Model-data comparison tools using DART and CrocoLake

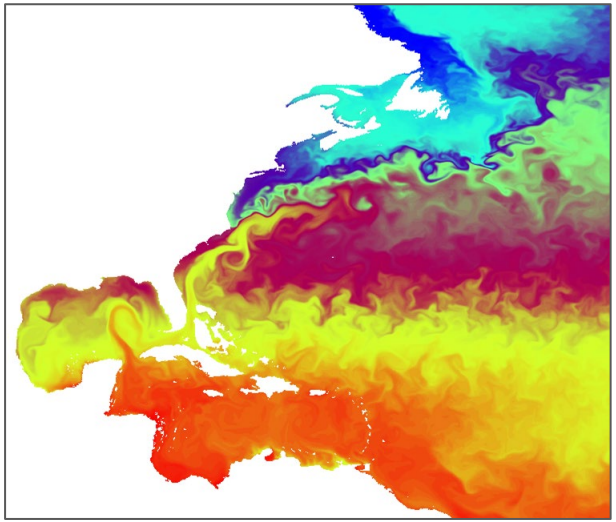
Enrico Milanese

Ocean Model Working Group Meeting 2026

NSF NCAR Mesa Lab, Boulder, CO

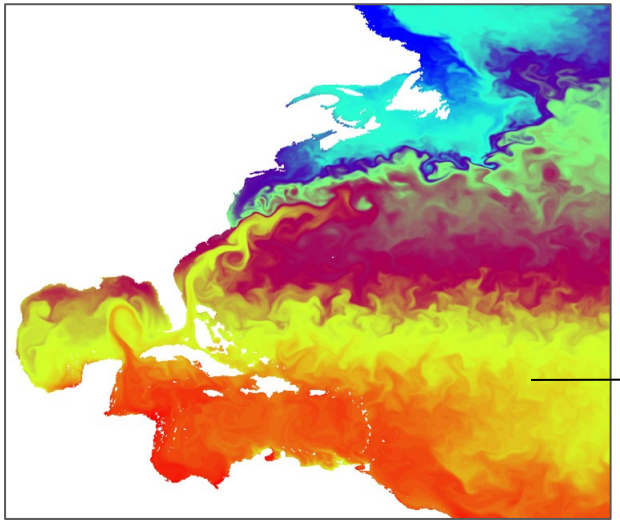
February 6, 2025

You just ran your MOM6 simulation



NWA12, sea surface temperature, 2010-05-01

You just ran your MOM6 simulation



NWA12, sea surface temperature, 2010-05-01

correct?

https://argo.ucsd.edu/
(photo by Adam Ü)

https://anibos.com/

https://bios.asu.edu/oleander

https://spraydata.ucsd.edu/

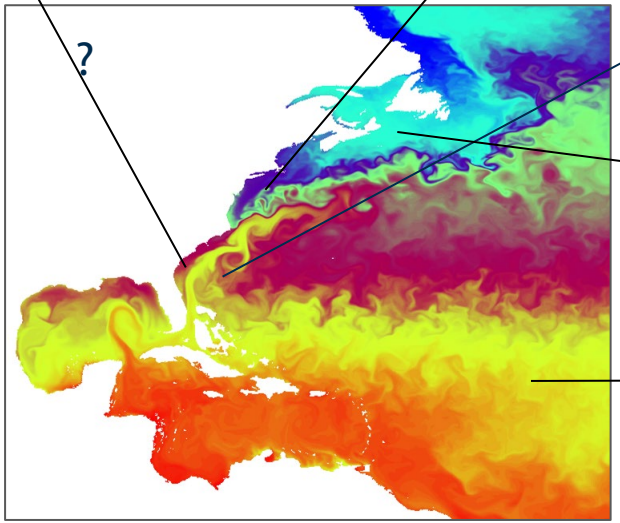
?

?

?

?

?



glodap

NWA12, sea surface temperature, 2010-05-01

https://argo.ucsd.edu/
(photo by Adam Ü)

Next: loading NWA temperature observations from 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)

Next: loading NWA temperature observations from 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)

Next: loading NWA temperature observations from 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)

Next: loading NWA temperature observations from 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?]

Next: loading NWA temperature observations from different products (Argo, GLODAP, Spray Gliders)

need to process one file per float

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) → csv is inefficient
filter data (tabular)

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?]

lack of uniform access

bottlenecks:

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

how can we make
your life easier?

CrocoLake: cloud-optimized collection of 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)
- formats: parquet, (DART's obs sequence)

CrocoLake: cloud-optimized collection of 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)
- formats: parquet, (DART's obs sequence)

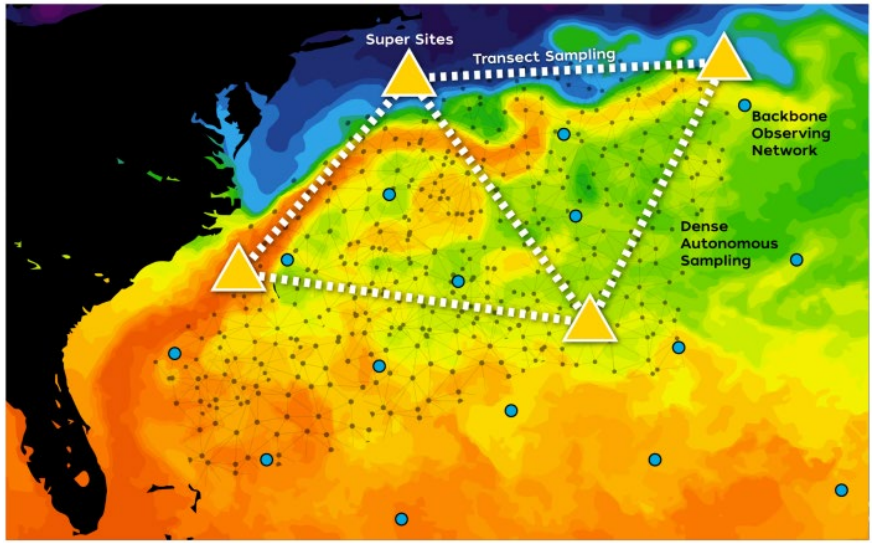
CrocoLake: cloud-optimized collection of observations

