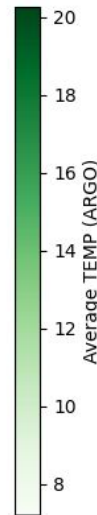
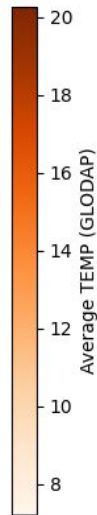
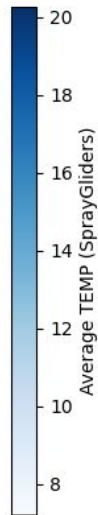
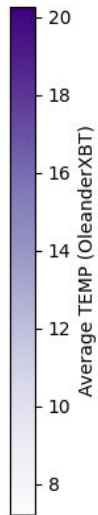
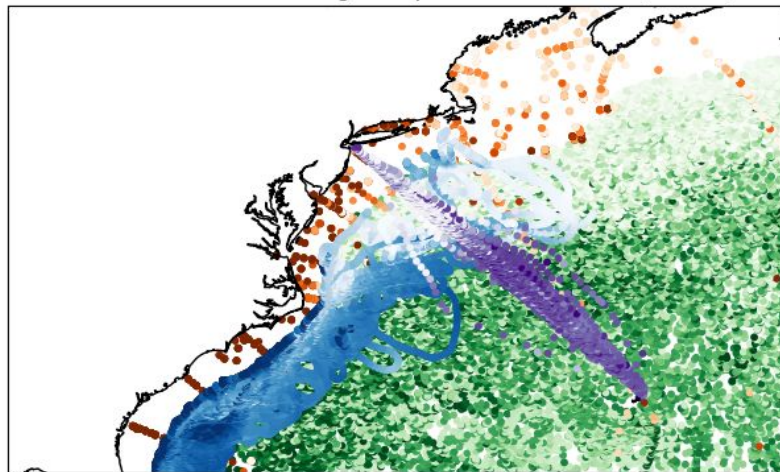


# Extending Crocolake's available datasets

North-West Atlantic average temperature 2010-2019 (Celsius)



[Enrico Milanese](#) (mentor), David Nagy (contributor)



WOODS HOLE  
OCEANOGRAPHIC  
INSTITUTION

## Background: CROCODILE

NSF CSSI project:

**C**ESM **R**egional **O**cean and **C**arbon  
**c**onfigurator with **D**ata assim**I**lation  
and **E**MBEDDING

### **Goal:**

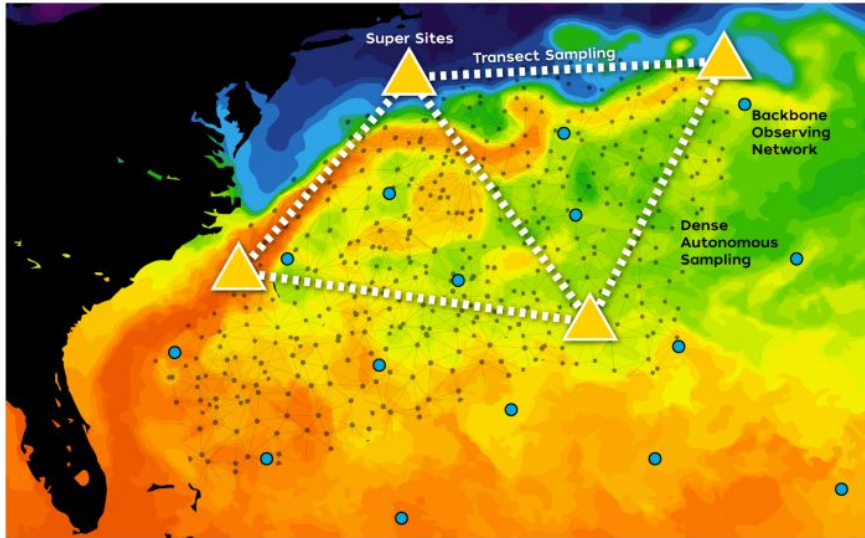
**How can we reduce the technological burden associated with new regional model configurations?**

Dan Amrhein, Manish Venumuddula, Gustavo Marques, Keith Lindsay, Helen Kershaw, Mike Levy, Alper Altuntas, Fred Castruccio, Susan Wijffels, David Nicholson, Enrico Milanese, Ashley Barnes, Helen Macdonald, Giovanni Seijo, Aidan Janney, Hung Nguyen, Andrew Kwong, Kate Boden



## Background: CROCODILE

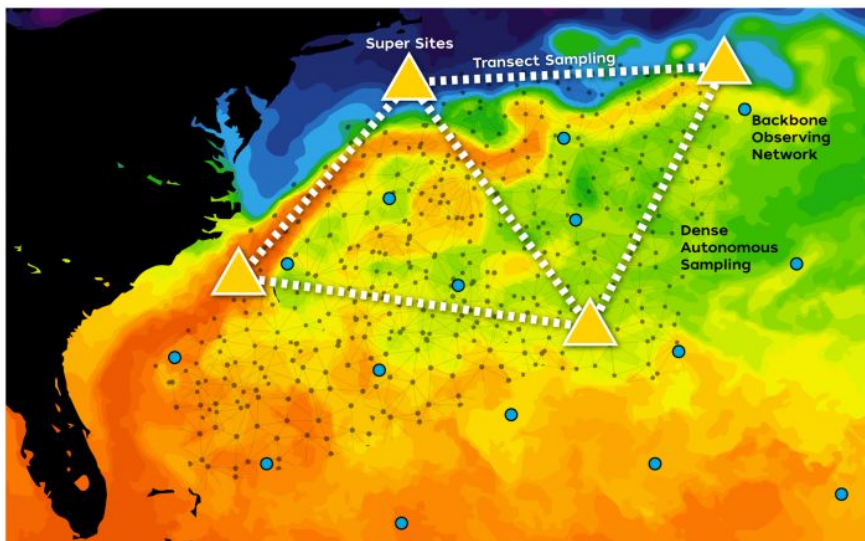
Let's say we want to **model** the NWA



(from WHOI-OVSN)

