argopy

a Python library to focus on Argo science

G. Maze and K. Balem, Ifremer/LOPS

Dedicated to Argo data access, visualisation and manipulation, for regular users as well as experts

https://github.com/euroargodev/argopy

Argo Science Workshop #7, 11-13 Oct. 2022

https://github.com/euroargodev/argopy

Import the library:

```
# Import the main fetcher:
from argopy import DataFetcher as ArgoDataFetcher
```

Define what you want to fetch:

```
# a region:
ArgoSet = ArgoDataFetcher().region([-85, -45, 10., 20., 0, 1000.])
ArgoSet = ArgoDataFetcher().region([-85, -45, 10., 20., 0, 4000.,'20220901','20221001'])

# floats:
ArgoSet = ArgoDataFetcher().float([6902746, 6902747, 6902757, 6902766])

# or specific profiles:
ArgoSet = ArgoDataFetcher().profile(6902746, 34)
ArgoSet = ArgoDataFetcher().profile(6902746, np.arange(1,10))
```

More access points in dev:

- `around` : a specific x,y (z,t) location
- `along`: a trajectory (eg: of another float, cruise or hurricane track, etc)

argopy https://github.com/euroargodev/argopy

Fetch and get data as array dataset:

```
ds = ArgoSet.load().data
# or
ds = ArgoSet.to_xarray()
```

xarray.Dataset

▸ Dimensions: (**N_POINTS**: 76671)

▾ Coordinates:

N_POINTS	(N_POINTS)	int64	0 1 2 3 ... 76667 76668 76669 76670
LATITUDE	(N_POINTS)	float64	19.39 19.39 19.39 ... 14.12 14.12
LONGITUDE	(N_POINTS)	float64	-53.82 -53.82 ... -56.09 -56.09
TIME	(N_POINTS)	datetime64[ns]	2022-09-01T06:29:03 ... 2022-09-...

▾ Data variables:

CONFIG_MISSI...	(N_POINTS)	int32	6 6 6 6 6 6 6 6 ... 4 4 4 4 4 4 4 4
CYCLE_NUMBER	(N_POINTS)	int32	108 108 108 108 108 ... 58 58 58 58
DATA_MODE	(N_POINTS)	<U1	'R' 'R' 'R' 'R' ... 'R' 'R' 'R' 'R'
DIRECTION	(N_POINTS)	<U1	'A' 'A' 'A' 'A' ... 'A' 'A' 'A' 'A'
PLATFORM_NU...	(N_POINTS)	int32	4903225 4903225 ... 4903339 4903339
POSITION_QC	(N_POINTS)	int32	1 1 1 1 1 1 1 1 ... 1 1 1 1 1 1 1 1
PRES	(N_POINTS)	float64	1.04 2.04 ... 2.012e+03 2.013e+03
PRES_QC	(N_POINTS)	int32	1 1 1 1 1 1 1 1 ... 1 1 1 1 1 1 1 1
PSAL	(N_POINTS)	float64	36.11 36.11 36.11 ... 34.97 34.97
PSAL_QC	(N_POINTS)	int32	1 1 1 1 1 1 1 1 ... 1 1 1 1 1 1 1 1
TEMP	(N_POINTS)	float64	28.18 28.18 28.18 ... 3.608 3.607
TEMP_QC	(N_POINTS)	int32	1 1 1 1 1 1 1 1 ... 1 1 1 1 1 1 1 1
TIME_QC	(N_POINTS)	int32	1 1 1 1 1 1 1 1 ... 1 1 1 1 1 1 1 1

▸ Attributes: (8)

Only science-ready data are returned, by default.

Data are returned:

- as a collection of points,
- and curated, i.e. QC filters have been applied and Data Mode has been taken care of

https://github.com/euroargodev/argopy

They are many usages and fine-tuning to allow you to access and manipulate Argo data:

- filters at fetch time (standard vs expert users, automatically select QC flags or data mode, ...)
- select data sources (erddap, ftp, local, ...)
- manipulate data (points, profiles, index, interpolations, binning, TEOS-10, …)
- visualisation (trajectories, topography, histograms, ...)
- tools for Quality Control (OWC, figures, ...)
- improve performances (caching, parallel data fetching)

Just check out the documentation for more !

Incoming new features:

BGC variables, more visualisation and more access points.