



argopy

(One)Argo data
for everyone



Guillaume Maze, Kevin Balem, Jade Dogo

Laboratory for Ocean Physics and Satellite remote sensing (LOPS)
IFREMER

*Mastering the argopy library for
Argo Ocean Data Analysis*

Presentation and Training Session

Heraklion, September 22nd 2025



14h00: Session 1

- General presentation of Argopy
- Demonstration notebook
- Introduction to session 2 themes

15h00: Coffee break and self grouping by Session 2 themes interests

15:30: Session 2

- Parallel sub-sessions around thematic set of notebooks

17h00: End of the training

The project, the team

Why argopy ?

Getting started
For those more familiar with Argo
(for experts and operators)



The argopy project, the team

argopy was created 5 years ago with the idea of making
Argo data easily accessible to everyone
(Maze & Balem, 2020: 10.21105/joss.02425)

argopy is an **open source** and **collaborative** software project

The codebase is licensed under the EUPL v1.2



 euroargodev/argopy is licensed under the
European Union Public License 1.2

The European Union Public Licence (EUPL) is a copyleft free/open source software license created on the initiative of and approved by the European Commission in 23 official languages of the European Union.

Permissions

- ✓ Commercial use
- ✓ Modification
- ✓ Distribution
- ✓ Patent use
- ✓ Private use

Limitations

- ✗ Liability
- ✗ Trademark use
- ✗ Warranty

Conditions

- ⓘ License and copyright notice
- ⓘ Disclose source
- ⓘ State changes
- ⓘ Network use is distribution
- ⓘ Same license



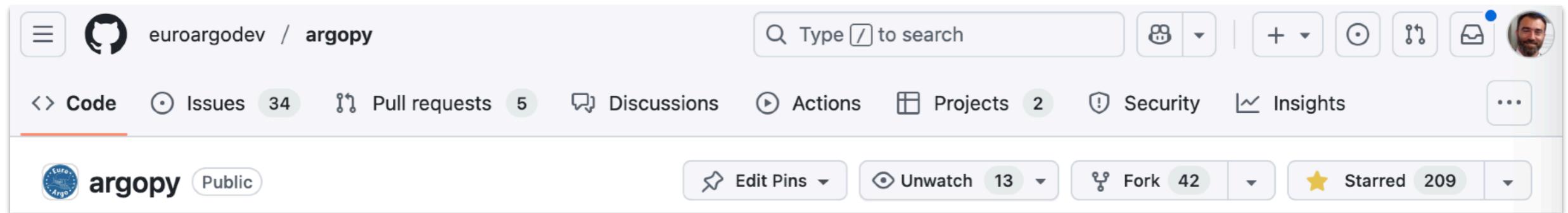
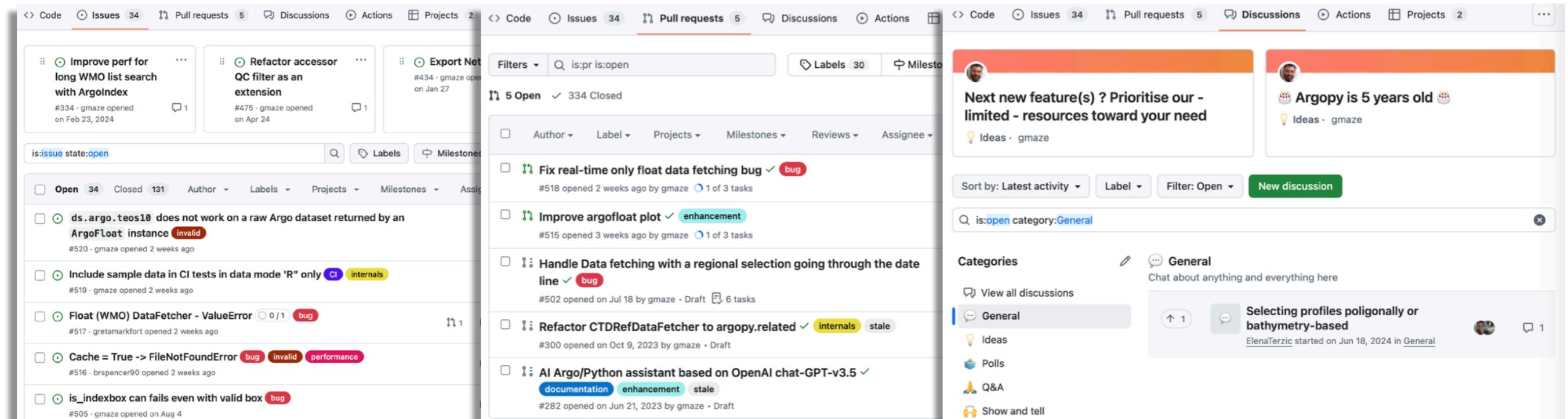
The argopy project, the team



argopy was created 5 years ago with the idea of making
Argo data easily accessible to everyone
(Maze & Balem, 2020: 10.21105/joss.02425)

argopy is an **open source** and **collaborative** software project

The project is hosted on <https://github.com/euroargodev/argopy>



The argopy project, the team

argopy was created 5 years ago with the idea of making
Argo data easily accessible to everyone
(Maze & Balem, 2020: 10.21105/joss.02425)

argopy is an **open source** and **collaborative** software project

The team

Developers and maintainers:

- Guillaume Maze (PI, 60%)
- Kevin Balem (20%)
- Jade Dogo (80%)
- Community enthusiasts



Support and consulting:

- Filipe Fernandes (distribution support)
 - Raphaëlle Sauzède and LOV (consulting BGC)
 - Catherine Schmechtig (consulting BGC)
 - Léo Le Lonquer, Thierry Carval (data server support)
-
- Ifremer, SNO Argo-France, CNRS - LEFE/GMMC
 - Euro-Argo RISE, Euro-Argo ONE

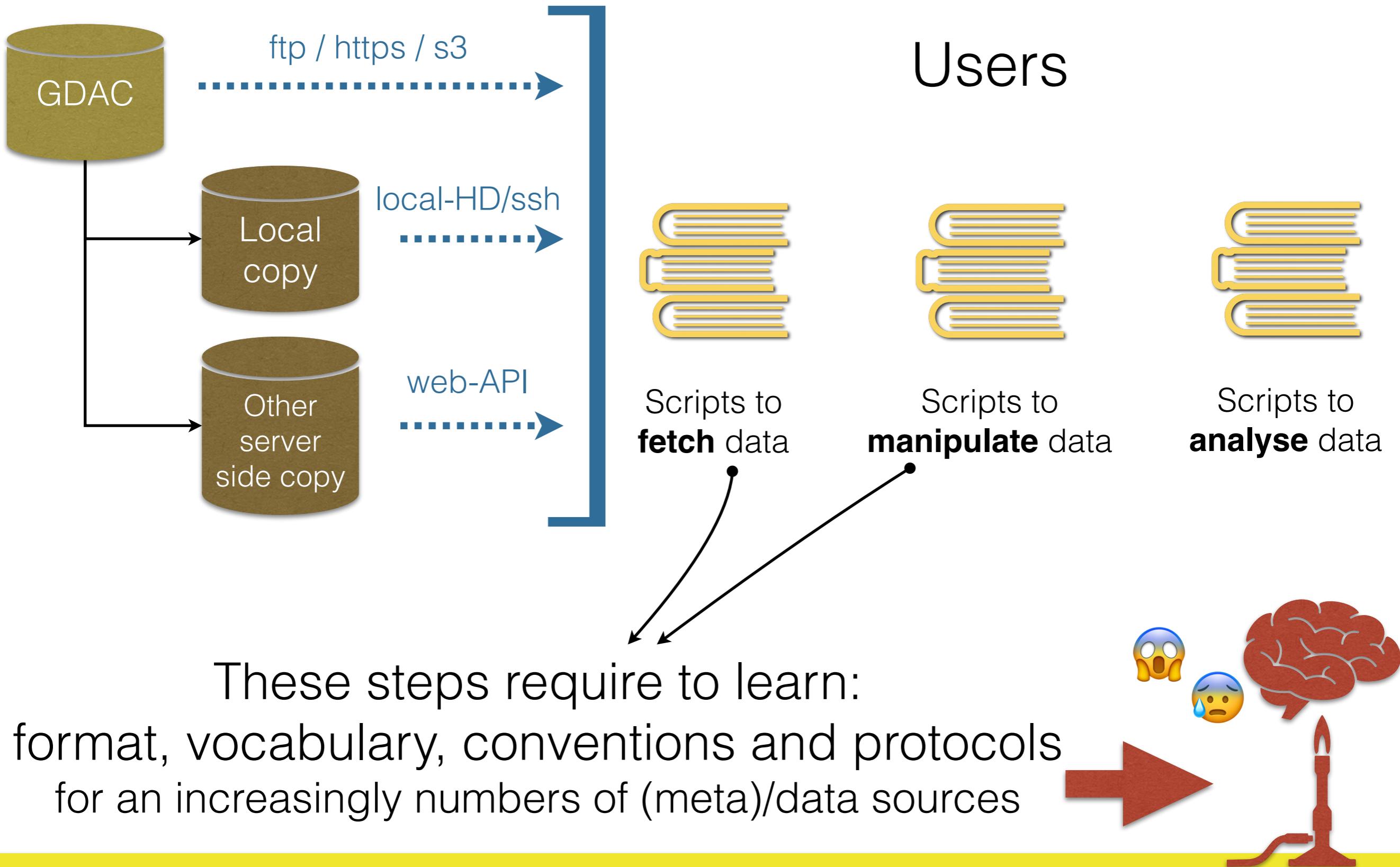
The project, the team **Why argopy ?**

Getting started
For those more familiar with Argo
(for experts and operators)

