x4c: Xarray for efficient CESM postprocessing, analysis, and visualization

Motivation, Design, and Features

Feng Zhu
Paleoclimate Software Engineer II
PPC, CGD, NSF NCAR

Nov 10, 2025

This material is based upon work supported by the National Center for Atmospheric Research, which is a major facility sponsored by the National Science Foundation under Cooperative Agreement No. 1852977.

Self-Introduction

MS in AOS
(2016; UW-Madison)

PhD in Earth Sciences
(2021; USC)

Research Interests:

- Paleoclimate Modeling, Data Assimilation, and Machine Learning (led two NSF-funded proposals on ML-enhanced Earth system sciences)
- Scientific Programming for facilitating Scientific Discovery

Motivation

A typical workflow for a CESM-based research:

CESM Modeling

Postprocessing
(History to Timeseries)

- Analysis
- Visualization

CESM_postprocessing

- Out of maintenance
- Not very flexible to use and debug
- Small issues (e.g., timestamps, metadata/variable handling)

xarray

Motivation

Xarray is a great tool that has provided powerful, fundamental blocks for scientific analysis and visualization tasks. However, it still requires nontrivial programming skills and efforts.

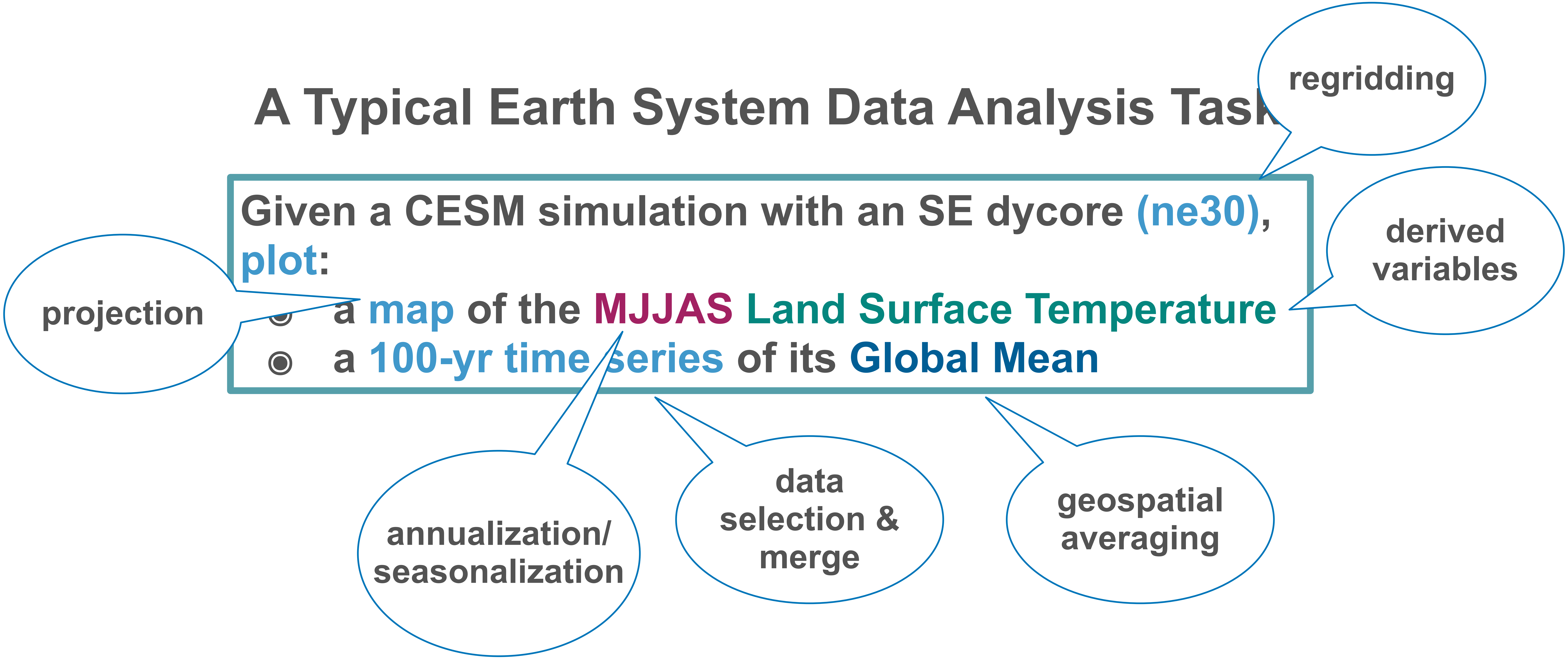
an **interruption** to the scientific mind flow

lengthy & **error-prone**

what scientists really need is a tool that works, but with minimal programming skills and efforts, and thus minimal interruption to the mind flow, so as to better focus on scientific thinking

Motivation

A Typical Earth System Data Analysis Task



Given a CESM simulation with an SE dycore (ne30), plot:

- **a map of the MJJAS Land Surface Temperature**
- **a 100-yr time series of its Global Mean**

Motivation

x4c

— Xarray for CESM —

x4c is an Xarray extension that features intuitive, flexible, concise, and easy-to-use workflows for CESM postprocessing, analysis, and visualization.

x4c aims to liberate scientists from technical details and facilitate scientific thinking.

Motivation

A typical workflow for a CESM-based research:

CESM Modeling

⇩

Postprocessing
(History to Timeseries)

⇩

- **Analysis**
- **Visualization**

CESM_postprocessing

xarray

}

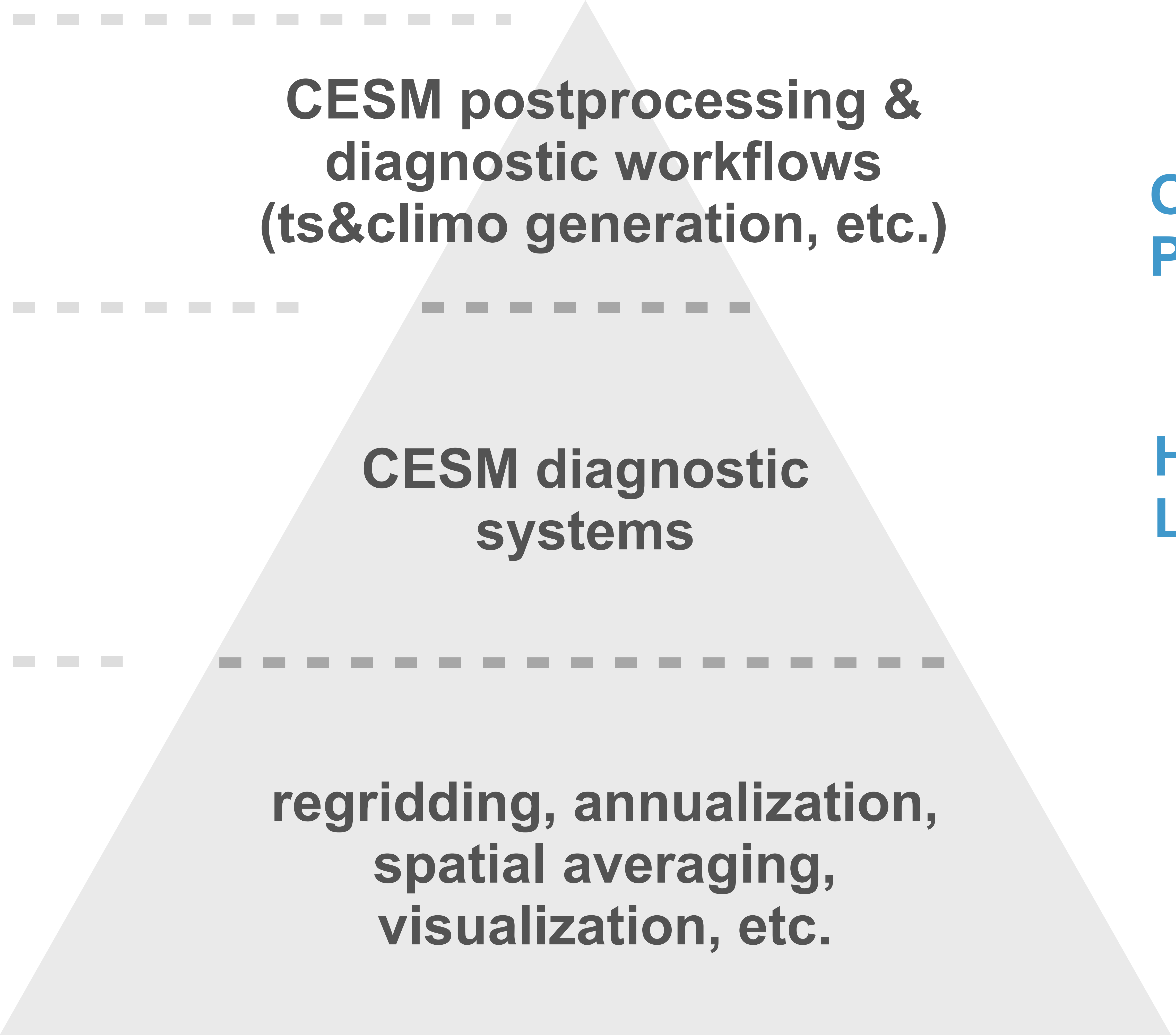
NCAR CGD

CGD-SIF Award on CESM Postprocessing (under the guidance of Michael Levy & Brian Dobins)

x4c

— Xarray for CESM —

Design



High-level Workflows (History, Timeseries)

CESM postprocessing & diagnostic workflows (ts&climo generation, etc.)

Object-Orient Programming (OOP)

Advanced Features (Timeseries)

CESM diagnostic systems

High Cohesion, Low Coupling

Fundamental Features (Xarray Extension)

regridding, annualization, spatial averaging, visualization, etc.

Intuitive, Concise, Flexible

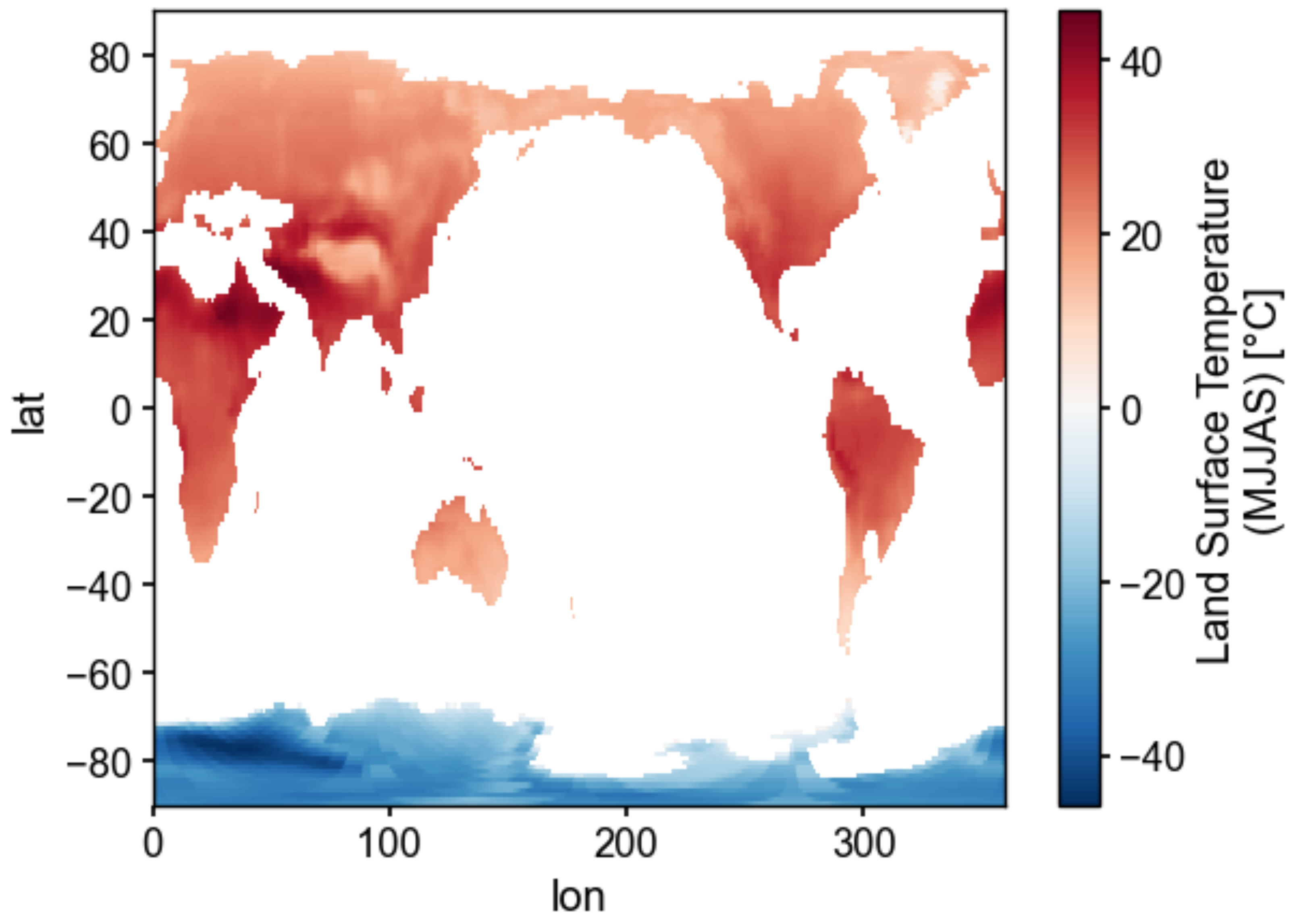
A Bottom-up Design

▣ Fundamental Features (Xarray Extension)

Given a CESM simulation with an SE dycore (ne30), plot:

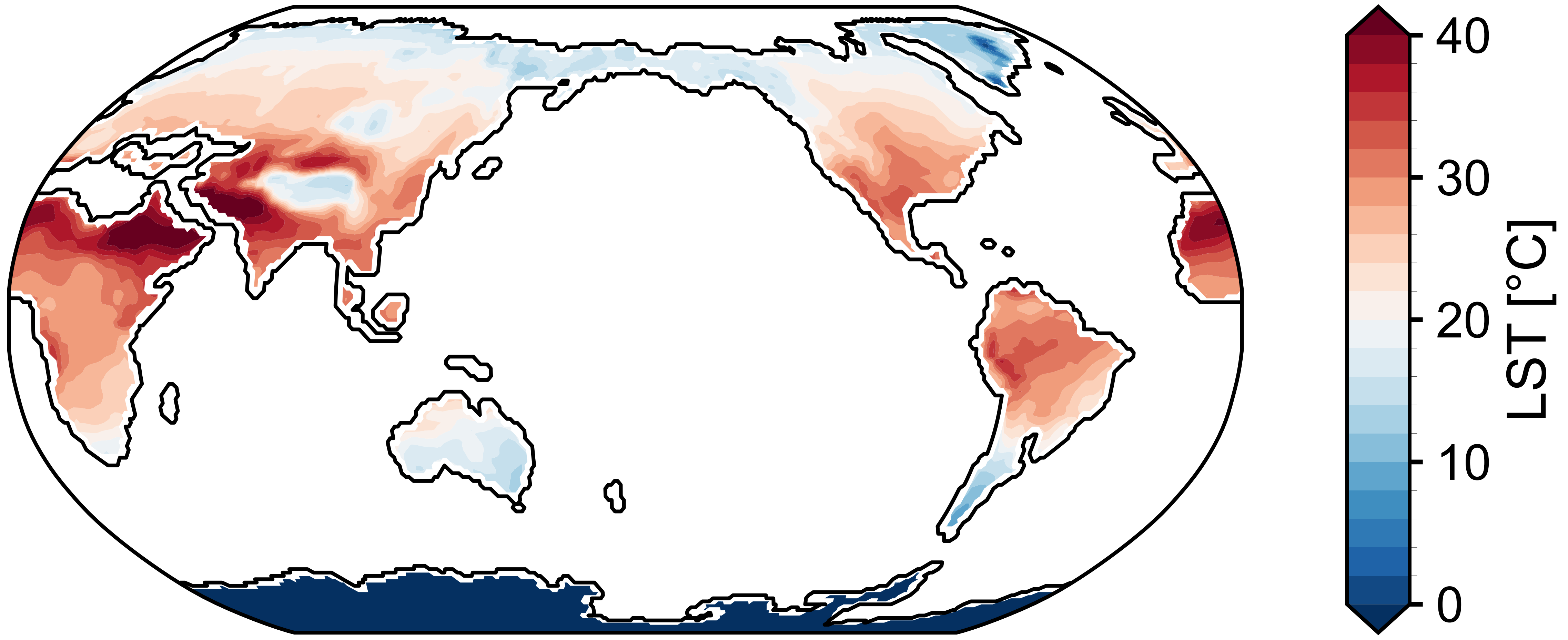
- a map of the MJJAS Land Surface Temperature

da.plot()



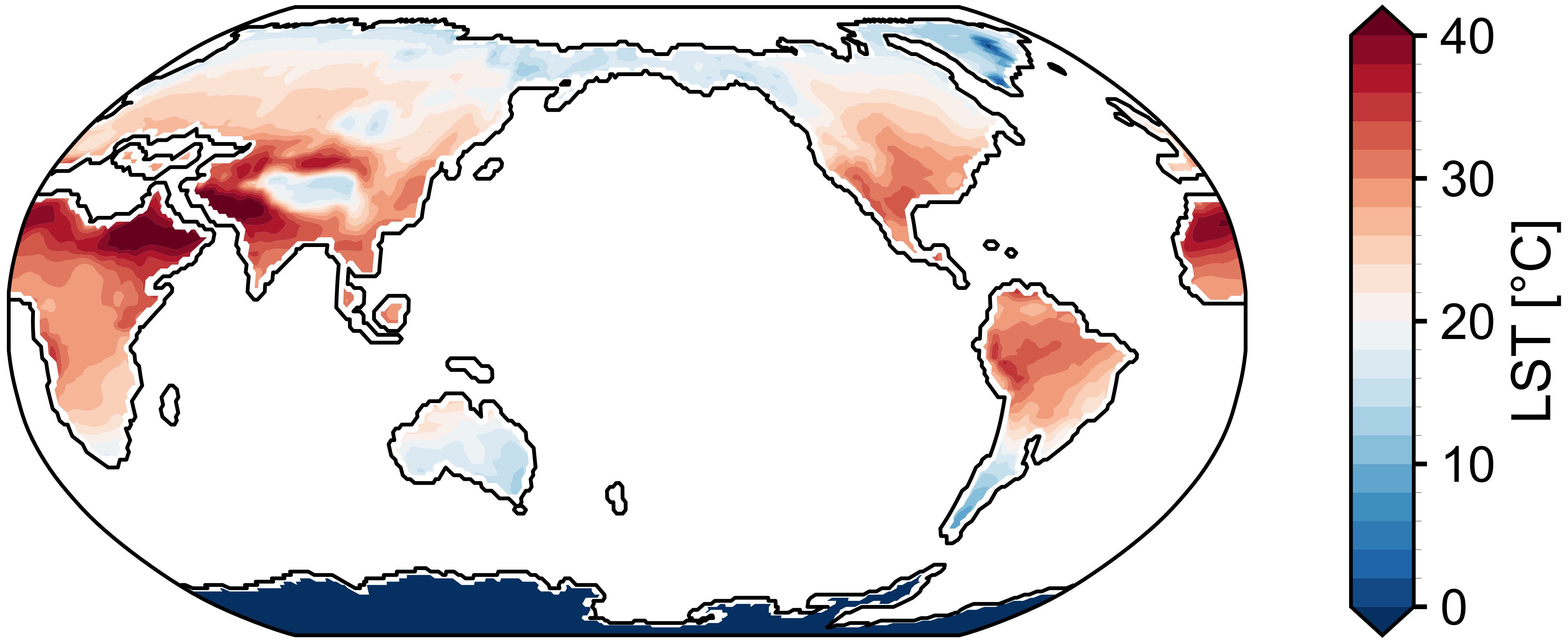
da.x.plot()

Land Surface Temperature (MJJAS)



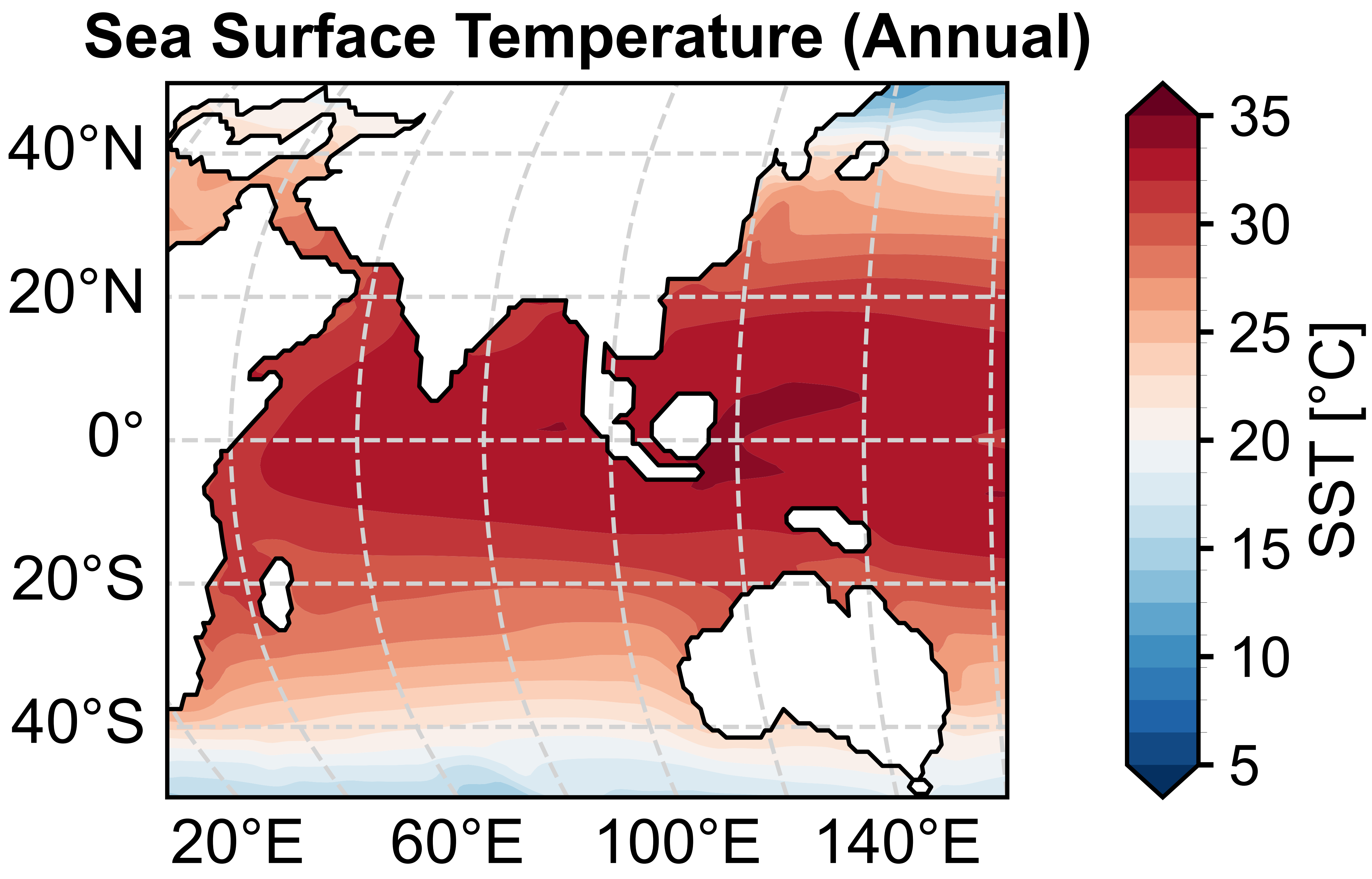
▣ Fundamental Features (Xarray Extension) Visualization

da.x.plot()


