Institut Universitaire de Technologie,
Aix-Marseille Université

RAPPORT DE STAGE de fin de deuxième année
Bachelor Universitaire de Technologie
Spécialité Réseaux et Télécommunications
parcours cybersécurité

Gestion de données scientifiques au format NetCDF
Interopérabilité et Programmation Python

Killian PAVY

OSU Institut Pythéas

Responsable entreprise : Maurice LIBES

Responsable académique : Djamal MERAD

2023

Table des matières

1. Introduction

Ce rapport porte sur mon stage de 10 semaines réalisé du 17 avril au 23 juin au sein de l'Observatoire des Sciences de l'Univers (OSU) Institut Pythéas. J'ai eu l'opportunité de contribuer activement au traitement et à la gestion des données scientifiques dans le cadre du projet "European Multidisciplinary Seafloor and water column Observatory" (EMSO) du M.I.O (Institut Méditerranéen d'Océanologie) et du service d'observation de l'OSU.

Mon stage s'est concentré sur la mise en place d'outils pour garantir la qualité des données et faciliter leur interopérabilité. Dans ce contexte j'ai été chargé :

- Dans un premier temps, d'améliorer la qualité des données existantes depuis 2017 au format NetCDF en leur ajoutant des métadonnées standardisées au niveau européen.

- Dans un second temps, j'ai été chargé d'analyser différents formats de fichiers bruts provenant des capteurs de la station instrumentées EMSO Ligure Ouest (située à 2500m de profondeur au large de Toulon). J'ai traité en Python les fichiers bruts provenant des capteurs afin de les uniformiser en format NetCDF et d'améliorer la cohérence et l'interopérabilité des données collectées.

Dans un premier temps, nous fournirons une présentation de l'Institut et de ses activités. Ensuite, nous aborderons le contexte technique global du sujet avant de détailler le travail effectué.

2. Présentation de l'institut

2.1 Ses laboratoires et ses missions

L'Observatoire des Sciences de l'Univers (OSU) Pythéas est un institut qui dépend de l'université d'Aix-Marseille. Il est affilié au Centre National de la Recherche Scientifique (CNRS), à (Institut de Recherche pour le Développement (IRD) et à l'Institut national de recherche pour l'agriculture, l'alimentation et l'environnement (INRAE). Il se consacre à des domaines de recherche majeurs tels que les sciences de la Terre, l'environnement et l'univers.

L'Observatoire des Sciences de l'Univers (OSU) Institut Pythéas a été créé en janvier 2012 et fédère six grandes unités mixtes de recherche :

- CEREGE - Centre Européen de Recherche et d'Enseignement des Géosciences de l'Environnement,
- IMBE - Institut Méditerranéen de Biodiversité et d'Écologie marine et continentale,
- LAM - Laboratoire d'Astrophysique de Marseille,
- LPED - Laboratoire Population Environnement Développement,
- MIO - Institut Méditerranéen d'Océanologie.
- RECOVER - Risques, ECOsystèmes, Vulnérabilité, Environnement, Résilience

Les objectifs du M.I.O sont de mieux comprendre le système océanique et son évolution en réponse au changement global. Il constitue un pôle de compétences en biologie, écologie, biodiversité, microbiologie, halieutique, physique, chimie, biogéochimie et en sédimentologie marines. Ses cadres d'exercice sont l'océan mondial, ses interfaces avec le continent, l'atmosphère et le sédiment. Quant à l'OSU, ses missions sont de fournir le soutien aux programmes de recherche des laboratoires et de s'occuper de la gestion des données.

Le laboratoire se divise en 5 équipes :

- Océanographie Physique, Littorale et Côtière
- Chimie des Environnements Marins
- Microbiologie Environnementale Biotechnologie
- Cycle Biogéochimiques et rôle fonctionnel des assemblages de micro-organismes planctoniques
- Ecologie Marine et Biodiversité

Personnellement, j'ai travaillé au M.I.O avec le responsable de la gestion des données, M. Maurice LIBES Ingénieur en Système d'Information Géographiques du service d'Observation de l'OSU et le responsable scientifique M. Dominique LEFEVRE.

3. Sujet du Stage

3.1 Contexte : le projet EMSO

De nos jours dans le milieu scientifique, la collecte et l'analyse des données océanographiques revêtent une importance capitale pour comprendre et préserver les écosystèmes marins. Cependant, les données océanographiques sont souvent issues de capteurs aux conceptions et marques très différentes qui fournissent des données brutes très hétérogènes, stockées dans des formats différents et provenant de sources variées. Dans ce cas il est illusoire de pouvoir utiliser et partager ces données. Il faut donc utiliser des formats de fichiers standardisés (comme NetCDF) ainsi que de nombreuse métadonnées, qui vont fournir tous les renseignements sur la mesure des données.

Le projet EMSO est un réseau Européen d'observatoires du fond de mer et de la colonne d'eau au point fixe qui a pour objectif scientifique d'observer en temps réel les processus environnementaux liés avec les interactions entre géosphère, biosphère et hydrosphère. EMSO vise à acquérir des séries temporelles longues (10 à 20 ans au moins) et à offrir des données de haute qualité et interopérables dans les mers du pourtour européen, de l'Arctique à l'Atlantique, de la Méditerranée à la mer Noire avec pour objectifs principaux :

- L'étude de l'impact du réchauffement climatique sur les océans entourant l'Europe
- L'étude des écosystèmes marins profonds dans une optique de recherche fondamentale mais aussi de gestion durable, en s'intéressant particulièrement aux facteurs anthropogéniques et climatiques
- L'étude des processus tectoniques, volcaniques, hydrothermaux et gravitaires et la surveillance des risques naturels associés (séismes, tsunamis, instabilité des pentes) pour les zones côtières à forte densité de population.

EMSO est constitué en ERIC (European Research Infrastructure Consortium) depuis septembre 2016, compte 8 membres : Italie, France, Irlande, Espagne, Roumanie, Grèce, Royaume-Uni et Portugal.

Afin de collecter ces données d'observations, le M.I.O participe et utilise l'observatoire sous-marin "Mediterranean Eurocentre For Underwater Sciences and Technologies" (MEUST). Cet observatoire sous-marin profond accueille :

- Une ligne de mouillage instrumentée inductive (ALBATROSS, Voir Annexes, Figure 1) a été installée à 2400m de profondeur au large de Toulon
- Un module d'interface instrumenté fixe posé au fond de la mer appelée MII (Voir Annexes, Figure 2). Voici à quoi ressemble l'installation :

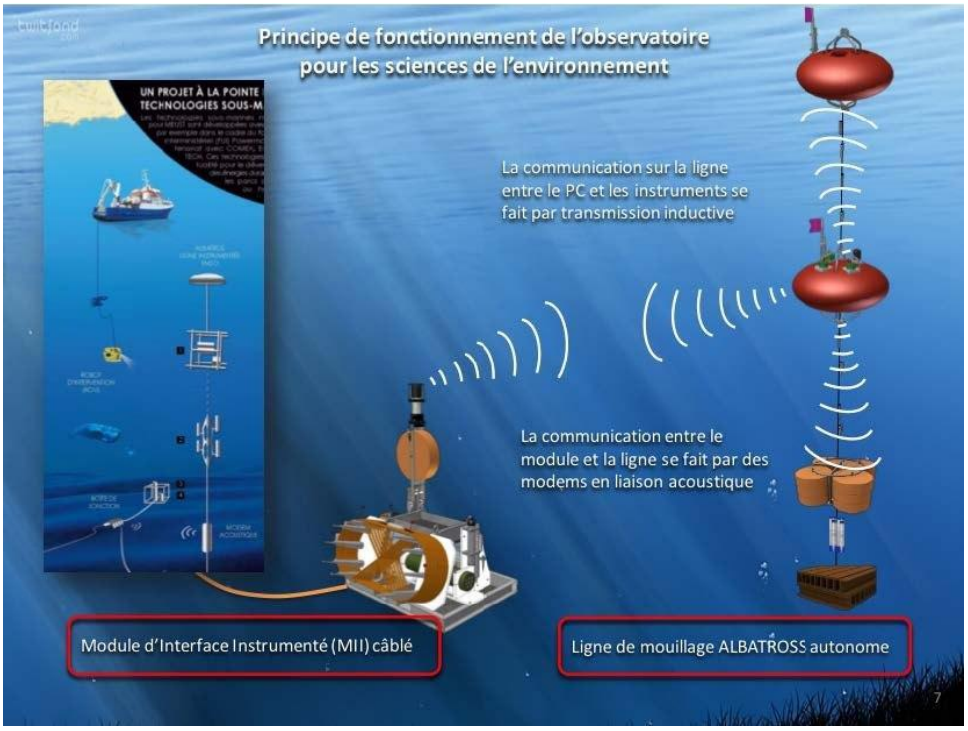


Figure 3 : Plateforme MEUST

Mon travail dans ce projet a été de convertir des fichiers de données bruts provenant de capteurs "in situ", au format NetCDF avec des métadonnées standardisées provenant de thesaurus Européens "OceanSite" http://www.oceansites.org/data/index.html et NERC http://vocab.nerc.ac.uk/collection/

Attributes	Attribute name/value	Notes	Required
date_created	2020-05-19	specific for each dataset	X
Conventions	OceanSITES v1.4,SeaDataNet_1.0,COARDS, CF-1.6	prefiled if we decide to go along with these proposed standards	x
institution_edmo_code	3917	each institution has a EDMO code and must be search through	x
institution_edmo_uri	https://edmo.seadatanet.org/report/3917		x
insitution_ror_uri	https://ror.org/04xkqms46		x
geospatial_lat_min	37.54765	specific for each dataset	X
geospatial_lat_max	37.54765	specific for each dataset	X
geospatial_lon_min	15.3975	specific for each dataset	X
geospatial_lon_max	15.3975	specific for each dataset	X
geospatial_vertical_min	2036	specific for each dataset	X
geospatial_vertical_max	2036	specific for each dataset	X
time_coverage_start	2012-06-10T00:00:47Z	specific for each dataset	X