CrocoLake

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

- close collaboration with Susan Wijffels and David Nicholson @WHOI:
  - Argo Core and BGC teams
  - Ocean Vital Signs Network (OVSN)

⇒ special interest in NorthWest Atlantic



CrocoLake: cloud-optimized collection of observations

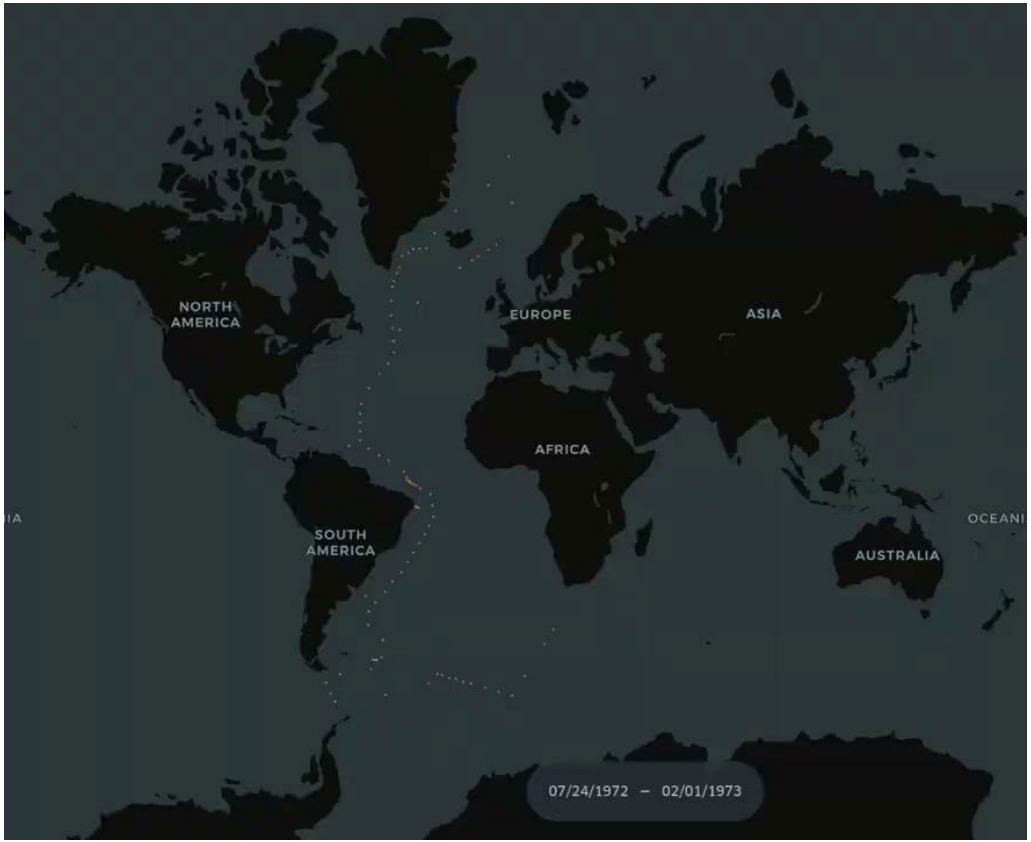
CrocoLake

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

- multi-language support:
  - ✓ crocolake-python
  - ✓ crocolake-matlab
  - ✓ crocolake-julia
- on-disk CrocoLake is the same
- code to access changes with language

CrocoLake: cloud-optimized collection of observations

CrocoLake



CrocoLake: cloud-optimized collection of observations

```
%%time
cols = ["DB_NAME","LATITUDE","LONGITUDE","PRES","JULD","TEMP"]
filter_coords_time_temp = [
    ("LATITUDE",">",lat0), ("LATITUDE","<",lat1),
    ("LONGITUDE",">",lon0), ("LONGITUDE","<",lon1),
    ("JULD",">",date0), ("JULD","<",date1),
    ("TEMP",">=",-1e30),("TEMP","<=",+1e30)
]
ds = pq.ParquetDataset(parquet_dir, schema=crocolake_schema, filters=filter_coords_time_temp)
df = ds.read(columns=cols).to_pandas()
df
```

```
CPU times: user 21.5 s, sys: 894 ms, total: 22.4 s
Wall time: 2.42 s
```