The problem to solve

Argo data sources and access protocols

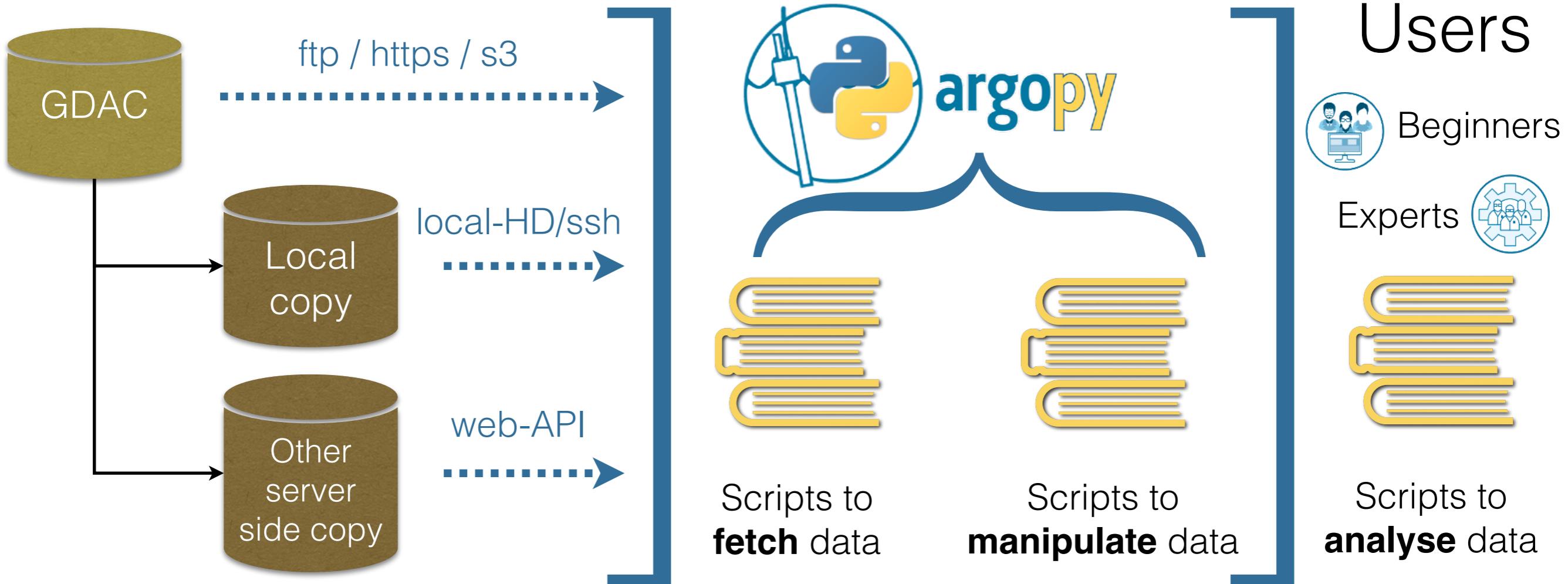
End-user standard workflow



Solution proposed by argopy

Argo data sources and access protocols

End-user standard workflow

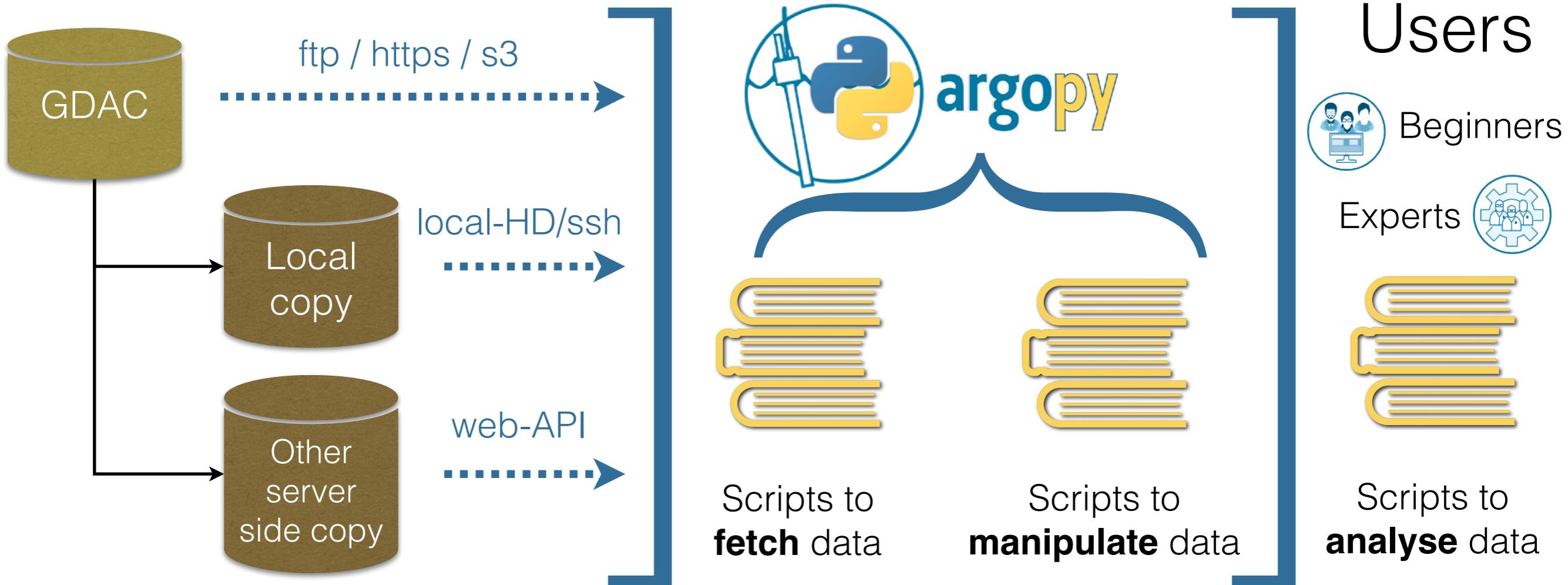


argopy let users decide how much of the Argo machinery to learn/see:
Less data wrangling, more analysis

Solution proposed by argopy

Argo data sources and access protocols

End-user standard workflow



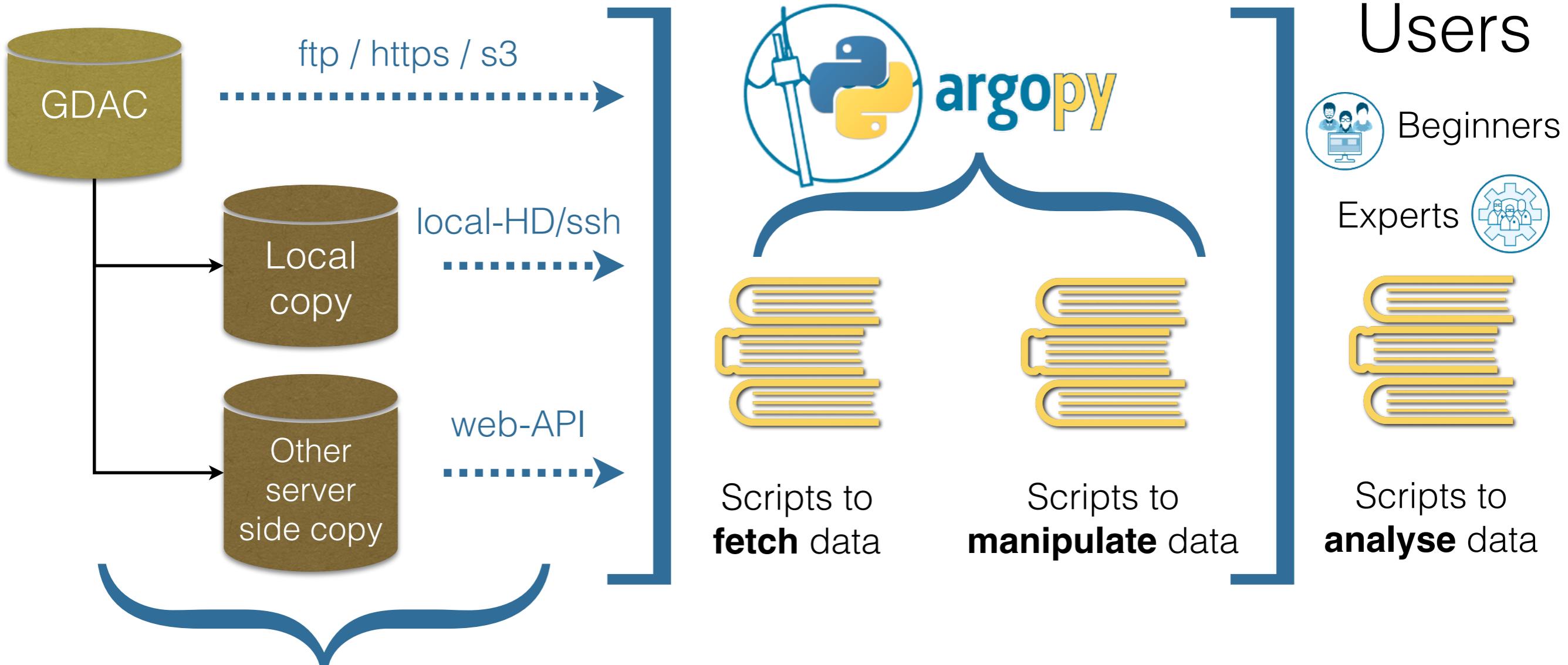
argopy let users decide how much of the Argo machinery to learn/see:
Less data wrangling, more analysis

argopy can simplify Argo dataset with state of the art Argo Data Management recommendations for You !

Solution proposed by argopy

Argo data sources and access protocols

End-user standard workflow



1. <ftp://usgodaе.org/pub/outgoing/argo> (data, index)
2. <ftp://ftp.ifremer.fr/ifremer/argo> (data, index)
3. <https://data-argo.ifremer.fr> (data, index)
4. <http://erddap.ifremer.fr/erddap> (data + Reference Data)
5. <https://argovis.colorado.edu> (data)
6. <https://fleetmonitoring.euro-argo.eu> (meta-data, dashboard)
7. <https://dataselection.euro-argo.eu> (trajectories)
8. <https://co-insitucharts.ifremer.fr> (dashboard)
9. <https://vocab.nerc.ac.uk/collection> (meta-data, vocabulary)
10. <https://coastwatch.pfeg.noaa.gov/erddap> (topography)
11. <https://maps.biogeochemical-argo.com/bgcargo> (dashboard)
12. <https://www.ocean-ops.org> (dashboard, deployment plan)
13. <https://archimer.ifremer.fr/> (ADMT documentation manuals)
14. <https://www.seanoe.org/api/> (GDAC snapshot DOIs)
15. <s3://argo-gdac-sandbox> (data, index) > experimental

argopy keeps track with ALL data sources and protocols for You !

argopy provides high-level methods to access Argo meta and related datasets
(as well as access protocols)

The project, the team Why argopy ?

Getting started
For those more familiar with Argo
(for experts and operators)



Fetch data

```
from argopy import DataFetcher

fetcher = DataFetcher().region([-75, -45, 20, 30, 0, 100, '2011-01', '2011-06'])
fetcher = DataFetcher().float([6902746, 6902755])
fetcher = DataFetcher().profile(6902746, [1,12])

ds = fetcher.to_xarray() # Xarray Dataset (default argopy data model)
ds = fetcher.to_dataframe() # Pandas Dataframe
ds = fetcher.to_dataset() # netCDF4 dataset for legacy

ds = fetcher.data # Xarray Dataset
df = fetcher.index # Pandas DataFrame
```

Select dataset

argopy provides access to **physical** or **BGC** parameters:

```
from argopy import DataFetcher

fetcher = DataFetcher(ds='phy')

# or

fetcher = DataFetcher(ds='bgc')
fetcher = DataFetcher(ds='bgc', params='DOXY')
fetcher = DataFetcher(ds='bgc', params='all', measured=['DOXY', 'BBP700'])
```



Select data source

argopy can get access to Argo data from the following sources:

- ★ the Ifremer **erddap** server (Default)
- 🌐 all Argo **gdac** servers or any other GDAC-compliant **local folder**
- 👁 the **argovis** server

```
from argopy import DataFetcher

fetcher = DataFetcher(src='erddap').region([-75, -45, 20, 30, 0, 100])
```

Select user mode

argopy provides 3 user modes:

- 🏄 **expert** mode: return all the Argo data, without post-processing
- 🏊 **standard** mode: simplifies the dataset, remove most of its jargon and return a priori good data
- ⛵ **research** mode: simplifies the dataset to its heart, preserving only data of the highest quality for research studies

```
from argopy import DataFetcher

fetcher = DataFetcher(mode='research').region([-75, -45, 20, 30, 0, 100])
```



How to use argopy ?