Land Surface Temperature (MJJAS)

```
x4c.set_style('journal_spines', font_scale=1.2)

fig, ax = da.x.plot()

x4c.showfig(fig)
x4c.savefig(fig, './figs/LST_MJJAS.pdf')
```

Fundamental Features (Xarray Extension) Visualization


Sea Surface Temperature (Annual)

```
x4c.set_style('journal_spines', font_scale=1.2)

fig, ax = da.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),
)

x4c.showfig(fig)
x4c.savefig(fig, './figs/SST_ann_regional.pdf')
```

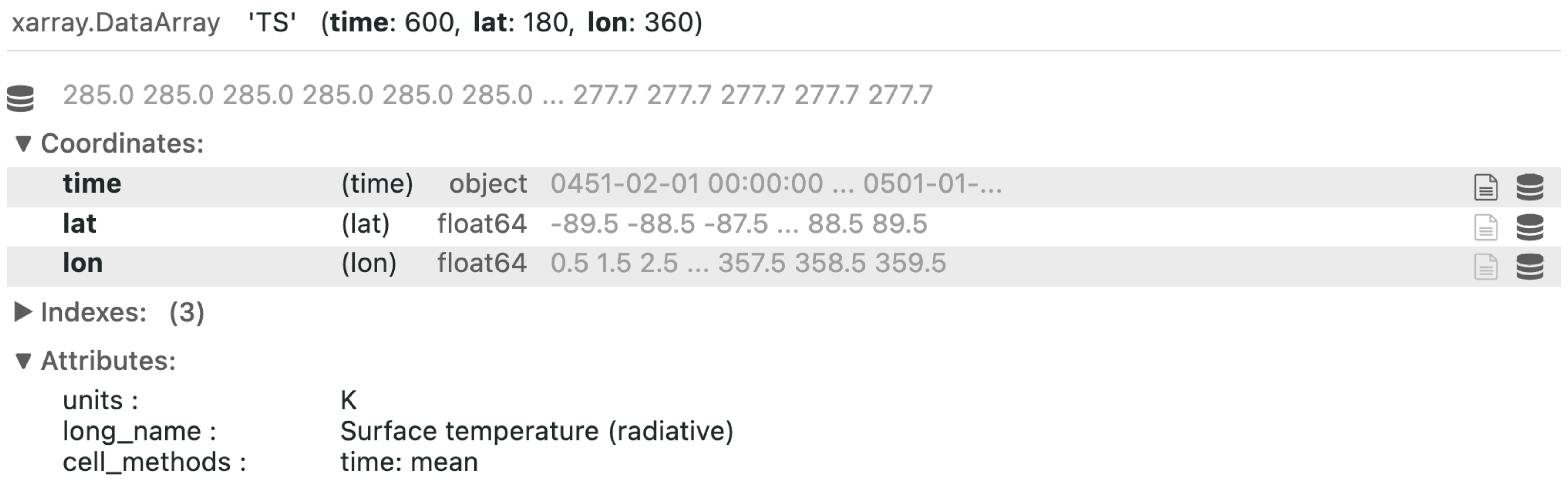
Fundamental Features (Xarray Extension) Regridding

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: ne16/30/120np4/pg3 to 1x1 / 2x2
- ocn: gXX to any regular grid

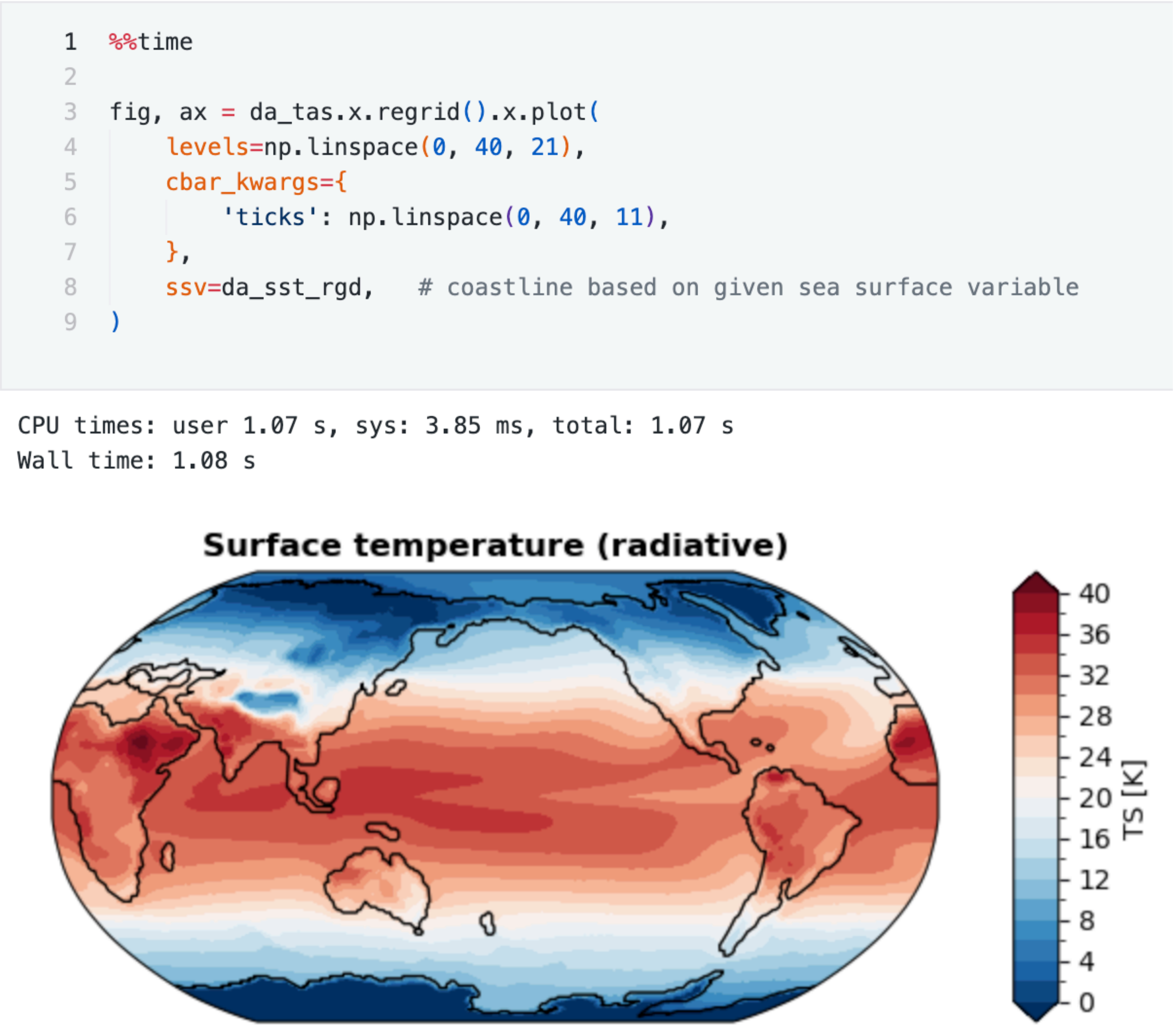
```
ds = x4c.open_dataset(fpath, comp='atm', grid='ne30np4')
ds_rgd = ds.x.regrid()
ds_rgd['TS']
```



```
xarray.DataArray   'TS'   (time: 600,  lat: 180,  lon: 360)

285.0 285.0 285.0 285.0 285.0 285.0 ... 277.7 277.7 277.7 277.7 277.7
▼ Coordinates:
    time          (time)    object   0451-02-01 00:00:00 ... 0501-01-...
    lat           (lat)     float64  -89.5 -88.5 -87.5 ... 88.5 89.5
    lon           (lon)     float64  0.5 1.5 2.5 ... 357.5 358.5 359.5
▶ Indexes:   (3)
▼ Attributes:
    units :           K
    long_name :       Surface temperature (radiative)
    cell_methods :    time: mean
```

Fundamental Features (Xarray Extension)

1. x4c w/ regridding



```
%%time

fig, ax = da_tas.x.regrid().x.plot(
    levels=np.linspace(0, 40, 21),
    cbar_kwargs={
        'ticks': np.linspace(0, 40, 11),
    },
    ssv=da_sst_rgd,    # coastline based on given sea surface variable
)
```

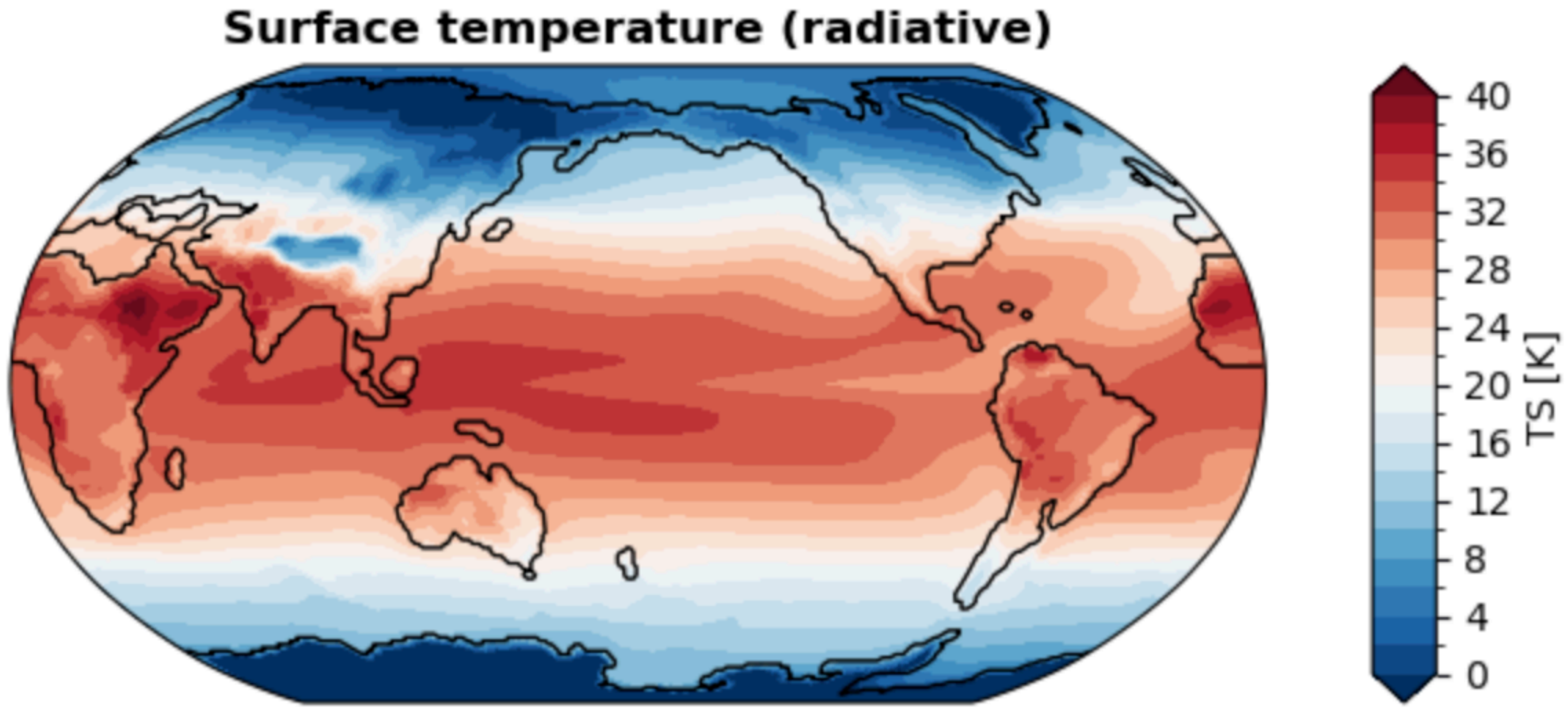
```
CPU times: user 1.07 s, sys: 3.85 ms, total: 1.07 s
Wall time: 1.08 s
```

Fundamental Features (Xarray Extension) Visualization (ne30)

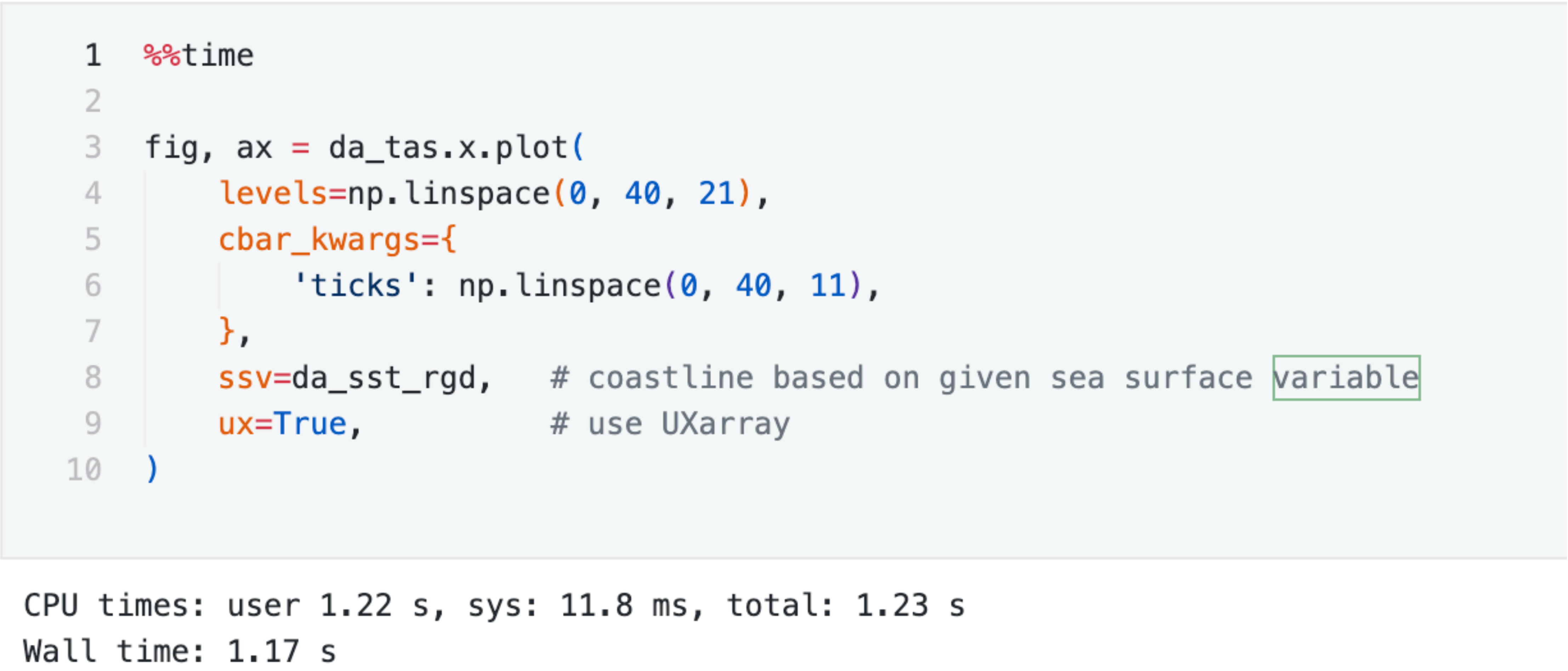
2.1 x4c w/o regridding

```
%%time
fig, ax = da_tas.x.plot(
    levels=np.linspace(0, 40, 21),
    cbar_kwargs={
        'ticks': np.linspace(0, 40, 11),
    },
    ssv=da_sst_rgd,   # coastline based on given sea surface variable
)
```

```
CPU times: user 287 ms, sys: 0 ns, total: 287 ms
Wall time: 286 ms
```

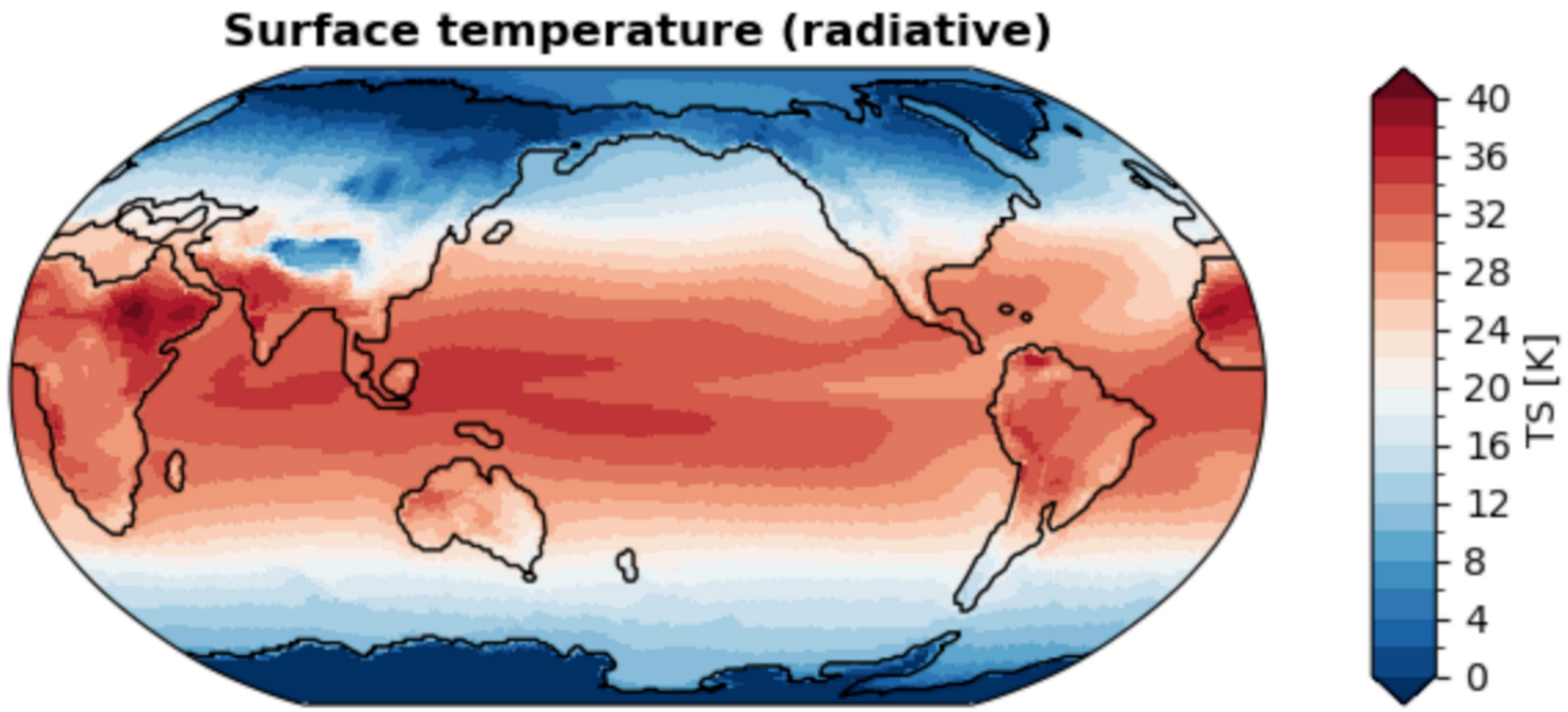


2.2 uxarray



```
%%time

fig, ax = da_tas.x.plot(
    levels=np.linspace(0, 40, 21),
    cbar_kwargs={
        'ticks': np.linspace(0, 40, 11),
    },
    ssv=da_sst_rgd,   # coastline based on given sea surface variable
    ux=True,          # use UXarray
)
```

```
CPU times: user 1.22 s, sys: 11.8 ms, total: 1.23 s
Wall time: 1.17 s
```

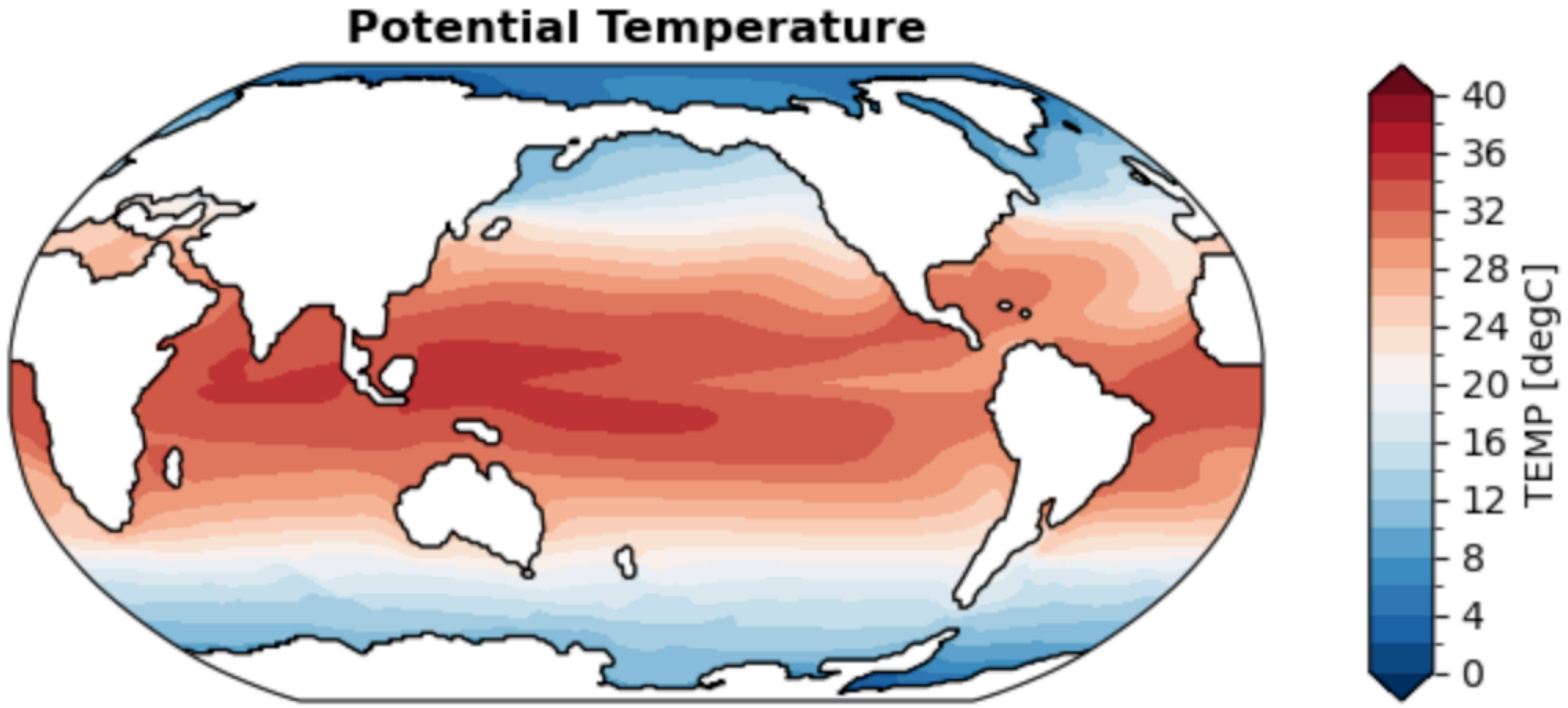


Fundamental Features (Xarray Extension) Visualization (g16)