	DB_NAME	LATITUDE	LONGITUDE	PRES	JULD	TEMP
0	ARGO	32.115	-64.427	4.36	2010-01-02 03:16:13.999788544	20.747
1	ARGO	32.115	-64.427	5.96	2010-01-02 03:16:13.999788544	20.747999
2	ARGO	32.115	-64.427	7.96	2010-01-02 03:16:13.999788544	20.747
3	ARGO	32.115	-64.427	10.160001	2010-01-02 03:16:13.999788544	20.740999
4	ARGO	32.115	-64.427	12.160001	2010-01-02 03:16:13.999788544	20.742001
...	...	...	...	...	...	...
8150312	OleanderXBT	32.670734	-64.745903	911.322876	2019-11-04 07:35:04.999995136	9.16
8150313	OleanderXBT	32.670734	-64.745903	911.929749	2019-11-04 07:35:04.999995136	9.15
8150314	OleanderXBT	32.670734	-64.745903	912.546814	2019-11-04 07:35:04.999995136	9.15
8150315	OleanderXBT	32.670734	-64.745903	913.153687	2019-11-04 07:35:04.999995136	9.14
8150316	OleanderXBT	32.670734	-64.745903	913.770691	2019-11-04 07:35:04.999995136	9.13

8150317 rows × 6 columns

CrocoLake: cloud-optimized collection of observations

few lines!
no treatment for each db!

```
%%time
cols = ["DB_NAME","LATITUDE","LONGITUDE","PRES","JULD","TEMP"]
filter_coords_time_temp = [
    ("LATITUDE",">",lat0), ("LATITUDE","<",lat1),
    ("LONGITUDE",">",lon0), ("LONGITUDE","<",lon1),
    ("JULD",">",date0), ("JULD","<",date1),
    ("TEMP",">=",-1e30),("TEMP","<=",+1e30)
]
ds = pq.ParquetDataset(parquet_dir, schema=crocolake_schema, filters=filter_coords_time_temp)
df = ds.read(columns=cols).to_pandas()
df
```

filters

load filtered data (similar syntax in Matlab, Julia)

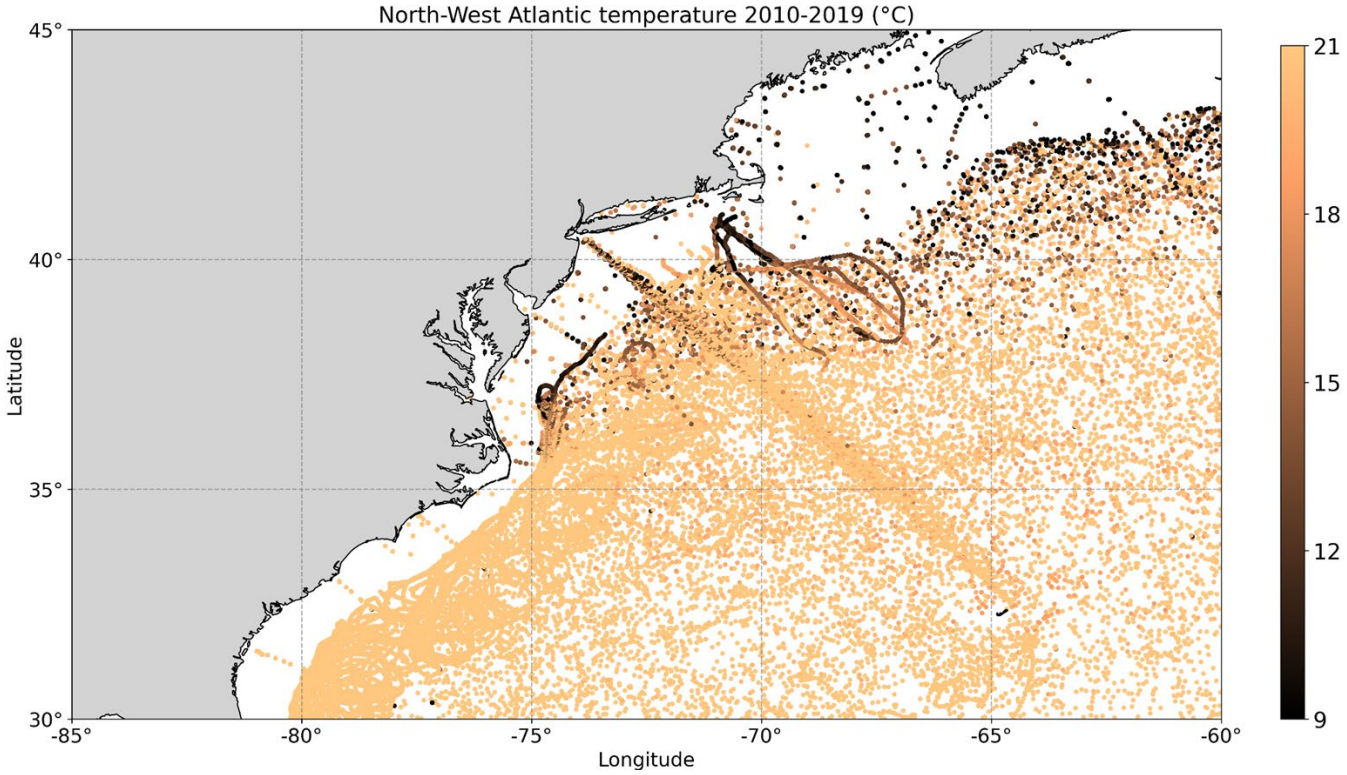
```
CPU times: user 21.5 s, sys: 894 ms, total: 22.4 s
Wall time: 2.42 s
```

