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

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 multiple tools, is inefficient over the cloud, and presents an overall high entrance barrier in terms of knowledge and time required to effectively access these resources. CrocoLakeTools is a Python package that address those shortcomings by providing workflows to convert several 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 different formats (e.g. netCDF, CSV) to parquet. CrocoLakeTools take 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 and data scientists and engineers in oceanograhy. CrocoLake was designed to be accessed by the wider oceanographic community.

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

39 datasets requires some general functionality (e.g. renaming the original variables to the final
40 schema), each conversion requires specific tools for the specific dataset (e.g. the map used to
41 rename the variable). CrocoLakeTools then hosts a converter for each dataset that implement
42 the specific needs of that datasets and inherits from `Converter`, which contains the shared
43 methods.

Workflow

45 The first step in the workflow is to retrieve the original files. This is generally left to the user,
46 although we provide methods to download Argo data and we wish to be able in the future to
47 provide this type of tools for other datasets as well.

Sources

49 These are the original data provided by the project, mission, scientist, etc. The format, schema,
50 nomenclature and conventions are those defined by the individual project and are unaware of
51 CrocoLake's workflow.

Storage (original)

53 Modules to download the original data are optional. They should inherit from the `Downloader`
54 class and be called downloader<ProjectName>, e.g. downloaderArgoGDAC. Whether a down-
55 loader module exists or the user downloads the data themselves, the original data is stored on
56 disk and this is the starting point for the converter.

Storage (converted)

58 The core of `CrocoLakeTools` are the modules in the `Converter` class and its subclasses. Each
59 project has its own subclass called converter<ProjectName>, e.g. converterGLODAP; further
60 specifiers can be added as necessary (e.g. at this time there a few different converters for Argo
61 data to prepare different datasets). The need for a dedicated converter for each project despite
62 the usage of common data formats (e.g. netCDF, CSV) is due to differences in the schema,
63 e.g. variable names, units, etc., while the steps that can be generalized for each format are
64 usually already included in other libraries that CrocoLakeTools rely on (e.g. pandas, dask,
65 pyarrow). Depending on the dataset, multiple converters can be applied. For example, to create
66 CrocoLake, Argo data goes through two converters: 1. converterArgoGDAC, which converts
67 the original Argo GDAC preserving most of its original conventions; 2. converterArgoQC,
68 which takes the output of the previous step and applies some filtering based on Argo's QC
69 flags and makes the data conforming to CrocoLake's schema.

CrocoLake

71 CrocoLake contains each converted dataset. The first step to build it is to create a direc-
72 tory containing symbolic links to the converted datasets (an example script is provided).
73 The submodule `CrocoLakeLoader` then allows to seamlessly load all the converted datasets
74 into memory as one dask dataframe with a uniform schema, using just a few lines. The
75 script merge_crocolake.py exploits CrocoLakeLoader's capabilities to generate one merged
76 CrocoLake dataset that contains all the converted datasets and is stored back to disk (in
77 parquet).

Accessing CrocoLake

79 The examples folder contains examples for how to access parquet datasets with several
80 programming languages: Python, Matlab, Julia. We hope to include soon R too. Separate
81 open repositories contains more examples for each language, see: CrocoLake-Python, CrocoLake-
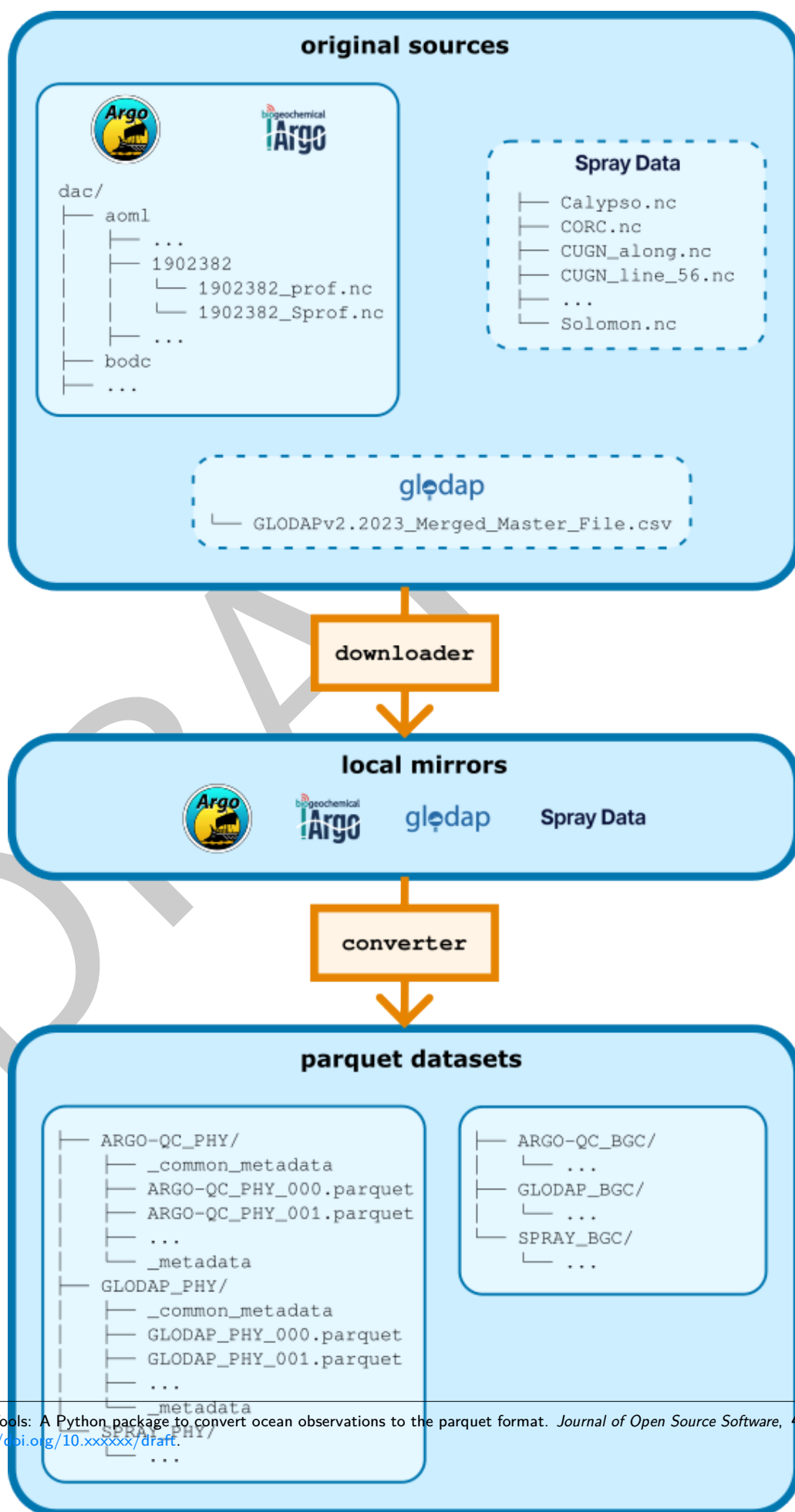82 Matlab, CrocoLake-Julia.

Example

[TD ADD FIGURES]

Figures

I am also trying this, maybe it works better with Fig Figure 1.

Documentation

Documentation is available at [TD add link]. It describes the CrocoLake dataset, the sub-datasets that it is made, and how they are obtained. It provides references to the original files origins for download before converting them.

Citation

If you use CrocoLakeTools and/or CrocoLake, do not limit yourself to citing this manuscript. 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 for each product to track their impact.

Acknowledgements

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.