Figure 31 : Exemple de métadonnées demandées issue d'un thesaurus Européen

3.2 Les travaux effectués : Outils et méthodologie utilisée

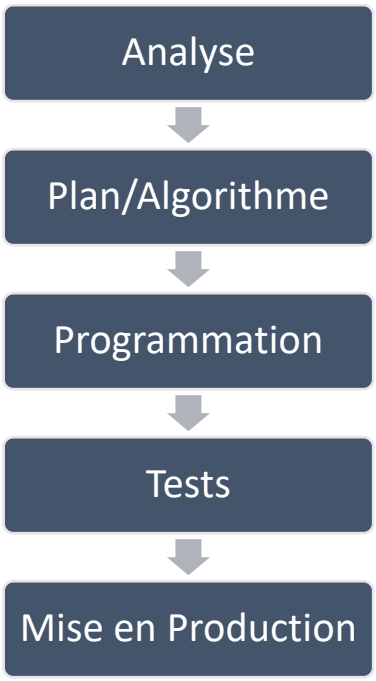
Lors de mon stage, j'ai utilisé divers outils et technologies pour mener à bien mon travail. Pour la gestion de version et la collaboration, j'ai utilisé GitLab, une plateforme de développement qui m'a permis de travailler efficacement et de suivre l'évolution de mon code, et de le partager avec le responsable du projet.

Pour traiter les données au format NetCDF, j'ai exploité la bibliothèque NetCDF4 pour Python, qui m'a offert une interface pratique pour manipuler et analyser les données NetCDF dans mon environnement de programmation. Une fois les fichiers NetCDF produits, il est possible de les visualiser en ligne de commande avec la commande "ncdump" ou bien graphiquement avec le

logiciel “panoply” ou sur le serveur de données ERDDAP (Environmental Research Division Data Access Protocol). J'ai également utilisé d’autres bibliothèques comme « xarray » car elle offre des fonctionnalités que NetCDF4 ne propose pas comme la suppression de variables.

De plus, j'ai utilisé deux validateurs/checkers NetCDF qui m'ont aidé à vérifier la conformité de mes fichiers par rapport aux conventions internationales et à détecter d'éventuelles erreurs.

Voici la Méthodologie de travail que j’ai adopté :



3.2.1 La Plateforme Gitlab

Ces commandes ont été essentielles pour maintenir une collaboration fluide et une gestion efficace des versions tout au long du projet sur la plateforme GitLab :

- **`git clone`** : Cette commande nous a permis de cloner le dépôt GitLab sur notre machine locale, établissant ainsi une copie intégrale du projet.
- **`git pull`** : Utilisée pour récupérer les dernières modifications du dépôt distant et mettre à jour l'environnement de travail local.
- **`git status`** : Permet de visualiser les fichiers modifiés, non suivis et les conflits éventuels dans le projet.
- **`git commit -a`** : Cette commande permet de valider tous les fichiers modifiés dans le dépôt local avec un message explicatif.
- **`git push`** : Utilisée pour pousser les commits vers le dépôt distant, permettant aux autres membres de l'équipe de récupérer les dernières mises à jour et de continuer à travailler.

3.2.2 Serveur de données ERDDAP

ERDDAP est un serveur de données qui offre un moyen simple et cohérent de diffuser et télécharger des sous-ensembles de données scientifiques dans des formats de fichiers courants et de créer des graphiques et des cartes.

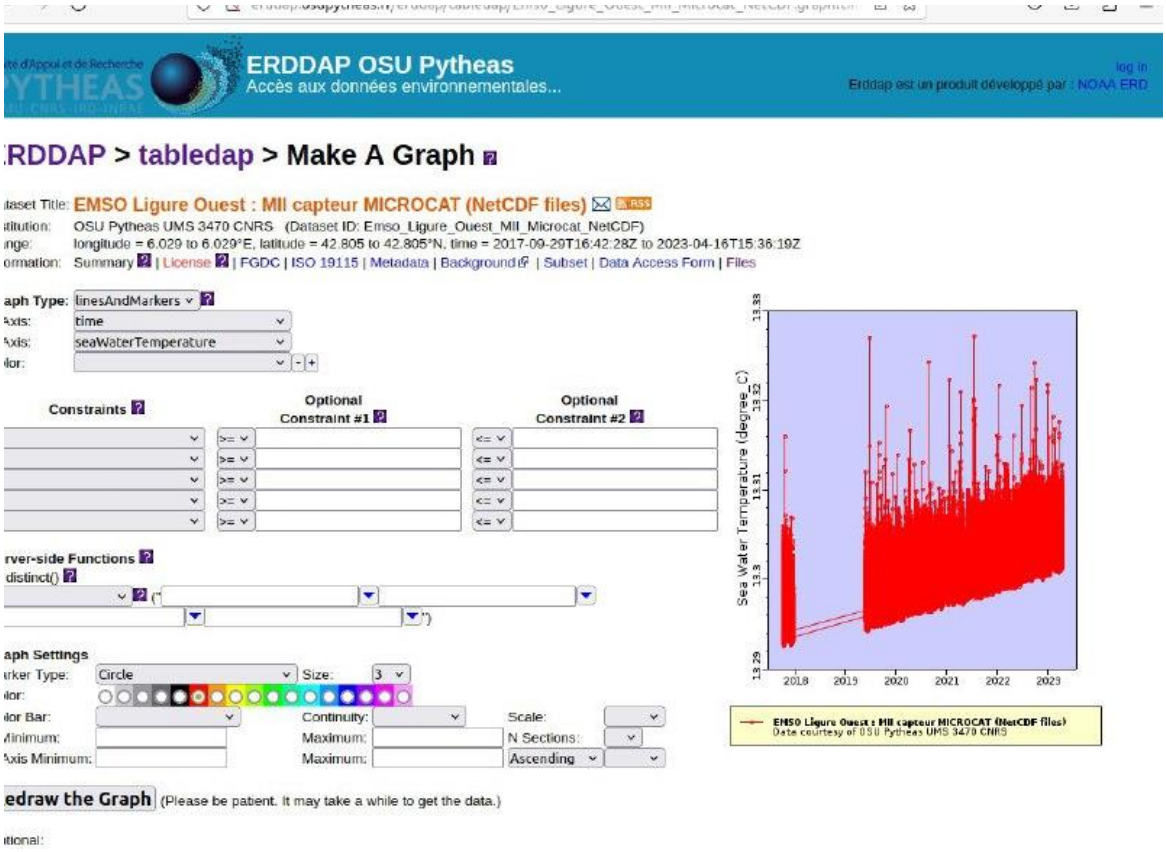


Figure 9 : Graphique Evolution Températures 2018-2023 à 2436m de profondeur

Au cours de mes travaux, j'ai été amené à transférer à plusieurs reprises des données retravaillées afin d'assurer leur bon fonctionnement et leur conformité aux exigences de mon étude. Pour cela, j'ai utilisé des outils tels que SSH (Secure Shell) et rsync (Remote Synchronization), qui m'ont permis d'effectuer des transferts sécurisés et efficaces entre mon PC et le serveur de données.

3.2.3 Le Format NetCDF

Durant tout le stage, j'ai travaillé sur des fichiers au format NetCDF, que ce soit pour convertir des fichiers issus de capteurs, d'appareils de mesures ou bien pour améliorer des fichiers NetCDF déjà existants.

Le format NetCDF est un format binaire autodescriptif, qui permet de représenter des données qui évoluent en fonction de certaines dimensions. On peut ainsi représenter des données en 1D y=f(x) qui évoluent en fonction du temps ou de la profondeur par exemple, ou bien 2D y=f(x,z) où les données évoluent en fonction d'une position géographique latitude/longitude par exemple

Il est largement utilisé dans le monde scientifique à l'international pour ses qualités descriptives qui permettent une interopérabilité et une réutilisation des données, chaque variable est décrite par des metadata et identifiée par un code renvoyant vers une documentation complète. Un fichier NetCDF se compose en plusieurs parties :

- Une première partie est consacrée aux dimensions selon lesquelles évolue le jeu de données, Par exemple quand il s'agit de séries temporelles, les données évoluent sur un seul axe, le temps.
- Une deuxième partie sert à décrire chaque variable est décrite avec ses attributs propres (type, unité, nom standardisé, identifiant).
- Une troisième partie sert à stocker des « attributs globaux » qui sont utiles pour ajouter des informations sur le jeu de données (auteur, licences, dates, résumé, méthodes, Digital Object Identifier DOI...)
- Enfin une dernière partie sert à stocker les données.

