The Journal of Open Source Software

CrocoLakeTools: A Python package to convert ocean observations to the parquet format

Enrico Milanese[1], **David Nicholson**[1], **Gaël Forget**[2], **and Susan Wijffels**[1]

1 Woods Hole Oceanographic Institution, United States of America **2** Massachusetts Institute of Technology, United States of America

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

Investigations of the ocean state are possible thanks to the ever growing number of measurements performed with multiple instruments by different research missions. The vast and variegated efforts have brought the community to define data storage conventions (e.g. CF-netCDF) and to assemble collections of datasets (e.g. the World Ocean Database). Yet, accessing these datasets often requires the usage of specialized tools, is generally inefficient over the cloud, and remains a major obstacle to new users in terms of pre-required knowledge and time. CrocoLakeTools is a Python package that addresses those challenges by providing workflows to convert various oceanographic datasets from their original format to a uniform parquet dataset with a shared schema.

Statement of need

`CrocoLakeTools` is a Python package to build workflows that convert ocean observations from their original formats (e.g. netCDF, CSV) to parquet. CrocoLakeTools takes advantage of Python's well-established and growing ecosystem of open tools: it uses `dask`'s parallel computing capabilities to convert multiple files at once and to handle larger-than-memory data. `dask` is already well-integrated with `xarray` and `pandas`, two widely used Python libraries for the treatment of array and tabular data, respectively, and with `pyarrow`, the API to the Apache Arrow library which is used to generate the parquet dataset.

Parquet is a data storage format for big tidy data which presents several advantages: it is language agnostic (it can be accessed with Python, Matlab, Julia and web developement technologies); it offers faster reading performances than other tabular formats such as CSV; it is optimized for cloud systems storage and operations; it is widespread in the data science community, leading to a multitude of freely accessible tools and educational material.

`CrocoLakeTools` was developed with the goal of building and serving CrocoLake, a regularly refreshed database of oceanographic observations that are pre-filtered to contain only quality-controlled measurements. `CrocoLakeTools` was designed to be used by researchers, engineers and data scientists in oceanograhy and to be accessed by the wider oceanographic community.

State of the field

The need for homogenized ocean data product has prompted several efforts.

The most comprehensive project to date is likely the World Ocean Database (WOD), which contains more than 40 variables gathered from several sources (buoys, gliders, floats, bathythermographs, etc.). The earliest record dates back to 1772, the data is quality-controlled, and the database is regularly updated with the latest measurements available. The data is public and freely available through different interfaces (WODselect web application, THREDDS, HTTPS, FTP) in ASCII, Comma Separated Value (CSV) and netCDF formats. WOD is maintained by the National Centers for Environmental Information (NCEI) of the United States' National Oceanic and Atmospheric Administration (NOAA).

The International Quality-controlled Ocean Database (IQuOD) is an effort by the oceanographic community "with the goal of producing the highest quality and complete single ocean profile repository along with (intelligent) metadata and assigned uncertainties". It includes subsurface ocean profiles of several variables, paying particular attention to temperature measurements. The database is prepared by NCEI, and it is freely available in netCDF format through multiple channels (THREDDS, HTTPS, FTP).

Argovis is a REST API and web application hosted at the University of Colorado, Boulder (United States) and serving profile data in JSON format from the Argo program, and from ship and drifters missions. #add argopy, R-oce, OceanRobots.jl/ArgoData.jl, CMEMS/Copernicus?

The Ocean Data Platform by the non-profit HUB Ocean is among the youngest projects in the community, and it allows to find and access datasets from a catalog. The user can interact with the platform through different interfaces (SDK, REST API, OQS, JupyterHub workspaces), loading the datasets in tabular format.

The above efforts often serve data in ASCII, CSV, JSON, or netCDF formats. For context, netCDF is a binary format that offers the advantage to be compact and efficient when dealing with multidimensional data, while the others have the advantage of being human-readable (but can be very inefficient for large datasets). None of these formats is optimized for cloud object storage, although there are ongoing efforts for netCDF (see Zarr and Icechunk). For this reason, Parquet has been drawing more and more attention from the earth sciences community recently.