	DB_NAME	LATITUDE	LONGITUDE	PRES	JULD	TEMP
0	ARGO	32.115	-64.427	4.36	2010-01-02 03:16:13.999788544	20.747
1	ARGO	32.115	-64.427	5.96	2010-01-02 03:16:13.999788544	20.747999
2	ARGO	32.115	-64.427	7.96	2010-01-02 03:16:13.999788544	20.747
3	ARGO	32.115	-64.427	10.160001	2010-01-02 03:16:13.999788544	20.740999
4	ARGO	32.115	-64.427	12.160001	2010-01-02 03:16:13.999788544	20.742001
...	...	...	...	...	...	...
8150312	OleanderXBT	32.670734	-64.745903	911.322876	2019-11-04 07:35:04.999995136	9.16
8150313	OleanderXBT	32.670734	-64.745903	911.929749	2019-11-04 07:35:04.999995136	9.15
8150314	OleanderXBT	32.670734	-64.745903	912.546814	2019-11-04 07:35:04.999995136	9.15
8150315	OleanderXBT	32.670734	-64.745903	913.153687	2019-11-04 07:35:04.999995136	9.14
8150316	OleanderXBT	32.670734	-64.745903	913.770691	2019-11-04 07:35:04.999995136	9.13

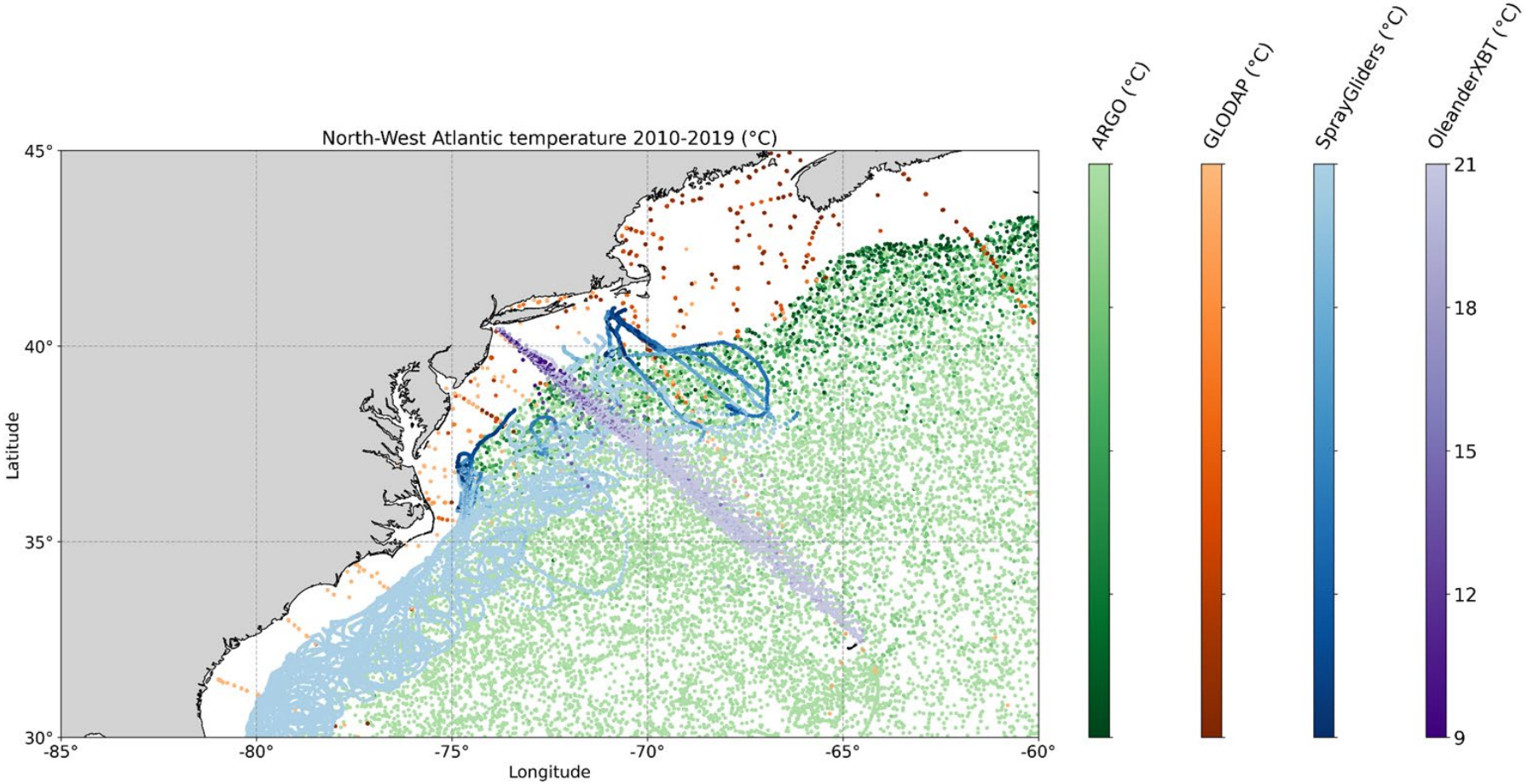
8150317 rows × 6 columns

data in same uniform structure (pandas dataframe)

CrocoLake: cloud-optimized collection of observations



CrocoLake: cloud-optimized collection of observations



CrocoLake: cloud-optimized collection of observations

World Ocean Database		CrocoLake
○ long established ○ ad hoc QC ○ largest records	*Pros*	○ weekly refreshed ○ download with link ○ fast and cloud -optimized ○ I'm here :) tell me what you need
○ updates every ~5 years (major) or quarterly (minor, Nov '24?) ○ fill in form to download ○ not cloud -optimized	*Cons*	○ fewer metadata (units) ○ fewer observations (for now…?) ○ generally "beta"