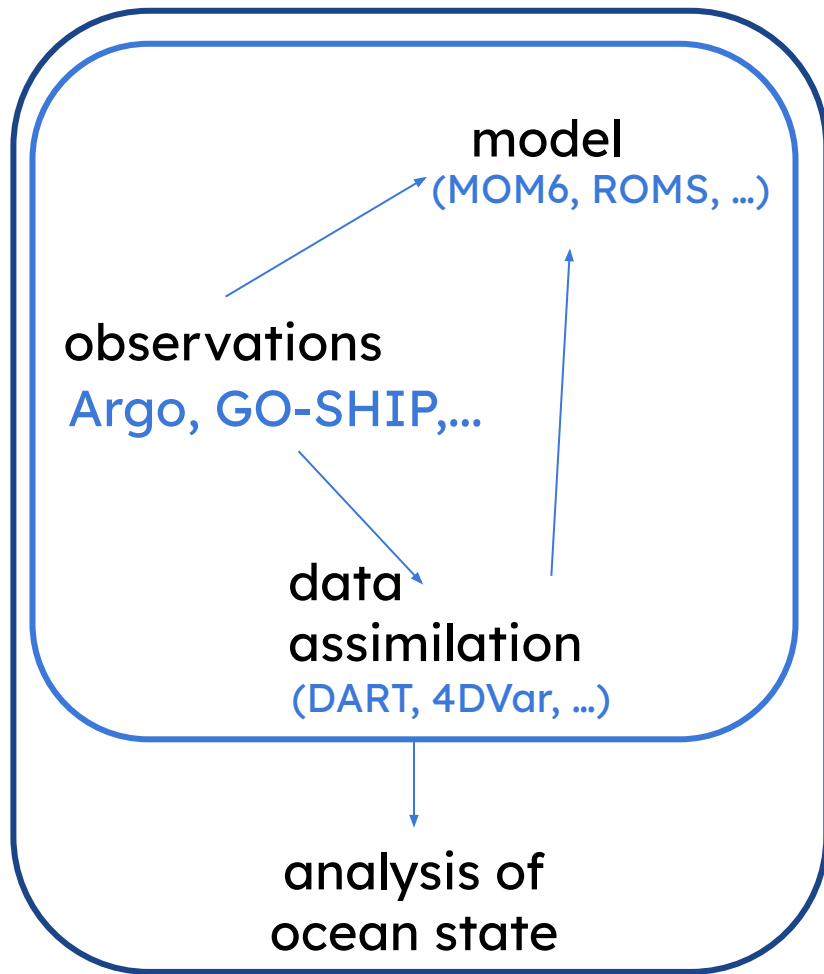
## Background: CROCODILE

Let's say we want to **model** the NWA



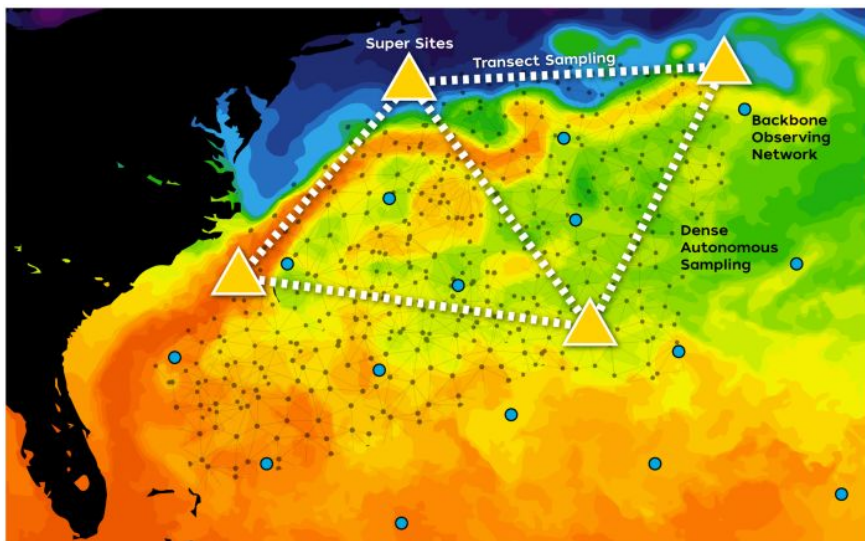
(from WHOI-OVSN)

repeatable workflow



## Background: CROCODILE

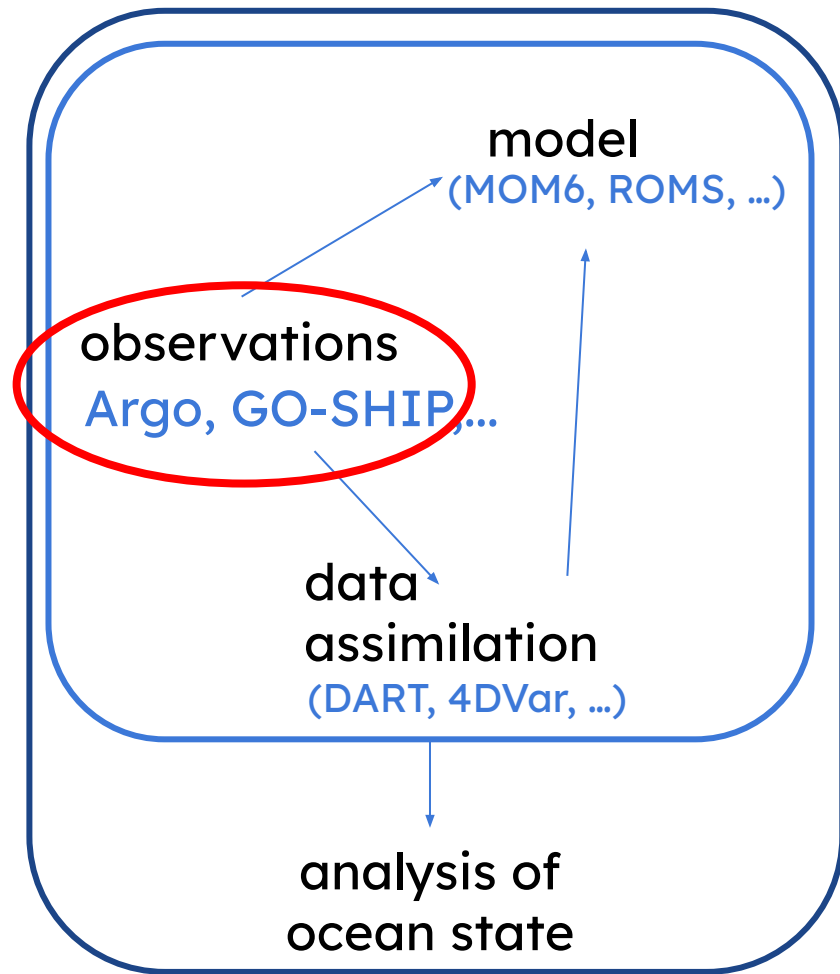
Let's say we want to **model** the NWA



(from WHOI-OVSN)