Manipulate data

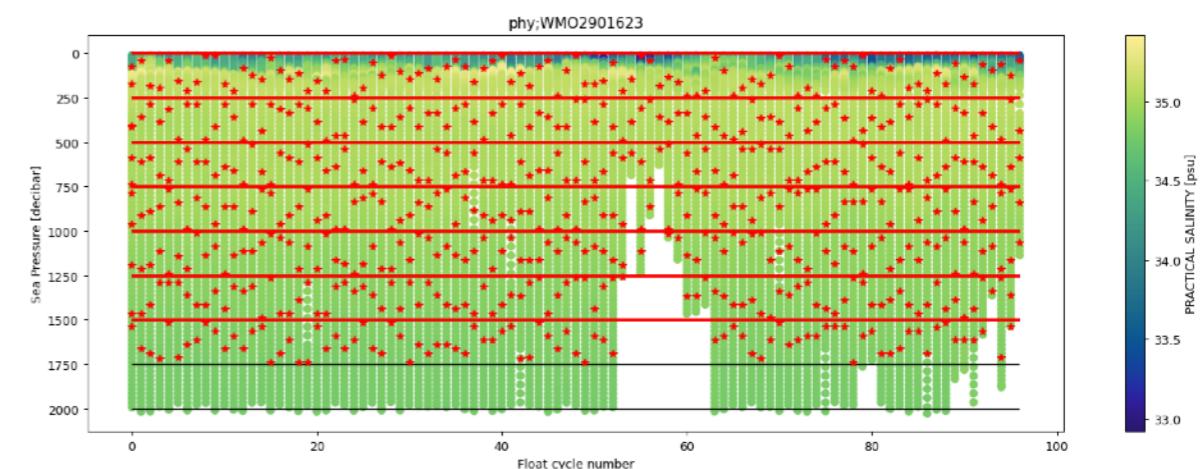
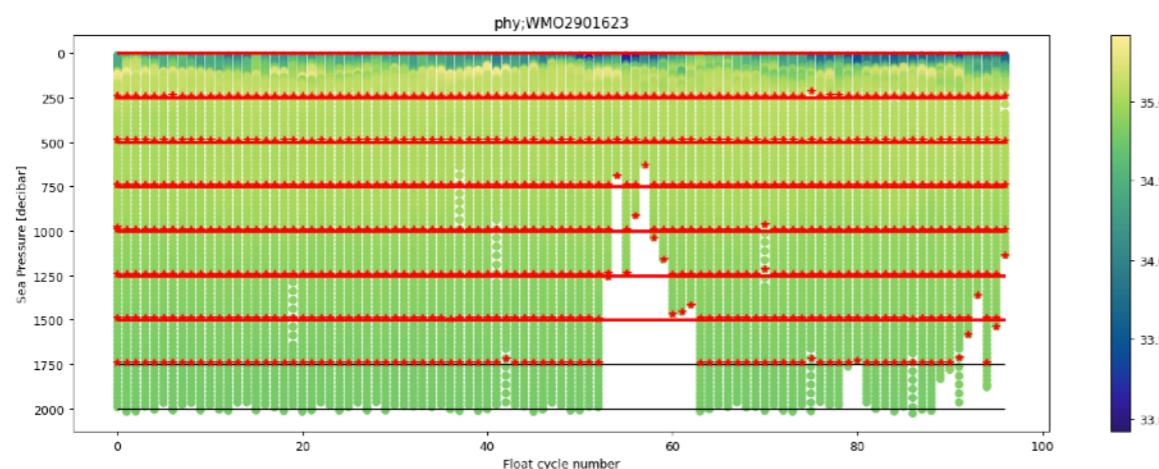
Work with a collection of points (default) vs a collection of profiles:

```
ds.argo.point2profile()
ds.argo.profile2point()
```

Vertical re-gridding:

```
std = [0,100,200,500] # in db
dsp.argo.interp_std_levels(std)

b = np.arange(0., 2000., 250.0) # in db
ds.argo.groupby_pressure_bins(bins=b, select='deep')
ds.argo.groupby_pressure_bins(bins=b, select='random')
```





Complement dataset

Compute standard variables:

```
ds.argo.teos10(['SA', 'CT', 'CNDC']) # 'SIG0', 'N2', 'PV', 'PTEMP', 'SOUND_SPEED'
```

Compute BGC optical modeling variables:

```
dsp.argo.optic.Zeu() # Depth of the euphotic zone
dsp.argo.optic.Zeu(method='percentage', max_surface=5.)
dsp.argo.optic.Zeu(method='KdPAR', layer_min=10., layer_max=50.)

dsp.argo.optic.Zpd() # First optical depth

dsp.argo.optic.Z_iPAR_threshold(threshold=15.) # Depth where PAR>=threshold

dsp.argo.optic.DCM() # Search and qualify a Deep Chlorophyll Maxima
dsp.argo.optic.DCM_depth() # Search and qualify a Deep Chlorophyll Maxima
```

Compute BGC nutrient and carbonate system variables:

CANYON-MED available:

```
dsp.argo.canyon_med.predict()
dsp.argo.canyon_med.predict('PO4') # "NO3", "DIC", "SiOH4", "AT", "pHT"
```

CANYON-B, ESPER and CONTENT coming up by the end of 2025



Complement dataset

Compute your own “per profile” diagnostic:

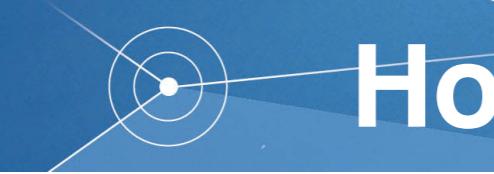
```
ds.argo.reduce_profile(your_diag, params=['PRES', 'TEMP', 'PSAL'])
```

argopy takes care of efficient, parallel, diagnostic on
your collection of Argo profiles

Example:

```
def max_salinity_depth(pres, psal, max_layer=1000.):
    # A dummy function returning depth of the maximum salinity above max_layer
    idx = ~np.logical_or(np.isnan(pres), np.isnan(psal))
    idx = np.logical_and(idx, pres<=max_layer)
    if np.any(idx):
        return pres[idx][np.argmax(psal[idx])]
    else:
        return np.NaN

# Apply reduce function on all profiles of the dataset:
ds.argo.reduce_profile(max_salinity_depth, params=['PRES', 'PSAL'], max_layer=700)
```



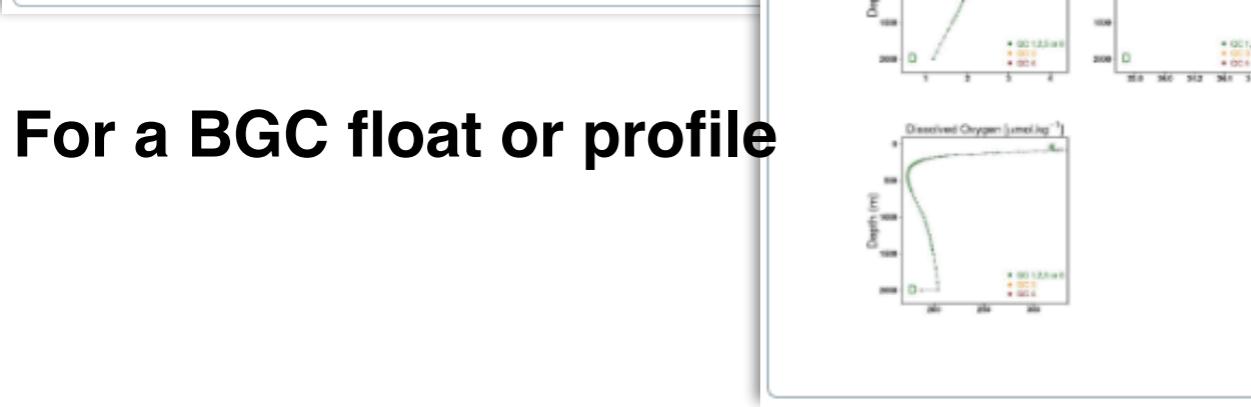
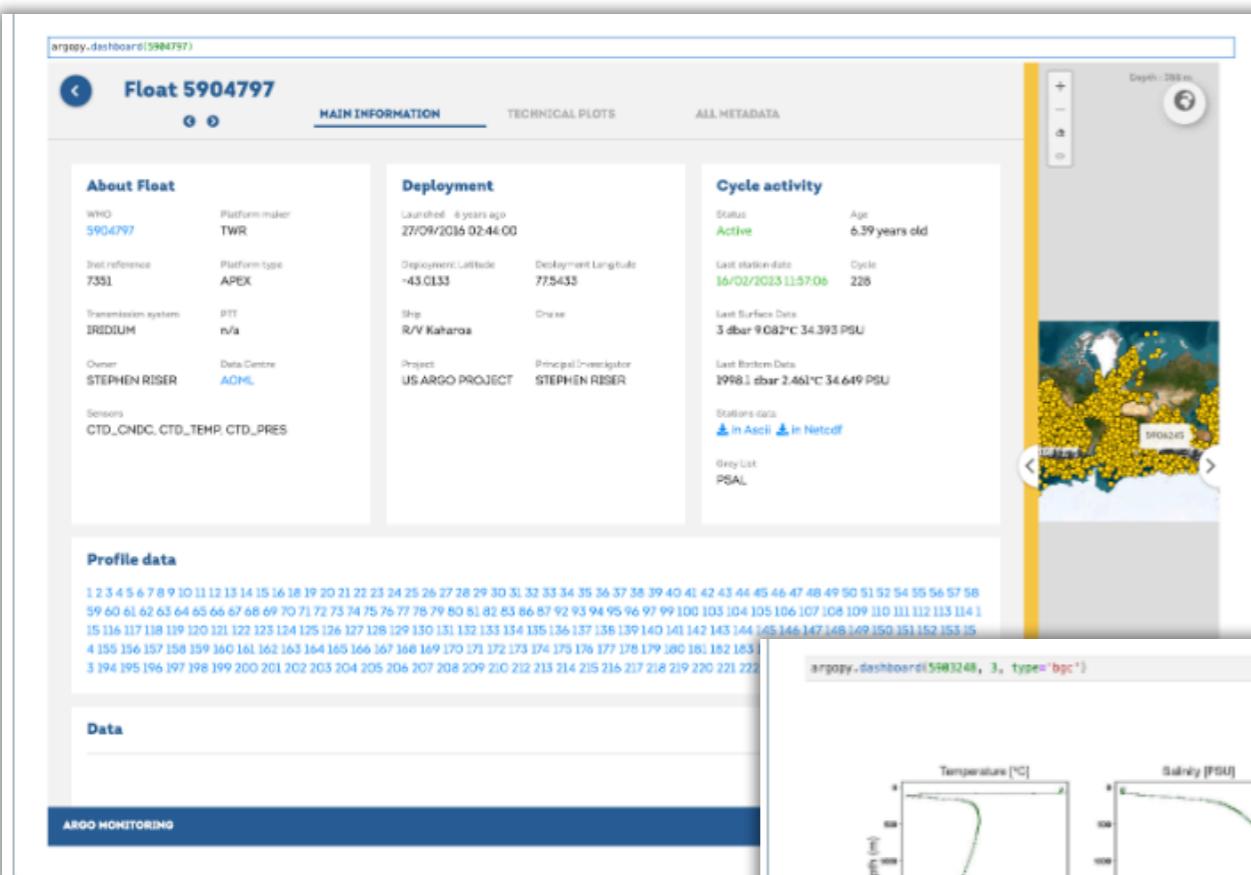
How to use argopy ?