Voici un schéma représentant la structure d'un fichier NetCDF :

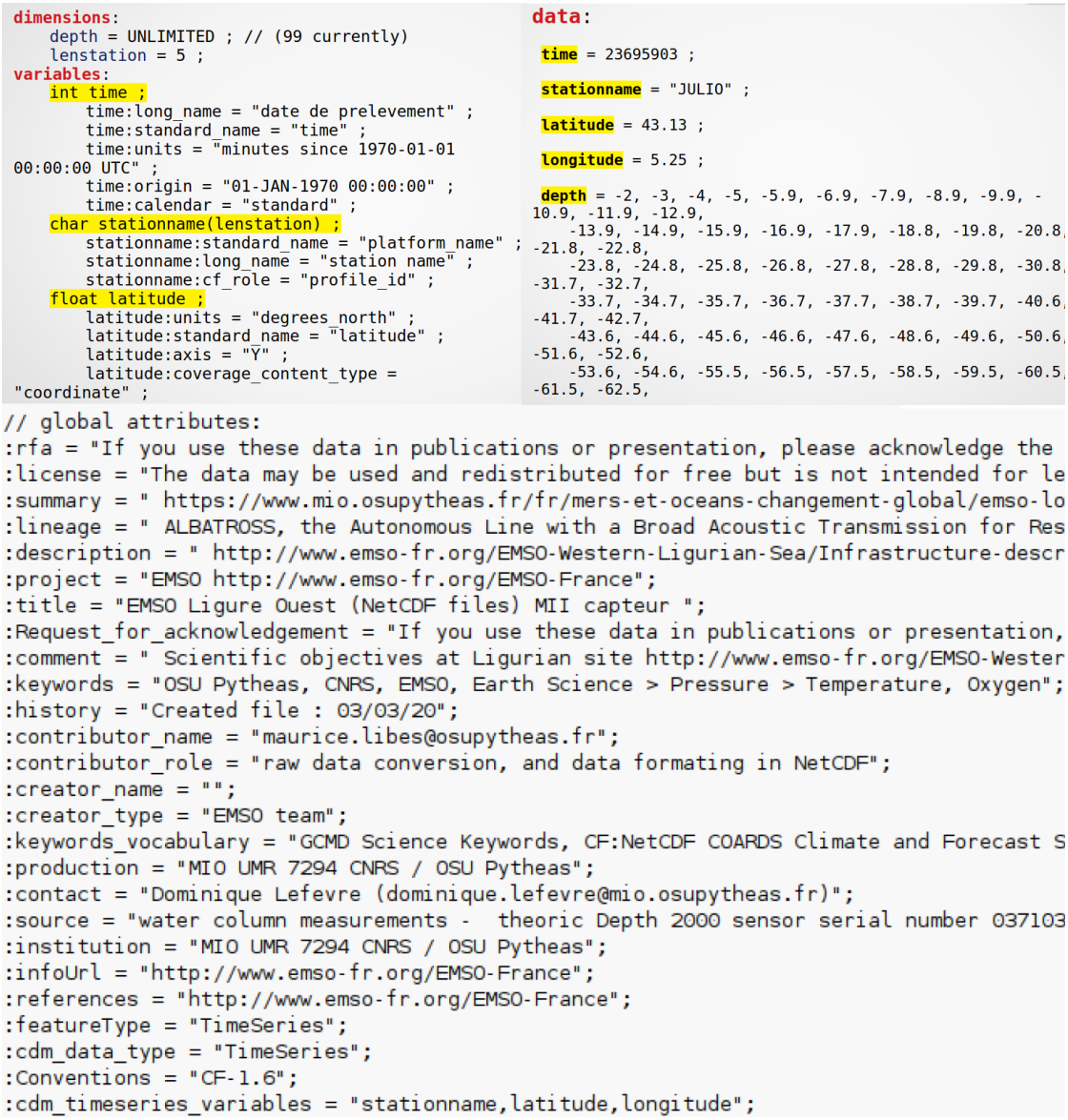


Figure 5 : Structure d'un fichier NetCDF

L'affichage d'un fichier NetCDF peut se faire en ligne de commande grâce à la commande « ncdump » :

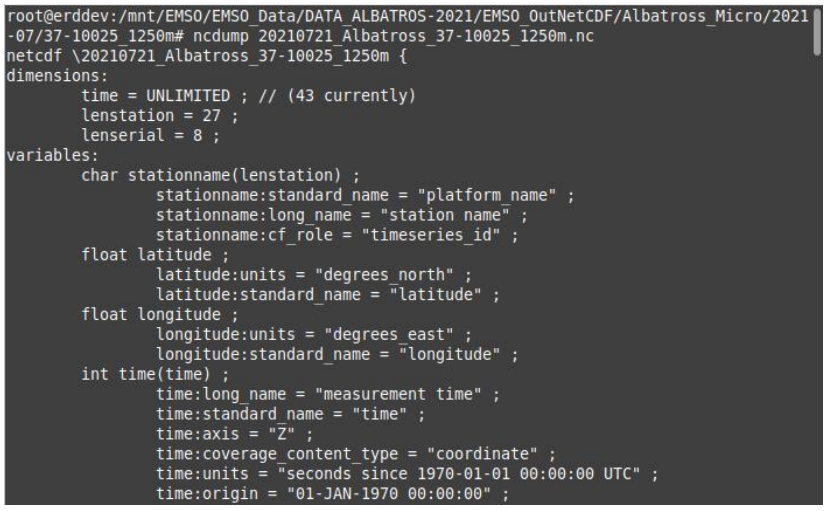


Figure 6 : Fichier NetCDF avec ncdump

Ou bien il peut se faire graphiquement avec « panoply », qui permet en plus l'affichage de graphiques :

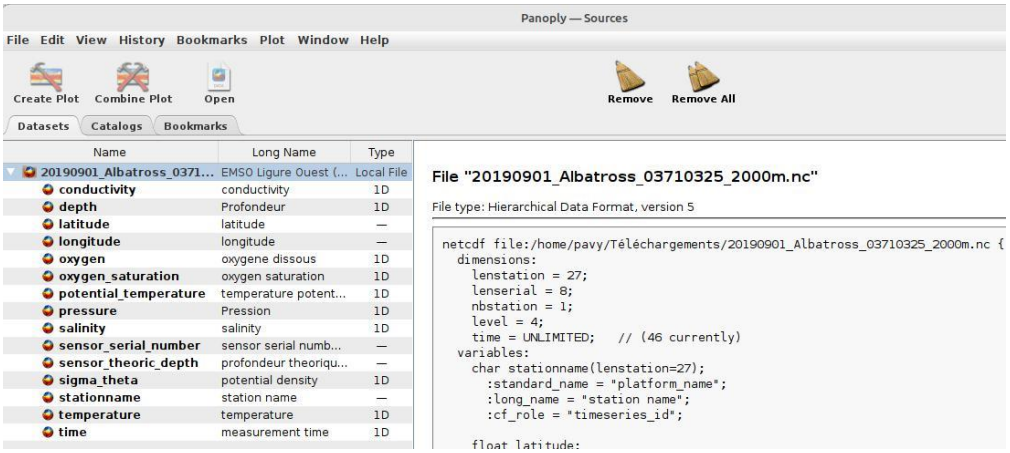


Figure 7 : Fichier NetCDF sur Panoply

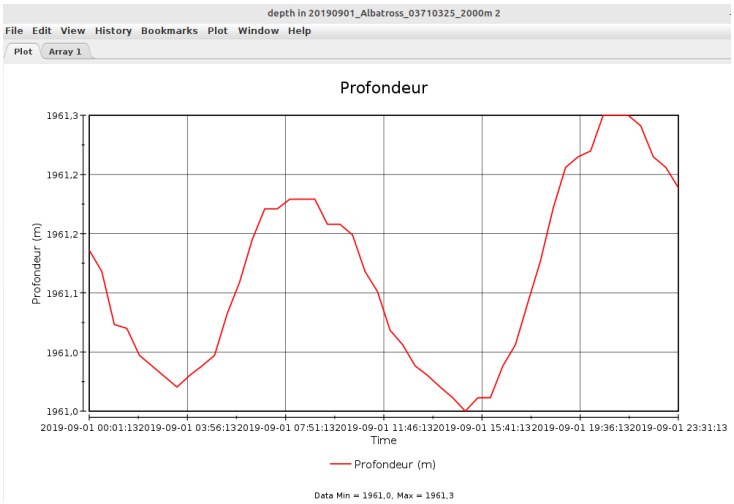


Figure 8 : Graphique Profondeur sur Panoply

3.2.4 Le Format CSV

En Complément du format NetCDF, le format CSV (Comma Separated Values) est mis à disposition sur ERDDAP pour avoir une seconde option, un format plus simple et moins détaillé, facile à utiliser avec des logiciels de bureautique courants. Les fichiers .csv sont des fichiers utilisant un séparateur, souvent une virgule.

Ce format CSV n'est pas approprié à une gestion FAIR des données, car ce format ne permet pas de décrire convenablement les mesures et valeurs du fichier. C'est pourquoi le projet EMSO demande que toutes les données soient dans des fichiers NetCDF avec des métadonnées standardisées.

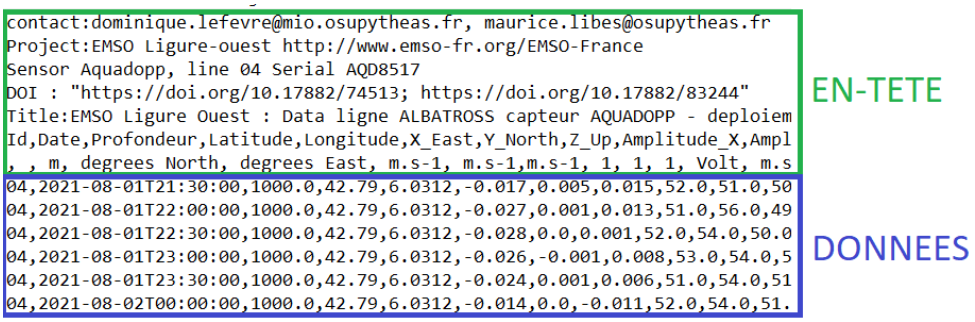


Figure 25 : Exemple Structure .csv

3.2.5 Les Checkers NetCDF

Ces checkers fonctionnent en examinant de manière automatisée les fichiers NetCDF pour s'assurer qu'ils respectent les règles et les conventions établies. Ils vérifient des éléments tels que la présence des variables et des dimensions requises, la cohérence des unités de mesure, la validité des métadonnées et la conformité aux conventions de nommage.