**We need  
observations from  
multiple sources**

repeatable workflow

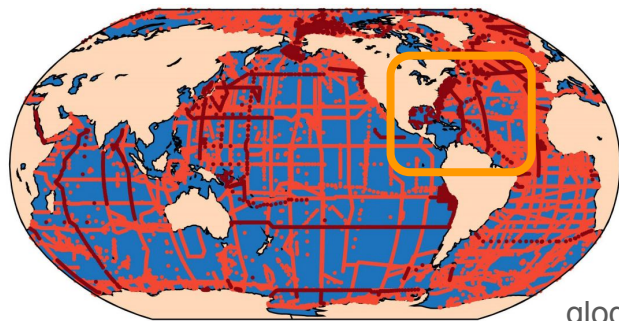
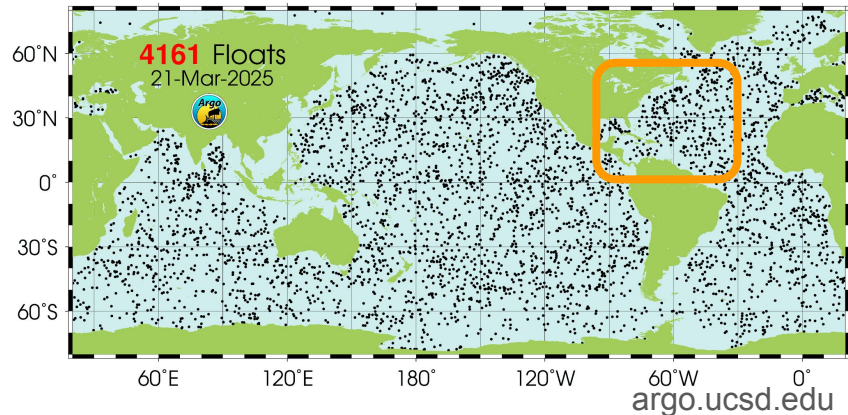


## Background: CrocoLake to facilitate access to ocean observations

Let's consider NWA temperature from three different products: Argo, GLODAP, Spray Gliders



[spraydata.ucsd.edu/projects/gs/](https://spraydata.ucsd.edu/projects/gs/)



[glodap.info](https://glodap.info)

## Background: CrocoLake to facilitate access to ocean observations

Let's consider NWA temperature from three different products: Argo, GLODAP, Spray Gliders

define filters in time, space, variables

fetch **Argo** data with filters (e.g. with argopy)

load filtered data into memory (array)

## Background: CrocoLake to facilitate access to ocean observations

Let's consider NWA temperature from three different products: Argo, GLODAP, Spray Gliders

define filters in time, space, variables

fetch **Argo** data with filters (e.g. with argopy)

load filtered data into memory (array)

download **GLODAP** master file (csv)

load all data into memory (**tabular**)

filter data (tabular)

download **Spray Glider** files (netCDF)

load filtered data into memory (**array**)

## Background: CrocoLake to facilitate access to ocean observations

Let's consider NWA temperature from three different products: Argo, GLODAP, Spray Gliders

define filters in time, space, variables

fetch Argo data with filters (e.g. with argopy)

load filtered data into memory (**array**)

download GLODAP master file (csv)

load all data into memory (tabular)

filter data (**tabular**)

download Spray Glider files (netCDF)

load filtered data into memory (**array**)

merge them into one structure (table) matching names, etc  
[perform QC]

## Background: CrocoLake to facilitate access to ocean observations

Let's consider NWA temperature from three different products: Argo, GLODAP, Spray Gliders

define filters in time, space, variables

fetch Argo data with filters (e.g. with argopy)

load filtered data into memory (**array**)

need to process  
one file per float

download GLODAP master file (csv)

load all data into memory (tabular)

filter data (**tabular**)

csv is inefficient

download Spray Glider files (netCDF)

load filtered data into memory (**array**)

“TEMP”,  
“temperature”,  
“G2temperature”

merge them into one structure (table) matching names, etc  
[perform QC]

## Background: CrocoLake to facilitate access to ocean observations

Let's consider NWA temperature from three different products: Argo, GLODAP, Spray Gliders

### **lack of uniform access**

bottlenecks:

- both knowledge-related (.csv, .netCDF; array vs tabular)
- and technological (slow read performance; long workflows)

**can we make  
a new grad's life easier?**



## Background: CrocoLake to facilitate access to ocean observations

### CrocoLake

CROCODILE's "best and freshest" collection of ocean observations

- maintained by me @ WHOI-NCAR
- ~1.2 billion observations
- both physical and biogeochemical measurements
- it's a collection: Argo (~97% of obs), GLODAP, Spray Gliders, Oleander XBT, OCS Saildrones
- only data *already QCed by provider*
- contains *instrument* errors (where available), not *representation* errors
- updated weekly (Argo, OleanderXBT)
- format: parquet

## Background: CrocoLake to facilitate access to ocean observations

### CrocoLake

CROCODILE's “**best** and **freshest**” collection of ocean observations

- maintained by me @ WHOI-NCAR
- ~1.2 billion observations
- both physical and biogeochemical measurements
- it's a collection: Argo (~97% of obs), GLODAP, Spray Gliders, Oleander XBT, OCS Saildrones
- **only data *already QCed by provider***
- contains *instrument* errors (where available), not *representation* errors
- **updated weekly (Argo, OleanderXBT)**
- format: parquet