1. x4c w/ regridding

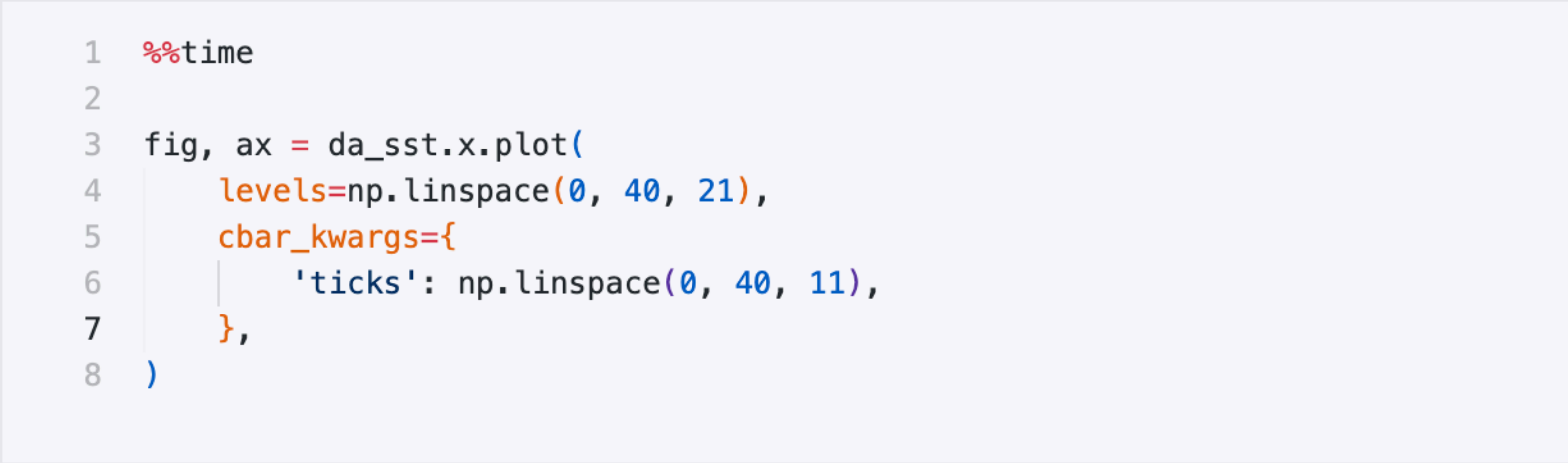
```
%%time

fig, ax = da_sst.x.regrid().x.plot(
    levels=np.linspace(0, 40, 21),
    cbar_kwargs={
        'ticks': np.linspace(0, 40, 11),
    },
)
```

```
CPU times: user 2.39 s, sys: 19.8 ms, total: 2.41 s
Wall time: 2.41 s
```



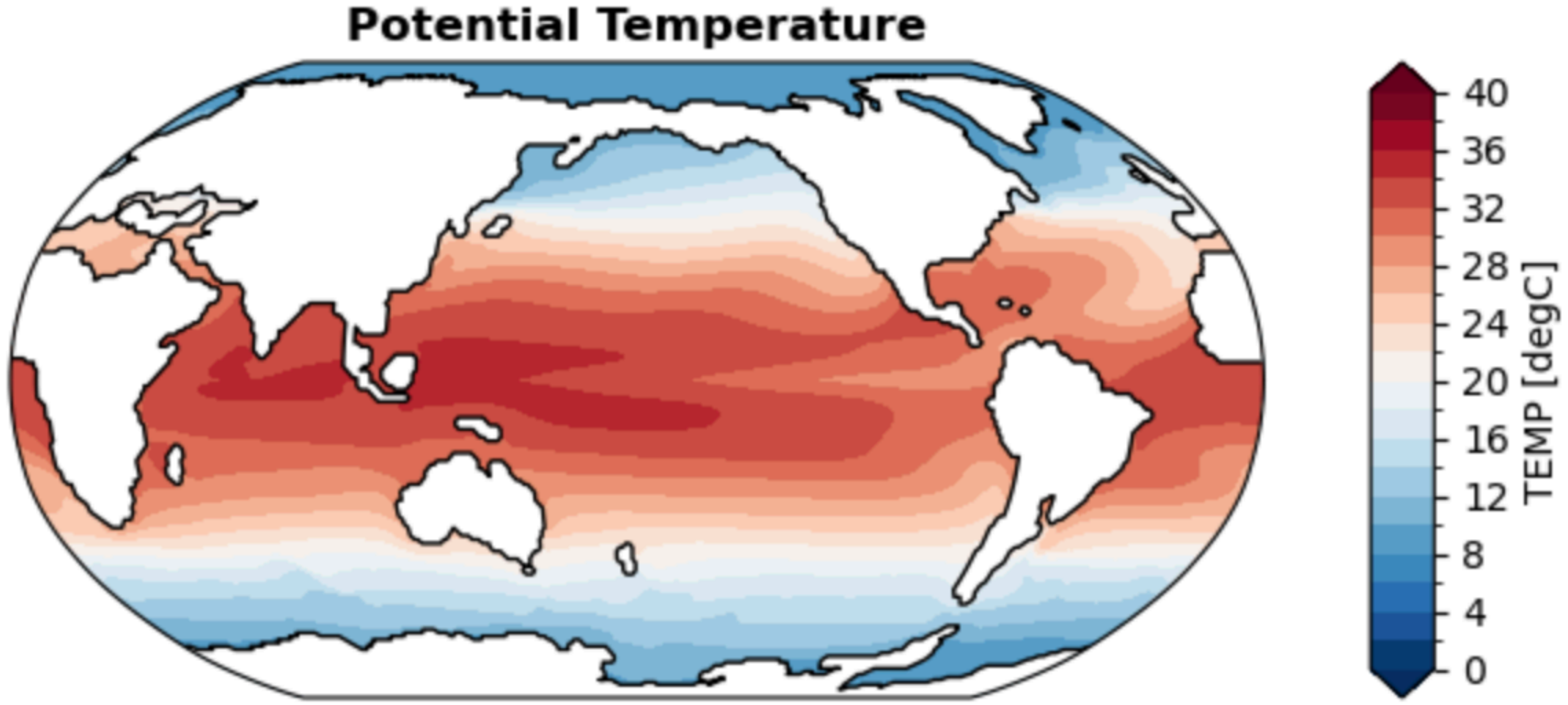
Potential Temperature

2. x4c w/o regridding



```
%%time

fig, ax = da_sst.x.plot(
    levels=np.linspace(0, 40, 21),
    cbar_kwargs={
        'ticks': np.linspace(0, 40, 11),
    },
)
```

```
CPU times: user 579 ms, sys: 183 µs, total: 579 ms
Wall time: 577 ms
```



Potential Temperature

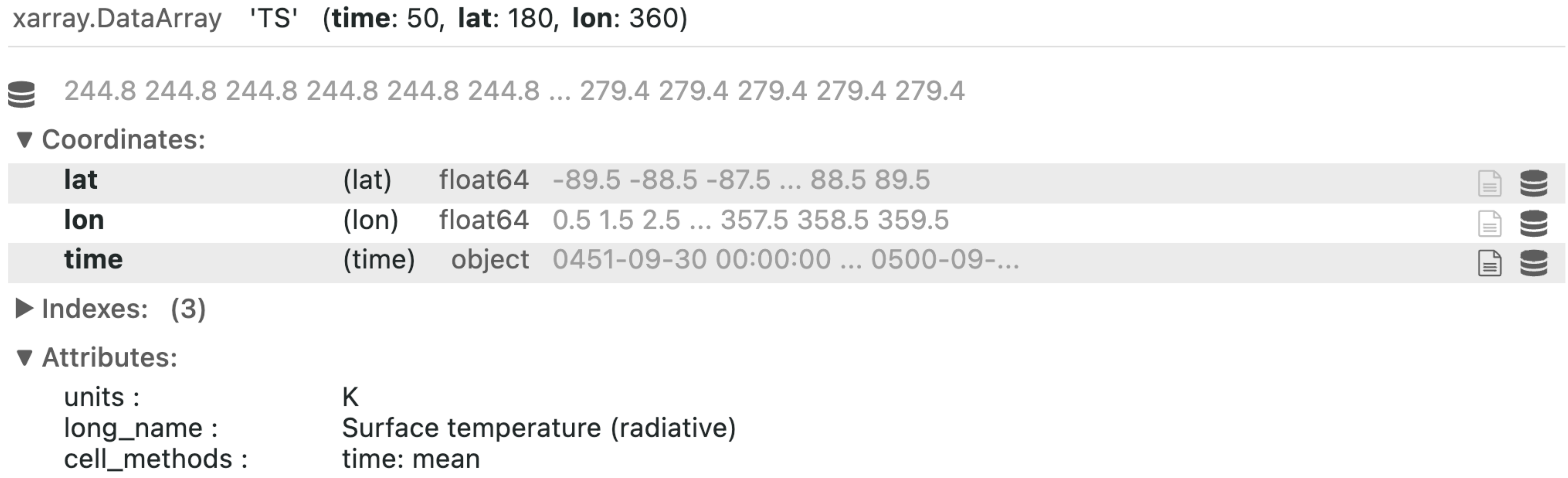
▣ Fundamental Features (Xarray Extension)

Annualization

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]: any arbitrary combination**

```
ds_ann = ds_rgd.x.annualize(months=[5, 6, 7, 8, 9], days_weighted=True)
ds_ann['TS']
```



```
xarray.DataArray   'TS'   (time: 50, lat: 180, lon: 360)

244.8 244.8 244.8 244.8 244.8 244.8 ... 279.4 279.4 279.4 279.4 279.4
▼ Coordinates:
    lat          (lat)    float64  -89.5 -88.5 -87.5 ... 88.5 89.5
    lon          (lon)    float64  0.5 1.5 2.5 ... 357.5 358.5 359.5
    time         (time)    object  0451-09-30 00:00:00 ... 0500-09-...
▶ Indexes:  (3)
▼ Attributes:
    units :            K
    long_name :        Surface temperature (radiative)
    cell_methods :     time: mean
```

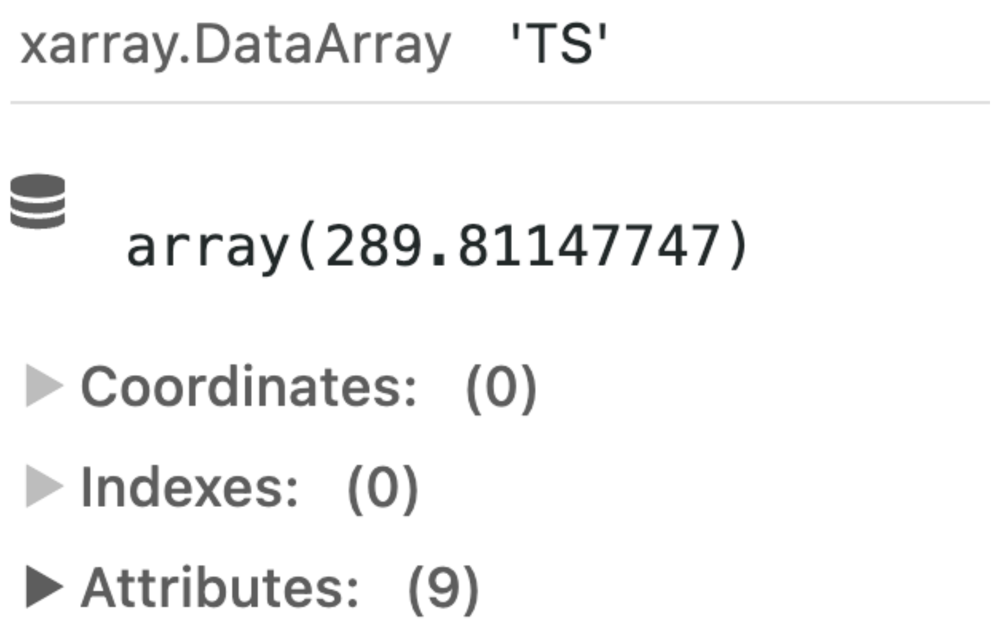
▣ Fundamental Features (Xarray Extension) Spatial Averaging

da.x.geo_mean([latlon_range=..., ind=...])

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

da.x.gm; da.x.nhm; da.x.shm; da.x.zm

```
da_gm = ds_ann['TS'].x.geo_mean()
da_gm
```



```
xarray.DataArray   'TS'

array(289.81147747)

▶ Coordinates:  (0)
▶ Indexes:  (0)
▶ Attributes:  (9)
```

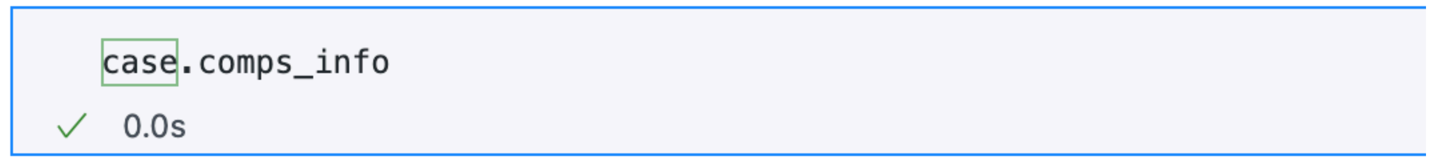
CESM Postprocessing (History)

A “Case” System

```
dirpath = '/glade/campaign/cesm/development/cross-wg/diagnostic_framework/CESM_output_for_testing/b.e30_beta02.BLT1850.ne30_t232.104'
case = x4c.History(dirpath)
```

✓ 4.0s

```
>>> case.root_dir: /glade/campaign/cesm/development/cross-wg/diagnostic_framework/CESM_output_for_testing/b.e30_beta02.BLT1850.ne30_t232.104
>>> case.casename: b.e30_beta02.BLT1850.ne30_t232.104
>>> case.paths["atm"]["cam.h0a"] created
>>> case.paths["atm"]["cam.h0i"] created
>>> case.paths["atm"]["cam.h2a"] created
>>> case.paths["atm"]["cam.h2i"] created
>>> case.paths["atm"]["cam.h3a"] created
>>> case.paths["atm"]["cam.h3i"] created
>>> case.paths["atm"]["cam.h4a"] created
>>> case.paths["atm"]["cam.h4i"] created
>>> case.paths["ocn"]["mom6.h.native"] created
>>> case.paths["ocn"]["mom6.h.rho2"] created
>>> case.paths["ocn"]["mom6.h.sfc"] created
>>> case.paths["ocn"]["mom6.h.z"] created
>>> case.paths["lnd"]["clm2.h0"] created
>>> case.paths["ice"]["cice.h"] created
>>> case.paths["rof"]["mosart.h0"] created
>>> case.vns["atm"]["cam.h0a"] created
>>> case.vns["atm"]["cam.h0i"] created
>>> case.vns["atm"]["cam.h2a"] created
>>> case.vns["atm"]["cam.h2i"] created
>>> case.vns["atm"]["cam.h3a"] created
>>> case.vns["atm"]["cam.h3i"] created
>>> case.vns["atm"]["cam.h4a"] created
>>> case.vns["atm"]["cam.h4i"] created
...
>>> case.vns["ocn"]["mom6.h.z"] created
>>> case.vns["lnd"]["clm2.h0"] created
>>> case.vns["ice"]["cice.h"] created
>>> case.vns["rof"]["mosart.h0"] created
```



```
case.comps_info
```

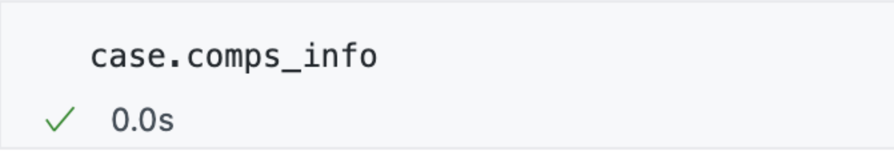
✓ 0.0s

```
{'atm': ['cam.h0a',
  'cam.h0i',
  'cam.h2a',
  'cam.h2i',
  'cam.h3a',
  'cam.h3i',
  'cam.h4a',
  'cam.h4i'],
 'ocn': ['mom6.h.native', 'mom6.h.rho2', 'mom6.h.sfc', 'mom6.h.z'],
 'lnd': ['clm2.h0'],
 'ice': ['cice.h'],
 'rof': ['mosart.h0']}
```

```
dirpath = '/glade/campaign/cesm/development/cross-wg/diagnostic_framework/CESM_output_for_testing/b.e30_beta02.BLT1850.ne30_t232.104'
case = x4c.History(
    dirpath,
    comps=['atm', 'ocn'],
    comps_info={
        'atm': ['cam.h0a'],
        'ocn': ['mom6.h.sfc'],
    }
)
```

✓ 0.3s

```
>>> case.root_dir: /glade/campaign/cesm/development/cross-wg/diagnostic_framework/CESM_output_for_testing/b.e30_beta02.BLT1850.ne30_t232.104
>>> case.casename: b.e30_beta02.BLT1850.ne30_t232.104
>>> case.paths["atm"]["cam.h0a"] created
>>> case.paths["ocn"]["mom6.h.sfc"] created
>>> case.vns["atm"]["cam.h0a"] created
>>> case.vns["ocn"]["mom6.h.sfc"] created
```



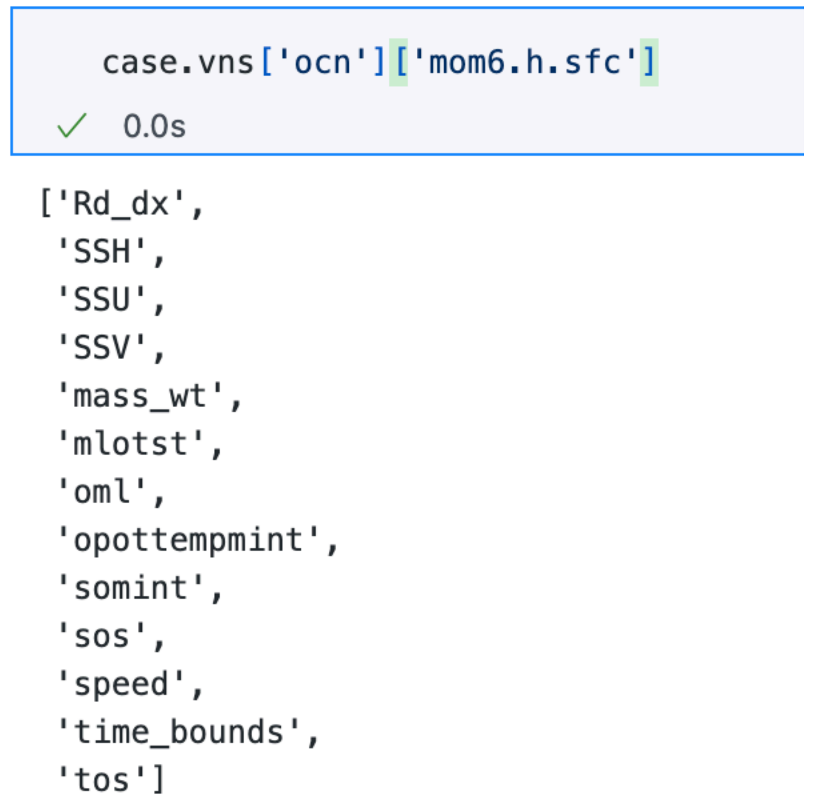
```
case.comps_info
```

✓ 0.0s

```
{'atm': ['cam.h0a'], 'ocn': ['mom6.h.sfc']}
```

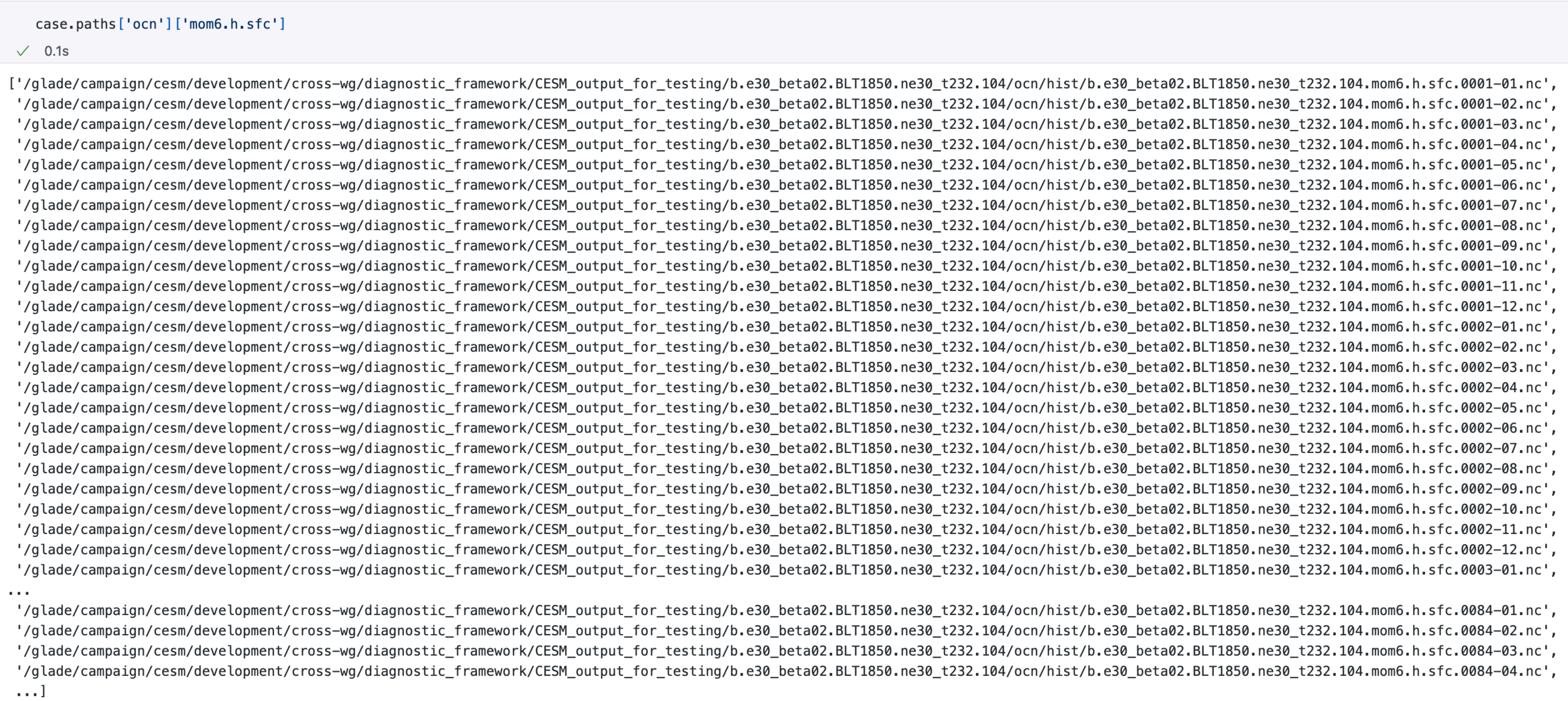
CESM Postprocessing (History)