Parquet is a cloud-optimized binary format for tidy and large data (i.e. large tables) that is language-agnostic. It is widely used in the data science and corporate worlds, and the software ecosystem around it is sound, mature, and growing. An overview of the characteristics of each format is in Table 1. We chose Parquet as the target format for `CrocoLake` because (1) it is optimized for cloud storage and cloud computing, (2) its mature software ecosystem includes packages in multiple coding languages to access it (Python, Julia, MATLAB, web technologies, etc.), and (3)) novel users are generally more familiar with tabular data than to specialized oceanographic data formats.

As we aim to make CrocoLake easily accessible from a technical standpoint, the main drawbacks of Parquet at present are that (1) many workflows in ocean modeling are based on multidimensional data structures (not tabular ones) and (2) attaching attributes to the data is not as straightfoward as it could be. However, we see CrocoLake as a major building bloc in workflows that require fast access to point-based ocean observations, responding to necessities of data storage with fast access both on disk and the cloud, rather than as an end-all solution for ocean observations.

Table 1. Comparison between different common file formats for oceanographic datasets.

Feature Name	CSV	netCDF	Parquet	ASCII	JSON	Zarr + Icechunk
Cloud-Optimized	No	No	Yes	No	No	Yes
Structure	Tabular	Array-based	Tabular	Tabular	Hierarchical	Array-based
Available tools in:						
Python	Yes	Yes	Yes	Yes	Yes	Yes
Julia	Yes	Yes	Yes	Yes	Yes	Zarr only
MATLAB	Yes	Yes	Yes	Yes	Yes	Zarr only

The Journal of Open Source Software

Feature Name	CSV	netCDF	Parquet	ASCII	JSON	Zarr + Icechunk
Attributes descriptors	Dedicated columns, header	Dictionary, accessed with data	Dictionary, separate access from data	Dedicated column	Dedicated field in object	Dictionary, accessed with data

Another key feature of CrocoLakeTools is that, unlike most aforementioned projects, it is fully open-source. Anyone can thus use CrocoLakeTools to build their own flavor of CrocoLake. We also welcome new contributors to CrocoLakeTools, for example by adding converters to support new datasets.

Code architecture

Converters

The core task of CrocoLakeTools is to take one or more files from a dataset and convert them to parquet, ensuring that CrocoLake's schema is followed. This is achieved through the methods contained in the `Converter` class and its subclasses. While the conversion of all datasets requires some general functionality (e.g. renaming the original variables to the final schema), each conversion requires specialized tools that differ betwee datasets (e.g. the map used to rename variables). CrocoLakeTools then contains the `Converter` class, which contains the methods shared across datasets, and from which converter subclasses inherit and implement the specific needs of each dataset.

Workflow

Local mirrors

The first step in our workflow is to retrieve original files from each data provider (Figure 1). The original source data follow the format, schema, nomenclature, and conventions defined by their side (project, mission, scientist, etc.) independently of CrocoLake's workflow. Modules to download the original data are optional. They should inherit from the `Downloader` class and be called `downloader<DatasetName>` (e.g. `downloaderArgoGDAC`). At the time of writing CrocoLakeTools is released with a downloader to build a local mirror of the Argo GDAC, and we hope to support more data providers in the future. Whether a downloader module exists or the user downloads the data themselves, the original data is stored on disk and this is the starting point for the converter.

Parquet datasets

The second step is to convert the data to parquet, and finally merge the datasets into CrocoLake (Figure 2). The core of `CrocoLakeTools` are the modules in the `Converter` class and its subclasses. Each original dataset has its own subclass called `converter<DatasetName>`, e.g. `converterGLODAP`; further specifiers can be added as necessary (e.g. at this time there a few different converters for Argo data to prepare different datasets). The need for a dedicated converter for each project despite the usage of common data formats (e.g. netCDF, CSV) is due to differences in schema (e.g. variable names or units). Depending on the dataset, multiple converters can be applied. For example, to create CrocoLake, Argo data goes through two converters: 1. `converterArgoGDAC`, which converts the original Argo GDAC preserving most of its original conventions; 2. `converterArgoQC`, which takes the output of the previous step and applies some filtering based on Argo's QC flags and makes the data conforming to CrocoLake's schema.