Visualise data

Get a dashboard link or inserted into your notebook, for your selection:

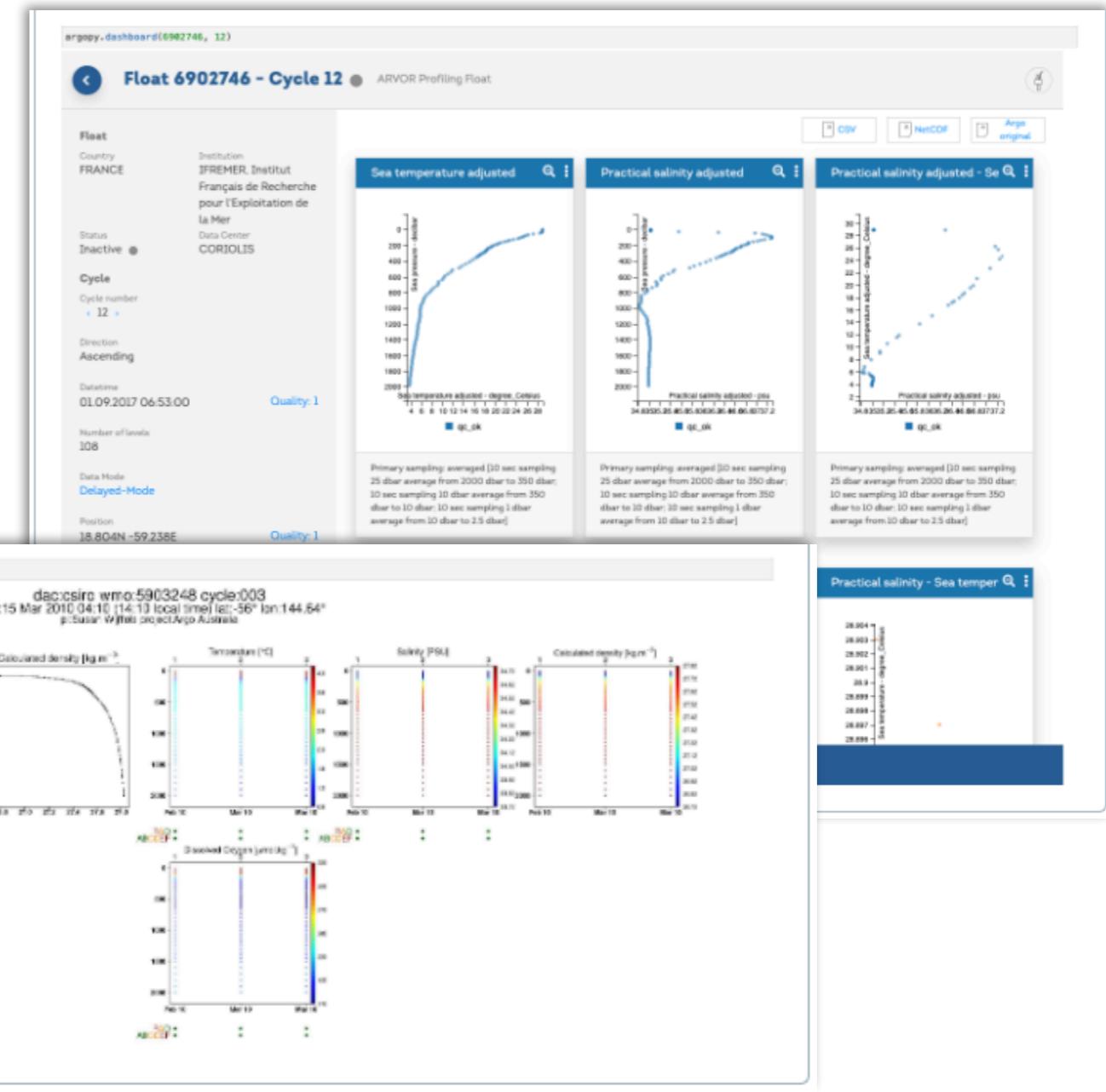
```
DataFetcher().float(6903091).dashboard()
DataFetcher().profile(6903091,12).dashboard()
```

For a float



For a BGC float or profile

For a profile



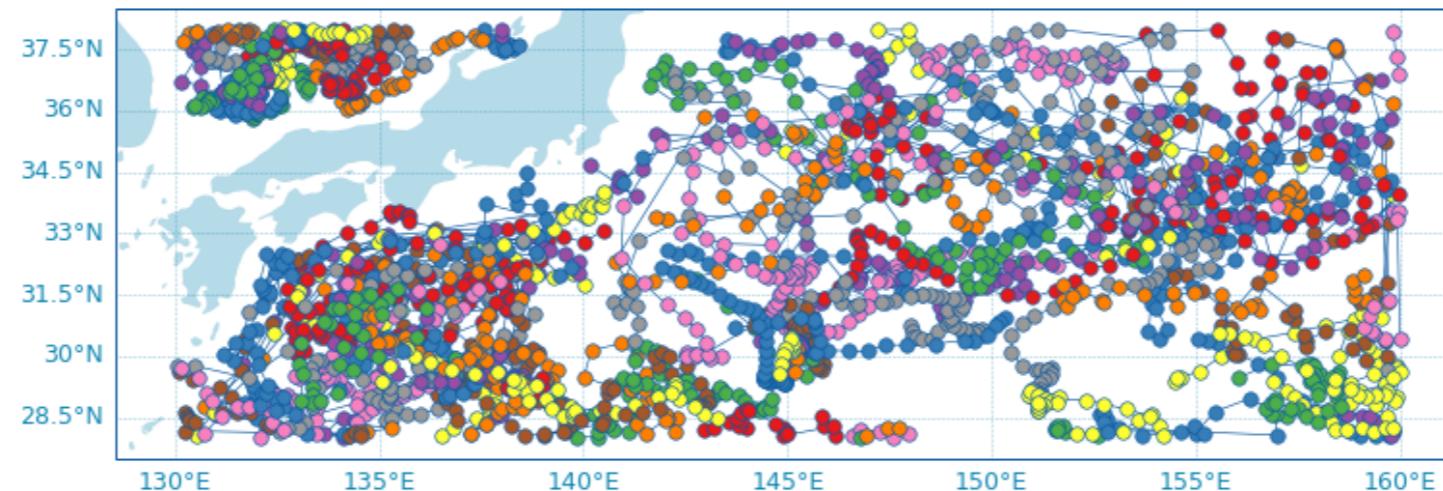


How to use argopy ?

Visualise data

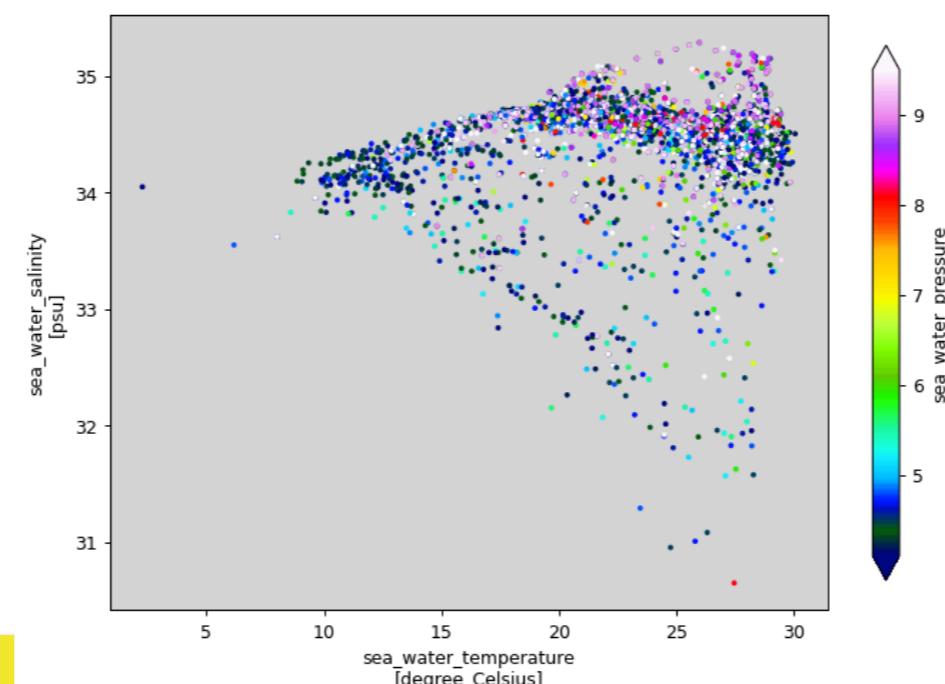
Look at trajectories:

```
fetcher = DataFetcher(src='argovis').region([130, 160, 28, 38, 0, 10, '2010','2011'])  
fetcher.plot('trajectory')
```



Or variables in scatter plots:

```
fetcher.plot('PRES', cbar=True, this_x='TEMP', this_y='PSAL', figsize=(9,6));
```



The project, the team Why argopy ?

Getting started

For those more familiar with Argo
For experts and operators



Direct access to float datasets

```
from argopy import ArgoFloat

af = ArgoFloat(6902746) # Can point to local or remote data source using `host` argument

# or
af = ArgoFloat(WMO, host='/home/ref-argo/gdac') # Use your local GDAC copy
af = ArgoFloat(WMO, host='https') # Shortcut for https://data-argo.ifremer.fr
af = ArgoFloat(WMO, host='ftp') # shortcut for ftp://ftp.ifremer.fr/ifremer/argo
af = ArgoFloat(WMO, host='s3') # Shortcut for s3://argo-gdac-sandbox/pub
```

```
>>> af

<argofloat.6902746.http.online>
GDAC host: https://data-argo.ifremer.fr
DAC name: coriolis
Network(s): ['CORE']
Deployment date: 2017-07-06 11:15 [2891 days ago]
Float type and manufacturer: ARVOR [NKE]
Number of cycles: 138
Dashboard: https://fleetmonitoring.euro-argo.eu/float/6902746
Netcdf dataset available: ['Rtraj', 'meta', 'prof', 'tech']
```



Direct access to float datasets

```
from argopy import ArgoFloat

af = ArgoFloat(6902746) # Can point to local or remote data source using `host` argument

# or
af = ArgoFloat(WMO, host='/home/ref-argo/gdac') # Use your local GDAC copy
af = ArgoFloat(WMO, host='https') # Shortcut for https://data-argo.ifremer.fr
af = ArgoFloat(WMO, host='ftp') # shortcut for ftp://ftp.ifremer.fr/ifremer/argo
af = ArgoFloat(WMO, host='s3') # Shortcut for s3://argo-gdac-sandbox/pub
```

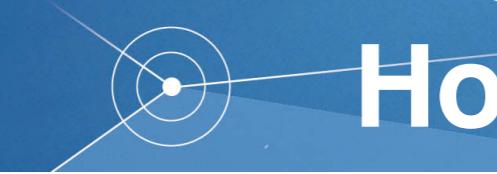
List and open netcdf files for this float:

```
>>> af.ls_dataset()

{'Rtraj': 'https://data-argo.ifremer.fr/dac/coriolis/6903091/6903091_Rtraj.nc',
 'Sprof': 'https://data-argo.ifremer.fr/dac/coriolis/6903091/6903091_Sprof.nc',
 'meta': 'https://data-argo.ifremer.fr/dac/coriolis/6903091/6903091_meta.nc',
 'prof': 'https://data-argo.ifremer.fr/dac/coriolis/6903091/6903091_prof.nc',
 'tech': 'https://data-argo.ifremer.fr/dac/coriolis/6903091/6903091_tech.nc'}
```

```
af.open_dataset('prof') # Return a Xarray dataset (argopy data model)
af.open_dataset('prof', netCDF4=True) # Return a legacy netCDF4 dataset
af.open_dataset('prof', lazy=True) # If supported by server
```

and more...



How to use argopy ?