# Background: CrocoLake to facilitate access to ocean observations

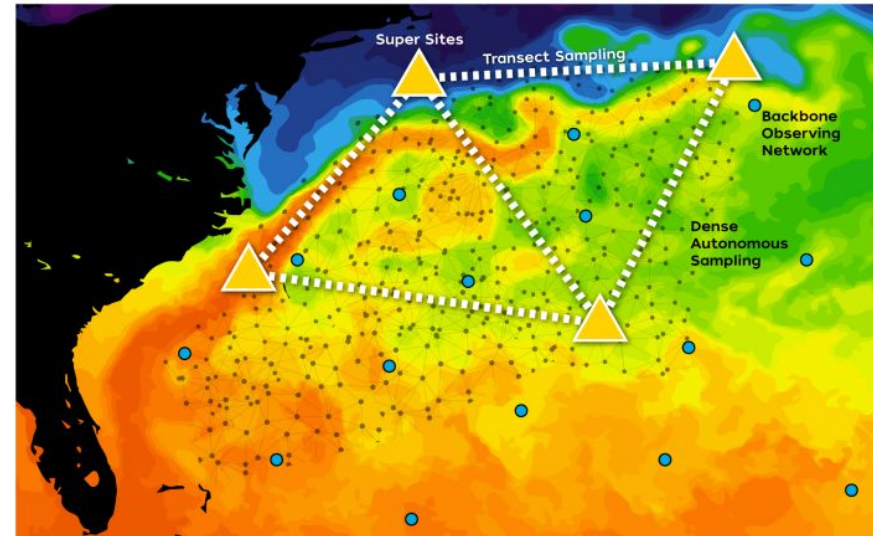
## CrocoLake

CROCODILE's “best and freshest” collection of ocean observations

- close collaboration with WHOI's projects led by Susan Wijffels and David Nicholson:

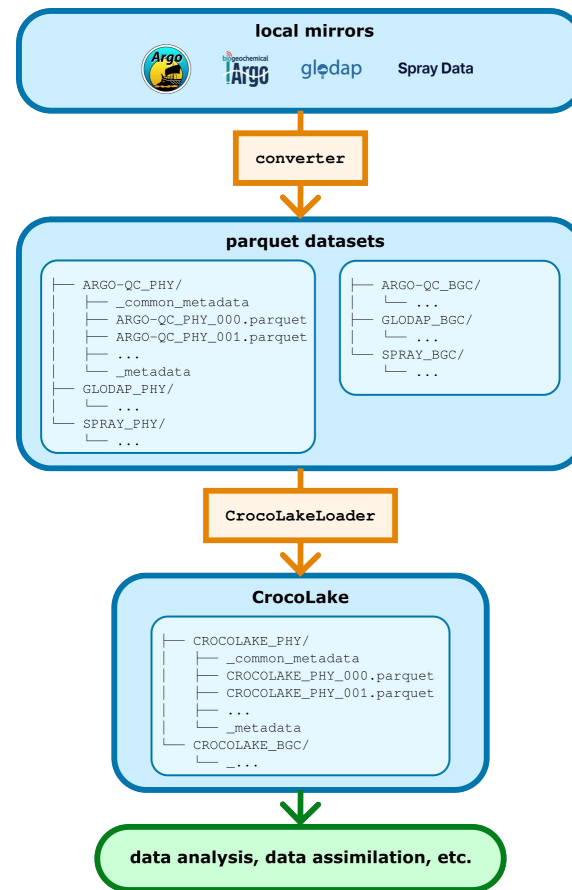
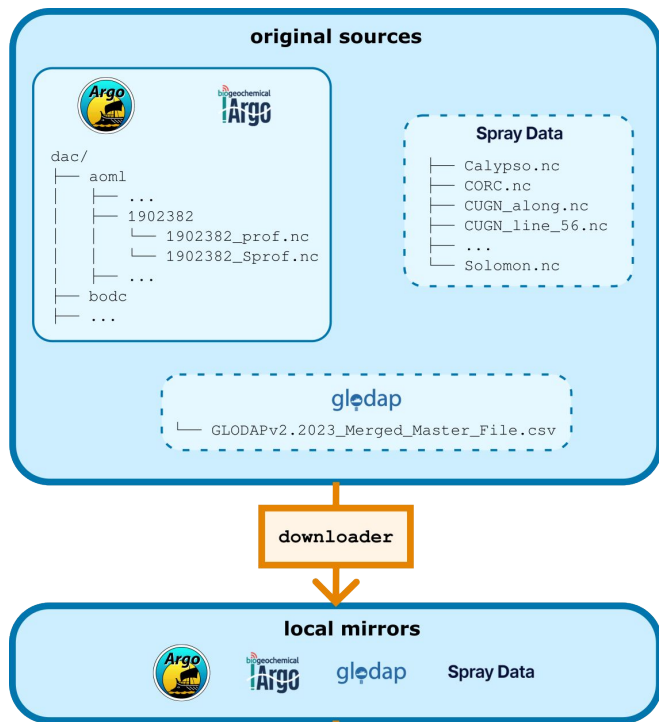
- ▷ Argo Core and BGC teams
- ▷ Ocean Vital Signs Network (OVSN)

⇒ special interest in  
NorthWest Atlantic



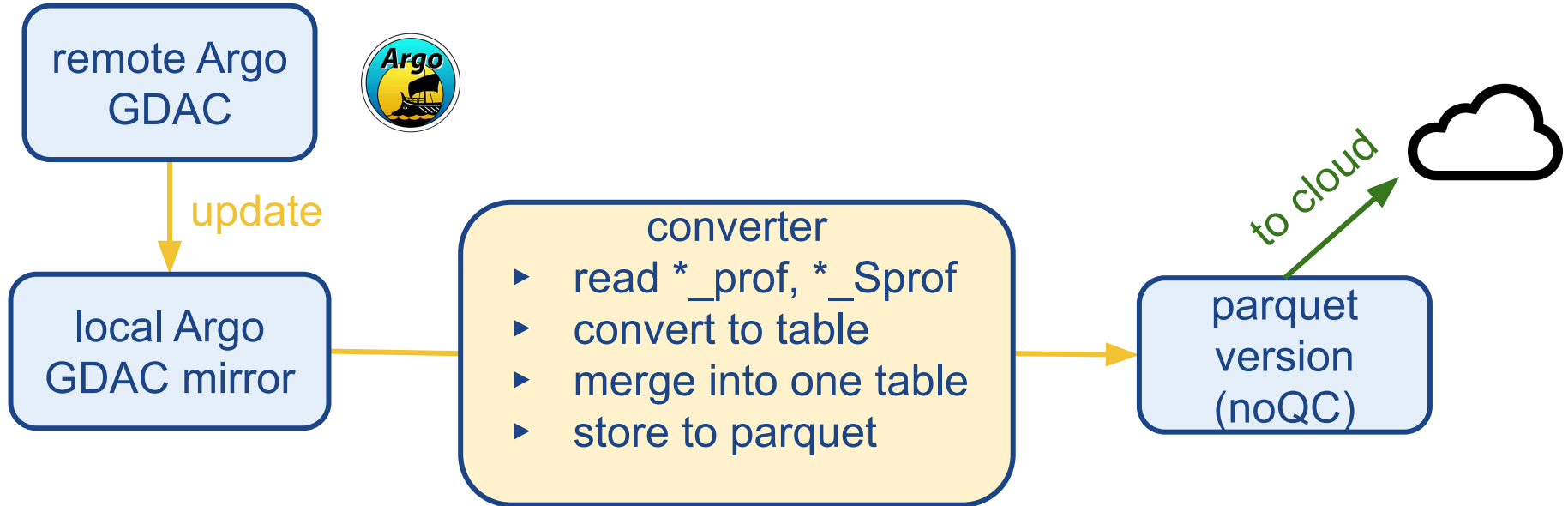
# CrocoLakeTools: workflow to generate CrocoLake