A “Case” System



```
case.vns['ocn']['mom6.h.sfc']
✓ 0.0s
```

```
['Rd_dx',
 'SSH',
 'SSU',
 'SSV',
 'mass_wt',
 'mlotst',
 'oml',
 'opottempmint',
 'somint',
 'sos',
 'speed',
 'time_bounds',
 'tos']
```



```
case.paths['ocn']['mom6.h.sfc']
✓ 0.1s
```

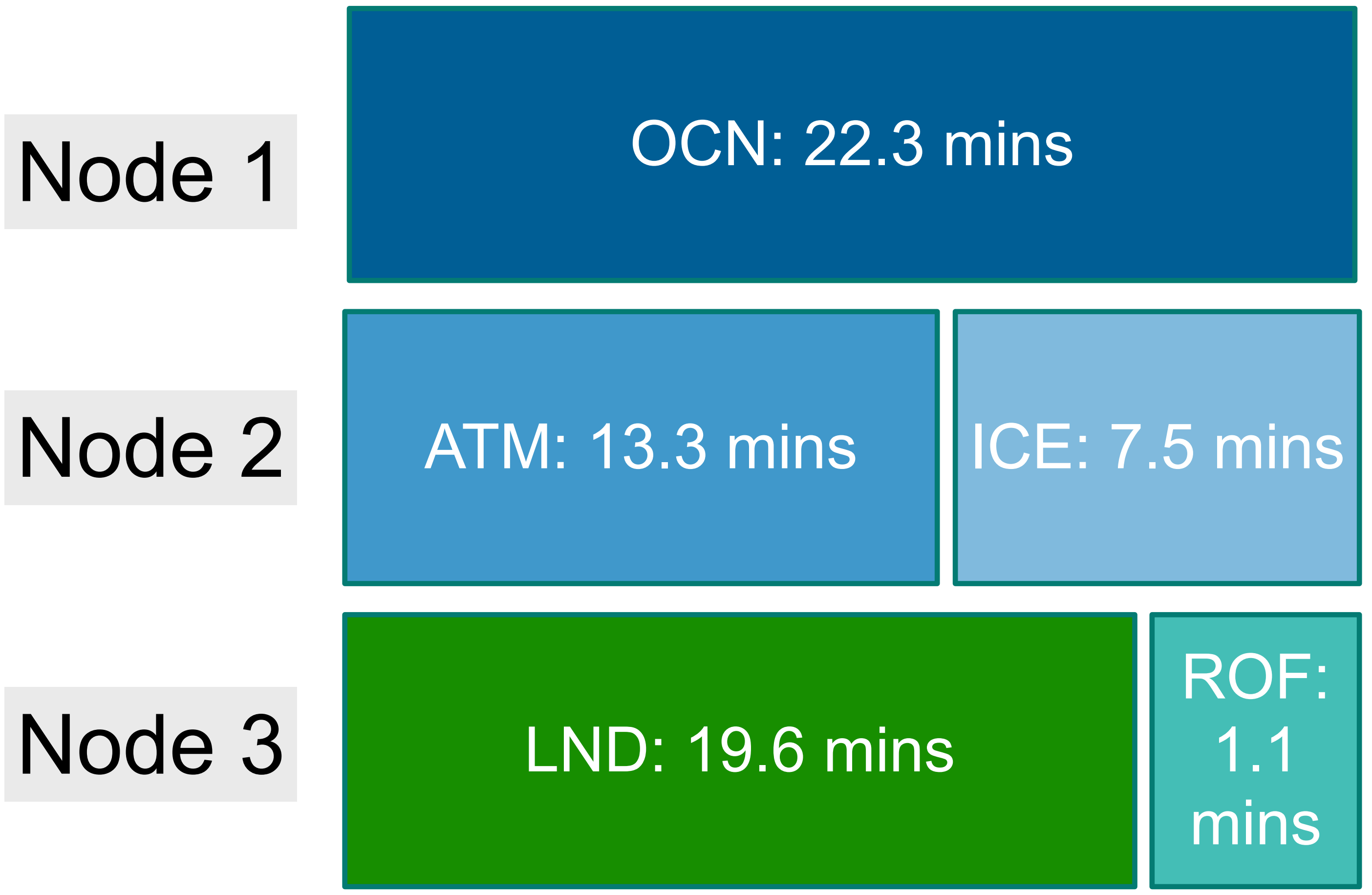
```
['/glade/campaign/cesm/development/cross-wg/diagnostic_framework/CESM_output_for_testing/b.e30_beta02.BLT1850.ne30_t232.104/ocn/hist/b.e30_beta02.BLT1850.ne30_t232.104.mom6.h.sfc.0001-01.nc',
 '/glade/campaign/cesm/development/cross-wg/diagnostic_framework/CESM_output_for_testing/b.e30_beta02.BLT1850.ne30_t232.104/ocn/hist/b.e30_beta02.BLT1850.ne30_t232.104.mom6.h.sfc.0001-02.nc',
 '/glade/campaign/cesm/development/cross-wg/diagnostic_framework/CESM_output_for_testing/b.e30_beta02.BLT1850.ne30_t232.104/ocn/hist/b.e30_beta02.BLT1850.ne30_t232.104.mom6.h.sfc.0001-03.nc',
 '/glade/campaign/cesm/development/cross-wg/diagnostic_framework/CESM_output_for_testing/b.e30_beta02.BLT1850.ne30_t232.104/ocn/hist/b.e30_beta02.BLT1850.ne30_t232.104.mom6.h.sfc.0001-04.nc',
 '/glade/campaign/cesm/development/cross-wg/diagnostic_framework/CESM_output_for_testing/b.e30_beta02.BLT1850.ne30_t232.104/ocn/hist/b.e30_beta02.BLT1850.ne30_t232.104.mom6.h.sfc.0001-05.nc',
 '/glade/campaign/cesm/development/cross-wg/diagnostic_framework/CESM_output_for_testing/b.e30_beta02.BLT1850.ne30_t232.104/ocn/hist/b.e30_beta02.BLT1850.ne30_t232.104.mom6.h.sfc.0001-06.nc',
 '/glade/campaign/cesm/development/cross-wg/diagnostic_framework/CESM_output_for_testing/b.e30_beta02.BLT1850.ne30_t232.104/ocn/hist/b.e30_beta02.BLT1850.ne30_t232.104.mom6.h.sfc.0001-07.nc',
 '/glade/campaign/cesm/development/cross-wg/diagnostic_framework/CESM_output_for_testing/b.e30_beta02.BLT1850.ne30_t232.104/ocn/hist/b.e30_beta02.BLT1850.ne30_t232.104.mom6.h.sfc.0001-08.nc',
 '/glade/campaign/cesm/development/cross-wg/diagnostic_framework/CESM_output_for_testing/b.e30_beta02.BLT1850.ne30_t232.104/ocn/hist/b.e30_beta02.BLT1850.ne30_t232.104.mom6.h.sfc.0001-09.nc',
 '/glade/campaign/cesm/development/cross-wg/diagnostic_framework/CESM_output_for_testing/b.e30_beta02.BLT1850.ne30_t232.104/ocn/hist/b.e30_beta02.BLT1850.ne30_t232.104.mom6.h.sfc.0001-10.nc',
 '/glade/campaign/cesm/development/cross-wg/diagnostic_framework/CESM_output_for_testing/b.e30_beta02.BLT1850.ne30_t232.104/ocn/hist/b.e30_beta02.BLT1850.ne30_t232.104.mom6.h.sfc.0001-11.nc',
 '/glade/campaign/cesm/development/cross-wg/diagnostic_framework/CESM_output_for_testing/b.e30_beta02.BLT1850.ne30_t232.104/ocn/hist/b.e30_beta02.BLT1850.ne30_t232.104.mom6.h.sfc.0001-12.nc',
 '/glade/campaign/cesm/development/cross-wg/diagnostic_framework/CESM_output_for_testing/b.e30_beta02.BLT1850.ne30_t232.104/ocn/hist/b.e30_beta02.BLT1850.ne30_t232.104.mom6.h.sfc.0002-01.nc',
 '/glade/campaign/cesm/development/cross-wg/diagnostic_framework/CESM_output_for_testing/b.e30_beta02.BLT1850.ne30_t232.104/ocn/hist/b.e30_beta02.BLT1850.ne30_t232.104.mom6.h.sfc.0002-02.nc',
 '/glade/campaign/cesm/development/cross-wg/diagnostic_framework/CESM_output_for_testing/b.e30_beta02.BLT1850.ne30_t232.104/ocn/hist/b.e30_beta02.BLT1850.ne30_t232.104.mom6.h.sfc.0002-03.nc',
 '/glade/campaign/cesm/development/cross-wg/diagnostic_framework/CESM_output_for_testing/b.e30_beta02.BLT1850.ne30_t232.104/ocn/hist/b.e30_beta02.BLT1850.ne30_t232.104.mom6.h.sfc.0002-04.nc',
 '/glade/campaign/cesm/development/cross-wg/diagnostic_framework/CESM_output_for_testing/b.e30_beta02.BLT1850.ne30_t232.104/ocn/hist/b.e30_beta02.BLT1850.ne30_t232.104.mom6.h.sfc.0002-05.nc',
 '/glade/campaign/cesm/development/cross-wg/diagnostic_framework/CESM_output_for_testing/b.e30_beta02.BLT1850.ne30_t232.104/ocn/hist/b.e30_beta02.BLT1850.ne30_t232.104.mom6.h.sfc.0002-06.nc',
 '/glade/campaign/cesm/development/cross-wg/diagnostic_framework/CESM_output_for_testing/b.e30_beta02.BLT1850.ne30_t232.104/ocn/hist/b.e30_beta02.BLT1850.ne30_t232.104.mom6.h.sfc.0002-07.nc',
 '/glade/campaign/cesm/development/cross-wg/diagnostic_framework/CESM_output_for_testing/b.e30_beta02.BLT1850.ne30_t232.104/ocn/hist/b.e30_beta02.BLT1850.ne30_t232.104.mom6.h.sfc.0002-08.nc',
 '/glade/campaign/cesm/development/cross-wg/diagnostic_framework/CESM_output_for_testing/b.e30_beta02.BLT1850.ne30_t232.104/ocn/hist/b.e30_beta02.BLT1850.ne30_t232.104.mom6.h.sfc.0002-09.nc',
 '/glade/campaign/cesm/development/cross-wg/diagnostic_framework/CESM_output_for_testing/b.e30_beta02.BLT1850.ne30_t232.104/ocn/hist/b.e30_beta02.BLT1850.ne30_t232.104.mom6.h.sfc.0002-10.nc',
 '/glade/campaign/cesm/development/cross-wg/diagnostic_framework/CESM_output_for_testing/b.e30_beta02.BLT1850.ne30_t232.104/ocn/hist/b.e30_beta02.BLT1850.ne30_t232.104.mom6.h.sfc.0002-11.nc',
 '/glade/campaign/cesm/development/cross-wg/diagnostic_framework/CESM_output_for_testing/b.e30_beta02.BLT1850.ne30_t232.104/ocn/hist/b.e30_beta02.BLT1850.ne30_t232.104.mom6.h.sfc.0002-12.nc',
 '/glade/campaign/cesm/development/cross-wg/diagnostic_framework/CESM_output_for_testing/b.e30_beta02.BLT1850.ne30_t232.104/ocn/hist/b.e30_beta02.BLT1850.ne30_t232.104.mom6.h.sfc.0003-01.nc',
...
 '/glade/campaign/cesm/development/cross-wg/diagnostic_framework/CESM_output_for_testing/b.e30_beta02.BLT1850.ne30_t232.104/ocn/hist/b.e30_beta02.BLT1850.ne30_t232.104.mom6.h.sfc.0084-01.nc',
 '/glade/campaign/cesm/development/cross-wg/diagnostic_framework/CESM_output_for_testing/b.e30_beta02.BLT1850.ne30_t232.104/ocn/hist/b.e30_beta02.BLT1850.ne30_t232.104.mom6.h.sfc.0084-02.nc',
 '/glade/campaign/cesm/development/cross-wg/diagnostic_framework/CESM_output_for_testing/b.e30_beta02.BLT1850.ne30_t232.104/ocn/hist/b.e30_beta02.BLT1850.ne30_t232.104.mom6.h.sfc.0084-03.nc',
 '/glade/campaign/cesm/development/cross-wg/diagnostic_framework/CESM_output_for_testing/b.e30_beta02.BLT1850.ne30_t232.104/ocn/hist/b.e30_beta02.BLT1850.ne30_t232.104.mom6.h.sfc.0084-04.nc',
 ...]
```

CESM Postprocessing (History)

History.gen_ts()

```
case.gen_ts(
    comps={'atm': ['PRECC']},
    output_dirpath=os.path.join('/glade/campaign/cgd/ppc/fengzhu/x4c/gen_ts/test', case.casename),
    staging_dirpath=os.path.join('/glade/derecho/scratch/fengzhu/x4c/gen_ts/test', case.casename),
    timespan=(1, 100),
    timestep=10,
    timestep_unit='year',
    nproc=4,
    overwrite=True,
)
```

```
● ➜  b.e30_beta02.BLT1850.ne30_t232.104 ll atm/proc/tseries/cam.h3a
  total 17G
  -rw-r--r-- 1 fengzhu ncar 1.7G Aug 21 11:07 b.e30_beta02.BLT1850.ne30_t232.104.cam.h3a.PRECC.0001010110800-0010122710800.nc
  -rw-r--r-- 1 fengzhu ncar 1.7G Aug 21 11:08 b.e30_beta02.BLT1850.ne30_t232.104.cam.h3a.PRECC.0011010110800-0020122710800.nc
  -rw-r--r-- 1 fengzhu ncar 1.7G Aug 21 11:10 b.e30_beta02.BLT1850.ne30_t232.104.cam.h3a.PRECC.0021010110800-0030122710800.nc
  -rw-r--r-- 1 fengzhu ncar 1.7G Aug 21 11:11 b.e30_beta02.BLT1850.ne30_t232.104.cam.h3a.PRECC.0031010110800-0040122710800.nc
  -rw-r--r-- 1 fengzhu ncar 1.7G Aug 21 11:12 b.e30_beta02.BLT1850.ne30_t232.104.cam.h3a.PRECC.0041010110800-0050122710800.nc
  -rw-r--r-- 1 fengzhu ncar 1.7G Aug 21 11:14 b.e30_beta02.BLT1850.ne30_t232.104.cam.h3a.PRECC.0051010110800-0060122710800.nc
  -rw-r--r-- 1 fengzhu ncar 1.7G Aug 21 11:15 b.e30_beta02.BLT1850.ne30_t232.104.cam.h3a.PRECC.0061010110800-0070122710800.nc
  -rw-r--r-- 1 fengzhu ncar 1.7G Aug 21 11:16 b.e30_beta02.BLT1850.ne30_t232.104.cam.h3a.PRECC.0071010110800-0080122710800.nc
  -rw-r--r-- 1 fengzhu ncar 1.7G Aug 21 11:17 b.e30_beta02.BLT1850.ne30_t232.104.cam.h3a.PRECC.0081010110800-0090122710800.nc
  -rw-r--r-- 1 fengzhu ncar 1.7G Aug 21 11:19 b.e30_beta02.BLT1850.ne30_t232.104.cam.h3a.PRECC.0091010110800-0100122710800.nc
```



Node 1

OCN: 22.3 mins

Node 2

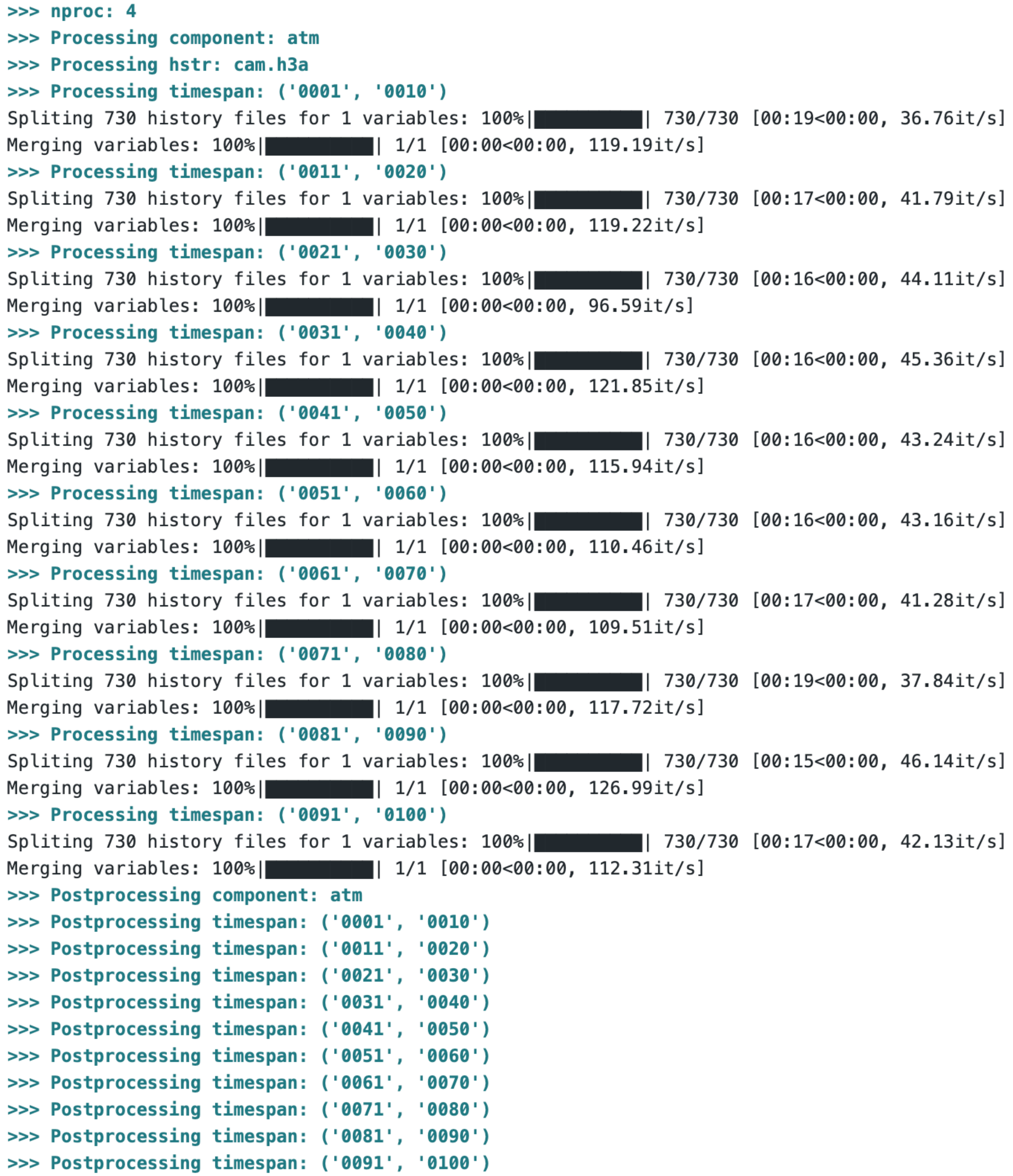
ATM: 13.3 mins

ICE: 7.5 mins

Node 3

LND: 19.6 mins

ROF: 1.1 mins



```
>>> nproc: 4
>>> Processing component: atm
>>> Processing hstr: cam.h3a
>>> Processing timespan: ('0001', '0010')
Spliting 730 history files for 1 variables: 100%|██████████| 730/730 [00:19<00:00, 36.76it/s]
Merging variables: 100%|██████████| 1/1 [00:00<00:00, 119.19it/s]
>>> Processing timespan: ('0011', '0020')
Spliting 730 history files for 1 variables: 100%|██████████| 730/730 [00:17<00:00, 41.79it/s]
Merging variables: 100%|██████████| 1/1 [00:00<00:00, 119.22it/s]
>>> Processing timespan: ('0021', '0030')
Spliting 730 history files for 1 variables: 100%|██████████| 730/730 [00:16<00:00, 44.11it/s]
Merging variables: 100%|██████████| 1/1 [00:00<00:00, 96.59it/s]
>>> Processing timespan: ('0031', '0040')
Spliting 730 history files for 1 variables: 100%|██████████| 730/730 [00:16<00:00, 45.36it/s]
Merging variables: 100%|██████████| 1/1 [00:00<00:00, 121.85it/s]
>>> Processing timespan: ('0041', '0050')
Spliting 730 history files for 1 variables: 100%|██████████| 730/730 [00:16<00:00, 43.24it/s]
Merging variables: 100%|██████████| 1/1 [00:00<00:00, 115.94it/s]
>>> Processing timespan: ('0051', '0060')
Spliting 730 history files for 1 variables: 100%|██████████| 730/730 [00:16<00:00, 43.16it/s]
Merging variables: 100%|██████████| 1/1 [00:00<00:00, 110.46it/s]
>>> Processing timespan: ('0061', '0070')
Spliting 730 history files for 1 variables: 100%|██████████| 730/730 [00:17<00:00, 41.28it/s]
Merging variables: 100%|██████████| 1/1 [00:00<00:00, 109.51it/s]
>>> Processing timespan: ('0071', '0080')
Spliting 730 history files for 1 variables: 100%|██████████| 730/730 [00:19<00:00, 37.84it/s]
Merging variables: 100%|██████████| 1/1 [00:00<00:00, 117.72it/s]
>>> Processing timespan: ('0081', '0090')
Spliting 730 history files for 1 variables: 100%|██████████| 730/730 [00:15<00:00, 46.14it/s]
Merging variables: 100%|██████████| 1/1 [00:00<00:00, 126.99it/s]
>>> Processing timespan: ('0091', '0100')
Spliting 730 history files for 1 variables: 100%|██████████| 730/730 [00:17<00:00, 42.13it/s]
Merging variables: 100%|██████████| 1/1 [00:00<00:00, 112.31it/s]
>>> Postprocessing component: atm
>>> Postprocessing timespan: ('0001', '0010')
>>> Postprocessing timespan: ('0011', '0020')
>>> Postprocessing timespan: ('0021', '0030')
>>> Postprocessing timespan: ('0031', '0040')
>>> Postprocessing timespan: ('0041', '0050')
>>> Postprocessing timespan: ('0051', '0060')
>>> Postprocessing timespan: ('0061', '0070')
>>> Postprocessing timespan: ('0071', '0080')
>>> Postprocessing timespan: ('0081', '0090')
>>> Postprocessing timespan: ('0091', '0100')
```

CESM Postprocessing (History)

History.gen_ts()

netCDF Operators (NCO) Software Stack

For production run, x4c supports multiple nodes leveraging MPI.
See *https://ncar.github.io/x4c/notebooks/post-pbs.html#CESM3-Example*

```
History: /glade/derecho/scratch/cmip7/archive/b.e30_beta06.B1850C_LTso.ne30_t232_wgx3.192.wrkflw.1
x4c Timeseries: /glade/campaign/cesm/development/cross-wg/diagnostic_framework/x4c/timeseries/b.e30_beta06.B1850C_LTso.ne30_t232_wgx3.192.wrkflw.1
Model years: 1 - 10
```

x4c

Task	Nodes	NCPUs	Mem	CPU	Elap
mom6.h.sfc+z	1	128	129.26	17.23	0.22
mom6.h.native +rho2+mosart	1	128	134.22	23.63	0.20
cam+cice.h+clm2	1	128	138	56.10	0.22
cice.h1	13	13*128	1655.82	34.93	0.25