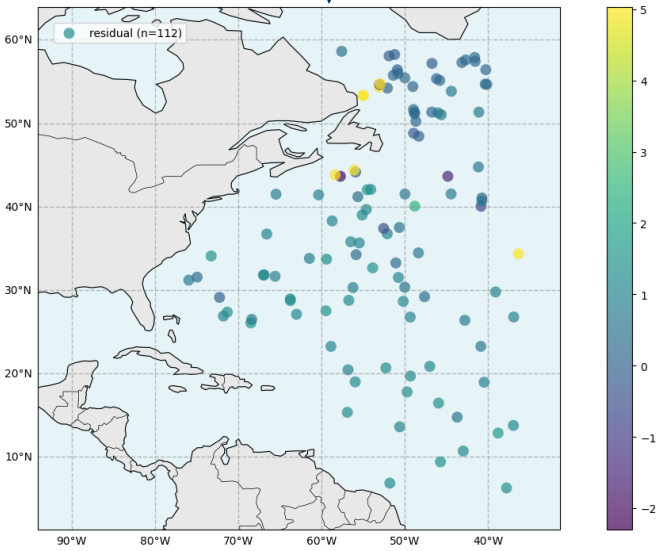
Comparing model output to observations (CrocoLake, WOD)

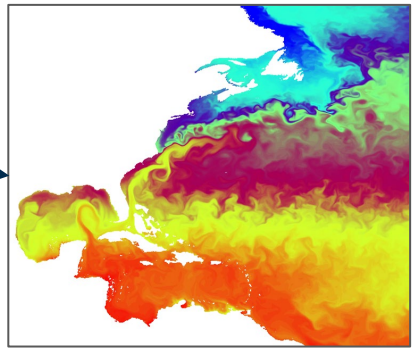
```python
from crococamp.workflows import WorkflowModelObs
from crococamp.viz import InteractiveMapWidget

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

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

Spray Data

glodap

Argo





Plotted var… residual

Observatio… FLOAT_TEMPERATURE

Window (hrs): 239

Override window: e.g. 4 weeks, 3 days, 48 hour

Center time: 2010-05-05 11:

Colorbar limits: −2.30 – 5.03

Comparing model output to observations (CrocoLake, WOD)

```python
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)
```

Spray Data

gl◉dap

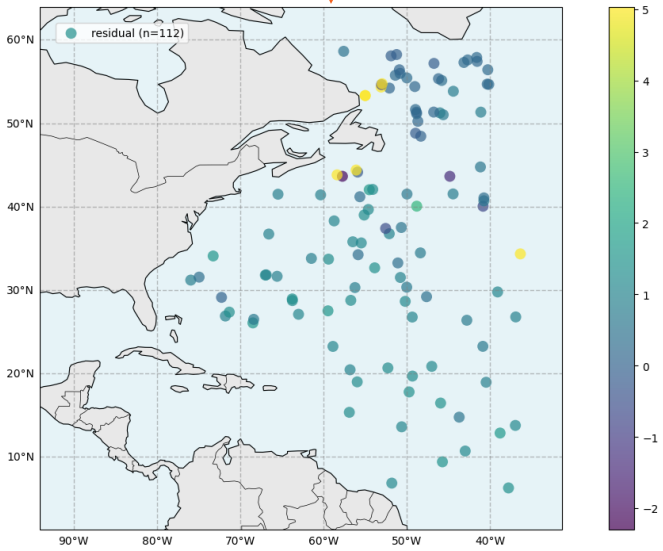
Argo

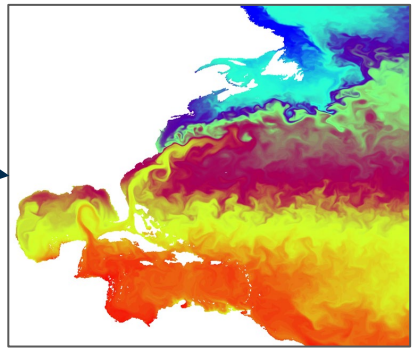
DART tools

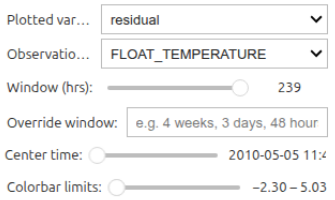
(“perfect_model_obs”)

The Data Assimilation Research Testbed (DART) is an open -source, freely available community facility for ensemble data assimilation (DA).

dev & maintained by NCAR







Comparing model output to observations (CrocoLake, WOD)

CrocoCamp

runs it for you!

