11 (>= 4.3.3), libsz2, zlib1g (>= 1:1.1.4)

  Description-en: Python 3 interface to the netCDF4 (network Common Data Form) library

NetCDF version 4 has many features not found in earlier versions of the library and is implemented on top of HDF5. This module can read and write files in both the new netCDF 4 and the old netCDF 3 format, and can create files that are readable by HDF5 clients. The API is modelled after Scientific.IO.NetCDF, and should be familiar to users of that module.

Most new features of netCDF 4 are implemented, such as multiple unlimited dimensions, groups and zlib data compression. All the new numeric data types (such as 64 bit and unsigned integer types) are implemented. Compound and variable length (vlen) data types are supported, but the enum and opaque data types are not. Mixtures of compound and vlen data types (compound types containing vlens, and vlens containing compound types) are not supported.

Rappel Entete netCDF

```
dimensions:
    depth = UNLIMITED ; // (99 currently)
    lenstation = 5 ;
variables:
    int time ;
        time:long_name = "date de prelevement" ;
        time:standard_name = "time" ;
        time:units = "minutes since 1970-01-01
00:00:00 UTC" ;
        time:origin = "01-JAN-1970 00:00:00" ;
        time:calendar = "standard" ;
    char stationname(lenstation) ;
        stationname:standard_name = "platform_name" ;
        stationname:long_name = "station name" ;
        stationname:cf_role = "profile_id" ;
    float latitude ;
        latitude:units = "degrees_north" ;
        latitude:standard_name = "latitude" ;
        latitude:axis = "Y" ;
        latitude:coverage_content_type =
"coordinate" ;
```

Rappel data netCDF

```
data:

 time = 23695903 ;

 stationname = "JULIO" ;

 latitude = 43.13 ;

 longitude = 5.25 ;

 depth = -2, -3, -4, -5, -5.9, -6.9, -7.9, -8.9, -9.9, -
10.9, -11.9, -12.9,
    -13.9, -14.9, -15.9, -16.9, -17.9, -18.8, -19.8, -20.8,
-21.8, -22.8,
    -23.8, -24.8, -25.8, -26.8, -27.8, -28.8, -29.8, -30.8,
-31.7, -32.7,
    -33.7, -34.7, -35.7, -36.7, -37.7, -38.7, -39.7, -40.6,
-41.7, -42.7,
    -43.6, -44.6, -45.6, -46.6, -47.6, -48.6, -49.6, -50.6,
-51.6, -52.6,
    -53.6, -54.6, -55.5, -56.5, -57.5, -58.5, -59.5, -60.5,
-61.5, -62.5,
```

- 4 phases principales (fonctions du programme)
- 1. Créer le dataset netCDF vierge
- 2. Créer les dimensions des variables (vecteurs, matrice)
  - *Create_Dimensions_NC(fileNC)*
- 3. Créer les variables dimensionnées
  - *Create_Variables_NC(fileNC)*
- 4. Ecrire les métadonnées dans l'entete netcdf
  - *Write_Metadata_NC(fileNC)*

5. Ecrire les données dans les variables

0. Création du dataset netcdf

- Importer la librairie netCDF4 dans le programme python
  - **`import netCDF4 as nc4`**

- 1. Créer le dataset netcdf “vide”
  - `outNetCDF=”/chemin/vers/outNetCDF”`
  - `NetCDFFileName=outNetCDF+”/”+nomfic+".nc"`
  - **`fileNC=nc4.Dataset(NetCDFFileName,"w", format="NETCDF4", encoding='latin-1')`**
    - `puis...`
    - *`Create_Dimensions_NC(fileNC)`*

1. Créer les dimensions des vecteurs de données

- *Cas d'un profil vertical*
  - *Point fixe, temps fixe*
  - *mesures le long de la profondeur*

- *def Create_Dimensions_NC(f):*
  - station_name="romarin"
  - lenstation=len(station_name)
  - **f.createDimension('depth',size=None)**
    - # unlimited pour recevoir les mesures sur le profil vertical
  - ***f.createDimension('lenstation',lenstation)***
    - # longueur chaine de caractere station

1. Créer les dimensions des vecteurs de données

- *Cas d'une série temporelle*
  - *Point fixe, mesures en fonction du temps*

- *def Create_Dimensions_NC(f):*
  - time='UNLIMITED'
  - station_name="julio"
  - lenstation=len(station_name)
  - **f.createDimension('*time*',size=None)** # unlimited pour les mesures de temps
  - **f.createDimension('*lenstation*',lenstation)** # longueur chaine de caractere station

2. Créer les variables dimensionnées

- **def Create_Variables_NC(f):**
- **stationname = f.createVariable('stationname', 'S1', ('lenstation',))**
  - stationname.***standard_name*** = 'platform_name'
  - stationname.long_name = 'station name'
  - ***stationname.cf_role = 'timeseries_id'*** *#obligatoire convention CF*

