epygram

A Python package to handle meteorological fields from various formats.

Alexandre Mary [1], *Sébastien Riette* [2]

1. Météo France : CNRM/GMAP/COOPE
2. Météo France : CNRM/GMME/MESO-NH

Py4ESS, Reading, Nov. 28th 2017

Outline

Handling NWP/Climate models output

- closest to the model (*historical* files, model variables & geometry) and/or post-processed output
- *proxy* for data processing (plots, extractions, aggregation, time series...)
- several data formats : GRIB, netCDF, FA (Arpege/AROME native), LFA (diagnostics)...
- meta-data
- Context : large heterogeneity in languages & graphical tools...

Handling NWP/Climate models output

- closest to the model (*historical* files, model variables & geometry) and/or post-processed output
- *proxy* for data processing (plots, extractions, aggregation, time series...)
- several data formats : GRIB, netCDF, FA (Arpege/AROME native), LFA (diagnostics)...
- meta-data
- Context : large heterogeneity in languages & graphical tools...

Goal

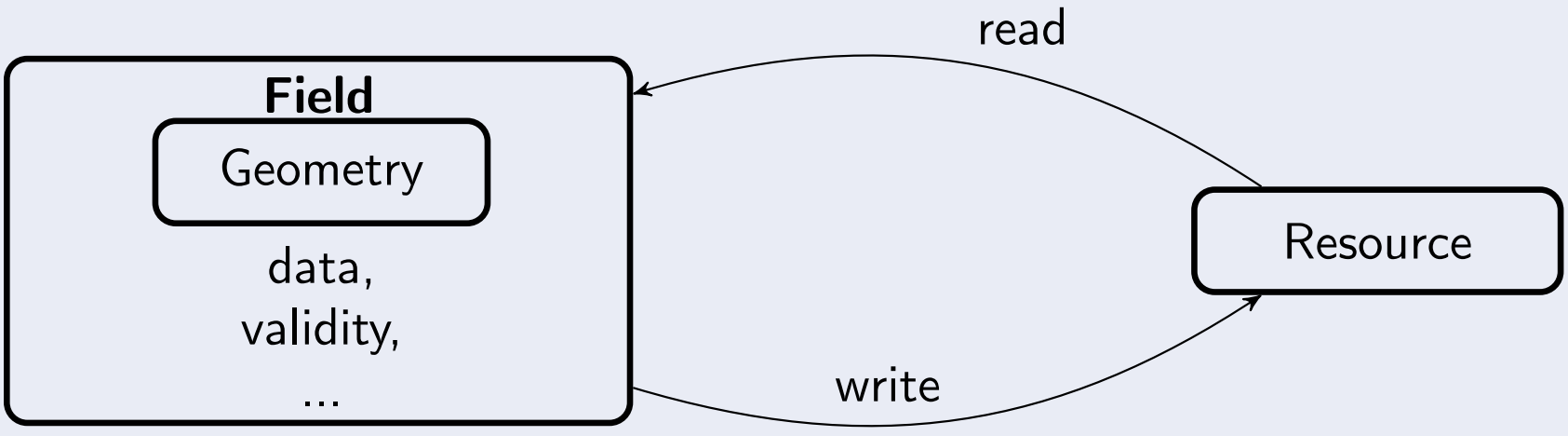
$\Rightarrow$ **one** library, with eased file/data, file/metadata, data/metadata interactions.
In a language of rich downstream applications...

Python

- lots of applicative libraries available (graphical, scientific, system, interfacing with Fortran/C, IO & data formats...)
- object-oriented : seemed mandatory for file/data/metadata interactions
- free, expanding community
- interactive/script duality
- interoperability with **Vortex** — Meteo France scripting system to run models for operational & research
- *easy-reading, easy-learning (...)*

3 main objects

- **Resource** : container of *fields* encoded in a *format*.
- **Field** : collection of values along a given *geometry*, with a given temporal validity.
  Most usual field = horizontal 2D field. (but also : vertical profile or section, transect, 3D field, local point(s)...)
- **Geometry** : collection of spatial positions in 3D (a "grid") on which is colocalized a *field*



Field identifier (*fid*)