CrocoLake

CrocoLake is one parquet dataset that contains all converted datasets merged together. This can be achieved with the script `merge_crocolake.py`. The script first creates a directory containing symbolic links to each converted dataset. It then uses the submodule `CrocoLakeLoader` to seamlessly load all the converted datasets into memory as one dask dataframe with a uniform schema, merges them into CrocoLake, and stores it back to disk.

CrocoLake can be accessed with several programming languages with just a few lines of codes: the submodules CrocoLake-Python, CrocoLake-Matlab, and CrocoLake-Julia contain tools and examples in some languages.

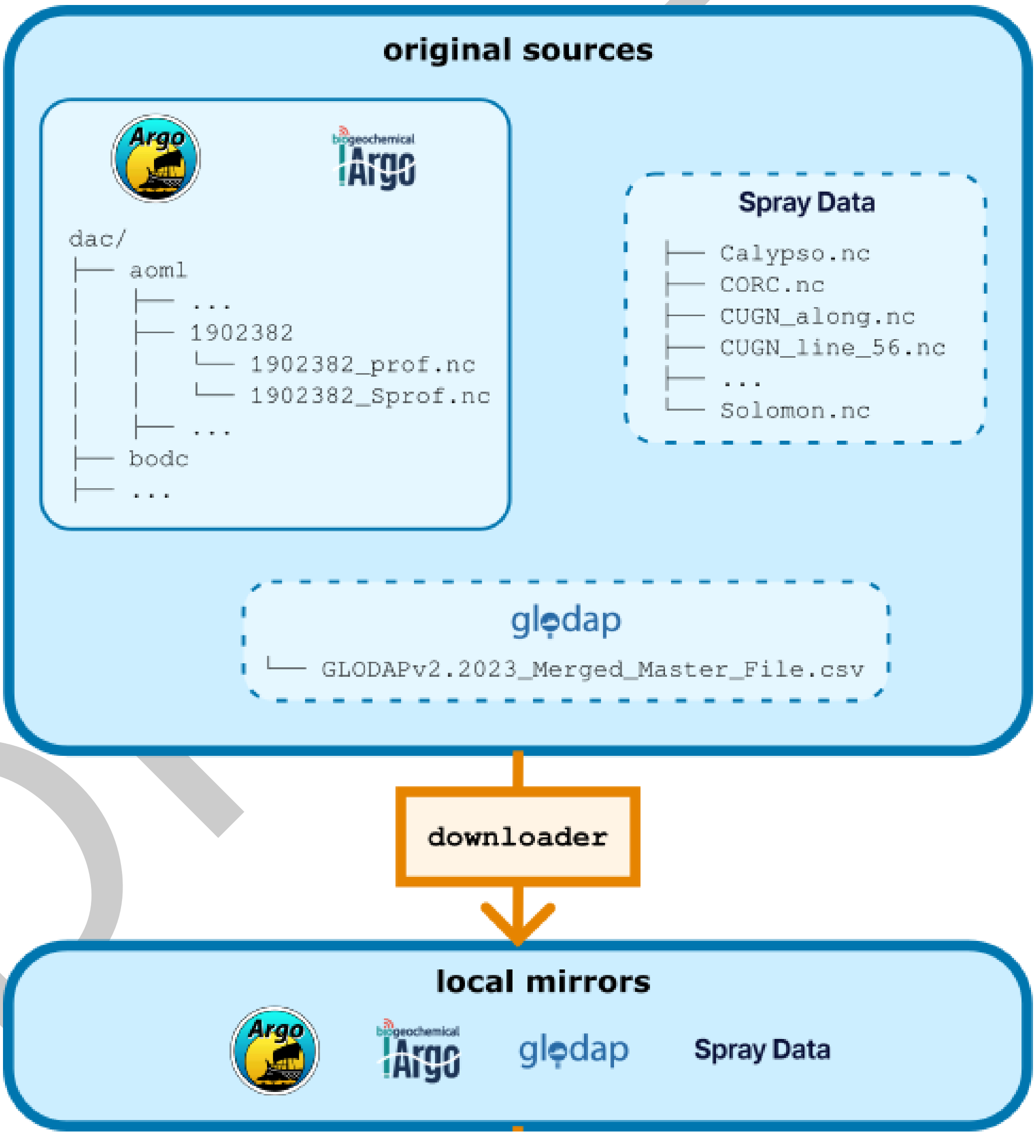

Figure 1: CrocoLake's workflow: 'downloader's. 'CrocoLakeTools' is set up to host modules that are dedicated to download the desired datasets from the web. It currently supports the download only of Argo data (solid line subset), and other datasets require the user to download them manually (dashed border subsets).