- **lats = f.createVariable('latitude','f4')**
  - lat.units = 'degrees_north'
  - lat.**standard_name** = 'latitude'
  - lat.axis = "Y"
  - lat.coverage_content_type = "coordinate" ;

- **lons = f.createVariable('longitude','f4')**
  - lon.units = 'degrees_east'
  - lon.standard_name = 'longitude'
  - lon.axis = "X"
  - lon.coverage_content_type = "coordinate" ;

2. Créer les variables dimensionnées

- **NC_TIME_FMT = 'minutes since 1970-01-01 00:00:00 UTC'**
- **time = f.createVariable('time', 'i4')**
  - time.long_name = "date de prelevement"
  - time.standard_name = "time"
  - ***time.units = NC_TIME_FMT***
  - time.origin = "01-JAN-1970 00:00:00"
  - time.calendar = 'standard'
- **temp = f.createVariable('temperature','f4',('depth',))**
  - temp.standard_name="sea_water_temperature"
  - temp.units = "Celsius"
  - temp.FillValue = -999.0
  - temp.long_name = "Temperature"
  - temp.source = "Seabird CTD"

3. Ecrire les métadonnées dans l'entete

- **def Write_Metadata_NC(f):**
-
- **f.description** = "CTD profile (NetCDF files) for station "+station_name
- f.title = "CTD profile (NetCDF files) station "+station_name+" - Service d'Observation en Milieu Littoral (SOMLIT) "
- ***f.keywords*** = "Seabird CTD temperature, conductivity, fluorimetry, salinity, transmission, density, oxygen "
- f.history = "Created " + today.strftime("%d/%m/%y")
- f.production = "UMR 7999 CNRS / OSU Pytheas UMS3470 CNRS"
- f.contact = "Christian Gronz (christian.gronz@labo.fr)"
- f.summary = "Mesures CTD : temperature, salinity, oxygen, fluorescence et turbidity 2 profils 0-50 m par jour "
- ***f.featureType = "profile"***
- ***f.cdm_data_type = 'profile'***
- ***f.cdm_profile_variables = "depth,latitude,longitude" ;***
- f.Conventions = 'CF-1.6, ACDD-1.3'

3. Ecrire les métadonnées dans l'entete

- f.creator_name = 'MIO Marine Institute of Oceanography UMR 7294 CNRS'
- f.creator_email = 'maurice.libes@osupytheas.fr'
- f.creator_url = 'http://www.osupytheas.fr'
- f.creator_type = 'institution'
- f.creator_institution = 'OSU Pytheas UMS 3470 CNRS'
- f.contributor_role = 'data formatting in netCDF
- f.contributor_url = 'http://www.osupytheas.fr'
- **f.license = " data are accessible freely and free of cost, for public research and teaching applications...etc..**
- f.contributor_institution = 'MIO UMR 7294 CNRS / OSU Pytheas UMS 3470 CNRS'
- f.institution = 'MIO UMR7294 CNRS / OSU Pytheas'
- f.project = 'SOMLIT'
- f.publisher_name = 'MIO Marine Institute of Oceanography - OSU Pytheas'
- f.publisher_email = 'maurice.libes@osupytheas.fr'
- f.publisher_url = 'http://www.osupytheas.fr'
- f.publisher_institution = 'MIO UMR 7294 CNRS / OSU Pytheas UMS 3470 CNRS'
- f.program = 'SOMLIT Service d'Observation en Milieu Littoral'

4. Ecrire les données dans les variables netcdf

- *variable_netcdf = vecteur/tableau de données*
  - *Syntaxe: variable_netcdf[:] = tableau_donnees*

- `nomstation="JULIO" #string`
- `stationname[:] = nc4.stringtoarr(nomstation,len(nomstation))`
- `dateprofile="2017-07-12T10:30:01"`
- `time[:] = nc4.date2num(dateprofile, units=NC_TIME_FMT)`
- `lats[:]=latitude`
- `lons[:]=longitude`

4. Ecrire les données dans les variables netcdf

- *On ecrit les tableaux/listes de données dans les variables netcdf*
- *variable_netcdf = vecteur/tableau de données*

  tab_temp=[37.2,37.3,38.1,36.4...]

  - `temp[:] = tab_temp`
  - `conductivity[:] = tab_conductivity`
  - `salinity[:] = tab_salinity`
  - `irradiance[:] = tab_irradiance`
  - `fluorescence[:] = tab_fluorescence`
  - Etc..
  - fileNC.close()

Liens utiles

- Tester votre fichier netCDF : CF-Convention Compliance Checker for NetCDF Format
  - https://pumatest.nerc.ac.uk/cgi-bin/cf-checker.pl
- Documentation de l' API Python interface à la librairie netCDF
  - http://unidata.github.io/netcdf4-python/netCDF4/index.html
- Les attributs netCDF
  - http://cfconventions.org/Data/cf-conventions/cf-conventions-1.7/build/apa.html
  -