Work with index files

```
idx = ArgoIndex() # Point to local or remote GDAC source using `host`
idx = ArgoIndex(index_file='bgc-s') # Select which file index to use

idx.load() # Load full index (2-3 seconds for core profiles index)
```

```
>>> idx

<argoindex.pyarrow>
Host: https://data-argo.ifremer.fr
Index: argo_synthetic-profile_index.txt.gz
Convention: argo_synthetic-profile_index (Synthetic-Profile directory file of the Argo GDAC)
In memory: True (348531 records)
Searched: False
```

```
>>> idx.to_dataframe() # Fully documented list of files (WMO, CYC, DAC, etc...)
```

	file	date	latitude	longitude	ocean	profiler_code	institution_code	parameters	parameter_data_mode	date_update	wmo	cyc	institution	dac	profiler
0	aoml/1900722/profiles/SD1900722_001.nc	2006-10-22 02:16:24	-40.316	73.389	I	846	AO	PRES TEMP PSAL DOXY	DDDD	2022-06-28 08:08:01	1900722	1	AOML, USA	aoml	Teledyne Webb Research float with SBE conducti...
1	aoml/1900722/profiles/SD1900722_002.nc	2006-11-01 06:44:23	-40.390	73.528	I	846	AO	PRES TEMP PSAL DOXY	DDDD	2022-06-28 08:08:13	1900722	2	AOML, USA	aoml	Teledyne Webb Research float with SBE conducti...
2	aoml/1900722/profiles/SD1900722_003.nc	2006-11-11 10:12:22	-40.455	73.335	I	846	AO	PRES TEMP PSAL DOXY	DDDD	2022-06-28 08:08:26	1900722	3	AOML, USA	aoml	Teledyne Webb Research float with SBE conducti...
3	aoml/1900722/profiles/SD1900722_004.nc	2006-11-21 07:50:21	-40.134	73.080	I	846	AO	PRES TEMP PSAL DOXY	DDDD	2022-06-28 08:08:39	1900722	4	AOML, USA	aoml	Teledyne Webb Research float with SBE conducti...
4	aoml/1900722/profiles/SD1900722_005.nc	2006-12-01 18:33:00	-39.641	73.158	I	846	AO	PRES TEMP PSAL DOXY	DDDD	2022-06-28 08:08:52	1900722	5	AOML, USA	aoml	Teledyne Webb Research float with SBE conducti...
...
348526	meds/4902688/profiles/SR4902688_085.nc	2025-05-10 15:31:00	57.664	-52.164	A	834	ME	PRES TEMP PSAL DOXY DOWN_IRRADIANCE380 DOWN_IR...	RRRRRRRRAR	2025-05-22 19:08:43	4902688	85	MEDS, Canada	meds	PROVOR V SBE
348527	meds/4902688/profiles/SR4902688_086.nc	2025-05-16 15:31:00	57.459	-53.125	A	834	ME	PRES TEMP PSAL DOXY DOWN_IRRADIANCE380 DOWN_IR...	RRRRRRRRAR	2025-05-22 19:08:53	4902688	86	MEDS, Canada	meds	PROVOR V SBE
348528	meds/4902688/profiles/SR4902688_087.nc	2025-05-22 15:29:00	57.289	-54.123	A	834	ME	PRES TEMP PSAL DOXY DOWN_IRRADIANCE380 DOWN_IR...	RRRRRRRRAR	2025-06-02 17:41:06	4902688	87	MEDS, Canada	meds	PROVOR V SBE
348529	meds/4902688/profiles/SR4902688_088.nc	2025-05-28 15:35:00	57.294	-54.127	A	834	ME	PRES TEMP PSAL DOXY DOWN_IRRADIANCE380 DOWN_IR...	RRRRRRRRAR	2025-06-02 17:41:16	4902688	88	MEDS, Canada	meds	PROVOR V SBE
348530	meds/4902688/profiles/SR4902688_089.nc	2025-06-03 15:33:00	57.440	-54.106	A	834	ME	PRES TEMP PSAL DOXY DOWN_IRRADIANCE380 DOWN_IR...	RRRRRRRRAR	2025-06-04 00:07:11	4902688	89	MEDS, Canada	meds	PROVOR V SBE



Work with index files

Search the index

```

idx.query.wmo(1901393)
idx.query.cyc(1)
idx.query.wmo_cyc(1901393, [1,12])

idx.query.lon([-60, -55, 40., 45., '2007-08-01', '2007-09-01'])
idx.query.lat([-60, -55, 40., 45., '2007-08-01', '2007-09-01'])
idx.query.date([-60, -55, 40., 45., '2007-08-01', '2007-09-01'])
idx.query.lon_lat([-60, -55, 40., 45., '2007-08-01', '2007-09-01'])
idx.query.box([-60, -55, 40., 45., '2007-08-01', '2007-09-01'])

idx.query.params(['C1PHASE_DOXY', 'DOWNWELLING_PAR']) # Only for BGC profile index
idx.query.parameter_data_mode({'BBP700': 'D'}) # Only for BGC profile index
idx.query.profiler_type(845)
idx.query.profiler_label('NINJA')

idx.query.compose({'box': BOX, 'wmo': WMOS})
idx.query.compose({'box': BOX, 'params': 'DOXY'})
idx.query.compose({'box': BOX, 'params': ('[DOXY', 'DOXY2'], {'logical': 'and'})})
idx.query.compose({'params': 'DOXY', 'profiler_label': 'ARVOR'})

```

```

idx.to_indexfile('myindex.csv')
idx.N_RECORDS
idx.N_MATCH

idx.read_wmo(); idx.read_dac_wmo(); idx.records_per_wmo()
idx.read_params() ; idx.read_domain()
idx.read_files()

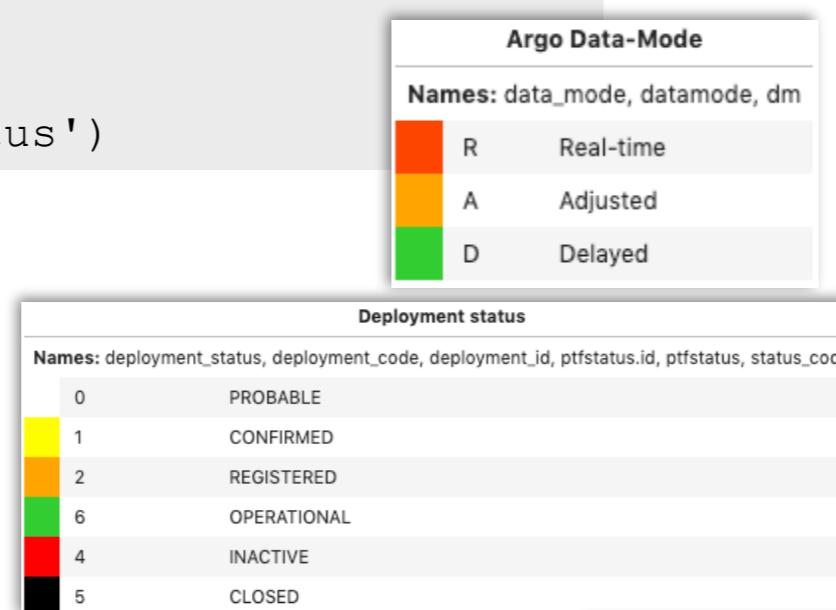
```

How to use argopy ?

Visualise expert data

Use standard colormaps:

```
from argopy.plot import ArgoColors
ArgoColors('data_mode')
ArgoColors('qc_flag')
ArgoColors('deployment_status')
```



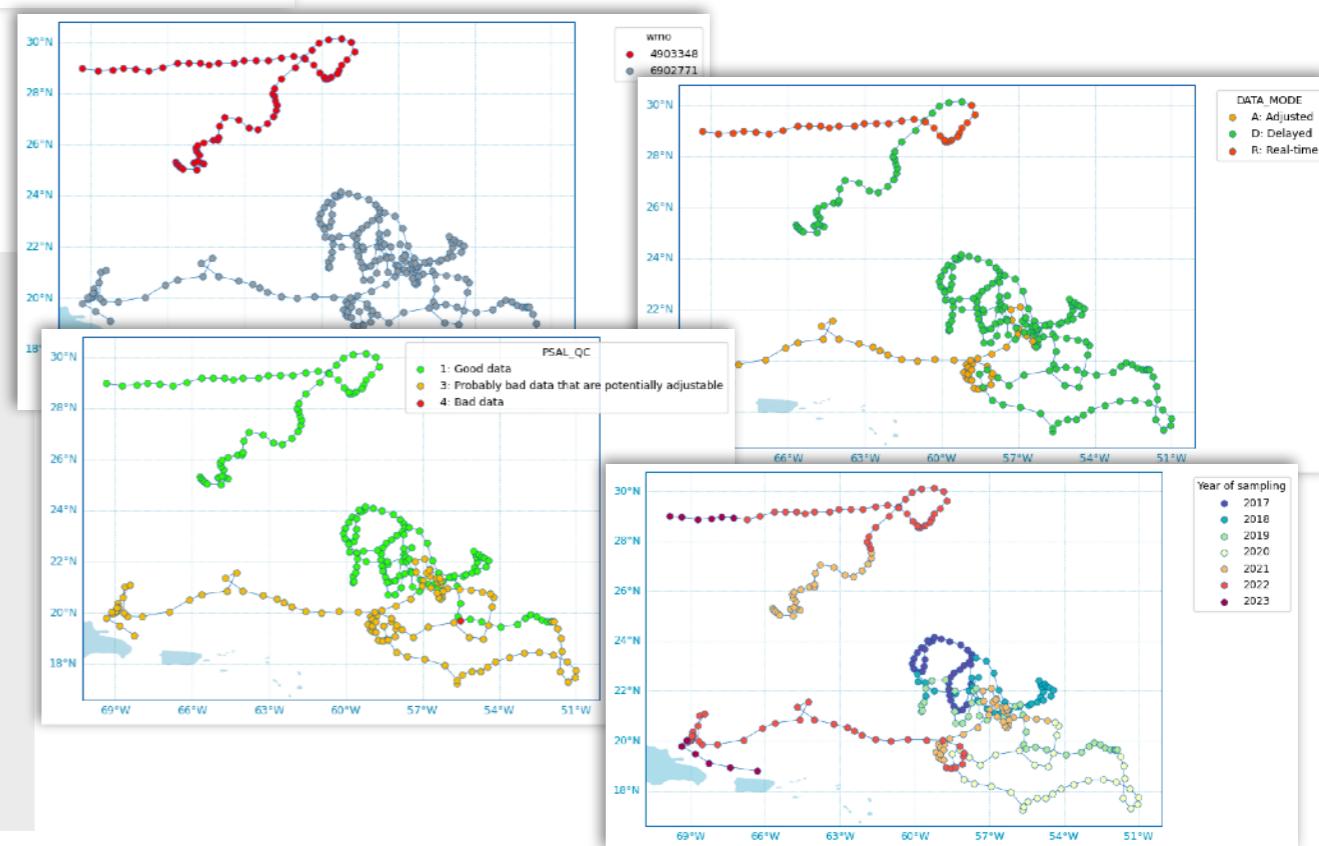
Quality control flag scale	
Names: qc, qc_flag, quality_control, quality_control_flag, quality_control_flag_scale	
0	No QC performed
1	Good data
2	Probably good data
3	Probably bad data that are potentially adjustable
4	Bad data
5	Value changed
6	Not used
7	Not used
8	Estimated value
9	Missing value

Plot your own scatter maps:

```
from argopy.plot import scatter_map

scatter_map(ds)
scatter_map(ds, hue='DATA_MODE')
scatter_map(ds.isel(N_LEVELS=0), hue='PSAL_QC')

ds['year'] = ds['TIME.year'] # Add a variable
scatter_map(ds.isel(N_LEVELS=0),
            hue='year',
            cmap='Spectral_r',
            legend_title='Year of sampling')
```



argopy in a nutshell

DataFetcher to load data

Select phy/bgc, data source and user mode with `ds`, `src` and `mode` options

Select profiles with `.region`, `.float` and `.profile` methods

`xarray.Dataset.argo` accessor to manipulate data

ArgoFloat to work with one float netcdf files

ArgoIndex to work with index of files

Got to:

<https://github.com/euroargodev/argopy-training>

[/content/hands-on](#)

[argopy-getting-started.ipynb](#)