```python
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)
```

Spray Data

gliders

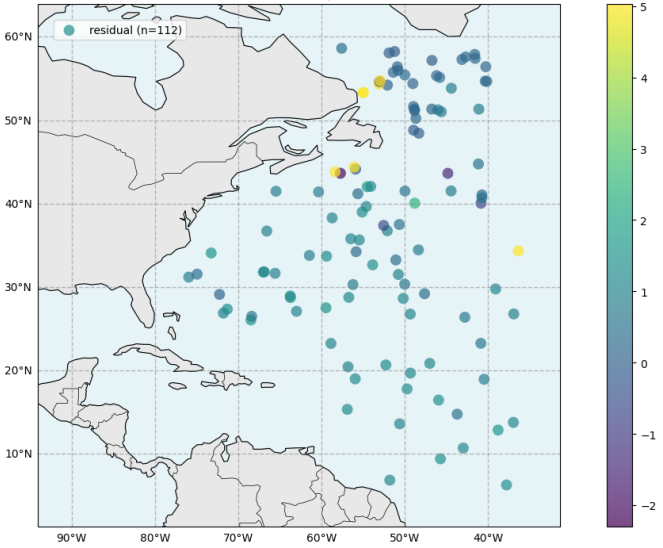
Argo

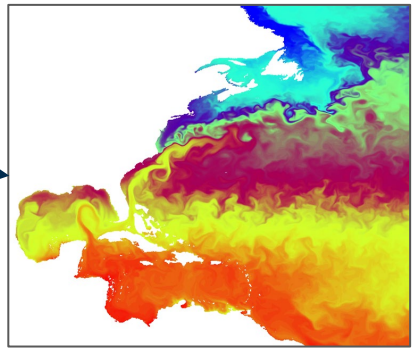
DART tools

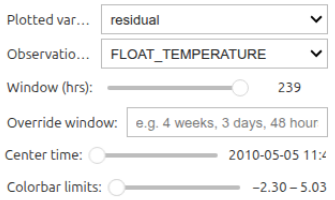
("perfect_model_obs")

The Data Assimilation Research Testbed (DART) is an open -source, freely available community facility for ensemble data assimilation (DA).

dev & maintained by NCAR







Comparing model output to observations (CrocoLake, WOD)

CrocoCamp

DART tools

("perfect_model_obs")

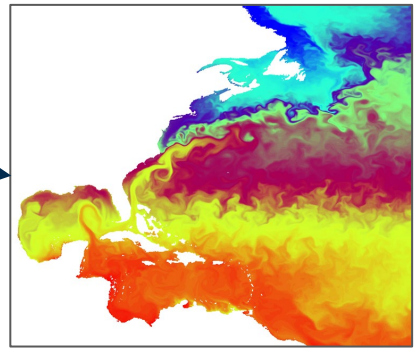
CrocoCamp

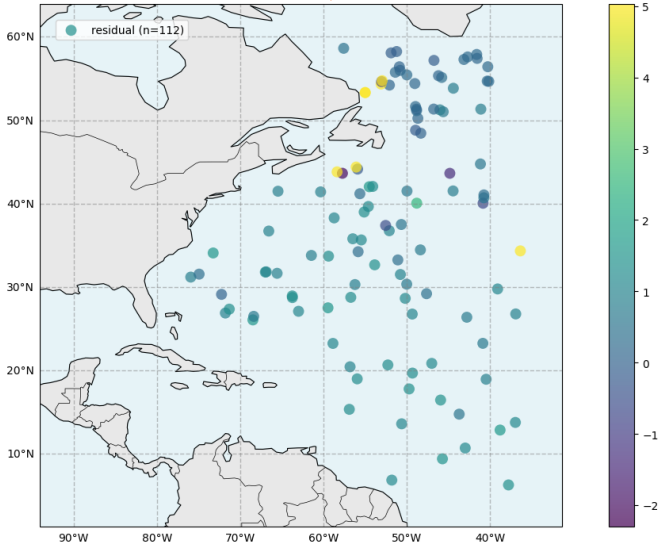
```python
from crococamp.workflows import WorkflowModelObs
from crococamp.viz import InteractiveMapWidget

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

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

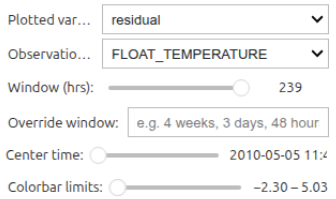
Spray Data

glodap

Argo







Comparing model output to observations (CrocoLake, WOD)

CrocoCamp

DART tools

("perfect_model_obs")

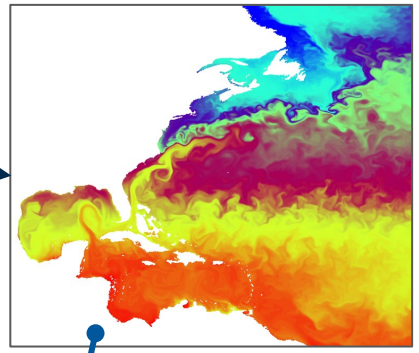
CrocoCamp

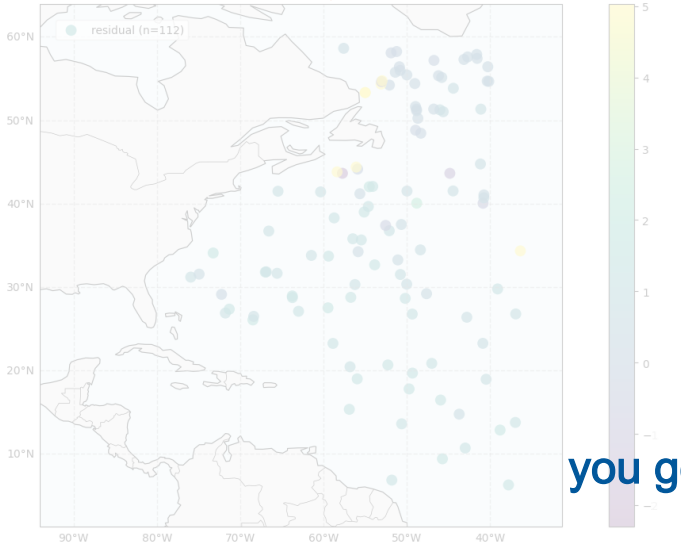
```python
from crococamp.workflows import WorkflowModelObs
from crococamp.viz import InteractiveMapWidget

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

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

Spray Data

glodap

Argo





you get this from MOM6

Comparing model output to observations (CrocoLake, WOD)

CrocoCamp

DART tools

("perfect_model_obs")

CrocoCamp

```python
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)
```

Spray Data

how about this?

Comparing model output to observations (CrocoLake, WOD)