L'avantage des checkers NetCDF est qu'ils permettent de détecter rapidement les éventuelles erreurs ou incohérences dans les fichiers de données, ce qui facilite la correction et garantit la qualité des données utilisées dans le cadre du projet EMSO. Grâce à ces outils, j'ai pu m'assurer que les fichiers de jeu de données que j'ai modifiés respectaient les standards requis, assurant ainsi une meilleure intégration et une meilleure interopérabilité des données au sein du projet.

J'ai utilisé deux checkers au début du stage cependant j'ai rapidement cessé d'en utiliser un des deux car il n'était pas cohérent dans ces erreurs et surtout qu'il n'était pas spécifique au projet EMSO. Contrairement au deuxième checker « metadata-harmonizer » qui lui est spécifique aux métadonnées demandées par le projet EMSO. Dans cet exemple ci-dessous, les résultats sont mauvais, les valeurs ne sont pas des labels reconnus :

Direction	units	True	0	Not a valid 'P06' prefered label	degrees
Direction	comment	False	False	not found	
Direction	coordinates	True	False	not found	
Direction	ancillary_variables	False	1	ok	Direction_QC
Direction	_FillValue	False	False	not found	
Direction	reference_scale	False	False	not found	
Direction	sdn_parameter_name	True	0	Not a valid 'P01' prefered label	Direction (towards) of water current (Eulerian measurement) in the water body

Figure 28 : Exemples mauvais résultats checker

4. Le Projet EMSO

4.1 L'architecture du flux de données du projet

Les données brutes des différents capteurs de MII et d'ALBATROSS sont transmises quotidiennement depuis les capteurs a 2500m de fond, jusqu'au site du MIO sur Luminy.

Le MII récupère les données de la ligne ALBATROSS par transmission acoustique puis le tout est envoyé en temps réel au serveur d'acquisition de données DT INSU (Division Technique de l'INSU Division Technique de l'INSU) qui se charge de produire les fichiers bruts en formats .dat et .cnv selon les capteurs. Ces données brutes sont reçues quotidiennement sur le serveur ERDDAP du M.I.O.

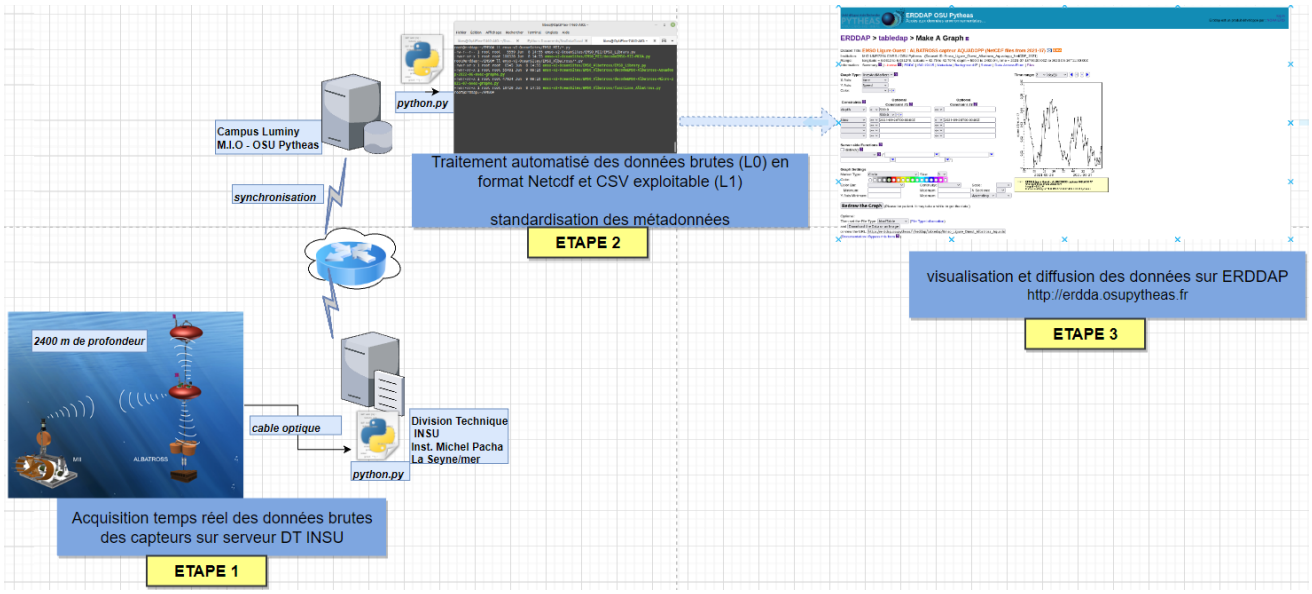


Figure 4 : Gestion des flux de données

En se référant au schéma, je suis intervenu sur l'étape 2 et 3 : programmation en Python pour ajouter les métadonnées et traiter les fichiers et placements des données sur la plateforme ERDDAP.

4.2 Etat initial du projet et ressources disponibles

4.2.1 Jeux de données à traiter par années

Les données de la plateforme instrumentée « MII » située à 2400m de profondeur se divisent en 4 capteurs : AQUADOPP, MICROCAT, CSTAR et OXYGEN. Elles commencent à partir de 2017 et ont déjà été traitées et converties en NetCDF et CSV, seulement les données étaient incomplètes ou mal remplies (ex : mauvais nom de variables).

La 1ère partie de mon projet a donc consisté à repasser sur des jeux de données déjà traités mais mal traités, c'est-à-dire que les fichiers de sortie étaient bien produits en .csv et .nc cependant plus ou moins tout le reste était à revoir : le nom des variables, attributs manquants, variables manquantes etc

Quant aux données de la ligne ALBATROSS, seuls les capteurs AQUADOPP et MICROCAT sont présents. Il y a une distinction entre avant et après 2019, en effet toutes les données brutes avant 2019 étaient récupérées manuellement et n'avait reçu aucun traitement. Après 2019 on retrouve dans la même situation que MII avec un processus de récupération des données en temps réel, où les données sont déjà passées dans les programmes de conversion qui sont à améliorer.

Pour le traitement des fichier pré-2019, j'ai dû traiter 3 formats de fichiers bruts différents :

Formats de fichiers bruts en entrée :

- .dat (fichiers Aquadopp)
- .cnv (fichiers Microcat)
- .csv (Pour les 2 capteurs de 2014 à 2016)

Formats de sorties :

- .nc (NetCDF)
- .csv

4.2.2 Programmes existants à compléter et améliorer

J'ai dû améliorer 3 programmes de traitement déjà existants à savoir :

- **MII Microcat, Aquadopp, Cstar, Oxygen :** .dat & .cnv → .nc & .csv
- **ALBATROSS Microcat >2019 :** → .cnv → .nc & .csv
- **ALBATROSS Aquadopp >2019 :** → .dat → .nc & .csv

4.2.3 Programmes manquants

Voilà les programmes que j'ai dû mettre au point en fonction du capteur, du déploiement car chaque déploiement a son type de fichier d'entrée :

- **ALBATROSS Microcat 2014-2016 :** .csv → .nc & .csv
- **ALBATROSS Aquadopp 2014-2016 :** .csv → .nc & .csv
- **ALBATROSS Microcat 2016-2019 :** .cnv → .nc & .csv
- **ALBATROSS Aquadopp 2016-2019 :** .dat → .nc & .csv

4.3 Programmes Python réalisés

J'ai dû compléter 3 programmes python et en réaliser 4 nouveaux pour traiter les fichiers avant 2019. Dans tous les scripts, l'algorithme reste plus ou moins le même :

1. Lire les données à partir des fichiers bruts
2. Éventuellement en convertir (notamment pour passer dans des unités internationales)
3. Ajouter les attributs, métadonnées manquantes
4. Éventuellement supprimer des variables ou des attributs inutiles
5. Exporter les nouveaux fichiers en .csv et .nc

4.3.1 Utilisation de l'API Python NetCDF4

1. Création du fichier vierge :

```
fileNC=nc4.Dataset("monfichier.nc”,"w",
format="NETCDF4", encoding='latin-1')
```

2. Création des dimensions

```
fileNC.createDimension('time',size=None)
```

3. Création des variables

```
temp = fileNC.createVariable('temperature','f4',('time',))
temp.standard_name="sea_water_temperature"
temp.units = "Celsius"
…
```

4. Création des attributs globaux

```
fileNC.description = "CTD profile (NetCDF files) for station"+station_name
fileNC.title = "CTD profile (NetCDF files) station "+station_name+" -
Service d'Observation en
Milieu Littoral (SOMLIT)"
fileNC.keywords = "Seabird CTD temperature, conductivity, fluorimetry,
salinity, transmission,
density, oxygen"
…
```

5. Remplissage des données

```
temp[:] = tab_temp
conductivity[:] = tab_conductivity
…
```

6. Fermeture et sauvegarde du fichier

```
fileNC.close()
```

4.3.2 Lecture des fichiers bruts

La lecture des fichiers bruts est une étape cruciale, en effet il a fallu s'assurer à de multiples reprises que les données étaient bien récupérées et sans erreurs. De plus, le fait que chaque format de fichier brut (.dat, .cnv, .csv) a son propre formatage complique la tâche.
Pour exemple, voyons la lecture du fichier de métadonnée propre à chaque déploiement. Ici, on charge le fichier .csv en utilisant la librairie python "pandas" qui permet de faire des opérations complexes simplement.

```
global_att = pd.read_csv(metadataFileCSV, delimiter='=', header=None, index_col=0)
indexlist = global_att.index

    # Creation des global attributs/metadata du fichier NC lues dans un fichier CSV externe
for index in indexlist:
    value = global_att.loc[index].values[0]
    value=value.strip()
    index=index.strip()
```