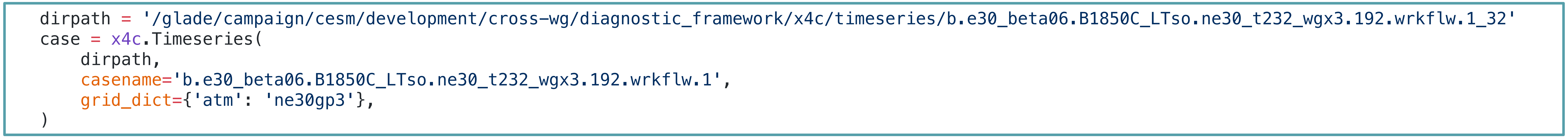
Baseline: CESM_postprocessing

Task	Nodes	NCPUs	Mem	CPU	Elap
all	16	1024	1952.26	24.66	0.21

▣ Diagnostic Features (Timeseries)   A “Case” System

case = x4c.Timeseries(dirpath, [grid_dict=…, casename=…, cesm_ver=…])

- **dirpath**: the History.gen_ts() generated timeseries directory
- **grid_dict**: the grid info; default: {'atm': 'ne30pg3', 'ocn': 'g16'}



```
dirpath = '/glade/campaign/cesm/development/cross-wg/diagnostic_framework/x4c/timeseries/b.e30_beta06.B1850C_LTso.ne30_t232_wgx3.192.wrkflw.1_32'
case = x4c.Timeseries(
    dirpath,
    casename='b.e30_beta06.B1850C_LTso.ne30_t232_wgx3.192.wrkflw.1',
    grid_dict={'atm': 'ne30gp3'},
)
```

```
>>> case.root_dir: /glade/campaign/cesm/development/cross-wg/diagnostic_framework/x4c/timeseries/b.e30_beta06.B1850C_LTso.ne30_t232_wgx3.192.wrkflw.1_32
>>> case.path_pattern: comp/proc/tseries/*/casename.hstr.vn.timespan.nc
>>> case.grid_dict: {'atm': 'ne30gp3', 'ocn': 'g16', 'lnd': 'ne30gp3', 'rof': 'ne30gp3', 'ice': 'g16'}
>>> case.casename: b.e30_beta06.B1850C_LTso.ne30_t232_wgx3.192.wrkflw.1
>>> case.paths["atm"]["cam.h0a"] created
>>> case.paths["atm"]["cam.h2a"] created
>>> case.paths["atm"]["cam.h1a"] created
>>> case.paths["ocn"]["mom6.h.sfc"] created
>>> case.paths["ocn"]["mom6.h.z"] created
>>> case.paths["ocn"]["mom6"] created
>>> case.paths["ocn"]["mom6.h.rho2"] created
>>> case.paths["ocn"]["mom6.h.native"] created
>>> case.paths["lnd"]["clm2.h0"] created
>>> case.paths["rof"]["mosart.h0"] created
>>> case.paths["ice"]["cice.h"] created
>>> case.paths["ice"]["cice.h1"] created
>>> case.vns["atm"]["cam.h0a"] created
>>> case.vns["atm"]["cam.h2a"] created
>>> case.vns["atm"]["cam.h1a"] created
>>> case.vns["ocn"]["mom6.h.sfc"] created
>>> case.vns["ocn"]["mom6.h.z"] created
>>> case.vns["ocn"]["mom6"] created
>>> case.vns["ocn"]["mom6.h.rho2"] created
>>> case.vns["ocn"]["mom6.h.native"] created
>>> case.vns["lnd"]["clm2.h0"] created
>>> case.vns["rof"]["mosart.h0"] created
>>> case.vns["ice"]["cice.h"] created
>>> case.vns["ice"]["cice.h1"] created
```

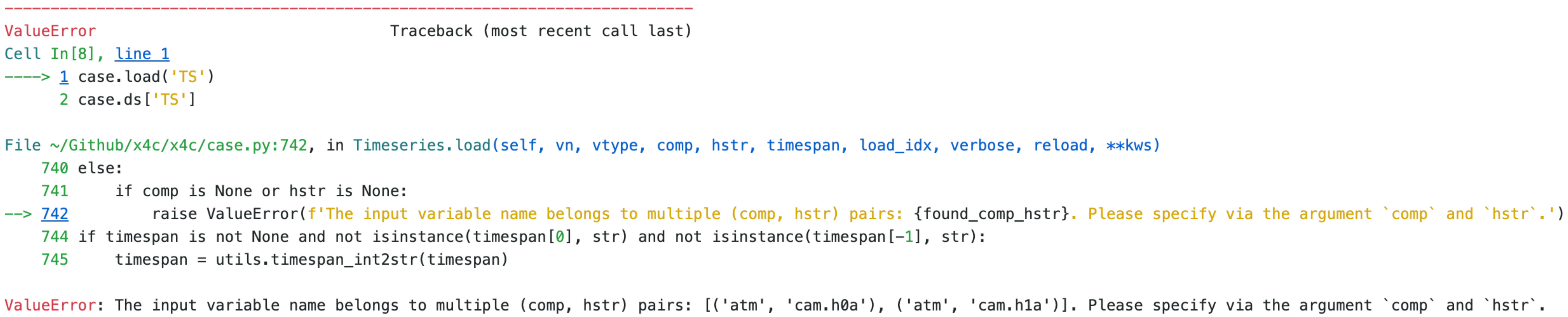
▣ Diagnostic Features (Timeseries)

Timeseries.load(raw)

case.load(vn, [timespan=..., load_idx=-1])

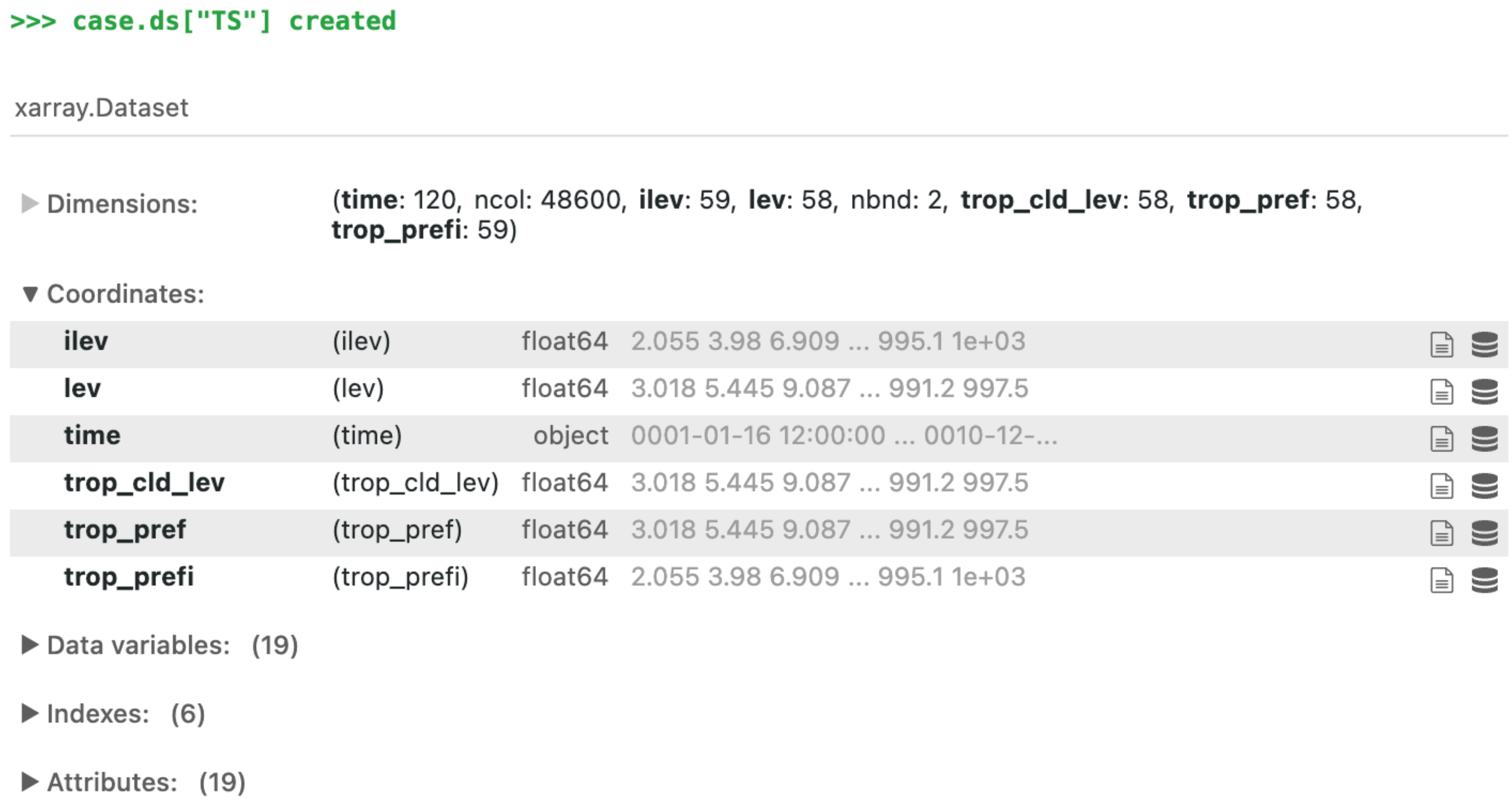
- vn: an existing variable name under the timeseries directory
- timespan: a timespan in (start, end) pointing to a single or multiple files
- load_idx: the file index to load; default: -1 (the last file)

```
case.load('TS')
case.ds['TS']
```



```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[8], line 1
----> 1 case.load('TS')
      2 case.ds['TS']

File ~/Github/x4c/x4c/case.py:742, in Timeseries.load(self, vn, vtype, comp, hstr, timespan, load_idx, verbose, reload, **kws)
    740 else:
    741     if comp is None or hstr is None:
--> 742         raise ValueError(f'The input variable name belongs to multiple (comp, hstr) pairs: {found_comp_hstr}. Please specify via the argument `comp` and `hstr`.')
    744 if timespan is not None and not isinstance(timespan[0], str) and not isinstance(timespan[-1], str):
    745     timespan = utils.timespan_int2str(timespan)

ValueError: The input variable name belongs to multiple (comp, hstr) pairs: [('atm', 'cam.h0a'), ('atm', 'cam.h1a')]. Please specify via the argument `comp` and `hstr`.
```

```
case.load('TS', comp='atm', hstr='cam.h0a')
case.ds['TS']
```



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

xarray.Dataset

▶ Dimensions:      (time: 120, ncol: 48600, ilev: 59, lev: 58, nbnd: 2, trop_cld_lev: 58, trop_pref: 58,
                   trop_prefi: 59)

▼ Coordinates:
    ilev           (ilev)          float64  2.055 3.98 6.909 ... 995.1 1e+03
    lev            (lev)           float64  3.018 5.445 9.087 ... 991.2 997.5
    time           (time)           object  0001-01-16 12:00:00 ... 0010-12-...
    trop_cld_lev   (trop_cld_lev)  float64  3.018 5.445 9.087 ... 991.2 997.5
    trop_pref      (trop_pref)     float64  3.018 5.445 9.087 ... 991.2 997.5
    trop_prefi     (trop_prefi)    float64  2.055 3.98 6.909 ... 995.1 1e+03

▶ Data variables:  (19)

▶ Indexes:  (6)

▶ Attributes:  (19)
```

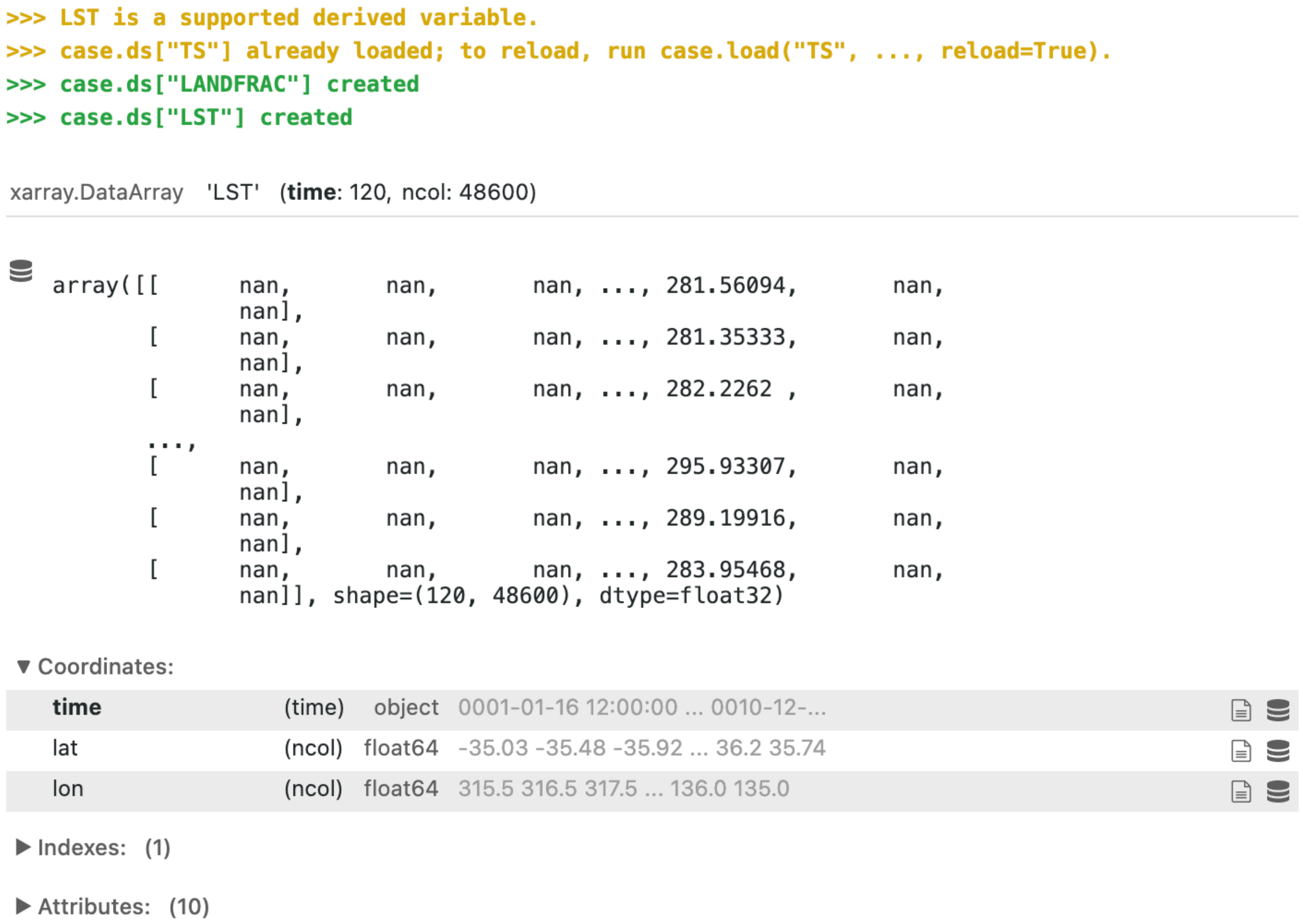
▣ Diagnostic Features (Timeseries)

Timeseries.load(derived)

case.load(vn, [timespan=..., load_idx=-1])

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

```
case.load('LST', comp='atm', hstr='cam.h0a')
case.ds['LST']
```



```
>>> LST is a supported derived variable.
>>> case.ds["TS"] already loaded; to reload, run case.load("TS", ..., reload=True).
>>> case.ds["LANDFRAC"] created
>>> case.ds["LST"] created
```

xarray.DataArray 'LST' (**time**: 120, ncol: 48600)

```
array([[        nan,         nan,         nan, ..., 281.56094,         nan,
                nan],
       [        nan,         nan,         nan, ..., 281.35333,         nan,
                nan],
       [        nan,         nan,         nan, ..., 282.2262 ,         nan,
                nan],
       ...,
       [        nan,         nan,         nan, ..., 295.93307,         nan,
                nan],
       [        nan,         nan,         nan, ..., 289.19916,         nan,
                nan],
       [        nan,         nan,         nan, ..., 283.95468,         nan,
                nan]], shape=(120, 48600), dtype=float32)
```

▼ Coordinates:

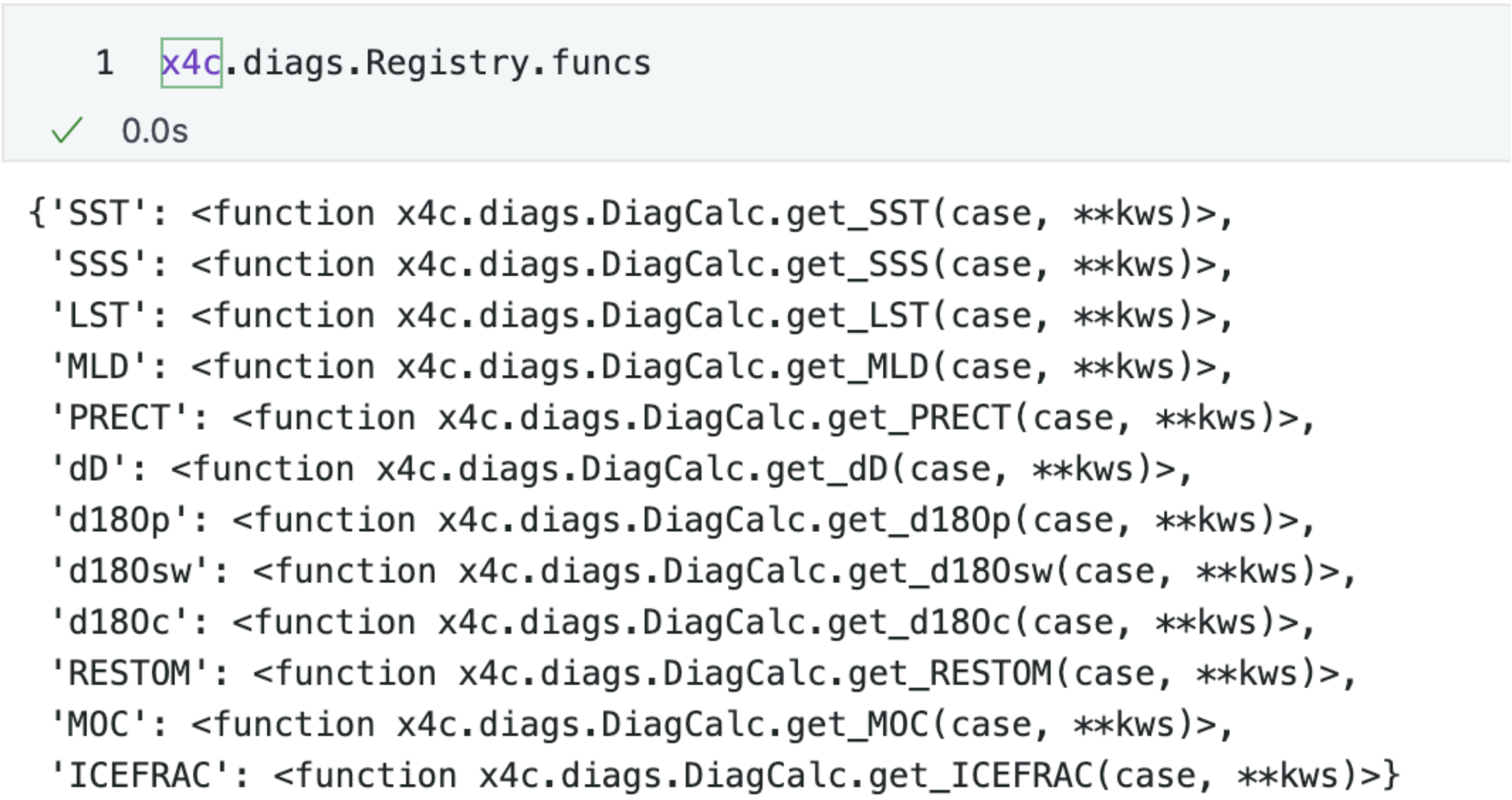
time	(time)	object	0001-01-16 12:00:00 ... 0010-12-...
lat	(ncol)	float64	-35.03 -35.48 -35.92 ... 36.2 35.74
lon	(ncol)	float64	315.5 316.5 317.5 ... 136.0 135.0

▶ Indexes: (1)

▶ Attributes: (10)

Diagnostic Features (Timeseries)

Timeseries.load(derived)

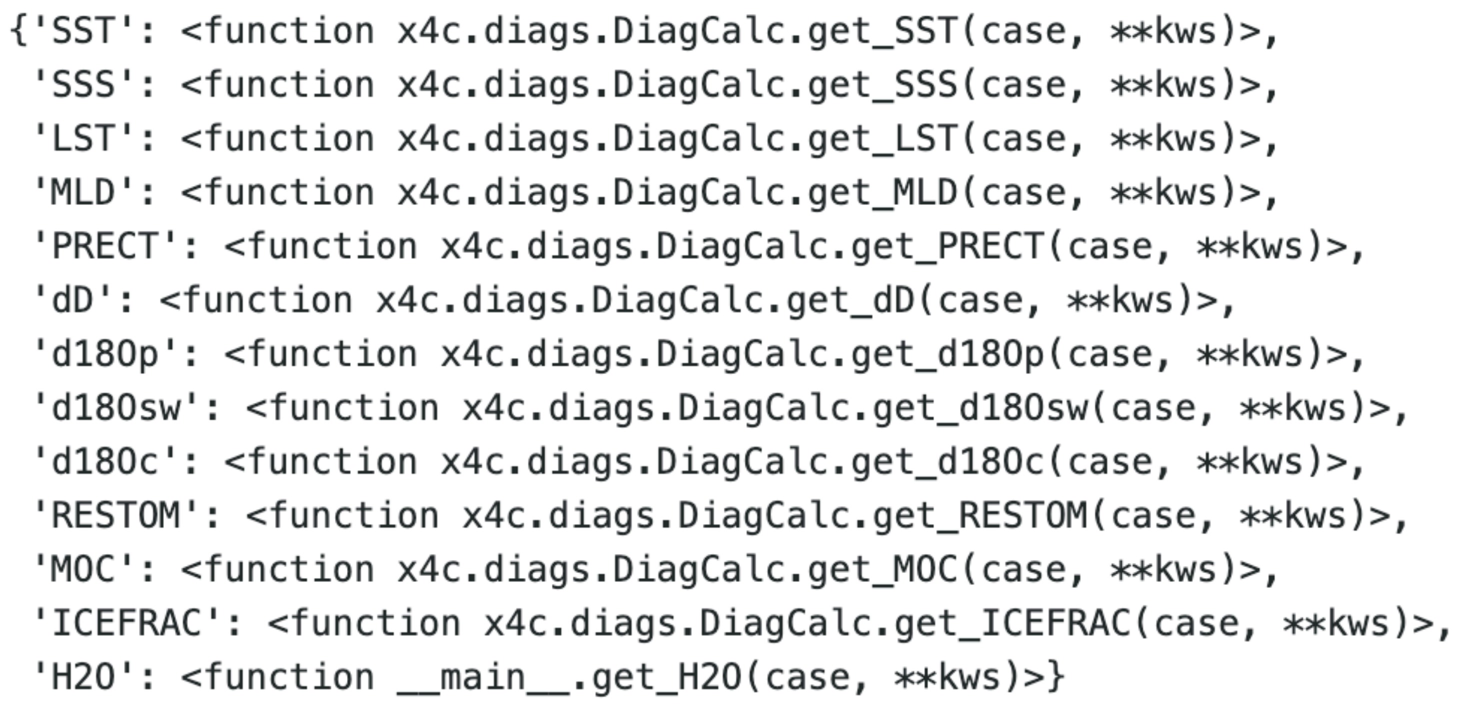