```python
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)
```

CrocoCamp

DART tools

("perfect_model_obs")

CrocoCamp

Spray Data

glodap

Argo

how about this?

CrocoLake (or WOD)

Comparing model output to observations (CrocoLake, WOD)

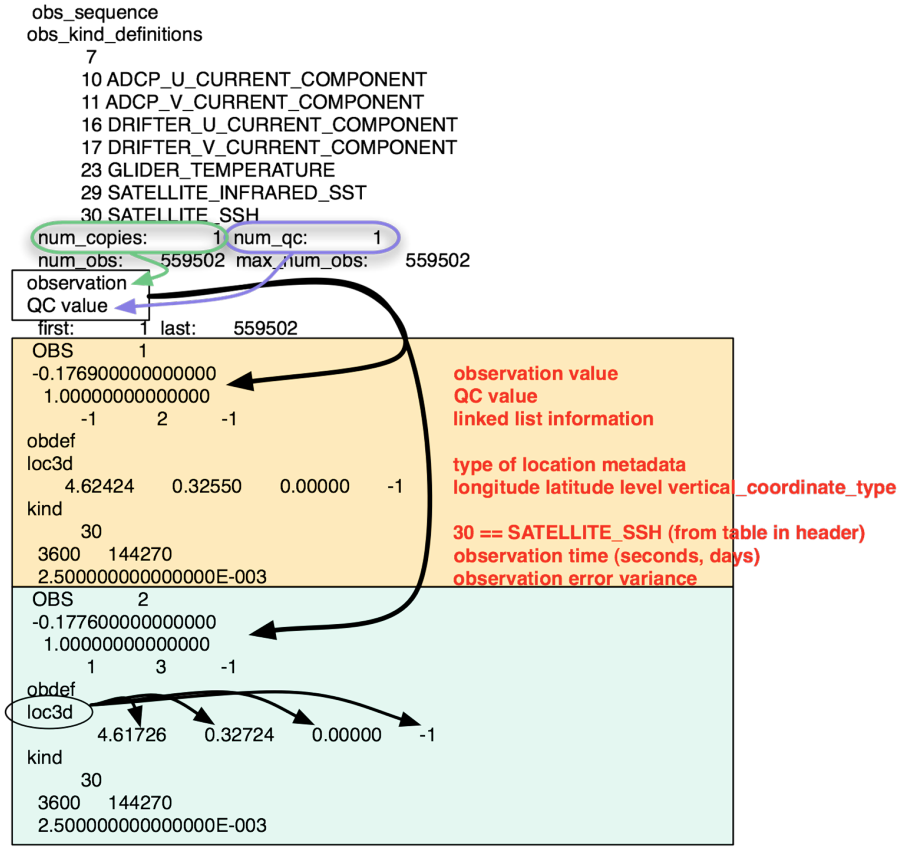
DART (“perfect_model_obs”)

interpolates model

onto obs space

⇒ we need observations in ‘observation sequence’ file format:

- CrocoLake:
  - script to build your own
    - can filter by any variable (e.g. platform number, region, time)



Workflow example #1

model output: one file with a 10-day run, with daily averages

CrocoLake obs_sequence files: 10 files with daily sparse observations

CrocoCamp:

- splits model output along 'time'
- matches snapshots with obs_seq.in files
- interpolates model snapshot to **ALL** obs in corresponding obs_seq.in file
- it does not re-use the same obs_seq.in file (assumes pairing is unique)