- What does the data represent ?
- ex : U-component of wind at (hybrid-pression) level 58
  - FA : `S058WIND.U.PHYS`
  - GRIB2 :

```
discipline=0, # meteorological products
parameterCategory=3, # momentum
parameterNumber=2, # U-component of wind
typeofFirstFixedSurface=119, # hybrid-pressure levels
level=58, # level 58
```

Field identifier (*fid*)

- What does the data represent ?
- ex : U-component of wind at (hybrid-pression) level 58
  - FA : `S058WIND.U.PHYS`
  - GRIB2 :

```
discipline=0, # meteorological products
parameterCategory=3, # momentum
parameterNumber=2, # U-component of wind
typeofFirstFixedSurface=119, # hybrid-pressure levels
level=58, # level 58
```

Resource, fid, Field, Geometry, Validity

```
r = epygram.formats.resource(filename, "r")
fld = r.readfield({"shortName":"2t"})
print(fld.validity, fld.geometry)
fld.data += 273.15
r.writefield(fld)
```

Various other useful classes and functions to deal with :

- Classes for temporal Validity (incl. cumulative processes), Angles, Spectra...
- default values & options ⇒ `epygram.config` // customizable in incremental *userconfig*
- externalised side-modules :
  - Python interfaces to subroutines from IFS-ARPEGE-AROME Fortran code (FA, LFI, LFA, spectral transforms : `arpifs4py`),
  - interface to `vortex` to get resources from research/operations
  - web interface for quick, basic, plotting of research/operations runs
- a set of "quicklook" command-line tools, e.g. `epy_plot.py`

Geometries

- localisation : grid $\Leftrightarrow$ Lon/Lat, including neighbours-search
- extraction of sub-geometries
- azimuth, distance, transects computation

Geometries

- localisation : grid $\Leftrightarrow$ Lon/Lat, including neighbours-search
- extraction of sub-geometries
- azimuth, distance, transects computation

Fields

- compute diagnostics : statistics, spectrum, histogram...
- data transformation : operations (incl. overload of operators), spectral transforms, distortions
- time operations : reductions, smoothing, decumulation
- interpolation : punctual, grid2grid resampling
- plot : maps, series, profiles, sections, Hovmöller, animations

Present status

- start=Jan.2014, β-test=Dec.2014, v1.0.0=Jun.2016, now=v1.2.13
- SCM = *GIT*, license = *CeCILL-C (open-source)*,
  project = **https ://opensource.umr-cnrm.fr/projects/epygram**
- Use in MF : expanding in research ($\sim$ 50p.) & operations (Vortex : date control, fields movements...)
- ALADIN, HIRLAM consortia : spreading...
- HTML *Sphinx* documentation, some Notebooks tutorial

Present status

- start=Jan.2014, β-test=Dec.2014, v1.0.0=Jun.2016, now=v1.2.13
- SCM = *GIT*, license = *CeCILL-C (open-source)*,
  project = **https ://opensource.umr-cnrm.fr/projects/epygram**
- Use in MF : expanding in research ($\sim$ 50p.) & operations (Vortex : date control, fields movements...)
- ALADIN, HIRLAM consortia : spreading...
- HTML *Sphinx* documentation, some Notebooks tutorial

Pending

- missing : read/write spectral fields in GRIB (FA only)
- some performance optimizations (profiles extractions, ...)
- ...

And tomorrow ?

- continuous integration of new functionalities
- observations support (ODB/BUFR) ? 3D-visualisation ?
- *more collaborative ?*

And tomorrow ?

- continuous integration of new functionalities
- observations support (ODB/BUFR) ? 3D-visualisation ?
- *more collaborative ?*

Portability

- designed for Python 2.7 (portability to Python3 under consideration...)
- dependances : `numpy`, `scipy`, `matplotlib`, `pyproj`, `argparse`, `gribapi`, `netCDF4`, `pyresample`
- formats activation in config $\Rightarrow$ no need to install unrequested formats libraries
- if required, key point is to compile `arpifs4py` Fortran interfaces (gmkpack)
- other dependances : side-packages from `vortex`, distributed along :`footprints`, `bronx`, `taylorism`

Demonstration

$\Rightarrow$ **Notebooks**