Figure 29 : Lecture du fichier de métadonnées avec pandas

4.3.3 Ajout de Métadonnées

En premier lieu, les métadonnées globales (description, date, auteurs, identifiant du jeu de donnée...) sont stockées dans un fichier "global_metadata.csv" à part. Lors de son exécution, le programme va lire chaque ligne du fichier de métadonnées pour ensuite les écrire dans les fichiers NetCDF.

```
global_att = pd.read_csv(metadataFile, delimiter='=', header=None, index_col=0)
indexlist = global_att.index

# Creation des global attributs/metadata du fichier NC lues dans un fichier CSV externe
for index in indexlist:
    value = global_att.loc[index].values[0]
    value=value.strip()
    index=index.strip()
    if index == 'Conventions':  # le libelle Convention doit avoir un "C" majuscule
        ncFile.setncattr(index, value)
    else:
        # print("value:"+value+":"); print("index:"+index+":")
        ncFile.setncattr(index.casefold(), value)
```

Figure 17 : Création des attributs globaux avec NetCDF4

Ici, la seule différence d'utilisation entre la librairie Python NetCDF4 et Xarray réside dans la manière d'assigner les valeurs, dans xarray on assigne la valeur à un dictionnaire "`attrs[index]=value`" tandis qu'avec NetCDF4, on utilise la fonction "`setncattr(index, value)`".

Après avoir rempli les attributs globaux, j'ai rempli les attributs des variables en cherchant dans les thesaurus européens et sur le site SeaDataNet, voyons un exemple pour la variable Température :

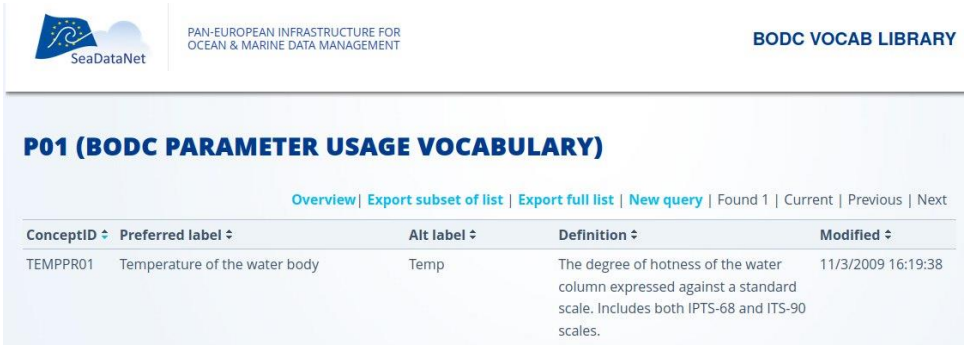


Figure 13 : Vocabulaire SeaDataNet - Nom Variable Température

Figure 16 : Vocabulaire SeaDataNet – Nom Unité Température

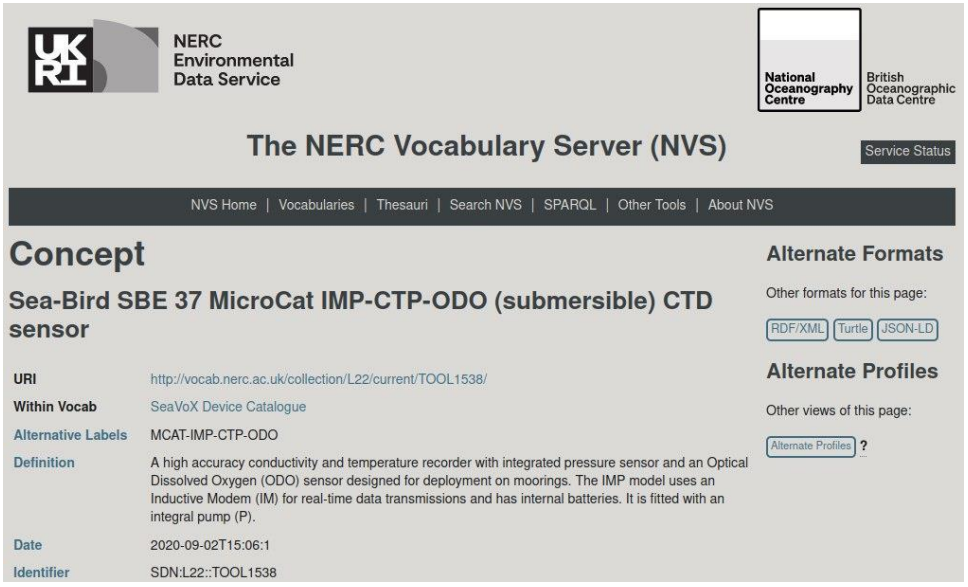


Figure 14 : NERC Vocabulary Server – Identifiant Capteur Microcat

```
# Variable Temperature
if 'TEMP' in ds.variables:
    TEMP_nc = ds['TEMP']
    TEMP_nc.attrs['long_name']  = "temperature"
    TEMP_nc.attrs['standard_name']  = "sea_water_temperature"
    TEMP_nc.attrs['units']  = "Degrees Celsius"
    TEMP_nc.attrs['coordinates'] = "TIME; LATITUDE; LONGITUDE; DEPH"
    TEMP_nc.attrs['ancillary_variables'] = "TEMP_QC"
    TEMP_nc.attrs['reference_scale'] = "ITS-90"
    TEMP_nc.attrs['sdn_parameter_name'] = "Temperature of the water body"
    TEMP_nc.attrs['sdn_parameter_urn'] = "SDN:P01::TEMPPR01"
    TEMP_nc.attrs['sdn_uom_name'] = "degree_Celsius"
    TEMP_nc.attrs['sdn_uom_urn'] = "SDN:P06::UPAA"
    TEMP_nc.attrs['sensor_SeaVoX_L22_code']=sensor_SeaVoX_L22_code
    TEMP_nc.attrs['sensor_model']=sensor_model
    TEMP_nc.attrs['sensor_manufacturer']=sensor_manufacturer
    TEMP_nc.attrs['sensor_manufacturer_uri']=sensor_manufacturer_uri
    TEMP_nc.attrs['sensor_manufacturer_urn']=sensor_manufacturer_urn
    TEMP_nc.attrs['sensor_reference']=sensor_reference
    TEMP_nc.attrs['sensor_serial_number']=sensor_serial
    TEMP_nc.attrs['sensor_mount']=sensor_mount
    TEMP_nc.attrs['sensor_orientation']=sensor_orientation
```

Figure 15 : Remplissage de la variable Température avec xarray

4.3.4 Modification des Attributs

Ensuite, il a fallu que je modifier les noms des variables et leurs attributs pour qu'ils correspondent à leur dénomination standardisée dans le vocabulaire P02 du NERC :
Par Exemple pour la variable « température » http://vocab.nerc.ac.uk/collection/P02/current/TEMP/

```
    # Renommage des variables
    if 'DEPTH' in variable:
        variables_to_rename[variable] = 'DEPH'
    elif 'cond0SPerm' in variable:
        variables_to_rename[variable] = 'CNDC'
    elif 'tv290C' in variable:
        variables_to_rename[variable] = 'TEMP'
    elif 'timeJV2' in variable or 'timeJ' in variable:
        variables_to_rename[variable] = 'TIME'
    elif 'prdM' in variable:
        variables_to_rename[variable] = 'PRES'
    elif 'sbeopoxPS' in variable:
        variables_to_rename[variable] = 'OSAT'
    elif 'sbeopoxMmPerKg' in variable:
        variables_to_rename[variable] = 'DOX2'
    elif 'sbeopoxMmPerL' in variable:
        variables_to_rename[variable] = 'DOX1'
    elif 'sigma' in variable:
        variables_to_rename[variable] = 'DENS'
    elif 'potemperature' in variable:
        variables_to_rename[variable] = 'TPOT'

ds = ds.rename_vars(variables_to_rename)
```

Figure 19 : Renommage des variables avec xarray

Avec NetCDF4, l'ajout et la modification se font au même moment :

```
Batterie_nc = f.createVariable('Battery', 'f4',('time'))
Batterie_nc.long_name = "Battery"
Batterie_nc.standard_name = "battery"
Batterie_nc.units = "Volts"
Batterie_nc.coordinates = "TIME; LATITUDE; LONGITUDE; DEPH"
Batterie_nc.ancillary_variables = "Battery_QC"
Batterie_nc.sdn_parameter_name = ""
Batterie_nc.sdn_parameter_urn = ""
Batterie_nc.sdn_uom_name = "Volts"
Batterie_nc.sdn_uom_urn = "SDN:P06::UVLT"
Batterie_nc.sensor_SeaVoX_L22_code=sensor_SeaVoX_L22_code
Batterie_nc.sensor_model=sensor_model
Batterie_nc.sensor_manufacturer="Nortek"
Batterie_nc.sensor_manufacturer_urn="SDN:L35::MAN0068"
Batterie_nc.sensor_manufacturer_uri="https://vocab.nerc.ac.uk/collection/L35/current/MAN0068/"
Batterie_nc.sensor_reference=sensor_reference
Batterie_nc.sensor_serial_number = serial
Batterie_nc.sensor_mount=sensor_mount
Batterie_nc.sensor_orientation=sensor_orientation
```

Figure 20 : Ajout/Modification de variable avec NetCDF4

4.3.5 Suppression de variables et d'attributs

Dans le programme de conversion des fichiers bruts de Microcat au format .cnv, j'ai utilisé une librarie python appelé « SeaBird » qui permet de convertir les fichiers .cnv en NetCDF afin de simplifier le traitement cependant nous nous sommes vite rendu compte que des variables étranges et inutiles étaient ajoutées.

