x4c

— Xarray for CESM —

Feng Zhu, Jiang Zhu, Dan Amrhein, Sophia Macarewich, Bette Otto-Bliersner, Esther Brady

Docs & Tutorials: https://fzhu2e.github.io/x4c

fengzhu@ucar.edu (CGD, NSF NCAR)

```
conda install -c conda-forge xesmf cartopy pop_tools && pip install x4c
```

Xarray has become an essential software infrastructure for the basic analysis and visualization of netCDF geoscientific datasets, including CESM-simulated climate fields. However, even with this great tool, advanced CESM diagnostics still require nontrivial programming skills and efforts.

x4c is an Xarray extension that features intuitive, flexible, concise, and easy-to-use workflows for general geoscientific data analysis and visualization, as well as CESM-dedicated diagnostics.

Core Features

Regridding

```
ds = x4c.open_dataset(path, [comp=..., grid=..., adjust_month=True])
```

- `path`: the path to the netCDF file
- `comp`: the component info; e.g., 'atm', 'ocn', 'lnd', 'ice', 'rof'
- `grid`: the grid info; e.g., 'ne16', 'g16'
- `adjust_month`: correct the timestamp in CESM Postprocessing timeseries files

```
ds_rgd = ds.x.regrid([dlon=1, dlat=1, weight_file=...])
```

- `dlon`: the longitude spacing
- `dlat`: the latitude spacing
- `weight_file`: the path to a user-provided ESMF weighting file for other regridding cases

Leveraging `xESMF`
- atm: ne16/30/120 to 1x1 / 2x2
- ocn: gXX to any regular grid

Spatial Averaging

```
ds_rgd = ds.x.regrid().x.geo_mean([ind=..., latlon_range=...])
```

- `ind`: a climate index; 'nino3.4', 'nino1+2', 'nino3', 'nino4', 'tpi', 'wp', 'dmi', 'iobw'
- `latlon_range`: a square range in [lat_min, lat_max, lon_min, lon_max] to perform an arbitrary geo-spatial averaging; default: global mean

Annualization / Seasonalization

```
ds_ann = ds.x.annualize([months=...])
```

- `months`: a list of months over which to annualize; default: calendar year. Examples:
  - [6, 7, 8]: JJA seasonalization
  - [-12, 1, 2]: DJF seasonalization
  - [3, 9, 10]: an arbitrary combination

Visualization

```
x4c.set_style(style, [font_scale=1.0])
```

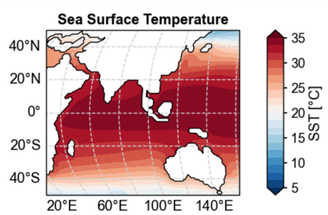
- `style`: set a style for the plots:
  - 'journal': journal paper like style
  - 'web': web-like style
  - In addition, the following strings can be appended to the base string
    - '_spines' / '_nospines': allow to show/hide spines
    - '_grid' / '_nogrid': allow to show/hide gridlines
- `font_scale`: set a global font size

```
fig, ax = ds.x.plot([...])
```

- an easy way to leverage Cartopy and plot maps without worrying about the details

```
x4c.set_style('journal_spines', font_scale=1.2)
fig, ax = sst_clim.x.plot(
    levels=np.linspace(5, 35, 21),
    cbar_kwargs={'ticks': np.linspace(5, 35, 7)},
    add_gridlines=True,
    latlon_range=[-50, 50, 30, 160],
)
```

0.9s



CESM Diagnostics

A "Timeseries Case" System

```
case = x4c.Timeseries(dirpath, [grid_dict=...])
```

- `dirpath`: the CESM Postprocessing generated timeseries directory
- `grid_dict`: the grid info; default:
  - {'atm': 'ne16', 'lnd': 'ne16', 'rof': 'ne16', 'ocn': 'g16', 'ice': 'g16'}

```
dirpath = '/glade/campaign/univ/ubrn0018/fengzhu/CESM_output/timeseries/b.e13.B1850C5.ne16_g16.icesm131_d180_fixer.Miocene.3xCO2.005'
case = x4c.Timeseries(dirpath)
case.vars_info
```