```python
x4c.diags.Registry.funcs
```

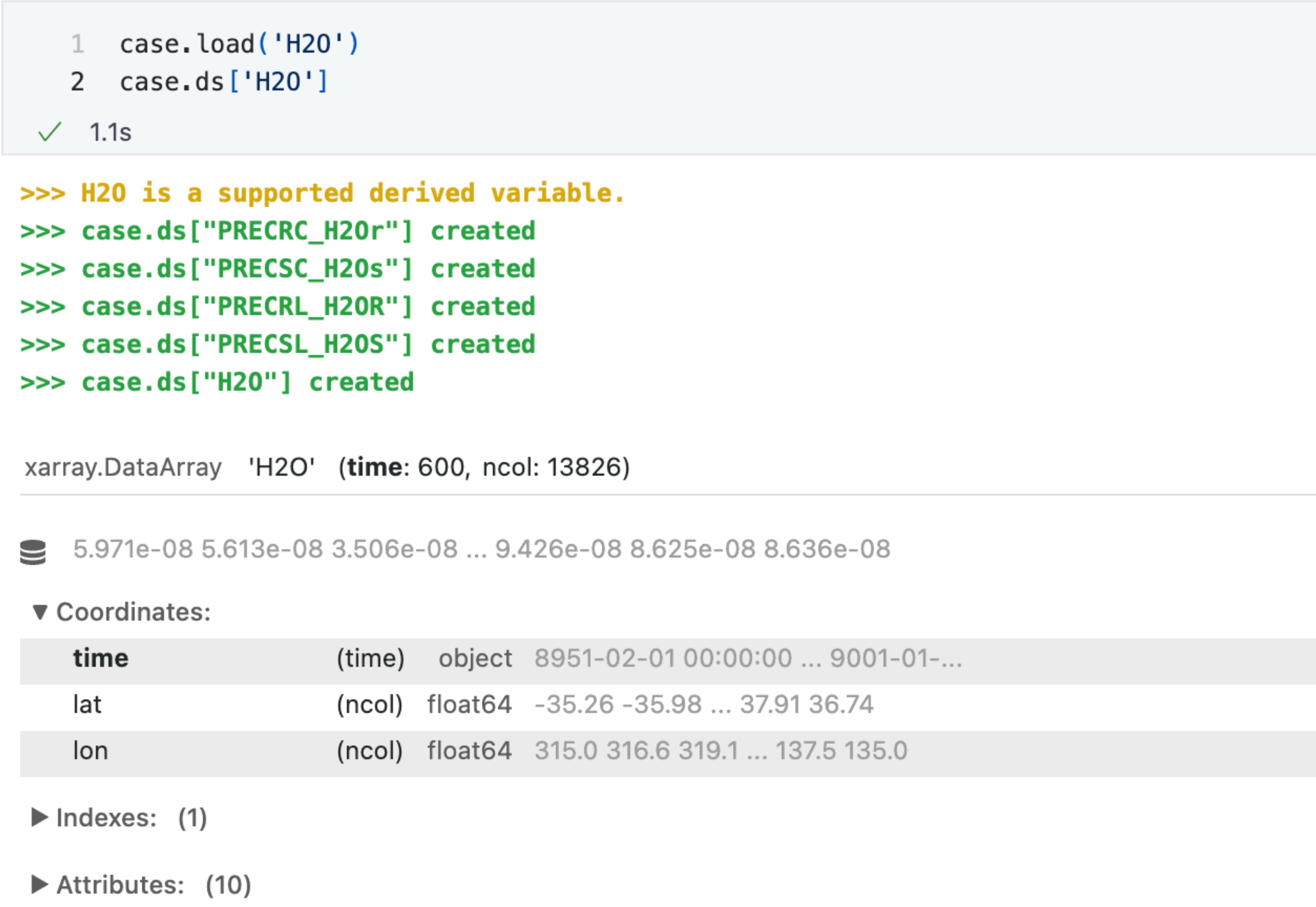
✓ 0.0s

```
{'SST': <function x4c.diags.DiagCalc.get_SST(case, **kws)>,
 'SSS': <function x4c.diags.DiagCalc.get_SSS(case, **kws)>,
 'LST': <function x4c.diags.DiagCalc.get_LST(case, **kws)>,
 'MLD': <function x4c.diags.DiagCalc.get_MLD(case, **kws)>,
 'PRECT': <function x4c.diags.DiagCalc.get_PRECT(case, **kws)>,
 'dD': <function x4c.diags.DiagCalc.get_dD(case, **kws)>,
 'd18Op': <function x4c.diags.DiagCalc.get_d18Op(case, **kws)>,
 'd18Osw': <function x4c.diags.DiagCalc.get_d18Osw(case, **kws)>,
 'd18Oc': <function x4c.diags.DiagCalc.get_d18Oc(case, **kws)>,
 'RESTOM': <function x4c.diags.DiagCalc.get_RESTOM(case, **kws)>,
 'MOC': <function x4c.diags.DiagCalc.get_MOC(case, **kws)>,
 'ICEFRAC': <function x4c.diags.DiagCalc.get_ICEFRAC(case, **kws)>}
```

```python
@x4c.diags.F
def get_H2O(case, **kws):
    case.load('PRECRC_H2Or', **kws)
    case.load('PRECSC_H2Os', **kws)
    case.load('PRECRL_H2OR', **kws)
    case.load('PRECSL_H2OS', **kws)
    h2o = case.ds['PRECRC_H2Or'].x.da + case.ds['PRECSC_H2Os'].x.da + case.ds['PRECRL_H2OR'].x.da + case.ds['PRECSL_H2OS'].x.da
    h2o = h2o.where(h2o > 1e-18, 1e-18)
    h2o.name = 'H2O'
    h2o.attrs['long_name'] = 'H2O'
    h2o.attrs['units'] = 'kg m-2 s-1'
    return h2o

x4c.diags.Registry.funcs
```



```
{'SST': <function x4c.diags.DiagCalc.get_SST(case, **kws)>,
 'SSS': <function x4c.diags.DiagCalc.get_SSS(case, **kws)>,
 'LST': <function x4c.diags.DiagCalc.get_LST(case, **kws)>,
 'MLD': <function x4c.diags.DiagCalc.get_MLD(case, **kws)>,
 'PRECT': <function x4c.diags.DiagCalc.get_PRECT(case, **kws)>,
 'dD': <function x4c.diags.DiagCalc.get_dD(case, **kws)>,
 'd18Op': <function x4c.diags.DiagCalc.get_d18Op(case, **kws)>,
 'd18Osw': <function x4c.diags.DiagCalc.get_d18Osw(case, **kws)>,
 'd18Oc': <function x4c.diags.DiagCalc.get_d18Oc(case, **kws)>,
 'RESTOM': <function x4c.diags.DiagCalc.get_RESTOM(case, **kws)>,
 'MOC': <function x4c.diags.DiagCalc.get_MOC(case, **kws)>,
 'ICEFRAC': <function x4c.diags.DiagCalc.get_ICEFRAC(case, **kws)>,
 'H2O': <function __main__.get_H2O(case, **kws)>}
```



```python
case.load('H2O')
case.ds['H2O']
```

✓ 1.1s

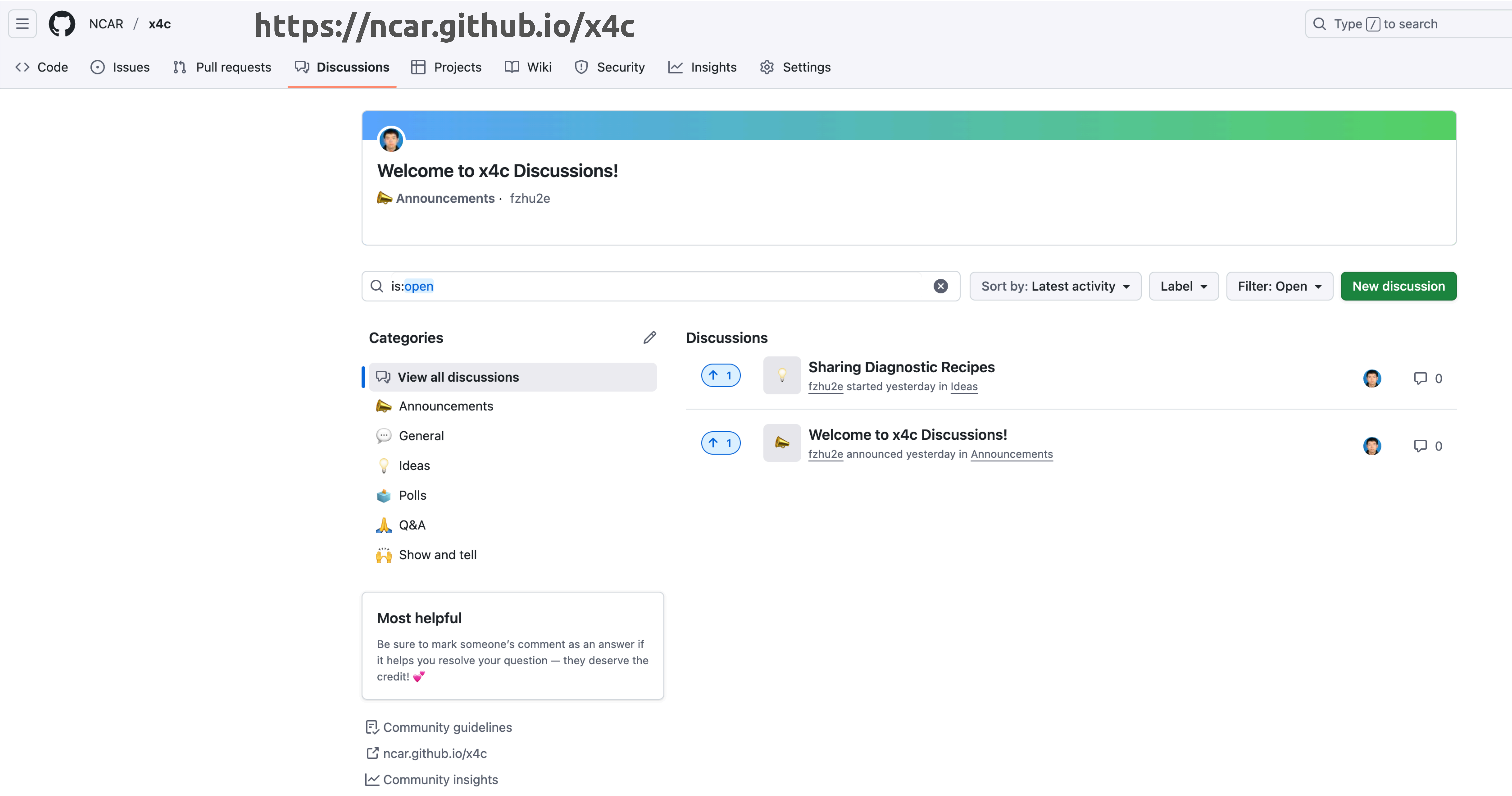
```
>>> H2O is a supported derived variable.
>>> case.ds["PRECRC_H2Or"] created
>>> case.ds["PRECSC_H2Os"] created
>>> case.ds["PRECRL_H2OR"] created
>>> case.ds["PRECSL_H2OS"] created
>>> case.ds["H2O"] created
```

```
xarray.DataArray  'H2O'  (time: 600, ncol: 13826)

5.971e-08 5.613e-08 3.506e-08 ... 9.426e-08 8.625e-08 8.636e-08

▼ Coordinates:
   time      (time)  object   8951-02-01 00:00:00 ... 9001-01-...
   lat       (ncol)  float64  -35.26 -35.98 ... 37.91 36.74
   lon       (ncol)  float64  315.0 316.6 319.1 ... 137.5 135.0

▶ Indexes:  (1)

▶ Attributes:  (10)
```

Diagnostic Features (Timeseries)

Contribution from the Community

https://ncar.github.io/x4c



NCAR / x4c

Type / to search

Code · Issues · Pull requests · **Discussions** · Projects · Wiki · Security · Insights · Settings

Welcome to x4c Discussions!

📣 Announcements · fzhu2e

is:open

Sort by: Latest activity ▾ | Label ▾ | Filter: Open ▾ | New discussion

Categories

- **View all discussions**
- 📣 Announcements
- 💬 General
- 💡 Ideas
- 🗳️ Polls
- 🙏 Q&A
- 🙌 Show and tell

Discussions

↑ 1	**Sharing Diagnostic Recipes** fzhu2e started yesterday in Ideas	0
↑ 1	**Welcome to x4c Discussions!** fzhu2e announced yesterday in Announcements	0

Most helpful

Be sure to mark someone's comment as an answer if it helps you resolve your question — they deserve the credit! 💕

Community guidelines

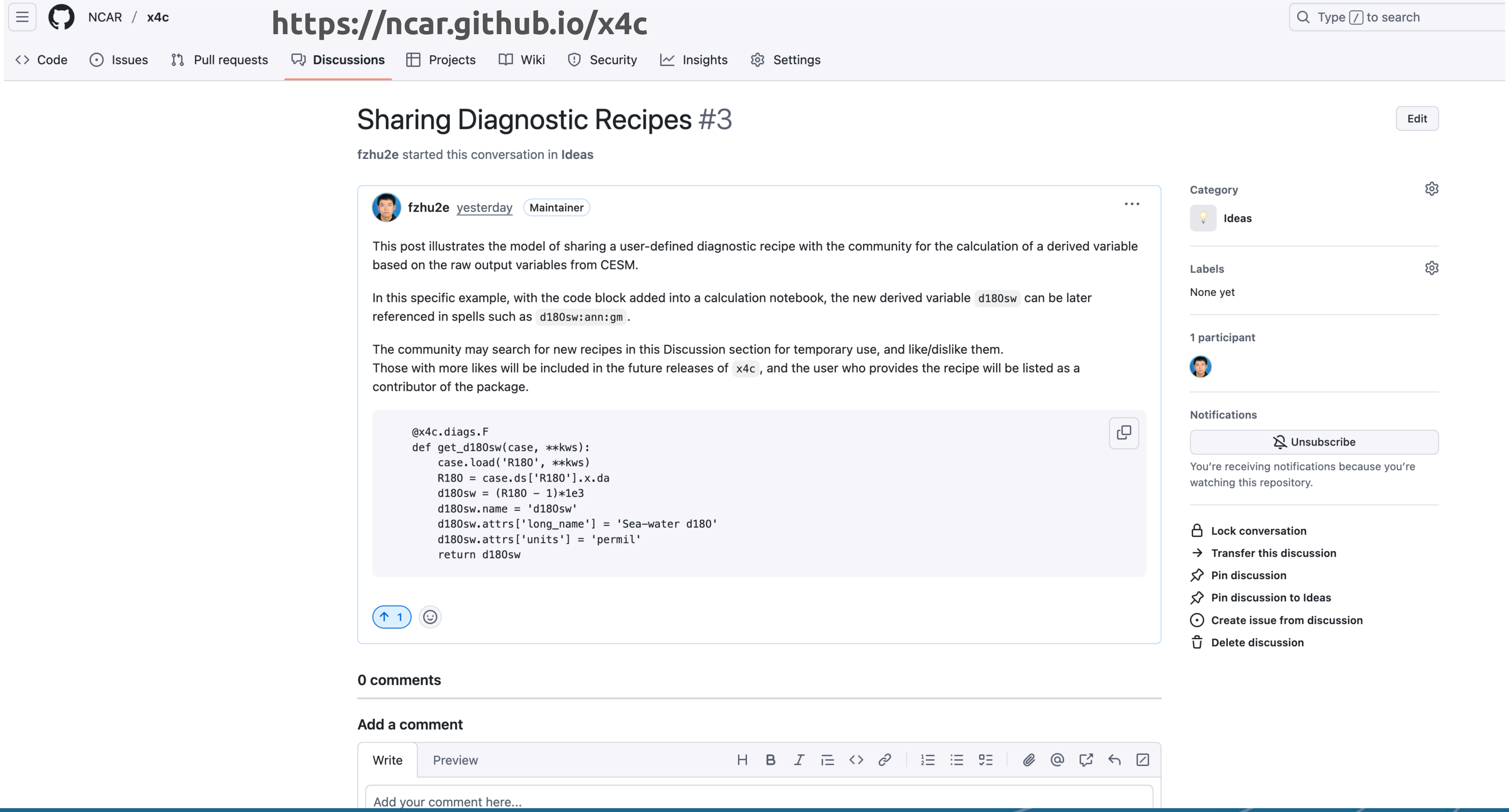
ncar.github.io/x4c

Community insights

Diagnostic Features (Timeseries)

Contribution from the Community

https://ncar.github.io/x4c



Sharing Diagnostic Recipes #3

fzhu2e started this conversation in **Ideas**

fzhu2e yesterday Maintainer

This post illustrates the model of sharing a user-defined diagnostic recipe with the community for the calculation of a derived variable based on the raw output variables from CESM.

In this specific example, with the code block added into a calculation notebook, the new derived variable `d18Osw` can be later referenced in spells such as `d18Osw:ann:gm`.

The community may search for new recipes in this Discussion section for temporary use, and like/dislike them.
Those with more likes will be included in the future releases of `x4c`, and the user who provides the recipe will be listed as a contributor of the package.

```
@x4c.diags.F
def get_d18Osw(case, **kws):
    case.load('R18O', **kws)
    R18O = case.ds['R18O'].x.da
    d18Osw = (R18O - 1)*1e3
    d18Osw.name = 'd18Osw'
    d18Osw.attrs['long_name'] = 'Sea-water d18O'
    d18Osw.attrs['units'] = 'permil'
    return d18Osw
```

0 comments

Add a comment

Add your comment here...

Category

Ideas

Labels

None yet

1 participant

Notifications

You're receiving notifications because you're watching this repository.

▣ Diagnostic Features (Timeseries)

A “Spell” System

Idea: to summarize a series of data processing steps with a spell that can be calculated and plotted

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

- vn: diagnostic variable name
- ann: annualization method
- sa: spatial averaging method

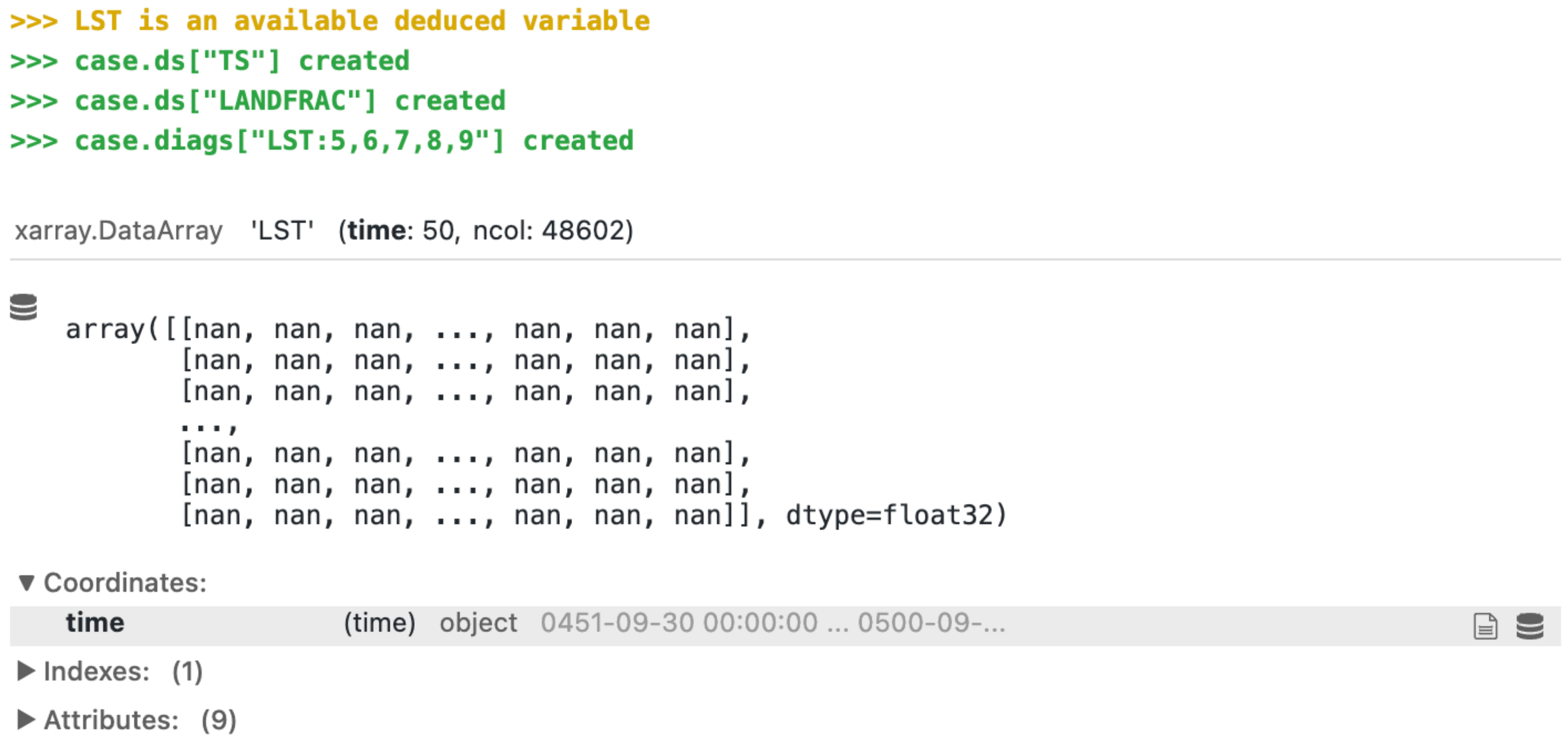
▣ Diagnostic Features (Timeseries)

A “Spell” System

Given a CESM simulation with an SE dycore (ne30), plot:

- **a map of the MJJAS Land Surface Temperature**