Pour cette raison, il a fallu supprimer des variables et par conséquent utiliser l'API python « xarray » car elle seule propose la suppression de variables.

Dans cet exemple, je recherche certaines variables qui ont été déterminées comme inutiles puis je les supprimes simplement en les ajoutant à une liste de variables à supprimer et en donnant finalement cette liste à la fonction de xarray "`drop_vars`" qui permet de supprimer des variables.

```
# Supprimer les variables (dernière colonne
variables_to_delete = []
if 'flag' in ds.variables:
    variables_to_delete.append('flag')
if 'timeS' in ds.variables:
    variables_to_delete.append('timeS')
if 'density' in ds.variables:
    variables_to_delete.append('density')
if 'sbeoxpd' in ds.variables:
    variables_to_delete.append('sbeoxpd')
if 'sbeoxpdv' in ds.variables:
    variables_to_delete.append('sbeoxpdv')
if 'sbeoxTC' in ds.variables:
    variables_to_delete.append('sbeoxTC')
if 'sbeoxtv' in ds.variables:
    variables_to_delete.append('sbeoxtv')

ds = ds.drop_vars(variables_to_delete)

if 'bad_flag' in ds.attrs:
    del ds.attrs['bad_flag']
if 'file_type' in ds.attrs:
    del ds.attrs['file_type']
if 'nquan' in ds.attrs:
    del ds.attrs['nquan']
```

Figure 30 : Suppression de variables et d'attributs avec xarray

4.4 Problèmes Rencontrés

Durant tout le stage j'ai rencontré toute une variété de problèmes que j'ai dû régler :

- Le checker plantait en cours d'exécution car il utilisait trop de mémoire, pour moi c'est à cause des chargements liés à la librairie pandas qui sont trop gourmand pour les fichiers avec 20 000 lignes de données

- Ce n'est pas un problème technique en soit mais il y a eu pas mal de confusion sur les unités (ex : dégrées ou dégrées true pour la direction), les noms des variables, les attributs et variables à conserver ou pas car certaines variables étaient en double quand d'autres se ressemblait fortement sans être identique (ex : 2 pressions mais a des salinités légèrement différentes)

- J'ai eu un problème sur le remplissage des codes qualités : normalement chaque variable contient un nombre égal de point : chaque variable qualité est ici pour indiquer si les données de sa variable associée est fiable ou non avec un code de 0 à 9. Pour un jeu de données sur 10 000 points, ce code se répète donc ici 10 000 fois normalement. Dans mon cas il y avait beaucoup plus de points seulement pour les codes qualités ce qui a eu pour conséquence d'étirer les tableaux de valeurs avec des valeurs vides pour toutes les variables (Voir Annexes, Figure 26) pour régler ce problème j'ai adapté le remplissage des codes qualités dans le code pour m'assurer que chaque variable qualité se remplisse bien avec le bon nombre de valeurs.

- Un autre problème en lien avec les variables qualités : les valeurs n'était pas dans le bon type ce qui causait des problèmes dans les checkers, pour régler ce problème j'ai dû chercher le type attendu et après avoir trouvé j'ai fait beaucoup de test pour finalement trouver qu'il fallait un type tableau numpy de type int8 (entier signé sur 8 bits)

- Problème de formatage différents entre différents fichiers csv du même déploiement, cela cause un gros problème au niveau de traitement automatisé par le programme car si chaque fichier est radicalement différent on ne peut rien faire, pour régler ce problème j'ai moi-même adapte les quelques fichiers qui étaient différents pour les formater comme les autres

- Fichiers bruts .dat .cnv corrompus, il a fallu trouver la ligne défaillante et la supprimer (Voir Annexes, Figure 27)

- ERDDEV refusait de charger le dataset xml à cause d'un problème sur l'unité de temps en secondes alors que les données sont en jour juliens (jours incrémentés à partir du lancement) pour résoudre cela j'ai dû faire une fonction qui converti les jours juliens en timestamps (secondes depuis 1970).

- ERDDEV ne chargeait toujours pas les xml car il manquait des attributs spécifiés nulle part. En essayant de les ajouter je me suis rendu compte que la variable "stationname" qui contient le nom de la station et un attribut nécessaire n'était pas présente, j'ai dû la créer et j'ai rencontré à nouveau des problèmes sur le type de la variable, la solution se trouvait encore dans un tableau numpy cette fois ci de type "string_".

- Parfois l'ordre et le nombre de variables faisait totalement échouer la conversion, pour résoudre cela j'ai créé des dictionnaires de correspondance pour indiquer quel fichier contient quelles variables.

- Problème sur la profondeur dans les fichiers NetCDF : Les valeurs étaient celles mesurés sauf qu'elles étaient quasiment identiques étant donné que la profondeur n'évolue presque pas, j'ai dû trouver un moyen pour avoir systématiquement n fois la bonne valeur arrondie (ex : 1700 au lieu de 1702.42). Il n'a pas été compliqué de faire cette modification mais cela a été un problème pour ERDDAP car il a besoin de valeurs fixes pour la profondeur afin d'effectuer du triage et des comparaisons.

- Certains attributs indésirables étaient impossibles à supprimer ni à modifier car ils étaient ajoutés au moment de l'encodage avec xarray, pendant la sauvegarde. Pour cela j'ai été contraint de rouvrir les fichiers plus tard dans le code avec NetCDF4 afin de corriger ces attributs

- J'ai rencontré des problèmes bloquants à cause de simples erreurs d'inattentions, par exemple une faute de frappe qui empêche un attribut d'être ajouté car j'ai mis 2 fois le même nom, par conséquent, la variable citée en double est écrasée (Voir Annexes, Figures 32 et 33).

- ERDDAP repassait sur certains attributs en modifiant des valeurs sans que je lui demande ce qui a causé des attributs sans valeur par exemple, ce problème n'a pas pu être réglé.

4.5 Mise en Production

La dernière étape est la mise en production sur le serveur ERDDAP, c'est-à-dire, mettre à jour les programmes, mettre à jour les anciennes données qui sont figées ainsi que les données actuelles qui sont complétées chaque jour dans un flux de données automatisé. Avant de mettre en production sur ERDDAP, les étapes suivantes ont été effectuées sur le serveur de développement ERDDEV afin de s'assurer du bon fonctionnement de l'ensemble.

La première étape consiste à copier les programmes sans oublier de modifier les chemins d'accès aux fichiers, pour cela j'ai utilisé rsync qui permet de faire de la copie de fichiers de manière synchrone (s'il manque qu'un fichier seul ce fichier est copié), par exemple :

```
rsync -av emso-2014-2019 root@erddap:/root/EMSO/
```

Ensuite il a fallu que je place dans les bons dossiers les données des anciens déploiements datant de 2014-2019 mais aussi que je régénère l'ensemble des données du déploiement actuel donc près de 6 ans de données quotidiennes (pour MII).

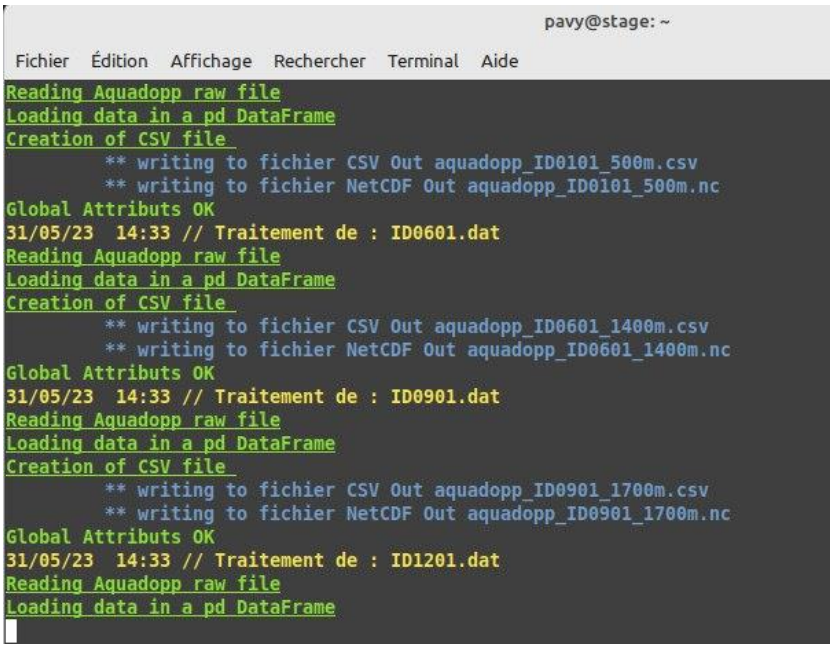


Figure 23 : Exemple de Génération

Une fois les générations effectuées, il faut inclure les jeux de données dans le fichier global du serveur “datasets.xml”. Ce fichier prend uniquement des datasets en format xml, pour lui fournir ce dont il a besoin j’ai utilisé un script de génération de datasets xml à partir de nombreux type de fichiers, dans mon cas, j’ai fait des datasets de fichiers .nc, .csv et également de .tar.gz afin de proposer une archive contenant l’ensemble des fichiers NetCDF de chaque déploiement.