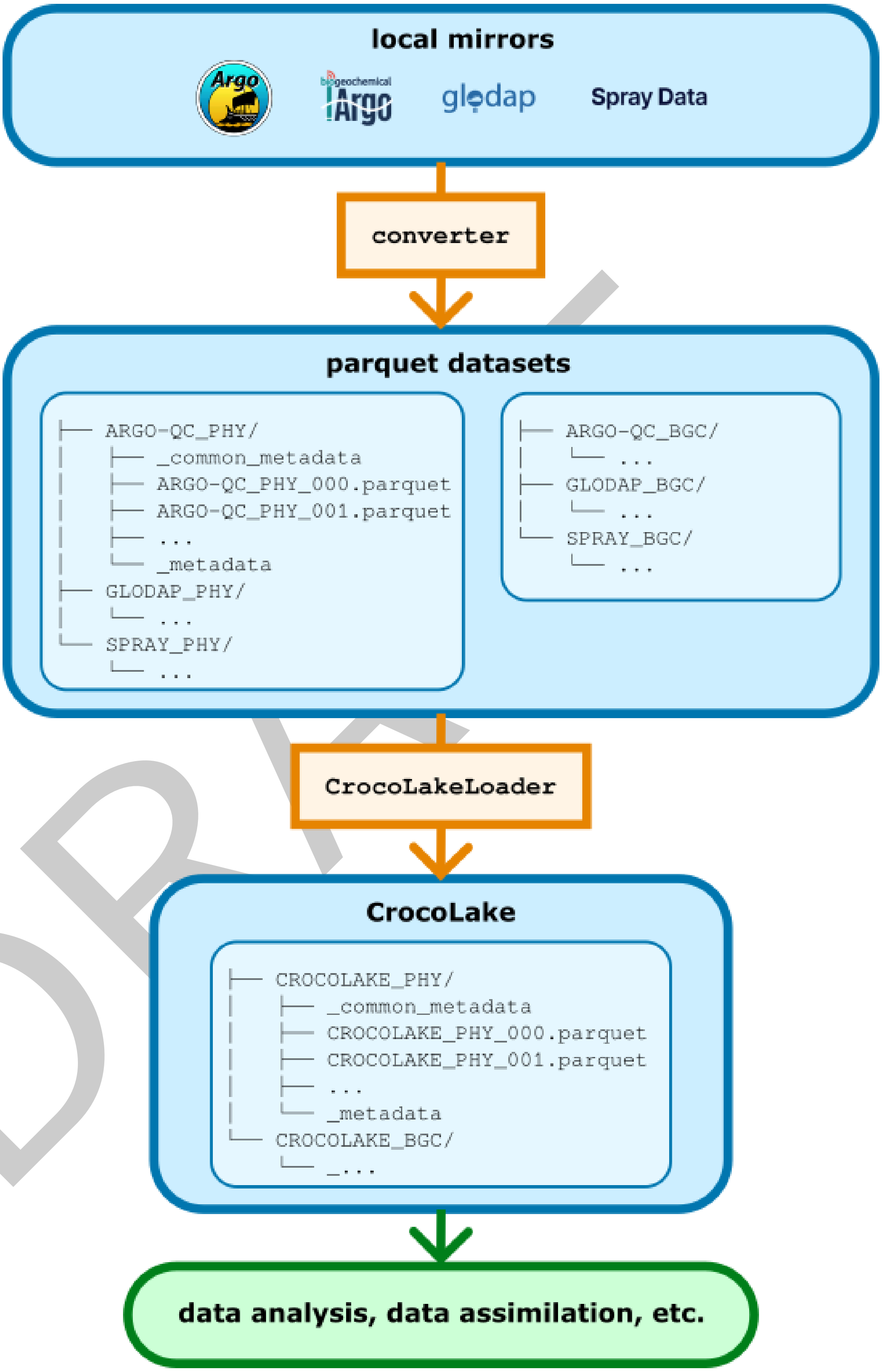

Figure 2: CrocoLake's workflowL 'converter's. 'converter's read the data in their original format, transform it following CrocoLake's conventions, converts it to parquet, and stores it back to disk. Each dataset is converted to its own parquet version. Thanks to the submodule 'CrocoLakeLoader', multiple parquet datasets are merged into a uniform dataframe which is saved to disk as CrocoLake. For each dataset, a version containing only physical variables ('PHY') or also biogeochemical variables ('BGC') can be generated.

Schema

The nomenclature, units and data types are generally based on Argo's (https://vocab.nerc.ac.uk/collection/R03/current/). For CrocoLake's variables that are not present in the Argo program, we provide new names mantaining consistency with Argo's style.

Profile numbering

Ocean data is often accessed by profiles and we provide this functionality for CrocoLake too: the user can retrieve the profiles through the CYCLE_NUMBER variable, which is unique and progressive for each PLATFORM_NUMBER of each subdataset (DB_NAME). As each original product uses its own conventions, the CYCLE_NUMBER of some datasets is generated ad hoc during their conversion if no obvious match with CYCLE_NUMBER exists. The procedure for each dataset is detailed in the online documentation.

Quality control

CrocoLake contains only quality-controlled (QC) measurements. We rely exclusively on QCs performed by the data providers and at the time we do not perform any additional QC ourselves (although this might change in the future). Each parameter <PARAM> has a corresponding <PARAM>_QC flag that is generally set to 1 to indicate that the data is reliable. For Argo measurements, the original QC value is preserved, and only measurements with QC values of 1, 2, 5, and 8 considered.