<https://github.com/euroargodev/argopy-training>

Session 2

Theme	Notebooks
 Argo data fetching	Select and fetch data Source and user mode BGC data specifics Direct access to one float dataset How to handle large data selection
 Argo data manipulation	Filtering (QC flags, data mode) Vertical interpolation & binning Compute (TEOS, Optic, CANYON-MED) Compute your own per-profile diagnostic
 Argo index and meta-data	Working with Argo index files Argo Reference tables lookup

For experts and operators

Data filtering (QC flag, data mode)
GDAC file store
Argo reference tables (NVS)
Argo DOIs (GDAC snapshot)
ADMT documentation
Deployment plan (OceanOPS)



Load some data:

```
from argopy import DataFetcher  
ds = DataFetcher(mode='expert').region([-75, -45, 20, 30, 0, 100]).to_xarray()
```

Filter points according to QC flags:

```
ds.argo.filter_qc(QC_list=[1, 2], QC_fields='all', drop=True, mode='all')  
ds.argo.filter_qc(QC_list=[1, 2], QC_fields=['PSAL_QC'], drop=True)
```

Filter points according to data mode:

```
ds.argo.datemode.filter(dm='D', params='all')
```

Merge PARAM and PARAM_ADJUSTED according to data mode:

```
ds.argo.datemode.merge(params='all')
```

Merging is done as follows:

- For measurements with data mode R: keep <PARAM>
- For measurements with data mode D or A: keep <PARAM>_ADJUSTED



Easy access to GDAC files with **gdacfs**

- **argopy** provides a convenient store for all GDAC files, a generic file system, for all possible GDAC hosts, which makes all path relative to the GDAC root level
- the **gdacfs** class will help you separate the logic in your code between data loading and access with specific host protocol handling

```
In [1]: from argopy import gdacfs  
  
In [2]: fs = gdacfs("https://data-argo.ifremer.fr")  
  
# or:  
# fs = gdacfs("https://usgodee.org/pub/outgoing/argo")  
# fs = gdacfs("ftp://ftp.ifremer.fr/ifremer/argo")  
# fs = gdacfs("s3://argo-gdac-sandbox/pub")  
# fs = gdacfs("/home/ref-argo/gdac")  
In [3]: fs  
Out[3]: <argopy.stores.implementations.http.httpstore at 0x7f8d8d6f8820>
```

- You can also use shortcut for all remote GDACs:

Full host	Shortcut
<code>https://data-argo.ifremer.fr</code>	<code>http</code> or <code>https</code>
<code>https://usgodee.org/pub/outgoing/argo</code>	<code>us-http</code> or <code>us-https</code>
<code>ftp://ftp.ifremer.fr/ifremer/argo</code>	<code>ftp</code>
<code>s3://argo-gdac-sandbox/pub/idx</code>	<code>s3</code> or <code>aws</code>



Easy access to GDAC files with **gdacfs**

- A **gdacfs** instance will provide most of the required methods to work with any file on a GDAC, **without the burden of handling access protocols and paths construction**
- paths are relative to the GDAC root folder (which is natively the case in Argo files index)

```
In [5]: fs.glob("dac/aoml/13857/*_meta.nc")
Out[5]: ['dac/aoml/13857/13857_meta.nc']

In [6]: fs.info("dac/aoml/13857/13857_meta.nc")
Out[6]:
{'name': 'dac/aoml/13857/13857_meta.nc',
 'size': 25352,
 'mimetype': 'application/x-netcdf',
 'partial': False,
 'url': 'https://data-argo.ifremer.fr/dac/aoml/13857/13857_meta.nc',
 'ETag': '"41c2167c-6308-577f9e9106041"',
 'type': 'file'}

In [7]: ds = fs.open_dataset("dac/coriolis/6903091/profiles/R6903091_001.nc")

In [8]: with fs.open("ar_index_this_week_meta.txt", "r") as f:
....:     data = f.readlines()
....:
```



Accessing the **NVS** - NERC Vocabulary Server - for Argo

```
from argopy import ArgoNVSReferenceTables
NVS = ArgoNVSReferenceTables()
```

If you don't know the ID of a reference table,
you can search in all tables name/definition:

```
NVS.search('quality')

[ ('R11',
  ('RTQC_TESTID',
   'List of real-time quality-control tests and corresponding binary identifiers, used as
reference to populate the Argo netCDF HISTORY_QCTEST variable.',
   'http://vocab.nerc.ac.uk/collection/R11/current/')),

  ('RD2',
  ('DM_QC_FLAG',
   "Quality flag scale for delayed-mode measurements. Argo netCDF variables
<PARAMETER>_ADJUSTED_QC in 'D' mode are populated by RD2 altLabel.",
   'http://vocab.nerc.ac.uk/collection/RD2/current/')),

  ('RP2',
  ('PROF_QC_FLAG',
   'Quality control flag scale for whole profiles. Argo netCDF variables PROFILE_<PARAMETER>_QC
are populated by RP2 altLabel.',
   'http://vocab.nerc.ac.uk/collection/RP2/current/')),

  ('RR2',
  ('RT_QC_FLAG',
   "Quality flag scale for real-time measurements. Argo netCDF variables <PARAMETER>_QC in 'R'
mode and <PARAMETER>_ADJUSTED_QC in 'A' mode are populated by RR2 altLabel.",
   'http://vocab.nerc.ac.uk/collection/RR2/current/'))]
```



Accessing the **NVS** - NERC Vocabulary Server - for Argo

Or/and directly load the reference table name or content:

```
NVS.tbl_name('R11')
```

```
('RTQC_TESTID',
 'List of real-time quality-control tests and corresponding binary identifiers, used as
 reference to populate the Argo netCDF HISTORY_QCTEST variable.',
 'http://vocab.nerc.ac.uk/collection/R11/current/')
```

```
NVS.tbl('R11') # Return a dataframe:
```

altLabel		prefLabel	definition	deprecated		id
0	Test 13	Stuck Value test	Stuck Value test. Test binary ID = 8192	false	http://vocab.nerc.ac.uk/collection/R11/current...	
1	Test 23	Interim rtqc flag scheme for data deeper than ...	Interim rtqc flag scheme for data deeper than ...	false	http://vocab.nerc.ac.uk/collection/R11/current...	
2	Test 25	MEDD test	MEDian with a Distance (MEDD) test. Test binar...	false	http://vocab.nerc.ac.uk/collection/R11/current...	
3	Test 9	Spike test	Spike test. Test binary ID = 512	false	http://vocab.nerc.ac.uk/collection/R11/current/9/	
4	Test 11	Gradient test	Gradient test. Test binary ID = 2048	true	http://vocab.nerc.ac.uk/collection/R11/current...	
5	Test 7	Regional Global Parameter test	Regional Global Parameter test. Test binary ID...	false	http://vocab.nerc.ac.uk/collection/R11/current/7/	
6	Test 2	Impossible Date test	Impossible Date test. Test binary ID = 4	false	http://vocab.nerc.ac.uk/collection/R11/current/2/	
7	Test 18	Frozen profile test	Frozen profile test. Test binary ID = 261144	false	http://vocab.nerc.ac.uk/collection/R11/current...	
8	Test 16	Gross Salinity or Temperature Sensor Drift test	Gross Salinity or Temperature Sensor Drift tes...	false	http://vocab.nerc.ac.uk/collection/R11/current...	
9	Test 17	Visual QC test	Visual QC test. Test binary ID = 131072	false	http://vocab.nerc.ac.uk/collection/R11/current...	
10	Test 26	TEMP_CNDC test applied to RBRargo3 2K	TEMP_CNDC test applied to RBRargo3 2K, test ad...	false	http://vocab.nerc.ac.uk/collection/R11/current...	
11	Test 3	Impossible Location test	Impossible Location test. Test binary ID = 8	false	http://vocab.nerc.ac.uk/collection/R11/current/3/	
12	Test 14	Density Inversion test	Density Inversion test. Test binary ID = 16384	false	http://vocab.nerc.ac.uk/collection/R11/current...	

ID and reference the most appropriate dataset with **ArgoDOI**

Promote analysis reproducibility ...

```
from argopy import ArgoDOI
ArgoDOI()
```

```
<argopy.DOIrecord>
DOI: 10.17882/42182
Title: Argo float data and metadata from Global Data Assembly Centre (Argo GDAC)
Date: 2023-10-09
Network: core+BGC+deep
File: 138 files in total
Files for core+BGC+deep:
- #105302 Global GDAC Argo data files (2023-10-09 snapshot) https://www.seanoe.org/data/00311/42182/data/105302.tar.gz (56.6GiB, openAccess=True)
- #104707 Global GDAC Argo data files (2023-09-09 snapshot) https://www.seanoe.org/data/00311/42182/data/104707.tar.gz (55.9GiB, openAccess=True)
- #104145 Global GDAC Argo data files (2023-08-09 snapshot) https://www.seanoe.org/data/00311/42182/data/104145.tar.gz (54.8GiB, openAccess=True)
- #103614 Global GDAC Argo data files (2023-07-09 snapshot) https://www.seanoe.org/data/00311/42182/data/103614.tar.gz (54.1GiB, openAccess=True)
- #103075 Global GDAC Argo data files (2023-06-09 snapshot) https://www.seanoe.org/data/00311/42182/data/103075.tar.gz (53.8GiB, openAccess=True)
- #102270 Global GDAC Argo data files (2023-05-09 snapshot) https://www.seanoe.org/data/00311/42182/data/102270.tar.gz (53.2GiB, openAccess=True)
- #101760 Global GDAC Argo data files (2023-04-10 snapshot) https://www.seanoe.org/data/00311/42182/data/101760.tar.gz (52.7GiB, openAccess=True)
- #100487 Global GDAC Argo data files (2023-03-10 snapshot) https://www.seanoe.org/data/00311/42182/data/100487.tar.gz (52.1GiB, openAccess=True)
- #99784 Global GDAC Argo data files (2023-02-10 snapshot) https://www.seanoe.org/data/00311/42182/data/99784.tar.gz (51.5GiB, openAccess=True)
- #98916 Global GDAC Argo data files (2023-01-10 snapshot) https://www.seanoe.org/data/00311/42182/data/98916.tar.gz (50.8GiB, openAccess=True)
Files for BGC only:
- #105307 BGC Sprof data files (2023-09-09 snapshot) https://www.seanoe.org/data/00311/42182/data/105307.tar.gz (3.7GiB, openAccess=True)
- #104705 BGC Sprof data files (2023-09-09 snapshot) https://www.seanoe.org/data/00311/42182/data/104705.tar.gz (3.6GiB, openAccess=True)
- #104148 BGC Sprof data files (2023-08-09 snapshot) https://www.seanoe.org/data/00311/42182/data/104148.tar.gz (3.4GiB, openAccess=True)
- #103643 BGC Sprof data files (2023-07-09 snapshot) https://www.seanoe.org/data/00311/42182/data/103643.tar.gz (844.2MiB, openAccess=True)
- #103088 BGC Sprof data files (2023-06-09 snapshot) https://www.seanoe.org/data/00311/42182/data/103088.tar.gz (3.3GiB, openAccess=True)
- #102271 BGC Sprof data files (2023-05-09 snapshot) https://www.seanoe.org/data/00311/42182/data/102271.tar.gz (3.3GiB, openAccess=True)
- #101753 BGC Sprof data files (2023-04-08 snapshot) https://www.seanoe.org/data/00311/42182/data/101753.tar.gz (2.6GiB, openAccess=False)
- #100490 BGC Sprof data files (2023-03-08 snapshot) https://www.seanoe.org/data/00311/42182/data/100490.tar.gz (3.0GiB, openAccess=False)
- #99793 BGC Sprof data files (2023-02-08 snapshot) https://www.seanoe.org/data/00311/42182/data/99793.tar.gz (1.4GiB, openAccess=False)
- #98921 BGC Sprof data files (2023-01-08 snapshot) https://www.seanoe.org/data/00311/42182/data/98921.tar.gz (2.6GiB, openAccess=False)
```