```
spell = 'LST:5,6,7,8,9'
case.calc(spell)
case.diags[spell]
```

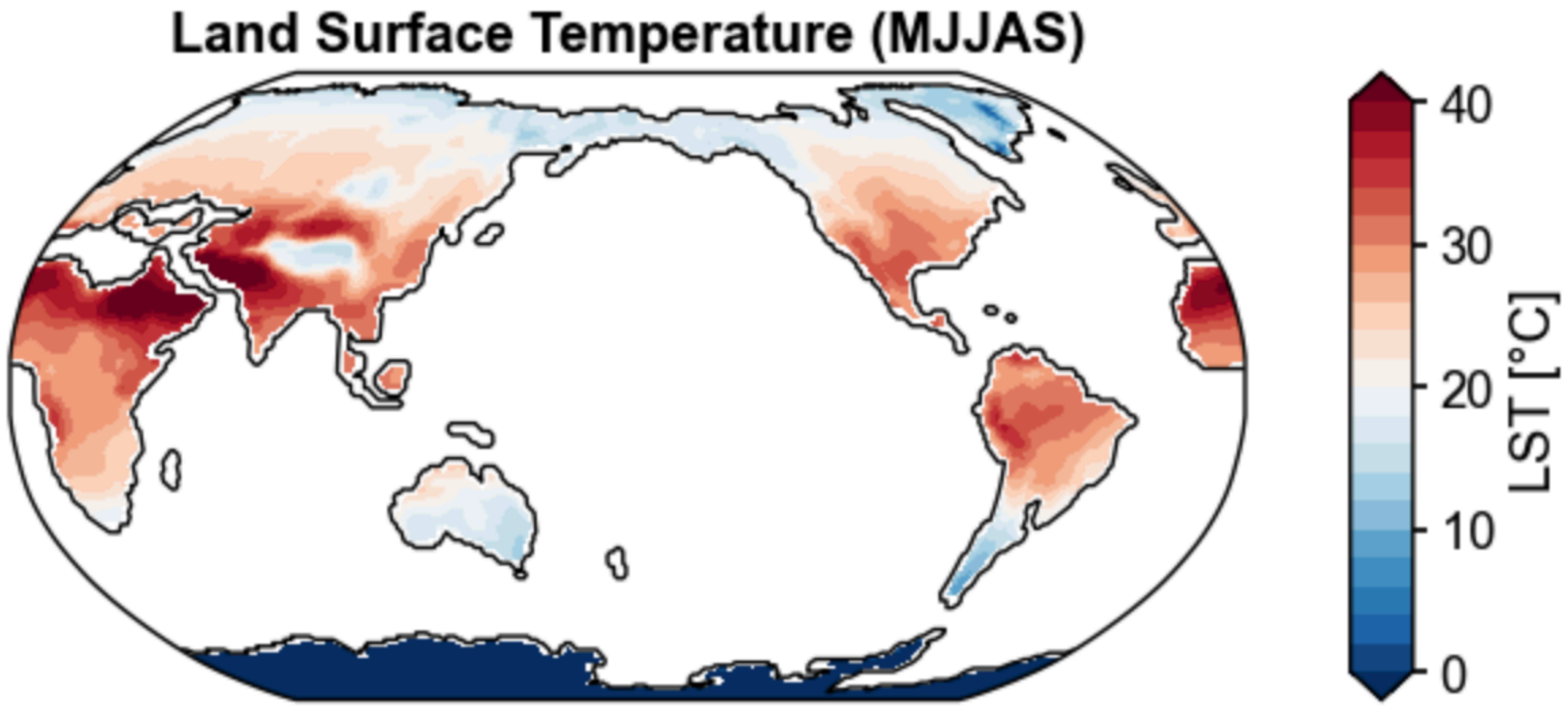


```
>>> LST is an available deduced variable
>>> case.ds["TS"] created
>>> case.ds["LANDFRAC"] created
>>> case.diags["LST:5,6,7,8,9"] created
```

```
xarray.DataArray   'LST'   (time: 50, ncol: 48602)

array([[nan, nan, nan, ..., nan, nan, nan],
       [nan, nan, nan, ..., nan, nan, nan],
       [nan, nan, nan, ..., nan, nan, nan],
       ...,
       [nan, nan, nan, ..., nan, nan, nan],
       [nan, nan, nan, ..., nan, nan, nan],
       [nan, nan, nan, ..., nan, nan, nan]], dtype=float32)

▼ Coordinates:
   time            (time)  object  0451-09-30 00:00:00 ... 0500-09-...
▶ Indexes:  (1)
▶ Attributes:  (9)
```

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

```
>>> case.ds["SSH"] created
```



Land Surface Temperature (MJJAS)

▣ Diagnostic Features (Timeseries)

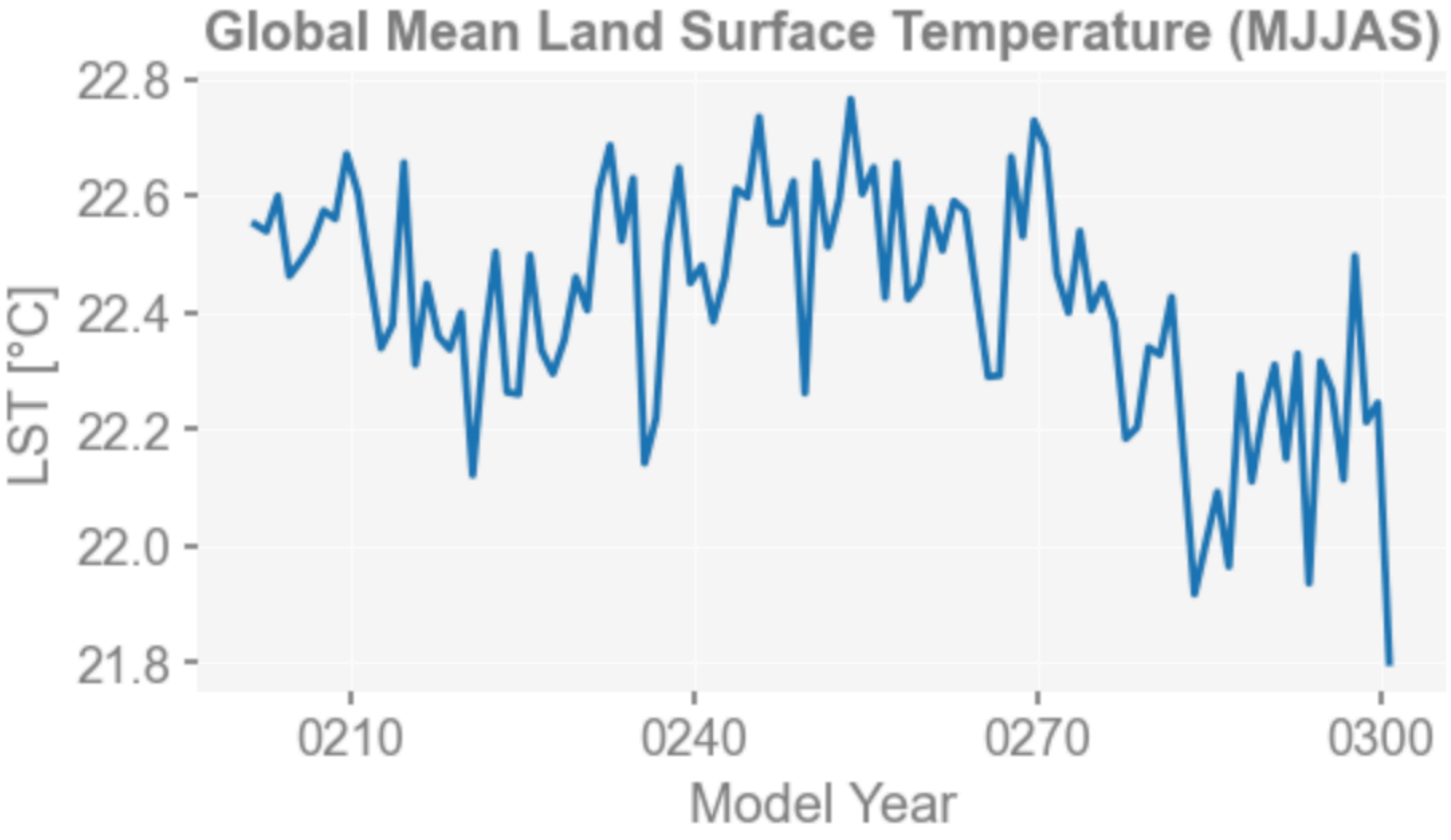
A “Spell” System

Given a CESM simulation with an SE dycore (ne30), plot:

- a **100-yr time series** of its **Global Mean**

```
x4c.set_style('web', font_scale=1.2)

spell = 'LST:5,6,7,8,9:gm'
fig, ax = case.plot(spell, timespan=(201, 300))
```

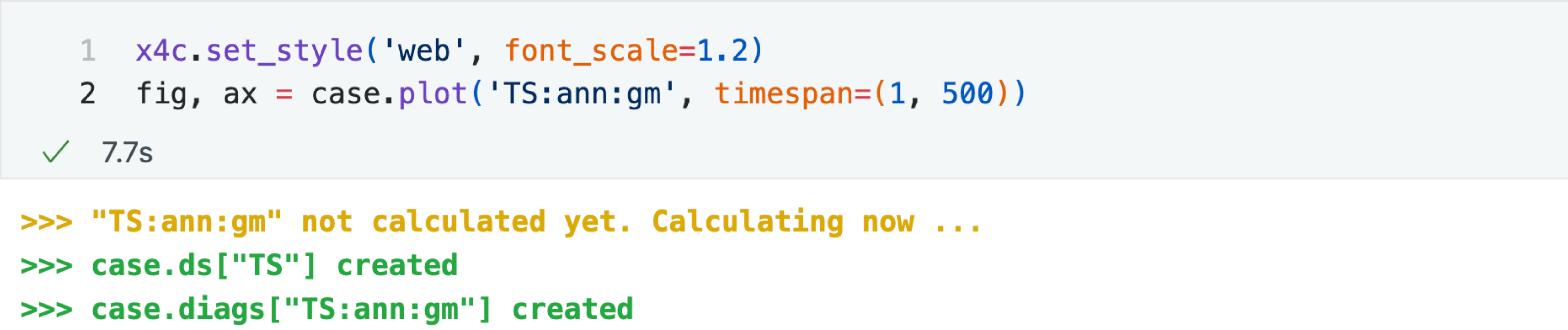
```
>>> "LST:5,6,7,8,9:gm" not calculated yet. Calculating now ...
>>> LST is an available deduced variable
>>> case.ds["TS"] created
>>> case.ds["LANDFRAC"] created
>>> case.diags["LST:5,6,7,8,9:gm"] created
```



Diagnostic Features (Timeseries)

A “Spell” System

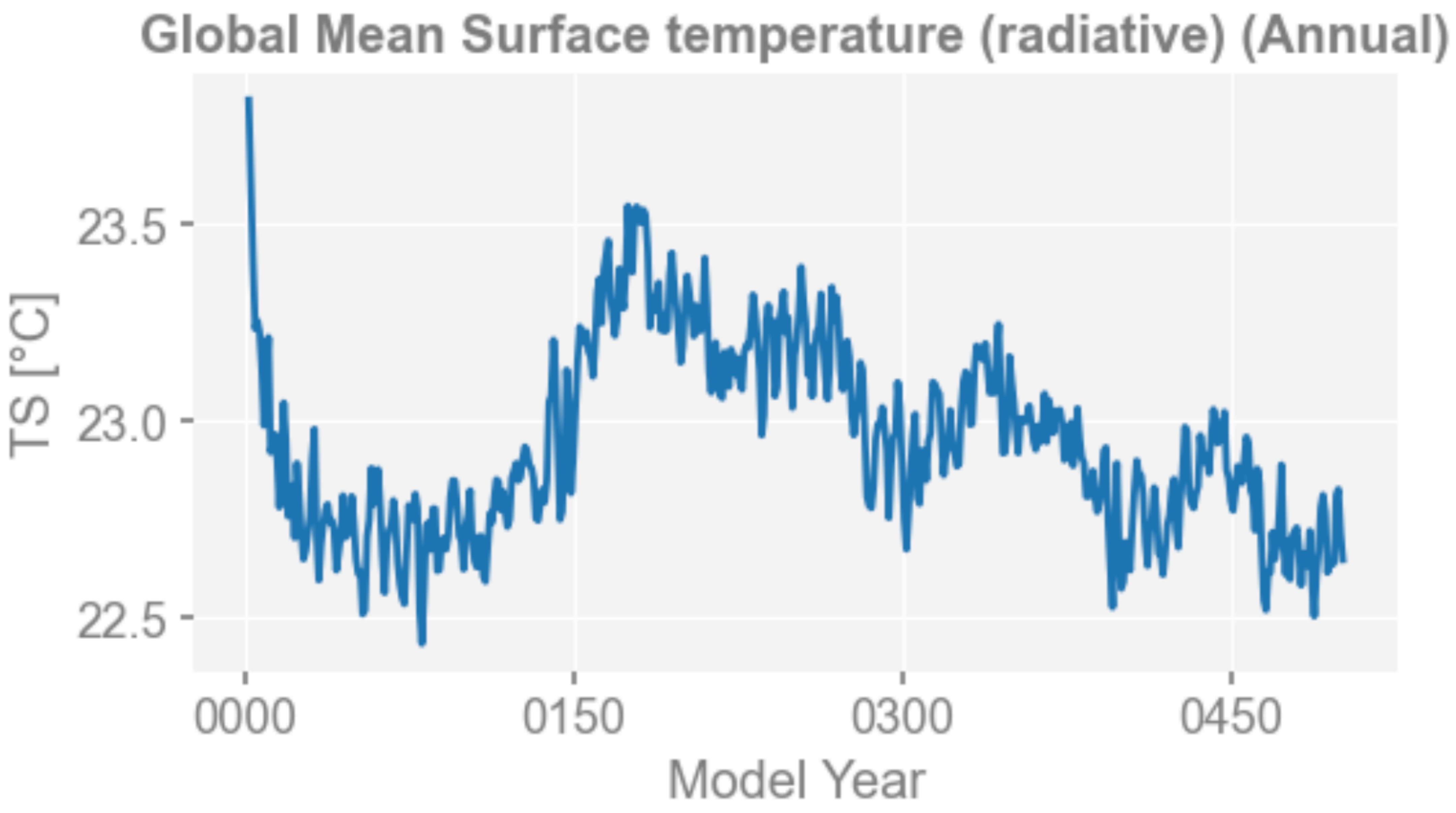
GMST 'TS:ann:gm'



```
x4c.set_style('web', font_scale=1.2)
fig, ax = case.plot('TS:ann:gm', timespan=(1, 500))
```

✓ 7.7s

```
>>> "TS:ann:gm" not calculated yet. Calculating now ...
>>> case.ds["TS"] created
>>> case.diags["TS:ann:gm"] created
```

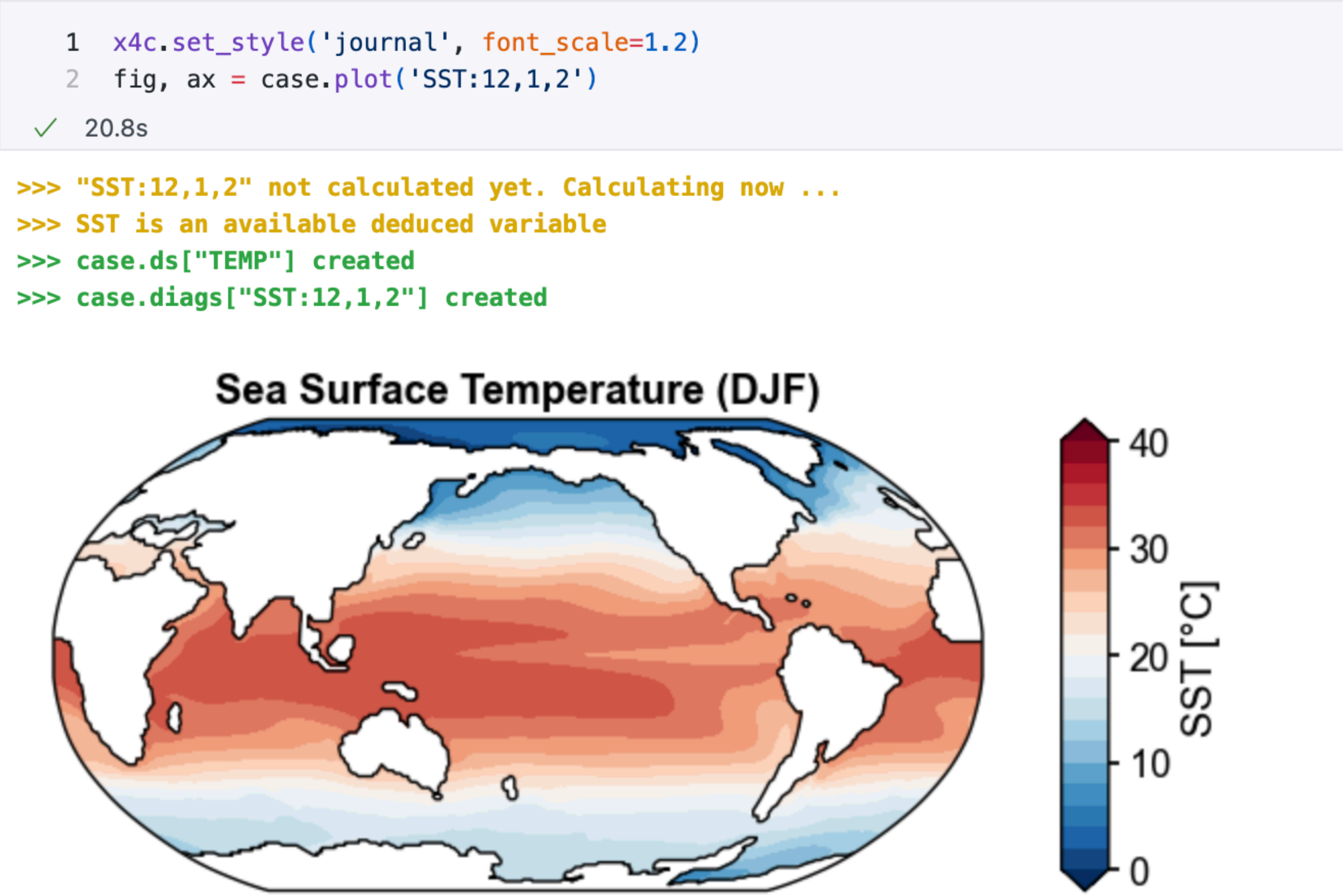


▣ Diagnostic Features (Timeseries)

A “Spell” System

A map of DJF SST

'SST:12,1,2'



```
x4c.set_style('journal', font_scale=1.2)
fig, ax = case.plot('SST:12,1,2')
✓ 20.8s
```

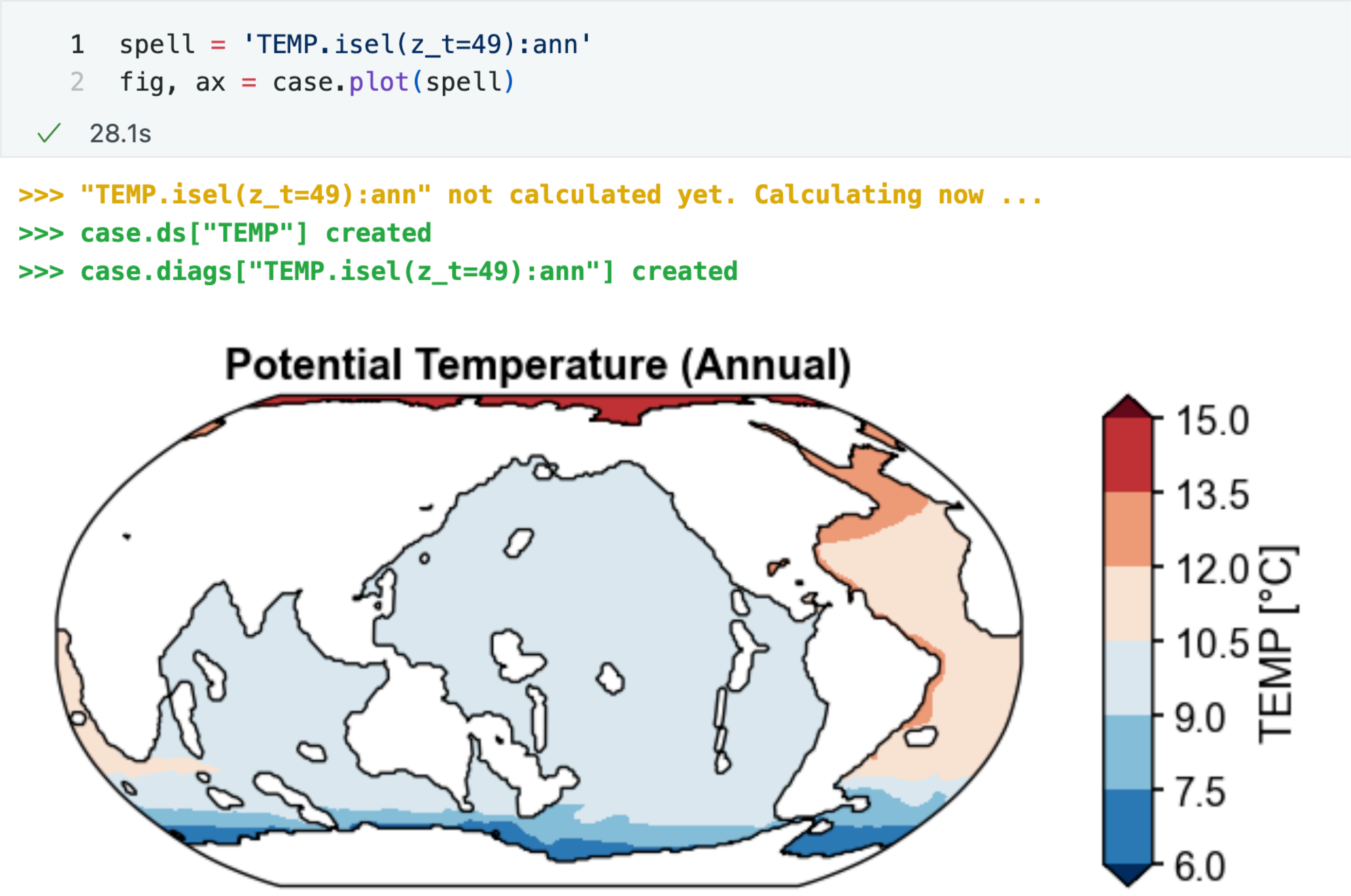
```
>>> "SST:12,1,2" not calculated yet. Calculating now ...
>>> SST is an available deduced variable
>>> case.ds["TEMP"] created
>>> case.diags["SST:12,1,2"] created
```

Diagnostic Features (Timeseries)

A “Spell” System

A map of annual Bottom Water Temperature (z_t=49)

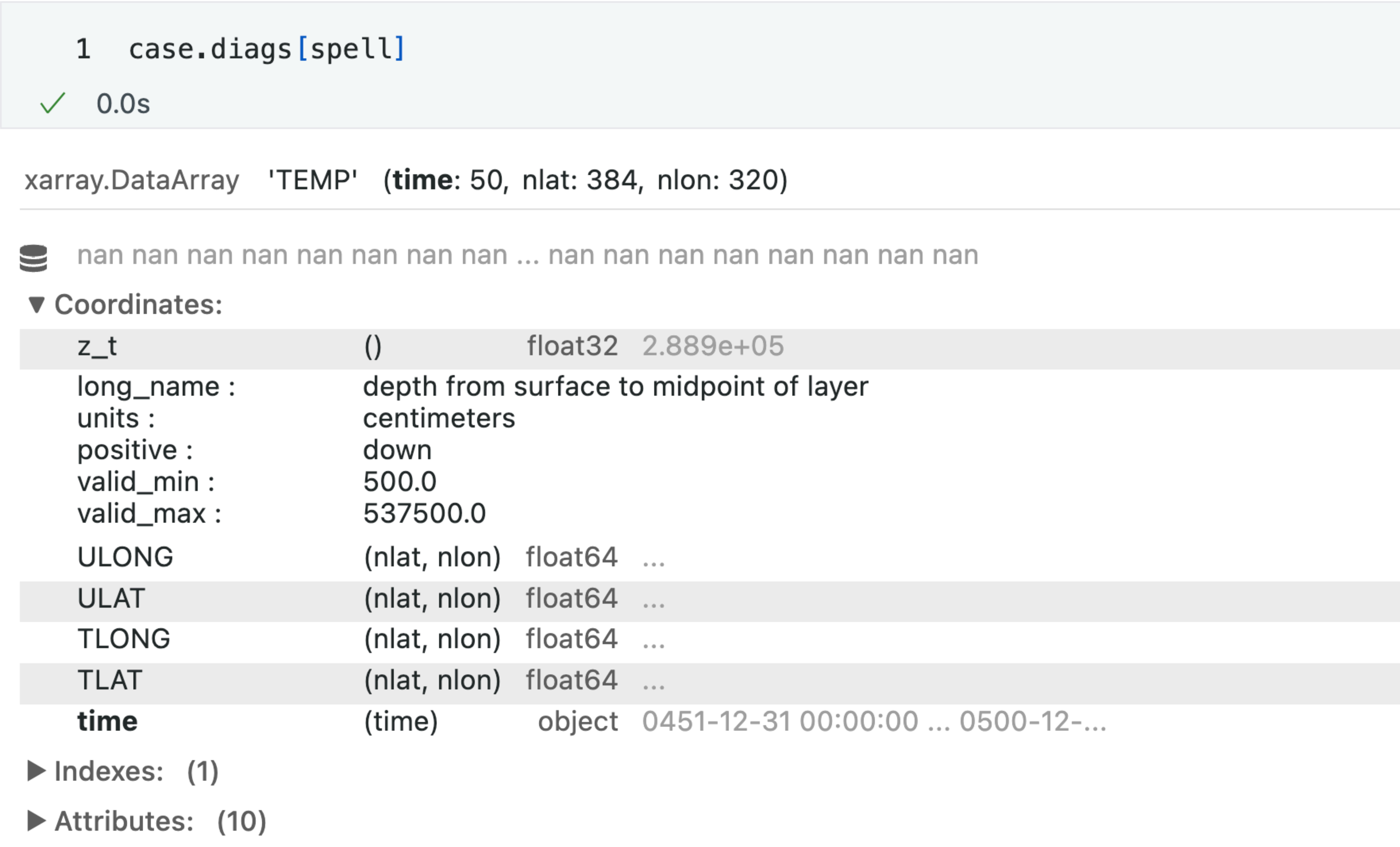
'TEMP.isel(z_t=49):ann'