Measurement errors

Each parameter <PARAM> has a corresponding <PARAM>_ERROR that indicates a measurement's error as provided in the original dataset. When no error is provided, <PARAM>_ERROR is set to null.

Documentation and updates

Documentation is available at (https://crocolakedocs.readthedocs.io/en/latest/index.html). It describes the specifics of each dataset (e.g. what quality-control filter we apply to each dataset, the procedure to generate the profile numbers, etc.), and get updated every time a new feature is made available.

Citation

If you use CrocoLakeTools and/or CrocoLake, please do not limit yourself to citing this manuscript but also remember to cite the datasets that you have used as indicated in the documentation. For example, if your work relies on Argo measurements, acknowledge Argo (Wong et al., 2020). This is important both for the maintainers of each product to track their impact and to acknowledge their efforts that made your work possible.

Acknowledgements

We acknowledge funding from [NSF CSSI CROCODILE details] and [NASA ECCO NSC…].

References

Wong, A. P., Wijffels, S. E., Riser, S. C., Pouliquen, S., Hosoda, S., Roemmich, D., Gilson, J., Johnson, G. C., Martini, K., Murphy, D. J., & others. (2020). Argo data 1999–2019: Two million temperature-salinity profiles and subsurface velocity observations from a global array of profiling floats. *Frontiers in Marine Science*, *7*, 700.