ALFRED-WEGENER-INSTITUT
HELMHOLTZ-ZENTRUM FÜR POLAR- UND MEERESFORSCHUNG

PyMOR Usage

Usage Examples: Combining multiple model output variables

Paul Gierz

HELMHOLTZ

Motivation

- In some cases, you might need more than one original model output to produce a requested CMOR variable.

Usage

- Provided that the variables are mergable in xarray (e.g. they share the same time axis), you can just open up several datasets at once by listing multiple inputs in your rule.

- If you need a non-mergable extra piece of information from a file (e.g. a grid description), you can use auxiliary files.

User Config Settings: Multiple Inputs

- `rules.[N].inputs` can be a list of dictionaries, so long as you have `path` and `pattern`

- `xarray` will try to merge these using standard `open_mfdataset` rules.

```
22    rules:
23      - name: Primary Organic Carbon Production
24        description: "Primary organic production. This example has several special cases!"
25        inputs:
26          - path: "/work/ab0246/a270077/SciComp/Projects/pymor/examples/04-multivariable-i
27            pattern: diags3d01_.*.nc
28          - path: "/work/ab0246/a270077/SciComp/Projects/pymor/examples/04-multivariable-i
29            pattern: diags3d02_.*.nc
```

User Config Settings: Multiple Inputs

```yaml
    - name: Primary Organic Carbon Production
      description: "Primary organic production. This example has several special cases!"
      inputs:
        - path: "/work/ab0246/a270077/SciComp/Projects/pymor/examples/04-multivariable-i
          pattern: diags3d01_.*.nc
        - path: "/work/ab0246/a270077/SciComp/Projects/pymor/examples/04-multivariable-i
          pattern: diags3d02_.*.nc
```

```
a270077 in levante6 in piControl_LUtrans1850/outdata/recom on main [?] via pymorize
❯ module load netcdf-c; ncdump -h diags3d01_fesom_29900101.nc
netcdf diags3d01_fesom_29900101 {
dimensions:
        time = UNLIMITED ; // (12 currently)
        nodes_3d = 3668773 ;
variables:
        double time(time) ;
                time:long_name = "time" ;
                time:units = "seconds since 2990-01-01 0:0:0" ;
                time:calendar = "standard" ;
        float diags3d01(time, nodes_3d) ;
                diags3d01:description = "Diagnostic tracer 01" ;
                diags3d01:units = "" ;
                diags3d01:grid_type = "unstructured" ;
                diags3d01:_FillValue = 1.e+30f ;

// global attributes:
                :output_schedule = "unit: m first: 1 rate: 1" ;
}
```

```
a270077 in levante6 in piControl_LUtrans1850/outdata/recom on main [?] via pymorize
❯ module load netcdf-c; ncdump -h diags3d02_fesom_29900101.nc
netcdf diags3d02_fesom_29900101 {
dimensions:
        time = UNLIMITED ; // (12 currently)
        nodes_3d = 3668773 ;
variables:
        double time(time) ;
                time:long_name = "time" ;
                time:units = "seconds since 2990-01-01 0:0:0" ;
                time:calendar = "standard" ;
        float diags3d02(time, nodes_3d) ;
                diags3d02:description = "Diagnostic tracer 02" ;
                diags3d02:units = "" ;
                diags3d02:grid_type = "unstructured" ;
                diags3d02:_FillValue = 1.e+30f ;

// global attributes:
                :output_schedule = "unit: m first: 1 rate: 1" ;
}
```

```
<xarray.Dataset> Size: 4GB
Dimensions:    (time: 120, nodes_3d: 3668773)
Coordinates:
  * time       (time) object 960B 2990-02-01 00:00:00 ... 3000-01-01 00:00:00
Dimensions without coordinates: nodes_3d
Data variables:
    diags3d01  (time, nodes_3d) float32 2GB dask.array<chunksize=(12, 3668773), meta=np.ndarray>
    diags3d02  (time, nodes_3d) float32 2GB dask.array<chunksize=(12, 3668773), meta=np.ndarray>
Attributes:
    output_schedule:  unit: m first: 1 rate: 1
```

custom_step.py

```python
#!/usr/bin/env python3
"""Example of a custom step script."""


def my_custom_step(data, rule):
    """
    A custom step must have the signature of:

    def func(data, rule):
        ...
    """
    # Do some manipulation using either data
    # or information from the rule (a dict-like object)
    ...
    return data
```

User Config Settings: Aux Files

ALFRED-WEGENER-INSTITUT
HELMHOLTZ-ZENTRUM FÜR POLAR- UND MEERESFORSCHUNG

- `rules.[N].aux` is a list of dictionaries, with `path` and `name`
- By default, `pymor` assumes you are just reading in files which it can process with `read`. However, you can also set a specific loader.

```yaml
rules:
  - name: My First Rule
    aux:
      - name: My Aux Data
        path: /path/to/aux/data.csv
```

```python
def my_step(data, rule):
  aux_data = rule.aux["My Aux Data"]
  print(aux_data)
  return data
```

Exercises

1. Try going through example 4 in the repository, dealing with multivariable input.

2. Try out aux data with a plain text source: read in a list of numbers to add an arbitrary offset to each timestep of a temperature time series

3. Use aux data with a NetCDF file to familiarize yourself with loaders