Une fois les datasets générés, j’y ai apporté quelques modifications pour rendre le tout plus propre et accessible, par exemple j’ai ajouté la directive `<accessibleViaFiles>true</accessibleViaFiles>` qui permet d’ajouter un lien cliquable pour accéder à l’ensemble des fichiers du dataset. À chaque modification, il a été très important de d’abord désactiver le dataset en mettant son attribut “active” à “false”, puis de rafraichir le dataset afin que le serveur prenne en compte en créant un fichier du nom du datasetID dans le dossier "flag/”. Enfin, après avoir fait des modifications, on peut le remettre à “true” et rafraîchir encore une fois le dataset.

```
<dataset                                    type="EDDTableFromMultidimNcFiles"
datasetID="Emso_Western_Ligurian_Albatross_Microcat_NetCDF_2018"
active="false">

touch flag/Emso_Western_Ligurian_Albatross_Microcat_NetCDF_2018
```

Enfin, la dernière étape consiste à automatiser l’exécution des programmes. Pour y arriver, j’ai ajouté l’exécution de mon programme en parallèle et juste avant l’exécution des anciens programmes afin de permettre une transition fluide entre la version 1 et la version 2. Les exécutions sont espacées de 5 minutes pour éviter toute surcharge sur le serveur :

```
00 7 * * * /root/EMSO/emso-v2-OceanSites/EMSO_Albatross/decodeEMSO-Albatross-Micro-2021-07-avec-graphe.py
05 7 * * * /root/EMSO/emso-v2-OceanSites/EMSO_Albatross/decodeEMSO-Albatross-Aquadopp-2022-06-avec-graphe.py
10 7 * * * /root/EMSO/emso-v2-OceanSites/EMSO_MII/decodeEMSO-MII-MCOA.py -a -g
15 7 * * * /root/EMSO/decodeEMSO-Albatross-Micro.py
20 7 * * * /root/EMSO/decodeEMSO-Albatross-Aquadopp.py
30 7 * * *  /root/EMSO/decodeEMSO-MII-MCO.py -a -g -d
```

Figure 10 : Exécution automatisée des programmes avec cron

J’ai également amélioré un script déjà existant qui permet d’effectuer une répartition des données dans des dossiers sous la forme années-mois, cela permet de rendre les grands jeux de données beaucoup plus lisibles.

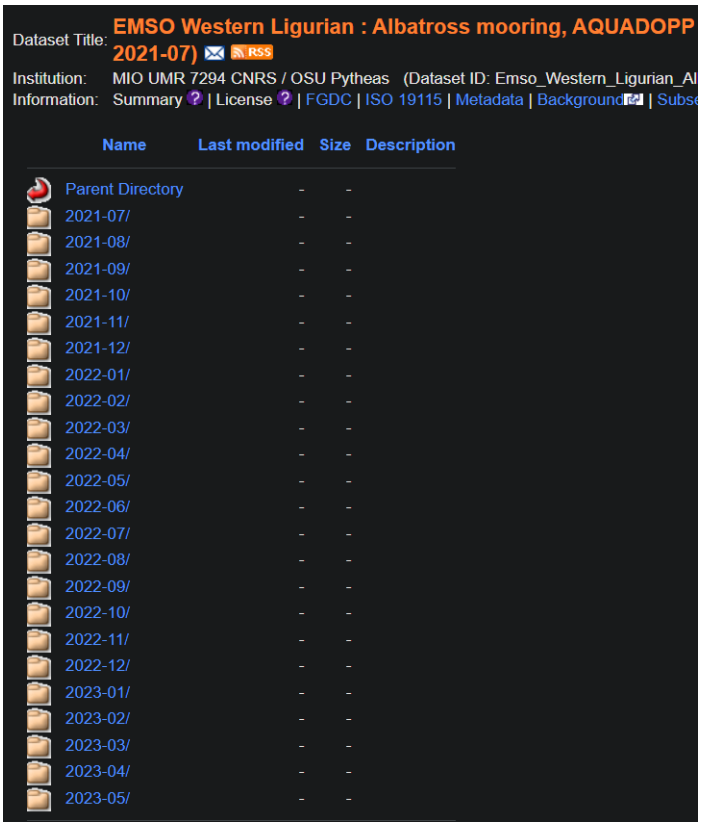


Figure 24 : Triage par dossier avec script

4.6 Résultats obtenus

Après ces traitements on obtient des fichiers .nc complets avec les bonnes références SeaDataNet et en accord avec les thesaurus européens. La page https://erddap.osupytheas.fr/erddap/search/index.html?page=1&itemsPerPage=1000&searchFor=western montre l'ensemble des fichiers que j'ai traités et produits :

data	graph		files	public	EMSO Western Ligurian : Albatross mooring, AQUADOPP sensor (CSV 2014-2015)			M	background	RSS
data	graph		files	public	EMSO Western Ligurian : Albatross mooring, AQUADOPP sensor (CSV 2015-2016)			M	background	RSS
data	graph		files	public	EMSO Western Ligurian : Albatross mooring, AQUADOPP sensor (CSV 2016-2017)			M	background	RSS
data	graph		files	public	EMSO Western Ligurian : Albatross mooring, AQUADOPP sensor (CSV 2018-2019)			M	background	RSS
data	graph		files	public	EMSO Western Ligurian : Albatross mooring, AQUADOPP sensor (CSV files from 2021-07)		F I	M	background	RSS
data	graph		files	public	EMSO Western Ligurian : Albatross mooring, AQUADOPP sensor (NetCDF 2014-2015)		F I	M	background	RSS
data	graph		files	public	EMSO Western Ligurian : Albatross mooring, AQUADOPP sensor (NetCDF 2015-2016)		F I	M	background	RSS
data	graph		files	public	EMSO Western Ligurian : Albatross mooring, AQUADOPP sensor (NetCDF 2016-2017)		F I	M	background	RSS
data	graph		files	public	EMSO Western Ligurian : Albatross mooring, AQUADOPP sensor (NetCDF 2018-2019)		F I	M	background	RSS
data	graph		files	public	EMSO Western Ligurian : Albatross mooring, AQUADOPP sensor (NetCDF files from 2021-07)		F I	M	background	RSS
data	graph		files	public	EMSO Western Ligurian : Albatross mooring, MICROCAT sensor (CSV 2014-2015)			M	background	RSS
data	graph		files	public	EMSO Western Ligurian : Albatross mooring, MICROCAT sensor (CSV 2015-2016)			M	background	RSS
data	graph		files	public	EMSO Western Ligurian : Albatross mooring, MICROCAT sensor (CSV 2016-2017)			M	background	RSS
data	graph		files	public	EMSO Western Ligurian : Albatross mooring, MICROCAT sensor (CSV 2018-2019)			M	background	RSS
data	graph		files	public	EMSO Western Ligurian : Albatross mooring, MICROCAT sensor (CSV files from 2021-07)			M	background	RSS
data	graph		files	public	EMSO Western Ligurian : Albatross mooring, MICROCAT sensor (NetCDF 2014-2015)		F I	M	background	RSS
data	graph		files	public	EMSO Western Ligurian : Albatross mooring, MICROCAT sensor (NetCDF 2015-2016)		F I	M	background	RSS
data	graph		files	public	EMSO Western Ligurian : Albatross mooring, MICROCAT sensor (NetCDF 2016-2017)		F I	M	background	RSS
data	graph		files	public	EMSO Western Ligurian : Albatross mooring, MICROCAT sensor (NetCDF 2018-2019)		F I	M	background	RSS
data	graph		files	public	EMSO Western Ligurian : Albatross mooring, MICROCAT sensor (NetCDF files from 2021-07)		F I	M	background	RSS
data	graph		files	public	EMSO Western Ligurian : MII, AQUADOPP sensor (CSV files)		F I	M	background	RSS
data	graph		files	public	EMSO Western Ligurian : MII, AQUADOPP sensor (NetCDF files)		F I	M	background	RSS
data	graph		files	public	EMSO Western Ligurian : MII, CSTAR sensor (CSV files)		F I	M	background	RSS
data	graph		files	public	EMSO Western Ligurian : MII, CSTAR sensor (NetCDF files)		F I	M	background	RSS
data	graph		files	public	EMSO Western Ligurian : MII, MICROCAT sensor (CSV files)		F I	M	background	RSS
data	graph		files	public	EMSO Western Ligurian : MII, MICROCAT sensor (NetCDF files)		F I	M	background	RSS
data	graph		files	public	EMSO Western Ligurian : MII, OXYGEN sensor (CSV files)		F I	M	background	RSS
data	graph		files	public	EMSO Western Ligurian : MII, OXYGEN sensor (NetCDF files)		F I	M	background	RSS

Figure 22 : Aperçu des datasets produits sur ERDDAP

Name	Long Name	Type
microcat_sbe...	microcat_sbe37imp-od...	Local File
cond0SPerm	Conductivity [S/m]	1D
DEPTH	Depth [salt water, m], ...	1D
flag	0.000e+00	1D
potempera...	Potential Temperature...	1D
prdM	Pressure, Strain Gaug...	1D
PSAL	Salinity, Practical [PSU]	1D
sbeopoxM...	Oxygen, SBE 63 [umol/...	1D
sbeopoxM...	Oxygen, SBE 63 [umol/l]	1D
sbeopoxPS	Oxygen, SBE 63 [% sat...	1D
sbeoxpd	Oxygen raw, SBE 63 p...	1D
sbeoxpdv	Oxygen raw, SBE 63 p...	1D
sbeoxTC	Oxygen Temperature, ...	1D
timeJ	Julian Days	1D
timeS	timeS	1D
tv290C	Temperature [ITS-90, ...	1D

File "microcat_sbe37imp-odo_03713018_2300m.nc"

File type: Hierarchical Data Format, version 5

```
netcdf file:/home/pavy/Téléchargements/microcat_sbe37imp-odo_03713018_2300m.nc {
  dimensions:
    scan = 18813;
  variables:
    double prdM(scan=18813);
      :missing_value = -9.99E-29; // double
      :id = "0";
      :span = "0.164", "50592.579";
      :long_name = "Pressure, Strain Gauge [db]";

    double potemperature(scan=18813);
      :span = "5.6260", "22.9368";
      :long_name = "Potential Temperature [ITS-90, deg C]";
      :missing_value = -9.99E-29; // double
      :id = "1";
```