⇒ your model-obs "resolution" is your run's averages
⇒ obs_seq.in files must have that "resolution"

Workflow example #2

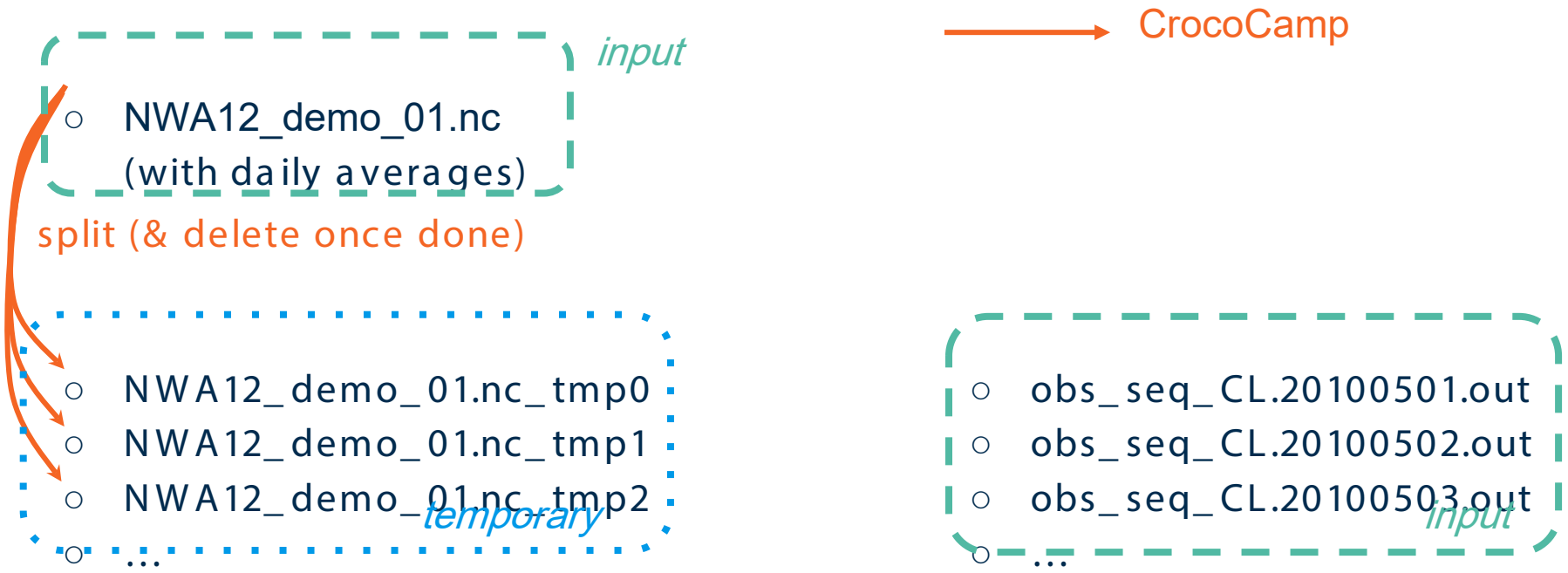
input

- NWA12_demo_01.nc
  (with daily averages)

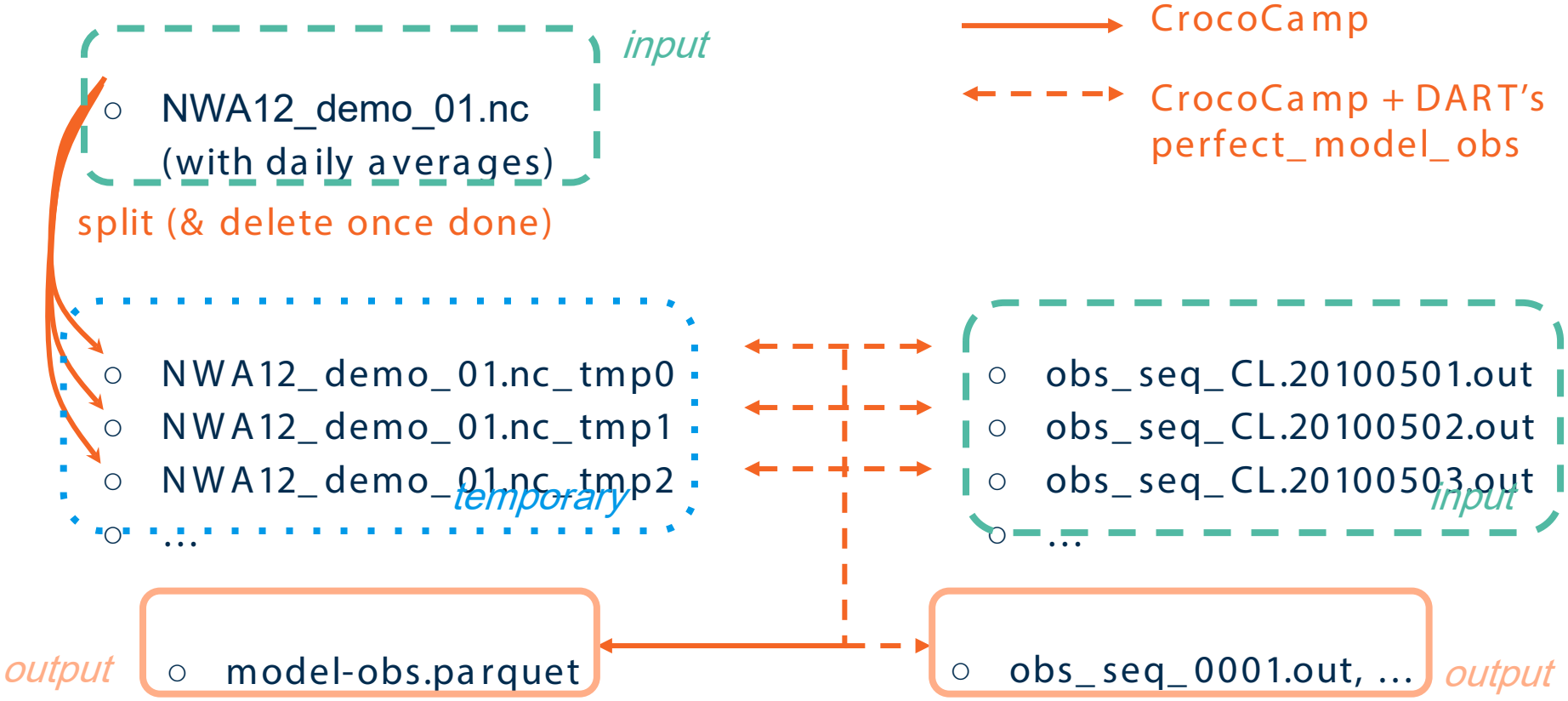
input

- obs_seq_CL.20100501.out
- obs_seq_CL.20100502.out
- obs_seq_CL.20100503.out
- ...

Workflow example #2

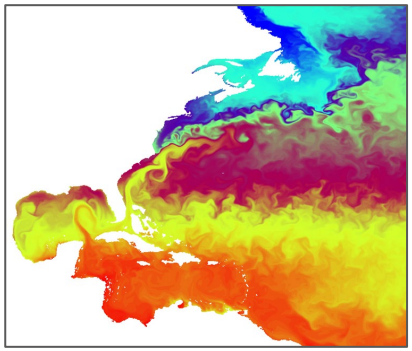


Workflow example #2



Comparing model output to observations (CrocoLake, WOD)

Recap of the ingredients:



1. MOM6/ ROMS model output

2. DART

3. CrocoCamp

4. Observations in

DART'S obs_sequence format:

a. CrocoLake: obs_converter in DART

b. Other: check DART / make your own

Final remarks + outlook

Future plans:

- customizable interpolation windows **without** changing files resolution
- tools for comparisons with **climatologies**
- improved efficiency