0.5s Python

```
>>> case.root_dir: /glade/campaign/univ/ubrn0018/fengzhu/CESM_output/timeseries/b.e13.B1850C5.ne16_g16.icesm131_d180_fixer.Miocene.3xCO2.005
>>> case.path_pattern: comp/proc/tseries/month_1/casename.mdl.h_str.vn.timespan.nc
>>> case.grid_dict: {'atm': 'ne16', 'lnd': 'ne16', 'rof': 'ne16', 'ocn': 'g16', 'ice': 'g16'}
>>> case.vars_info created

{'ADRAIN': ('atm', 'cam', 'h0'),
 'ADSNOW': ('atm', 'cam', 'h0'),
 'AEROD_v': ('atm', 'cam', 'h0'),
 'ANRAIN': ('atm', 'cam', 'h0'),
```

```
case.load(vn, [timespan=None, load_idx=-1])
```

- `vn`: an existing variable name under the timeseries directory
- `timespan`: a timespan in (start_year, end_year) pointing to a single or multiple files
- `load_idx`: the file index to load; default: -1 (the last file)

```
# the case of a single file
vn = 'TEMP'
case.load(vn, timespan=(6951, 7000))
print(case.ds[vn].path)
case.ds[vn]
```

0.5s Python

```
>>> case.ds["TEMP"] created
/glade/campaign/univ/ubrn0018/fengzhu/CESM_output/timeseries/b.e13.B1850C5.ne16_g16.icesm131_d180_fixer.Miocene.3xCO2.005/ocn/proc/tseries/month_1/b
```

xarray.Dataset

▶ Dimensions: (moc_comp: 3, transport_comp: 5, transport_reg: 1, **z_t**: 60, **z_t_150m**: 15, **z_w**: 60, **z_w_top**: 60, **z_w_bot**: 60, **lat_aux_grid**: 91, **moc_z**: 61, nlat: 384, nlon: 320, **time**: 600, d2: 2)

▼ Coordinates:

z_t	(z_t)	float32	500.0 1.5e+03 ... 5.375e+05
z_t_150m	(z_t_150m)	float32	500.0 1.5e+03 ... 1.35e+04 1.45e+04
z_w	(z_w)	float32	0.0 1e+03 2e+03 ... 5e+05 5.25e+05
z_w_top	(z_w_top)	float32	0.0 1e+03 2e+03 ... 5e+05 5.25e+05

A "Spell" System

```
spell = 'diag:[:ann:sa]'
case.calc(spell)
case.plot(spell)
```

- `diag`: diagnostic variable name
- `ann`: annualization / seasonalization method
- `sa`: spatial averaging method

1st order variables

```
spell = 'TS'
case.calc(spell)
case.diags[spell]
```

0.7s

```
>>> case.ds["TS"] created
>>> case.diags["TS"] created
```

xarray.DataArray 'TS' (**time**: 600, ncol: 13826)

[8295600 values with dtype=float32]

▼ Coordinates:

time	(time)	object	6951-01-31 00:00:00 ... 7000-12-

▶ Indexes: (1)

▼ Attributes:

deduced variables [contributions needed]

```
spell = 'd180p'
case.calc(spell)
case.diags[spell]
```

17.4s

```
>>> case.ds["PRECRC_H2160r"] created
>>> case.ds["PRECSC_H2160s"] created
>>> case.ds["PRECRL_H2160R"] created
>>> case.ds["PRECSL_H2160S"] created
>>> case.ds["PRECRC_H2180r"] created
>>> case.ds["PRECSC_H2180s"] created
>>> case.ds["PRECRL_H2180R"] created
>>> case.ds["PRECSL_H2180S"] created
>>> case.diags["d180p"] created
```

xarray.DataArray 'd180p' (**time**: 600, ncol: 13826)