ID and reference the most appropriate dataset with **ArgoDOI**

Promote analysis reproducibility ...

```
ArgoDOI().search('2022-10', network='BGC') # Return doi closest to a given date for a specific network
```

```
<argopy.DOIrecord>
DOI: 10.17882/42182#96562
Title: Argo float data and metadata from Global Data Assembly Centre (Argo GDAC) – Snapshot of BGC Sprof data fil
Date: 2022-10-08
Network: BGC
File: https://www.seanoe.org/data/00311/42182/data/96562.tar.gz (3.0GiB, openAccess=True)
```

```
ArgoDOI(hashtag='96562')
```

```
<argopy.DOIrecord>
DOI: 10.17882/42182#96562
Title: Argo float data and metadata from Global Data Assembly Centre (Argo GDAC) – Snapshot of BGC Sprof data fil
Date: 2022-10-08
Network: BGC
File: https://www.seanoe.org/data/00311/42182/data/96562.tar.gz (3.0GiB, openAccess=True)
```

```
ArgoDOI(hashtag='96562').dx
```

```
'https://dx.doi.org/10.17882/42182#96562'
```



Ever get lost with all Argo manuals ?

```
from argopy import ArgoDocs
ArgoDocs().list
```

	category		title	doi	id
0	Argo data formats		Argo user's manual	10.13155/29825	29825
1	Quality control	Argo Quality Control Manual for CTD and Trajec...	Argo quality control manual for dissolved oxyg...	10.13155/33951	33951
2	Quality control	Argo quality control manual for biogeochemical...	BGC-Argo quality control manual for the Chloro...	10.13155/46542	46542
3	Quality control	BGC-Argo quality control manual for nitrate co...	Quality control for BGC-Argo radiometry	10.13155/40879	40879
4	Quality control	Quality control for BGC-Argo radiometry	10.13155/35385	35385	
5	Quality control	Quality control for BGC-Argo radiometry	10.13155/84370	84370	
6	Quality control	Quality control for BGC-Argo radiometry	10.13155/62466	62466	
7	Cookbooks	Argo DAC profile cookbook	10.13155/41151	41151	
8	Cookbooks	Argo DAC trajectory cookbook	10.13155/29824	29824	
9	Cookbooks	DMQC Cookbook for Core Argo parameters	10.13155/78994	78994	
10	Cookbooks	Processing Argo oxygen data at the DAC level	10.13155/39795	39795	
11	Cookbooks	Processing Bio-Argo particle backscattering at...	10.13155/39459	39459	
12	Cookbooks	Processing BGC-Argo chlorophyll-A concentratio...	10.13155/39468	39468	
13	Cookbooks	Processing Argo measurement timing i...			
14	Cookbooks	Processing BGC-Argo CDOM concentrat...			
15	Cookbooks	Processing Bio-Argo nitrate concen...			
16	Cookbooks	Processing BGC-Argo Radiometric dat...			
17	Cookbooks	Processing BGC-Argo pH data at t...			
18	Cookbooks	Description of the Argo GDAC File Ch...			
19	Cookbooks	Description of the Argo GDAC File Me...			
20	Cookbooks	BGC-Argo synthetic profile file proc...			
21	Cookbooks	Argo GDAC File Changelog			
22	Cookbooks	Processing BGC-Argo pH data at t...			
23	Cookbooks	Processing BGC-Argo nitrate concen...			

```
id_list = ArgoDocs().search('user')
for id in id_list:
    print("-", id, ArgoDocs(id))
```

- 29825 <argopy.ArgoDocs>
Title: Argo user's manual
DOI: 10.13155/29825
url: <https://dx.doi.org/10.13155/29825>
last pdf: <https://archimer.ifremer.fr/doc/00187/29825/94819.pdf>
Abstract: This document is the Argo data user's manual. It contains the description of the formats and files produced by the Argo Data Assembly Centres (DACs).

```
ArgoDocs(29825).open_pdf()
# ArgoDocs(29825).open_pdf(page=12)
```

List and search all ADMT documents with **ArgoDocs**



Argo deployment plan

Connect to the OceanOPS database
to get the global or regional deployment plan

```
from argopy import OceanOPSDeployments

deployment = ArgoNVSReferenceTables()

<argo.deployment_plan>
API: https://www.ocean-ops.org/api/1/data/platform
Domain: [lon=-/-; lat=-/-; t=2025-06-/-]
Deployed only: False
Nb of floats in the deployment plan: - [Data not retrieved yet]
```

```
deployment.to_dataframe() # Trigger download of the full plan
```

	date	lat	lon	wmo	status_name	status_code	program	country	model
0	2025-06-10 00:00:00	-5.200	-81.650	7901039	CONFIRMED	1	Coriolis	FRANCE	ARVOR
1	2025-06-10 00:00:00	-6.500	-81.350	1902608	CONFIRMED	1	Coriolis	FRANCE	ARVOR
2	2025-06-10 00:00:00	-17.000	-74.000	5907068	CONFIRMED	1	Coriolis	FRANCE	ARVOR
3	2025-06-10 00:00:00	-18.000	-72.000	4903788	CONFIRMED	1	Coriolis	FRANCE	ARVOR
4	2025-06-10 00:00:00	-13.000	-77.500	2903900	CONFIRMED	1	Coriolis	FRANCE	ARVOR
...
526	2028-05-20 12:00:00	35.000	19.500	TMP-736322952	PROBABLE	0	Argo ITALY	ITALY	ARVOR
527	2028-06-14 08:56:58	36.710	17.402	TMP1190108211	PROBABLE	0	Argo ITALY	ITALY	ARVOR_D
528	2028-07-14 09:02:22	36.202	17.700	TMP2027352587	PROBABLE	0	Argo ITALY	ITALY	PROVOR_V - J
529	2028-10-14 08:58:10	39.690	11.917	TMP633847789	PROBABLE	0	Argo ITALY	ITALY	ARVOR_D
530	2028-10-14 09:04:07	42.830	8.180	TMP226293750	PROBABLE	0	Argo ITALY	ITALY	PROVOR_V - J

531 rows × 9 columns

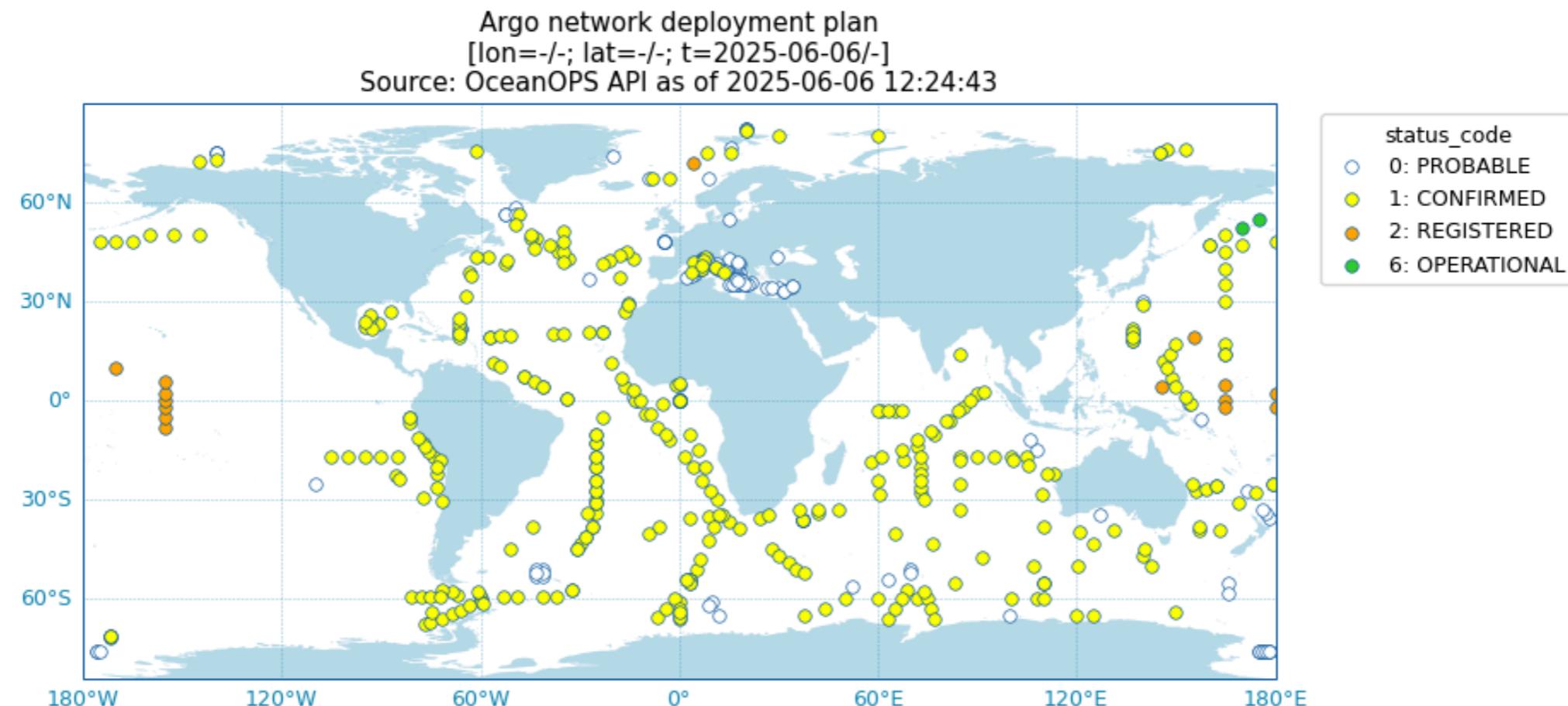
```
deployment = OceanOPSDeployments([-60, 0, 0, 90])
deployment = OceanOPSDeployments([-20, 0, 42, 51, '2020-01', '2021-01'])
deployment = OceanOPSDeployments([-180, 180, -90, 90, '2020-01', None])
```



Argo deployment plan

Connect to the OceanOPS database
to get the global or regional deployment plan

```
deployment.plot_status()
```



Advanced examples



Use the **parallel** option to improve performances on large selection

```
from argopy import DataFetcher  
  
box = [-60, -30, 40.0, 60.0, 0.0, 100.0, "2007-01-01", "2007-04-01"]  
  
ds = DataFetcher(parallel='thread', progress=True).region(box).to_xarray()
```

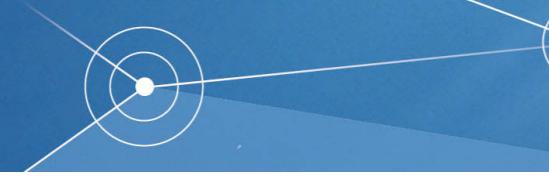
3 methods are available to set-up your data fetching requests in parallel:

- **thread**: multi-threading with a `concurrent.futures.ThreadPoolExecutor`
- **process**: multi-processing with a `concurrent.futures.ProcessPoolExecutor`
- **client**: a Dask Cluster

You can further control the parallel requests with **chunks** or **chunks_maxsize**

```
fetcher = DataFetcher(parallel=True,  
                      chunks={'lon': 5, 'lat':1, 'dpt':1, 'time':1}).region(box)
```

Access point dimension	Maximum chunk size
region / lon	20 deg
region / lat	20 deg
region / dpt	500 m or db
region / time	90 days
float / wmo	5
profile / wmo	5



Performances

or use a Dask client:

```
from dask.distributed import Client  
  
client = Client(processes=True)  
print(client)
```