## CrocoLakeTools



# CrocoLakeTools: workflow to generate CrocoLake

Saturday 00:00 UTC



PHY: ~1 hr

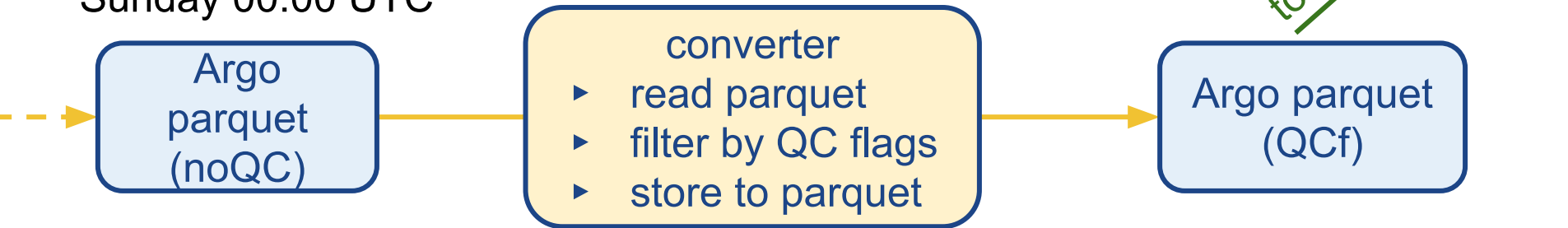
BGC: ~12 hrs

## Data products – workflow 2/3

PHY: ~10 mins

BGC: ~2 hrs

Sunday 00:00 UTC

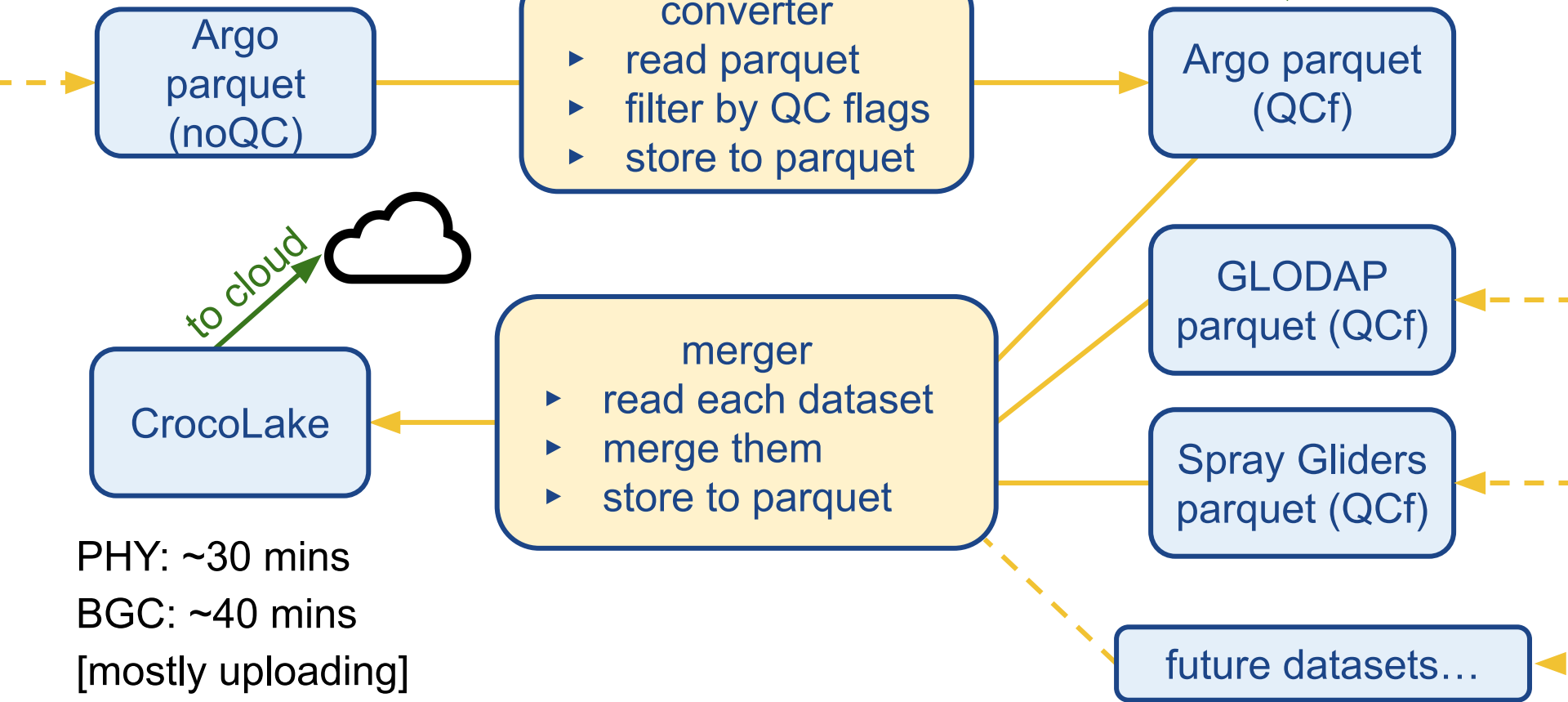


## Data products – workflow 3/3

PHY: ~10 mins

BGC: ~2 hrs

Sunday 00:00 UTC

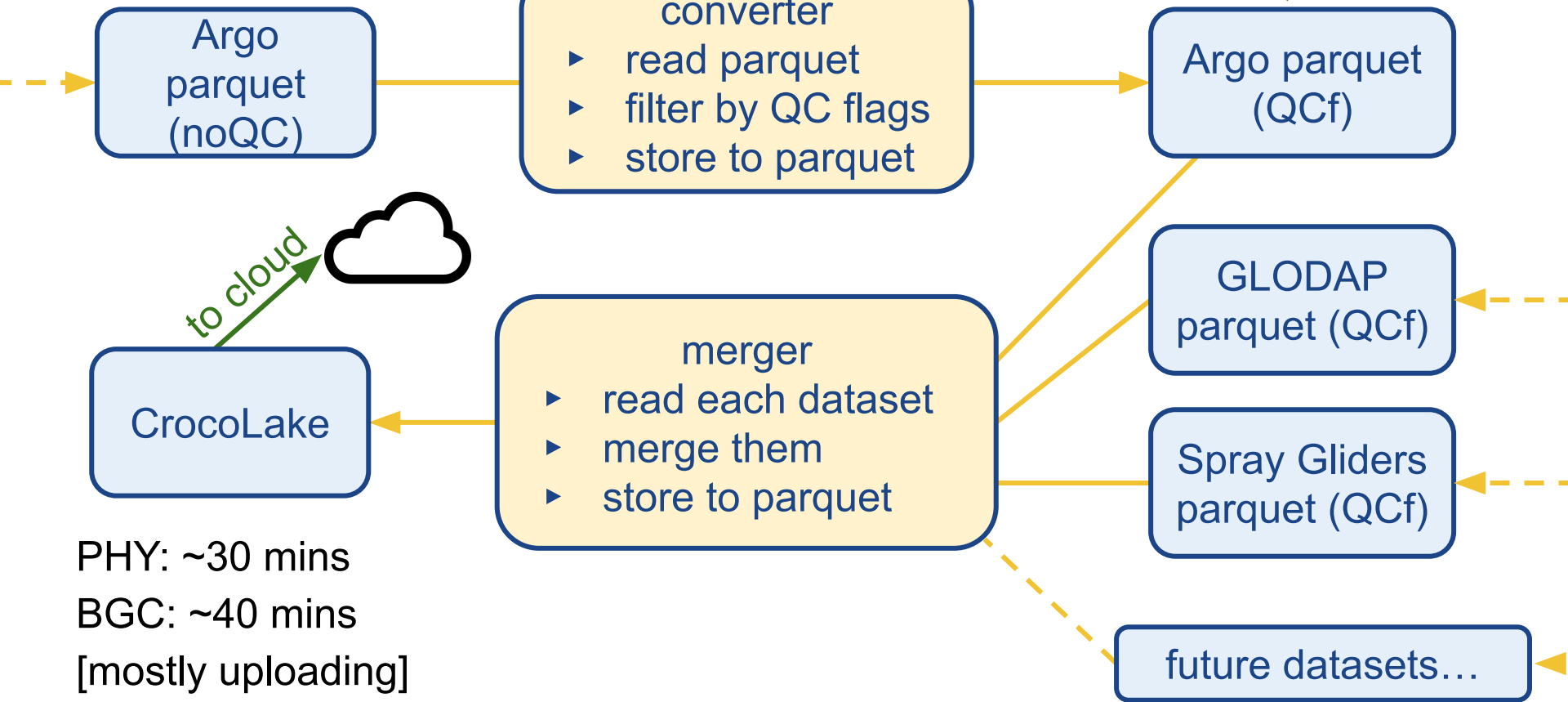


## Data products – workflow 3/3

PHY: ~10 mins

BGC: ~2 hrs

Sunday 00:00 UTC



## GSoC project: Extending Crocolake's available datasets

Contributor: David Nady, data science undergrad @ Minia University (Egypt)

## GSoC project: Extending Crocolake's available datasets

Contributor: David Nady, data science undergrad @ Minia University (Egypt)

Modular project structure:

- Datasets of interests: Saldrones, GO-SHIP, OleanderXBT, CPR

## GSoC project: Extending Crocolake's available datasets

Contributor: David Nady, data science undergrad @ Minia University (Egypt)

Modular project structure:

- Datasets of interests: Sairdrones, GO-SHIP, OleanderXBT, CPR
- For each ds:
  - Identifying variables to keep in CrocoLake's schema
  - Add downloader module
  - Add converter module
  - Add tests
  - Verify it merges with existing datasets
  - Add documentation

↪ for both physical and biogeochemical CrocoLake's flavours

## GSoC project: Extending Crocolake's available datasets

Contributor: David Nady, data science undergrad @ Minia University (Egypt)

**Modular** project structure:

- Datasets of interests: **Saildrones**, GO-SHIP, **OleanderXBT**, **(CPR)**
- For each ds:
  - Identifying variables to keep in CrocoLake's schema
  - Add downloader module
  - Add converter module
  - Add tests
  - Verify it merges with existing datasets
  - Add documentation