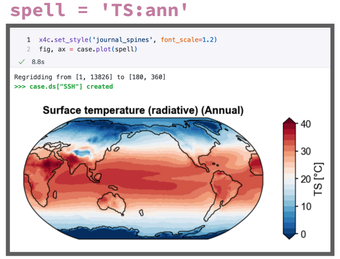
-1.743 -1.764 -2.177 -2.441 -1.713 ... -5.962 -6.126 -4.731 -4.798

spell = 'TS:ann'

```
x4c.set_style('journal_spines', font_scale=1.2)
fig, ax = case.plot(spell)
```

0.8s

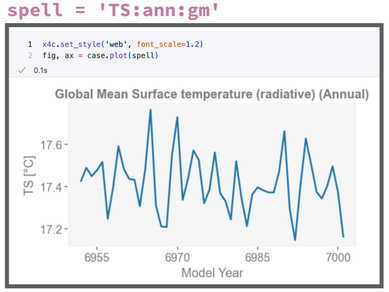
```
Regridding from [1, 13826] to [180, 360]
>>> case.ds["SSH"] created
```



spell = 'TS:ann:gm'

```
x4c.set_style('web', font_scale=1.2)
fig, ax = case.plot(spell)
```

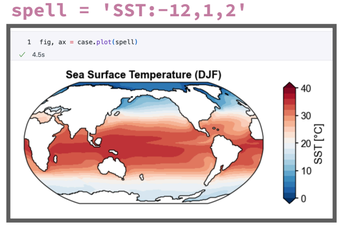
0.1s



spell = 'SST:-12,1,2'

```
fig, ax = case.plot(spell)
```

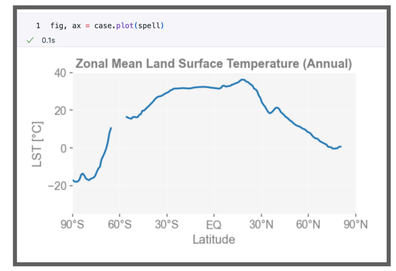
4.5s



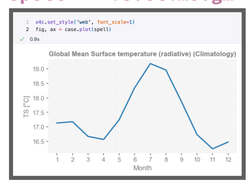
spell = 'LST:ann:zm'

```
fig, ax = case.plot(spell)
```

0.1s



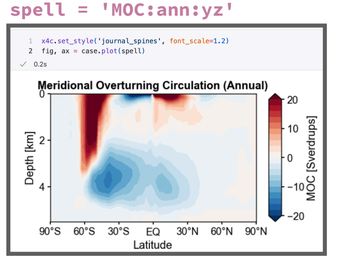
spell = 'TS:climo:gm'



spell = 'MOC:ann:yz'

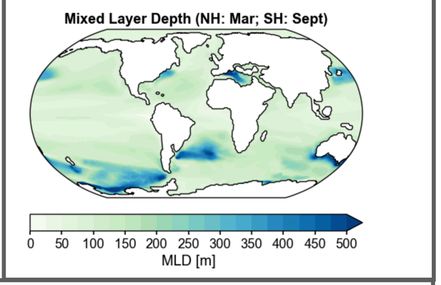
```
x4c.set_style('journal_spines', font_scale=1.2)
fig, ax = case.plot(spell)
```

0.2s




```
fig, ax = case.plot(
    'MLD:mix',
    figsize=(6, 6),
    cbar_kwargs={'orientation': 'horizontal', 'aspect': 20, 'pad': 0.05},
)
ax.set_title('Mixed Layer Depth (NH: Mar; SH: Sept)', weight='bold')
x4c.showfig(fig)
x4c.savefig(fig, './figs/map_MLD_mix.pdf')
```

5.2s



customized spells

spell = 'MLD:mix'

```
case.calc('MLD:3')
case.calc('MLD:9')

mask = case.diags['MLD:3'].lat > 0
case.diags['MLD:mix'] = case.diags['MLD:3'].mean('time').where(mask, other=case.diags['MLD:9'].mean('time'))
```

30.6s

```
>>> case.ds["XMXL"] created
>>> case.diags["MLD:3"] created
>>> case.ds["XMXL"] already loaded; to reload, run case.clear_ds("XMXL") before case.load("XMXL")
>>> case.diags["MLD:9"] created
```