Client

Client-592b498a-42d7-11f0-a744-acde48001122

Connection method: Cluster object Cluster type: distributed.LocalCluster

Dashboard: <http://127.0.0.1:8787/status>

▼ Cluster Info

LocalCluster

0a3f7434

Dashboard: <http://127.0.0.1:8787/status> Workers: 4

Total threads: 8 Total memory: 16.00 GiB

Status: running Using processes: True

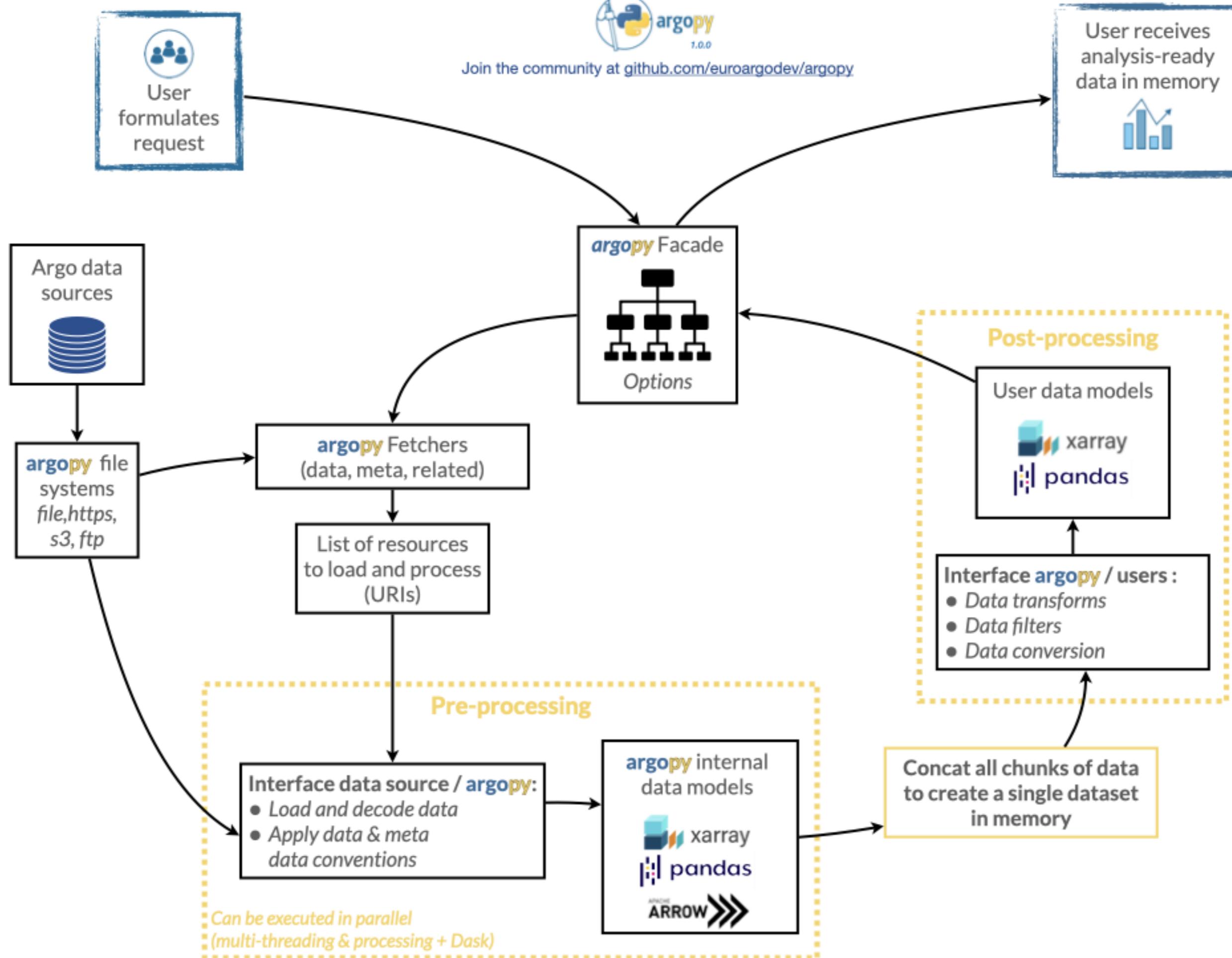
► Scheduler Info

```
from argopy import DataFetcher  
  
box = [-60, -30, 40.0, 60.0, 0.0, 100.0, "2007-01-01", "2007-04-01"]  
  
ds = DataFetcher(parallel=client).region(box).to_xarray()
```

Argopy data processing chain



Join the community at github.com/euroargodev/argopy





Data type casting

argopy provides Argo dataset with variables appropriately casted !

```
from argopy import DataFetcher
fetcher = DataFetcher(src='gdac', gdac='http', mode='expert').float(6903091)
ds = fetcher.to_xarray()
```

```
from argopy import ArgoFloat
af = ArgoFloat(6903091)
ds = af.open_dataset('prof')
```

```
from argopy import gdacfs
fs = gdacfs('http')
ds = fs.open_dataset('dac/coriolis/6903091/6903091_prof.nc', xr_opts={'engine':'argo'})
```

You can also use the `argo` xarray engine coming with argopy on any Argo netcdf file

```
import xarray as xr
ds = xr('6903091_prof.nc', engine='argo')
```



from argopy import ArgoFloat

In [4]: af = ArgoFloat(WMO)

```
# or:  
# af = ArgoFloat(WMO, host='/home/ref-argo/gdac') # Use your local GDAC copy  
# af = ArgoFloat(WMO, host='https') # Shortcut for https://data-argo.ifremer.fr ←  
# af = ArgoFloat(WMO, host='ftp') # shortcut for ftp://ftp.ifremer.fr/ifremer/argo  
# af = ArgoFloat(WMO, host='s3') # Shortcut for s3://argo-gdac-sandbox/pub
```

Ifremer https
GDAC is the
default choice

In [5]: af = ArgoFloat(WMO, aux=True)

In [6]: af

Out[6]:

```
<argofloat.6903091.http.online>  
GDAC host: https://data-argo.ifremer.fr  
DAC name: coriolis  
Network(s): ['BGC']  
Deployment date: 2021-03-05 22:55 [1469 days ago]  
Float type and manufacturer: PROVOR_III [NKE]  
Number of cycles: 330  
Dashboard: https://fleetmonitoring.euro-argo.eu/float/6903091  
Netcdf dataset available: ['Rtraj', 'Sprof', 'meta', 'meta_aux', 'prof', 'tech', 'tech_aux']
```

In [7]: af.ls_dataset()

Out[7]:

```
{'Rtraj': 'https://data-argo.ifremer.fr/dac/coriolis/6903091/6903091_Rtraj.nc',  
'Sprof': 'https://data-argo.ifremer.fr/dac/coriolis/6903091/6903091_Sprof.nc',  
'meta': 'https://data-argo.ifremer.fr/dac/coriolis/6903091/6903091_meta.nc',  
'meta_aux': 'https://data-argo.ifremer.fr/aux/coriolis/6903091/6903091_meta_aux.nc',  
'prof': 'https://data-argo.ifremer.fr/dac/coriolis/6903091/6903091_prof.nc',  
'tech': 'https://data-argo.ifremer.fr/dac/coriolis/6903091/6903091_tech.nc',  
'tech_aux': 'https://data-argo.ifremer.fr/aux/coriolis/6903091/6903091_tech_aux.nc'}
```



Iterate over ArgoFloat objects with an ArgoIndex

```
In [10]: from argopy import ArgoIndex

# Make a search on Argo index of profiles:
In [11]: idx = ArgoIndex().query.lon_lat([-70, -55, 20, 30], nrows=100)

# Then iterate over ArgoFloat matching the results:
In [12]: for a_float in idx.iterfloats():
.....    ds = a_float.open_dataset('meta')
.....    print(a_float.WMO, ds['LAUNCH_DATE'].data)
.....
1900022 2002-03-03T13:27:00.000000000
1900242 2003-06-03T17:10:00.000000000
1900639 2006-01-28T16:08:00.000000000
1900707 2006-06-02T22:19:00.000000000
```

Misc info



- 🏊 expert mode return all the Argo data, without any post-processing,
- 🎠 standard mode simplifies the dataset, remove most of its jargon and return a priori good data,
- 🚣 research mode simplifies the dataset to its heart, preserving only data of the highest quality for research studies, including studies sensitive to small pressure and salinity bias

Standard mode (default)

Table 2 Table of **argopy** data processing details in **standard** user mode

Parameters	Dataset	Level of assessment (data mode)	Level of quality (QC flags)	Pressure error	Return variables
Pressure, temperature, salinity	+	real time, adjusted and delayed mode data: [R,A,D] modes	good or probably good values (QC=[1,2])	<i>not used</i>	all without jargon [a]
Radiometry parameters [b] and BBP700 [c]		real time, adjusted and delayed mode data: [R,A,D] modes	good or probably good values, estimated or changed values (QC=[1,2,5,8])	<i>not used</i>	all without jargon [a]
CDOM [d]		None allowed	None allowed	<i>not used</i>	all without jargon [a]
All other BGC parameters [e]		real time data with adjusted values, delayed mode data: [A,D] modes	good or probably good data, estimated or changed values (QC=[1,2,5,8])	<i>not used</i>	all without jargon [a]

Research mode

Table 3 Table of **argopy** data processing details in **research** user mode

Parameters	Dataset	Level of assessment (data mode)	Level of quality (QC flags)	Pressure error	Return variables
Pressure, temperature, salinity	+	delayed mode data only: [D] mode	good values (QC=[1])	smaller than 20db	comprehensive minimum [a]
CDOM [d]		None allowed	None allowed	<i>not used</i>	comprehensive minimum [a]
All other BGC parameters [e]		delayed mode data only: [D] mode	good data, estimated or changed values (QC=[1,5,8])	<i>not used</i>	comprehensive minimum [a]



Argo index supported

Table 4 argopy GDAC index file support status

	Index file	Supported
Profile	ar_index_global_prof.txt	✓
Synthetic-Profile	argo_synthetic-profile_index.txt	✓
Bio-Profile	argo_bio-profile_index.txt	✓
Metadata	ar_index_global_meta.txt	✓
Auxiliary	etc/argo-index/argo_aux-profile_index.txt	✓
Trajectory	ar_index_global_traj.txt	✗
Bio-Trajectory	argo_bio-traj_index.txt	✗
Technical	ar_index_global_tech.txt	✗
Greylist	ar_greylist.txt	✗

shortcuts

Index file	Shortcut
ar_index_global_prof.txt	core
argo_bio-profile_index.txt	bgc-b
argo_synthetic-profile_index.txt	bgc-s
ar_index_global_meta.txt	meta
etc/argo-index/argo_aux-profile_index.txt	aux



More **index** features for experts

- Support search for profiler type or label:

```
from argopy import ArgoIndex

idx = ArgoIndex(host='https', index_file='bgc-b', cache=1)

idx.search_profiler_type(834) # PROVOR V SBE

idx.search_profiler_type([890, 891]) # PROVOR III (SBE and RBR)

idx.search_profiler_label('PROVOR V')
```

- Valid types are given by integers with values from the R8 Argo Reference table

```
from argopy import ArgoNVSReferenceTables

rt8 = argopy.ArgoNVSReferenceTables().tbl(8)
rt8.sample(5)
```

	altLabel	prefLabel	definition	deprecated
26	844	ARVOR float with SBE conductivity sensor	ARVOR profiling float with Sea-Bird Scientific...	false
50	882	XUANWU	XUANWU float (Code marked as 'Reserved' in WMO...	false
15	859	NEMO float with no conductivity	NEMO profiling float with no conductivity	false
2	841	PROVOR float with SBE conductivity sensor	PROVOR profiling float with Sea-Bird Scientific...	false