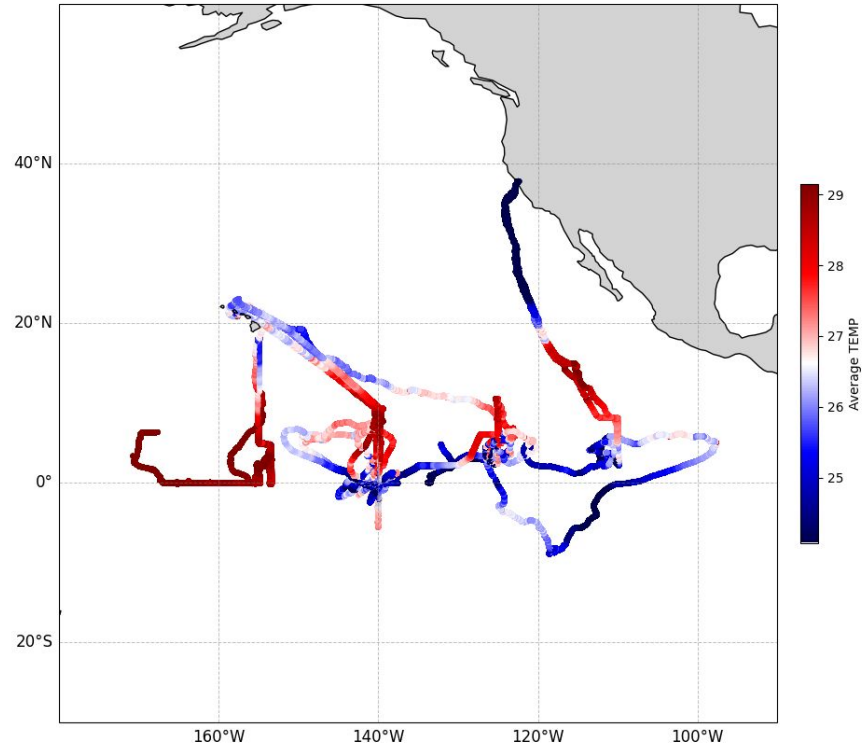
Allowed me to adjust based on contributor's interests/skillset

↪ for both physical and biogeochemical CrocoLake's flavours

## GSoC project: Extending Crocolake's available datasets

### OCS Saildrones:

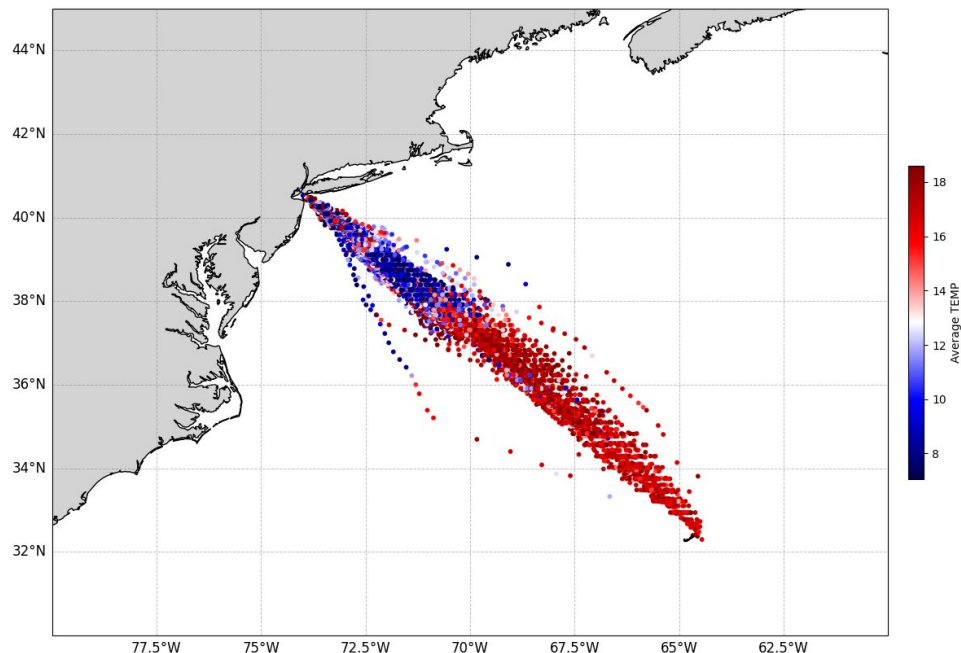
- Wind-powered autonomous surface vehicles
- NOAA PMEL + Saildrone Inc.
- Multi-month missions (2017-present)
- Pacific ocean
- Physical and BGC
- PHY: ~4.3 millions obs (0.35% of total; 11.7% non-Argo)



## GSoC project: Extending Crocolake's available datasets

### OleanderXBT:

- M/V Oleander container ship
- Weekly XBT deployments
- 1977-present (~2500 files)
- Route: New Jersey ↔ Bermuda
- North West Atlantic ocean
- Physical (temperature only)
- PHY: ~4.1 millions obs  
(0.33% of total; 11.0% non-Argo)



## GSoC project: Adding a dataset

Developers implements:

`converterNewDataset.py`

- `read_to_ddf()`: Parse source format → dask DataFrame
- `process_df()`: converts to table and CrocoLake schema
- `standardize_data()`: field mapping logic
- (optional) chunking of large files

`params.py`:

- variables list and mappings

Framework deals with:

`converter.py` (base class)

- `convert()`: top-level workflow (read → standardize → write)
- configuration management (yaml)
- generates schema
- common logic (apply mappings)
- Pandas↔PyArrow mapping
- compute derived vars (abs salinity, potential temperature, sigma)
- remove invalid rows, convert units...

## GSoC project: Adding a dataset

Parallelism: files download (pre-conversion), file read and file-chunking (during conversion), usage of Locks

Saildrones (more complex):

- Sensor merging (multiple TEMP sensors → single TEMP field)
- Add explicit depth dimension (in metadata in original files)
- DOXY:  $\mu\text{mol/L}$  →  $\mu\text{mol/kg}$  (density-based conversion)

Oleander XBT (simple):

- mostly uses existing framework
- temperature → TEMP (standardized naming)
- add default quality flags
- PSAL = NaN (XBT measures temperature only)

# CrocoLake's environment

[crocolaketools](#) : to generate CrocoLake

Language-agnostic -> examples to use CrocoLake:

[crocolake-python](#), [crocolake-julia](#), [crocolake-matlab](#)