Figure 11 : Fichier NetCDF avant traitement

Name	Long Name	Type
Micro_apres...	EMSO Western Liguri...	Local File
CNDC	conductivity	1D
CNDC_QC	Conductivity quality fl...	1D
DEPH	depth of measureme...	—
DOX1	Dissolved oxygen	1D
DOX2	Dissolved oxygen	1D
DOX_QC	Oxygen quality flag	1D
LATITUDE	Latitude of measure...	—
LONGITUDE	Longitude of measur...	—
OSAT	Oxygen saturation	1D
OSAT_QC	Saturation oxygene ...	1D
PRES	pressure	1D
PRES_QC	Pressure quality flag	1D
PSAL	Practical salinity	1D
PSAL_QC	Salinity quality flag	1D
stationna...	station name	—
TEMP	temperature	1D
TEMP_QC	Temperature quality ...	1D
TIME	time of measurements	1D
TIME_QC	Time quality flag	1D
TPOT	temperature potenti...	1D

File "Micro_apres_transfo.nc"

File type: Hierarchical Data Format, version 5

```
netcdf file:/home/pavy/Téléchargements/Micro_apres_transfo.nc {
  dimensions:
    time = 18813;
    string27 = 27;
  variables:
    double PRES(time=18813);
      :_FillValue = NaN; // double
      :long_name = "pressure";
      :standard_name = "sea_water_pressure";
      :units = "Decibars";
      :comment = "";
      :coordinates = "TIME; LATITUDE; LONGITUDE; DEPH";
      :ancillary_variables = "PRES_QC";
      :sdn_parameter_name = "Pressure (measured variable) exerted by the water body plus atmos
      :sdn_parameter_urn = "SDN:P01::PRSTPS01";
      :sdn_uom_name = "Decibars";
      :sdn_uom_urn = "SDN:P06::UPDB";
      :sensor_SeaVoX_L22_code = "SDN:L22::TOOL1538";
      :sensor_model = "Sea-Bird SBE 37 MicroCat IMP-CTP-ODO (submersible) CTD sensor";
      :sensor_manufacturer = "Sea-Bird Scientific";
      :sensor_manufacturer_uri = "https://vocab.nerc.ac.uk/collection/L35/current/MAN0013/";
      :sensor_manufacturer_urn = "SDN:L35::MAN0013";
      :sensor_reference = "https://vocab.nerc.ac.uk/collection/L22/current/TOOL1538";
      :sensor_serial_number = "03713018";
      :sensor_mount = "mounted_on_mooring_line";
      :sensor_orientation = "horizontal";
      :missing_value = NaN; // double
```

Figure 12 : Fichier NetCDF après traitement

	A	B	C	D	E	F	G	H	I	J	K
1	contact:dominique.lefevre@mio.osupytheas.fr	maurice.libes@osupytheas.fr									
2	Project:EMSO http://www.emso-fr.org/EMSO-France										
3	DOI:https://doi.org/10.17882/95264										
4	Depth:2300m										
5	Lat:42.803										
6	Lon:6.05										
7	TIME	TIME_QC	DEPH	CNDC	CNDC_QC	PRES	PRES_QC	TEMP	TEMP_QC	TPOT	PSAL
8	units:		meters	Siemens per metre		Decibars		degree_Celsius		degree_Celsius	
9	2017-08-26T14:10:01		2 2300.0	4.593097		2 2372.88		2 13.2767		2 12.9106	38.482
10	2017-08-26T14:11:01		2 2300.0	4.593076		2 2372.8		2 13.2768		2 12.9106	38.482
11	2017-08-26T14:12:02		2 2300.0	4.59309		2 2372.811		2 13.2765		2 12.9103	38.482
12	2017-08-26T14:13:01		2 2300.0	4.593069		2 2372.856		2 13.2762		2 12.9101	38.482
13	2017-08-26T14:14:02		2 2300.0	4.593083		2 2372.854		2 13.2767		2 12.9106	38.482
14	2017-08-26T14:15:01		2 2300.0	4.593076		2 2372.898		2 13.2765		2 12.9103	38.482
15	2017-08-26T14:16:02		2 2300.0	4.593083		2 2372.942		2 13.2766		2 12.9104	38.482
16	2017-08-26T14:17:01		2 2300.0	4.593097		2 2372.964		2 13.2767		2 12.9105	38.482
17	2017-08-26T14:18:02		2 2300.0	4.59309		2 2372.987		2 13.2767		2 12.9105	38.482

Figure 13 : Exemple de Fichier CSV produit

En passant le checker sur un dataset, le score obtenu est de 87%, sachant que les 13% manquants sont des « fausses » erreurs qui en réalité n'en sont pas vraiment, les données sont en accord avec les exigences du projet EMSO :

```
Required tests passed: 263 of 302
Required tests passed: 24 of 105
   Total tests passed: 287 of 407
Required tests...  87% -:---:--
Optional tests...  23% -:---:--
Total tests...     71% -:---:--
```

Figure 21 : Résultats checker metadata-harmonizer

5. Conclusion

Ce stage a été une véritable expérience enrichissante et formatrice. Il m'a notamment offert l'opportunité de développer mes compétences en programmation Python, en particulier dans le domaine de la gestion des données scientifiques. J'ai été plongé dans un projet scientifique européen d'envergure, exigeant une grande rigueur et un haut niveau de précision.

Au sein de l'équipe, j'ai pu contribuer activement à ce projet ambitieux. J'ai appris à manipuler et à analyser de vastes ensembles de données dans le format NetCDF que je ne connaissais pas, à mettre en œuvre des algorithmes adaptés et à automatiser des processus de traitement de données. La rigueur scientifique a été une composante essentielle de mon travail, puisque la qualité et la fiabilité des résultats étaient d'une importance primordiale.

Ce stage m'a sensibilisé à l'importance d'une communication précise et claire dans un projet scientifique international. J'ai également acquis une solide expérience de la gestion des délais et des priorités, ainsi que la capacité à m'adapter rapidement aux exigences changeantes du projet.

En somme, cette expérience m'a permis de progresser en tant que professionnel et de développer des compétences cruciales dans le domaine de la gestion des données scientifiques. Je suis reconnaissant d'avoir pu contribuer à ce projet et je suis prêt à relever de nouveaux défis dans le domaine de la recherche scientifique.

6. Remerciements

Je tiens à exprimer ma sincère gratitude à tous ceux qui ont contribué à la réussite de mon stage. Je remercie tout particulièrement mon tuteur Maurice LIBES, ainsi que Dominique LEFEVRE, pour leur accueil chaleureux, leur encadrement et leur collaboration tout au long de cette expérience. Ce fut une expérience enrichissante, et je suis reconnaissant envers toutes les personnes qui y ont contribué.

7. Glossaire

OSU, Observatoire des Sciences et de l'Univers

EMSO, European Multidisciplinary Seafloor and water column Observatory

MEUST, Mediterranean Eurocentre For Underwater Sciences and Technologies

CEREGE, Centre Européen de Recherche et d'Enseignement des Géosciences de l'Environnement

IMBE, Institut Méditerranéen de Biodiversité et d'Écologie marine et continentale

LAM, Laboratoire d'Astrophysique de Marseille

LPED, Laboratoire Population Environnement Développement

MIO, Institut Méditerranéen d'Océanologie

RECOVER, Risques Ecosystèmes Vulnérabilité Environnement Résilience

CNRS, Centre National de la Recherche Scientifique

IRD, Institut de Recherche pour le Développement

ERDDAP, Environmental Research Division Data Access Protocol

METADONNEES, Données qui fournissent de l'information sur une autre donnée

ALBATROSS, Autonomous Line with a Broad Acoustic Transmission for Research in Oceanography and Sea Sciences

MII, Module d'Interface Instrumenté

SSH, Secure Shell

RSYNC, Remote Synchronization

XML, Extensible Markup Language

DOI, Digital Object Identifier

NetCDF, Network Common Data Form

CSV, Comma Separated Values

Thesaurus, Répertoire structuré de termes (mots-clés) pour l'analyse de contenu et le classement de documents

API, Interface de programmation d'application

In-situ, Sur place, sur le site, dans son milieu naturel

FAIR, Facile à trouver Accessible Interopérable Réutilisable

UMR, Unité Mixte de Recherche

8. Sitographie

Plate-forme ERDDAP de gestion des données environnementales
OSU Pytheas [En Ligne] : https://erddap.osupytheas.fr/erddap/

Documentation Module Python Xarray [En Ligne] : https://docs.xarray.dev/en/stable/

Documentation Module Python Pandas [En Ligne] : http://www.python-simple.com/python-pandas/panda-intro.php

The NERC Vocabulary Server [En Ligne] : http://vocab.nerc.ac.uk/

Site SeaDataNet [En Ligne] : https://www.seadatanet.org/

Site EMSO-France [En Ligne] : https://www.emso-fr.org/EMSO-France

Checker NetCDF Cfchecker [En Ligne] : https://pypi.org/project/cfchecker/

Checker NetCDF [Réseau OSU] : Non Disponible

Site OSU Pythéas [En Ligne] : https://www.osupytheas.fr/

Site MIO [En Ligne] : https://www.mio.osupytheas.fr/

CF Metadata Conventions [En Ligne] : https://cfconventions.org/

Programme SeaBird [En Ligne] : https://github.com/castelao/seabird

Atelier NetCDF Python [En Ligne] :
https://sist19.sciencesconf.org/data/pages/SIST19_Atelier_NetCDF_python.pdf