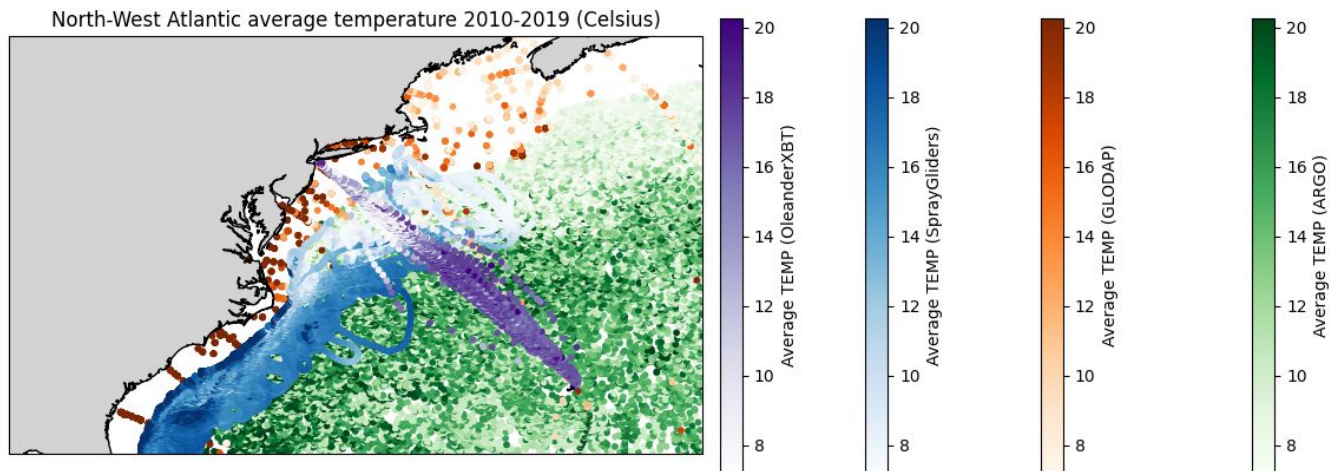
Example:

~8 mln obs

~700 MB

~2.6 s reading

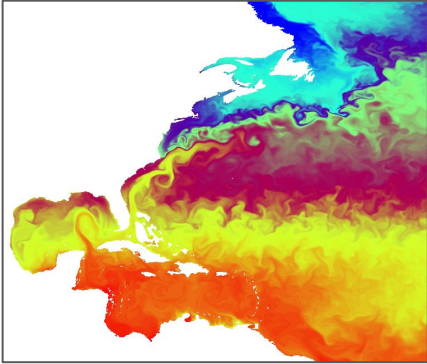
~3.8 s plotting



## CrocoLake's environment

CrocoCamp (to be re-named soon) : model-data comparison

**Ocean model output  
(MOM6, ROMS)**

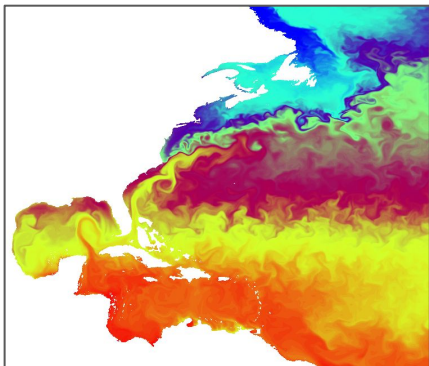


**Ocean obs (CrocoLake)**

# CrocoLake's environment

CrocoCamp (to be re-named soon) : model-data comparison

**Ocean model output  
(MOM6, ROMS)**



```
from crococamp.workflows import WorkflowModelObs
from crococamp.viz import InteractiveMapWidget

# interpolate model onto obs space
workflow = WorkflowModelObs.from_config_file("config.yaml")
workflow.run(clear_output=True)

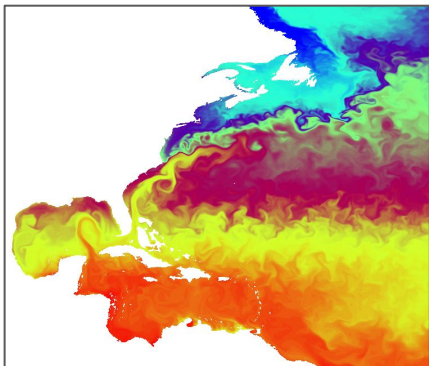
# interactive map
widget = InteractiveMapWidget(ddf)
```

**Ocean obs (CrocoLake)**

# CrocoLake's environment

**CrocoCamp** (to be re-named soon) : model-data comparison

**Ocean model output  
(MOM6, ROMS)**

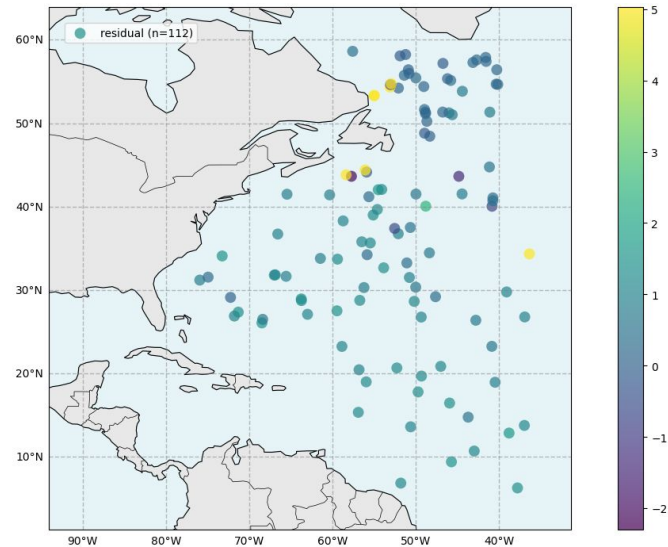


**model interpolated  
onto observation  
location**

```
from crococamp.workflows import WorkflowModelObs
from crococamp.viz import InteractiveMapWidget

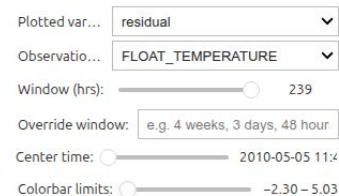
# interpolate model onto obs space
workflow = WorkflowModelObs.from_config_file("config.yaml")
workflow.run(clear_output=True)

# interactive map
widget = InteractiveMapWidget(ddf)
```



**Ocean obs (CrocoLake)**

**dataframes  
+  
interactive widgets**



## Ongoing and future work

- **Expanding CrocoLake (IOOS Gliders? Other IOOS DACs? GSoC?)**
- [BaskTemplate](#): regional model use-case with all CROCODILE's tools (CrocoLake, model-obs comparison, MOM6 case generator, etc.)
- JOSS manuscript
- Re-gridded CrocoLake version
- Regional observational and model climatologies studies

[enrico.milanese@whoi.edu](mailto:enrico.milanese@whoi.edu)



WOODS HOLE  
OCEANOGRAPHIC  
INSTITUTION



**Thank you!  
and thank IOOS for GSoC!**