```
spell = 'TEMP.isel(z_t=49):ann'
fig, ax = case.plot(spell)
```

✓ 28.1s

```
>>> "TEMP.isel(z_t=49):ann" not calculated yet. Calculating now ...
>>> case.ds["TEMP"] created
>>> case.diags["TEMP.isel(z_t=49):ann"] created
```



```
case.diags[spell]
```

✓ 0.0s

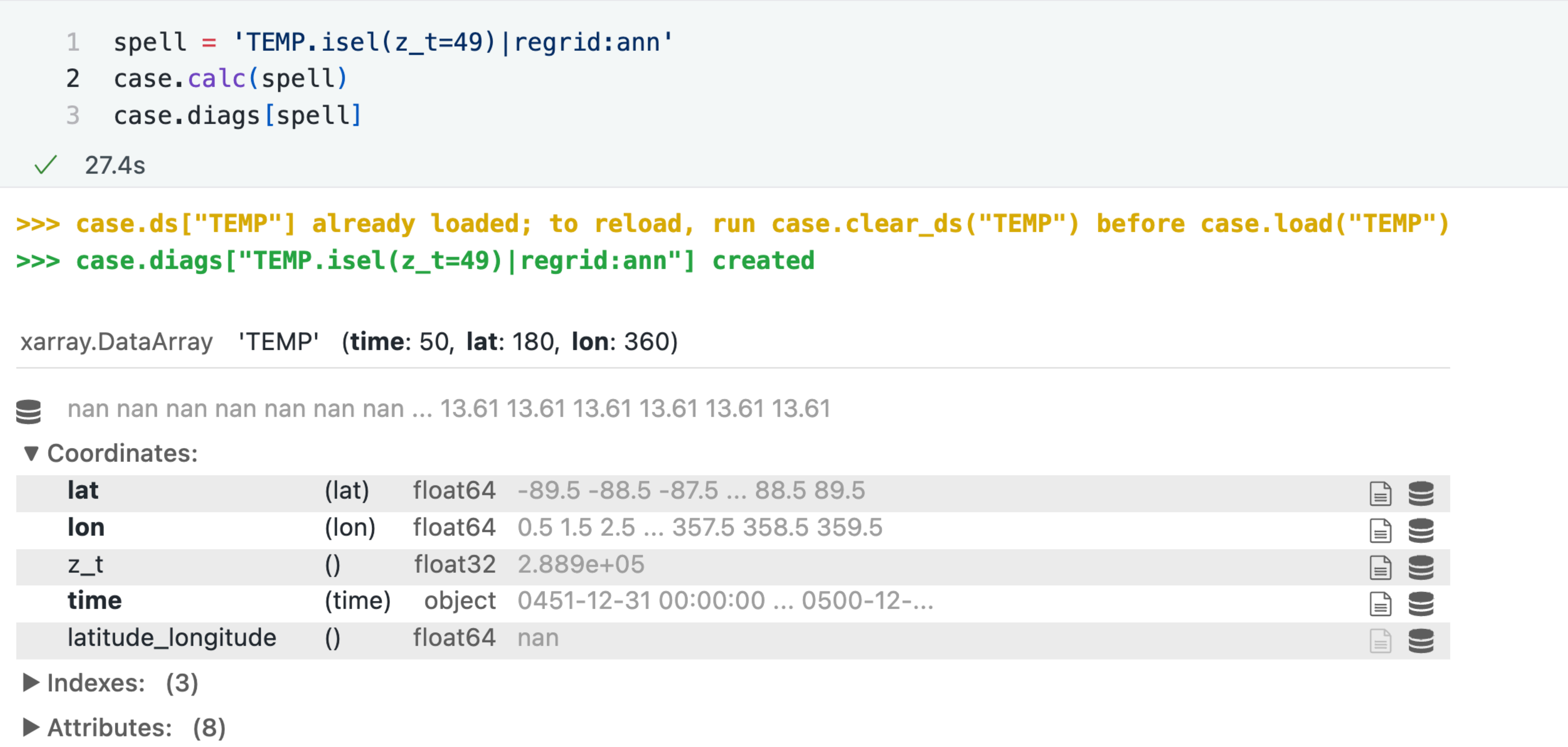
```
xarray.DataArray 'TEMP' (time: 50, nlat: 384, nlon: 320)

nan nan nan nan nan nan nan nan ... nan nan nan nan nan nan nan nan

▼ Coordinates:
   z_t            ()             float32  2.889e+05
   long_name :    depth from surface to midpoint of layer
   units :        centimeters
   positive :     down
   valid_min :    500.0
   valid_max :    537500.0
   ULONG          (nlat, nlon)   float64  ...
   ULAT           (nlat, nlon)   float64  ...
   TLONG          (nlat, nlon)   float64  ...
   TLAT           (nlat, nlon)   float64  ...
   time           (time)         object   0451-12-31 00:00:00 ... 0500-12-...
▶ Indexes: (1)
▶ Attributes: (10)
```

▣ Diagnostic Features (Timeseries)

A “Spell” System

A map of annual Bottom Water Temperature (z_t=49)

'TEMP.isel(z_t=49)|regrid:ann'



```python
spell = 'TEMP.isel(z_t=49)|regrid:ann'
case.calc(spell)
case.diags[spell]
```

✓ 27.4s

```
>>> case.ds["TEMP"] already loaded; to reload, run case.clear_ds("TEMP") before case.load("TEMP")
>>> case.diags["TEMP.isel(z_t=49)|regrid:ann"] created
```

xarray.DataArray 'TEMP' (**time**: 50, **lat**: 180, **lon**: 360)

nan nan nan nan nan nan nan ... 13.61 13.61 13.61 13.61 13.61 13.61

▼ Coordinates:

lat	(lat)	float64	-89.5 -88.5 -87.5 ... 88.5 89.5
lon	(lon)	float64	0.5 1.5 2.5 ... 357.5 358.5 359.5
z_t	()	float32	2.889e+05
time	(time)	object	0451-12-31 00:00:00 ... 0500-12-...
latitude_longitude	()	float64	nan

▶ Indexes: (3)

▶ Attributes: (8)

Diagnostic Features (Timeseries)

A “Spell” System

A map of annual Bottom Water Temperature (z_t=49)

'TEMP.isel(z_t=49)|regrid(dlon=2, dlat=2):ann'

```
spell = 'TEMP.isel(z_t=49)|regrid(dlon=2,dlat=2):ann'
case.calc(spell)
case.diags[spell]
```

✓ 26.4s

```
>>> case.ds["TEMP"] already loaded; to reload, run case.clear_ds("TEMP") before case.load("TEMP")
>>> case.diags["TEMP.isel(z_t=49)|regrid(dlon=2,dlat=2):ann"] created
```

```
xarray.DataArray   'TEMP'   (time: 50, lat: 90, lon: 180)

nan nan nan nan nan nan nan ... 13.61 13.61 13.61 13.61 13.61 13.61

▼ Coordinates:
    lat                  (lat)    float64   -89.0 -87.0 -85.0 ... 87.0 89.0
    lon                  (lon)    float64   1.0 3.0 5.0 ... 355.0 357.0 359.0
    z_t                  ()       float32   2.889e+05
    time                 (time)   object    0451-12-31 00:00:00 ... 0500-12-...
    latitude_longitude   ()       float64   nan
▶ Indexes:   (3)
▶ Attributes:   (8)
```

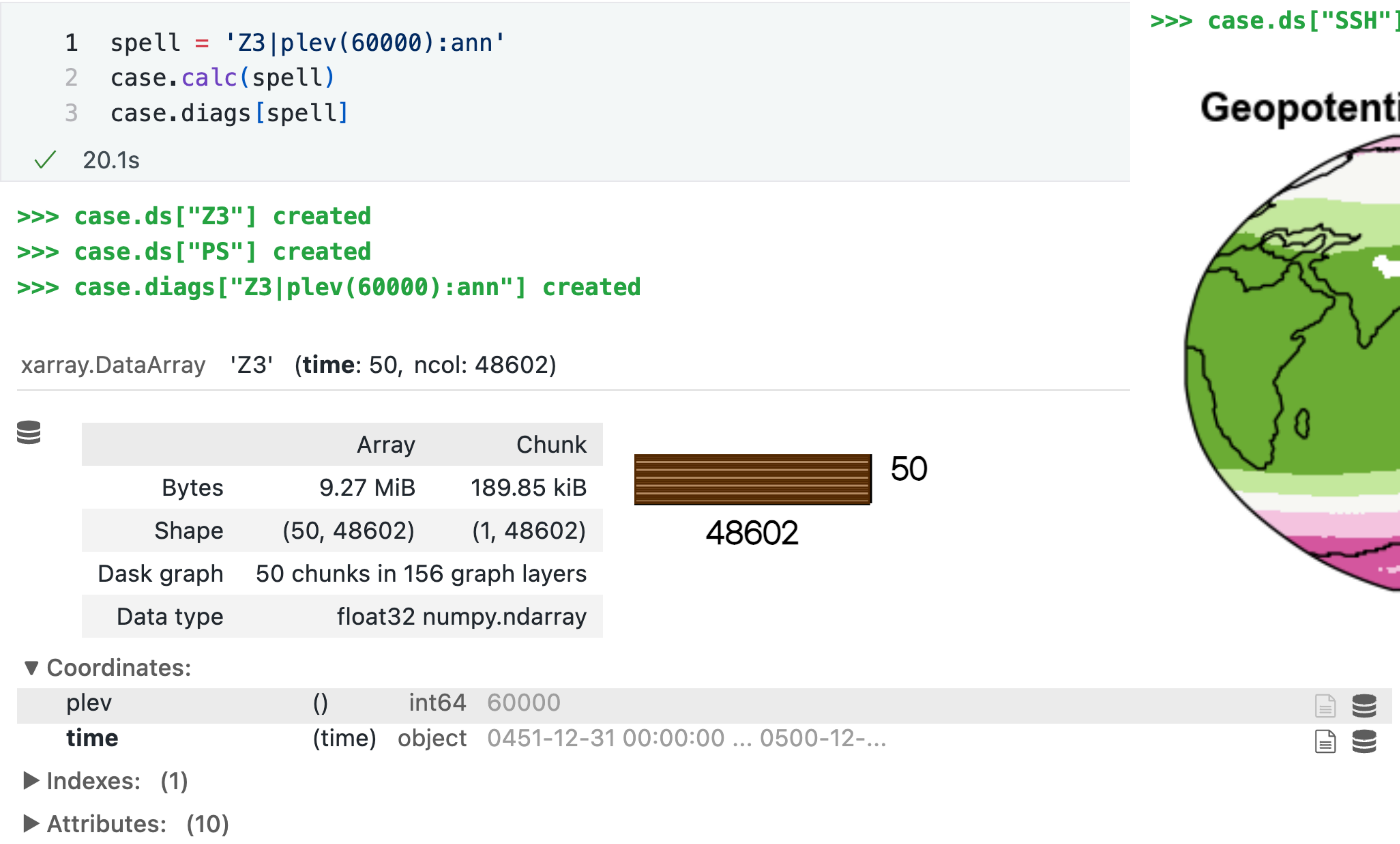
Diagnostic Features (Timeseries)

A “Spell” System

A map of annual 600 hPa Geopotential Height

GeoCAT

'Z3|plev(60000):ann'



```
spell = 'Z3|plev(60000):ann'
case.calc(spell)
case.diags[spell]
```

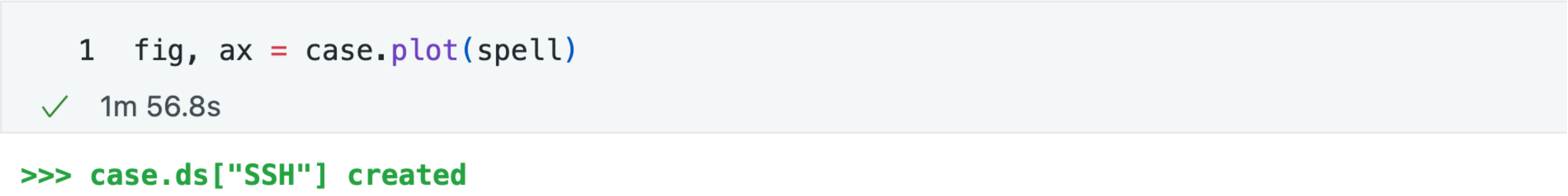
✓ 20.1s

```
>>> case.ds["Z3"] created
>>> case.ds["PS"] created
>>> case.diags["Z3|plev(60000):ann"] created
```

xarray.DataArray 'Z3' (**time**: 50, ncol: 48602)

	Array	Chunk
Bytes	9.27 MiB	189.85 kiB
Shape	(50, 48602)	(1, 48602)
Dask graph	50 chunks in 156 graph layers	
Data type	float32 numpy.ndarray	

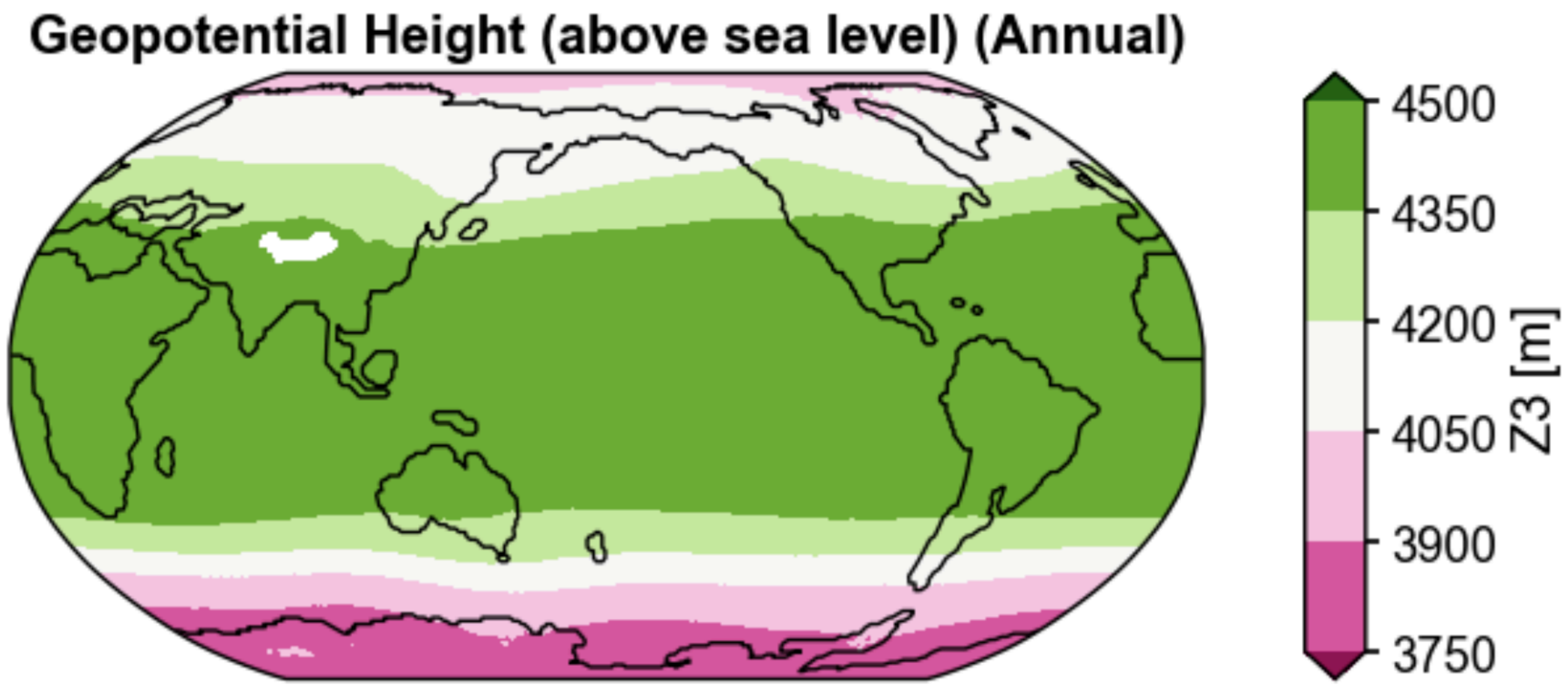
```
▼ Coordinates:
  plev      ()      int64   60000
  time      (time)  object  0451-12-31 00:00:00 ... 0500-12-...
▶ Indexes: (1)
▶ Attributes: (10)
```



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

✓ 1m 56.8s

```
>>> case.ds["SSH"] created
```



Geopotential Height (above sea level) (Annual)

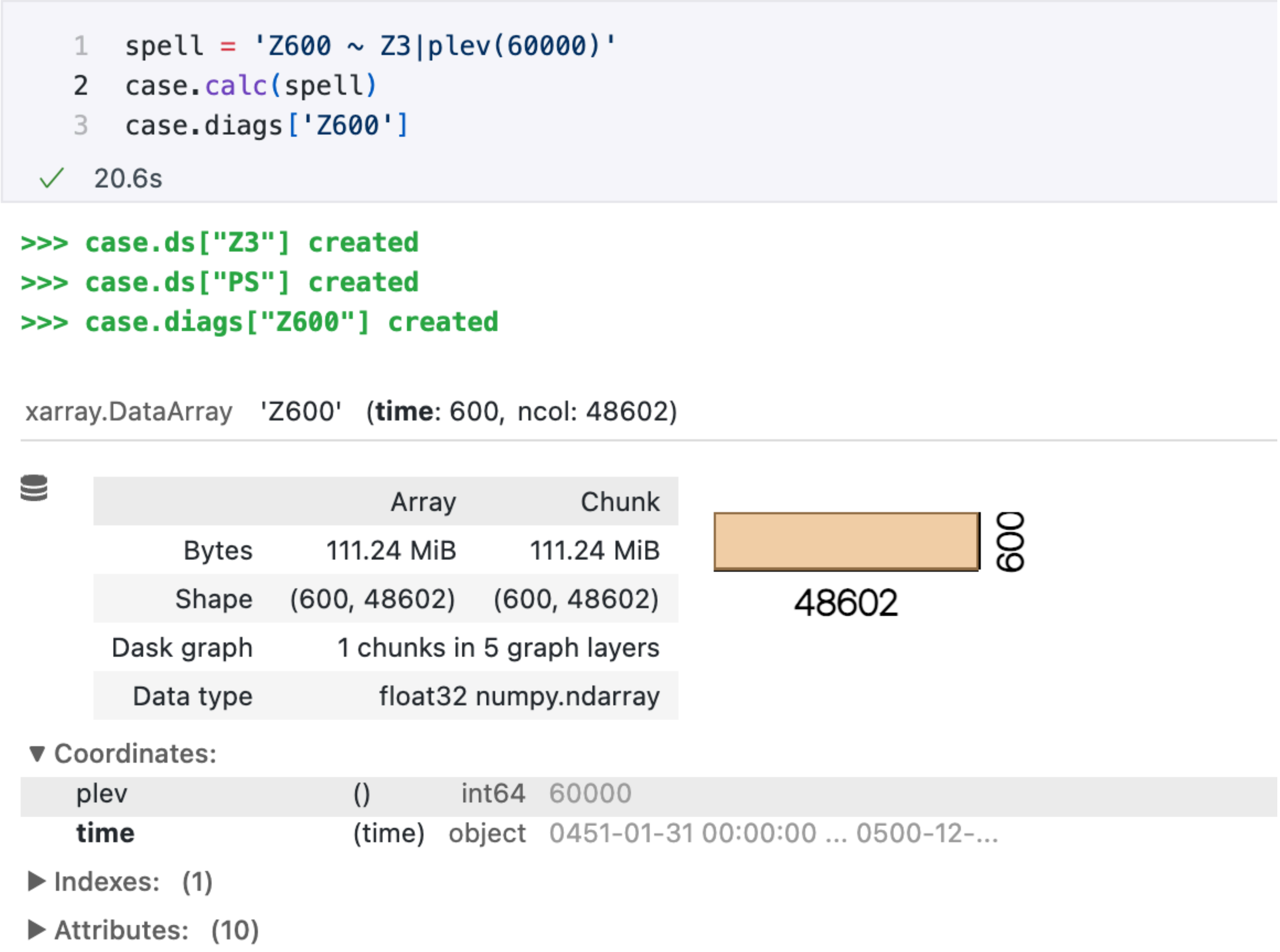
Diagnostic Features (Timeseries)

A “Spell” System

A map of annual 600 hPa Geopotential Height

'Z600 ~ Z3|plev(60000)'

'Z600:ann'



```
spell = 'Z600 ~ Z3|plev(60000)'
case.calc(spell)
case.diags['Z600']
```

✓ 20.6s

```
>>> case.ds["Z3"] created
>>> case.ds["PS"] created
>>> case.diags["Z600"] created
```

xarray.DataArray 'Z600' (**time**: 600, ncol: 48602)

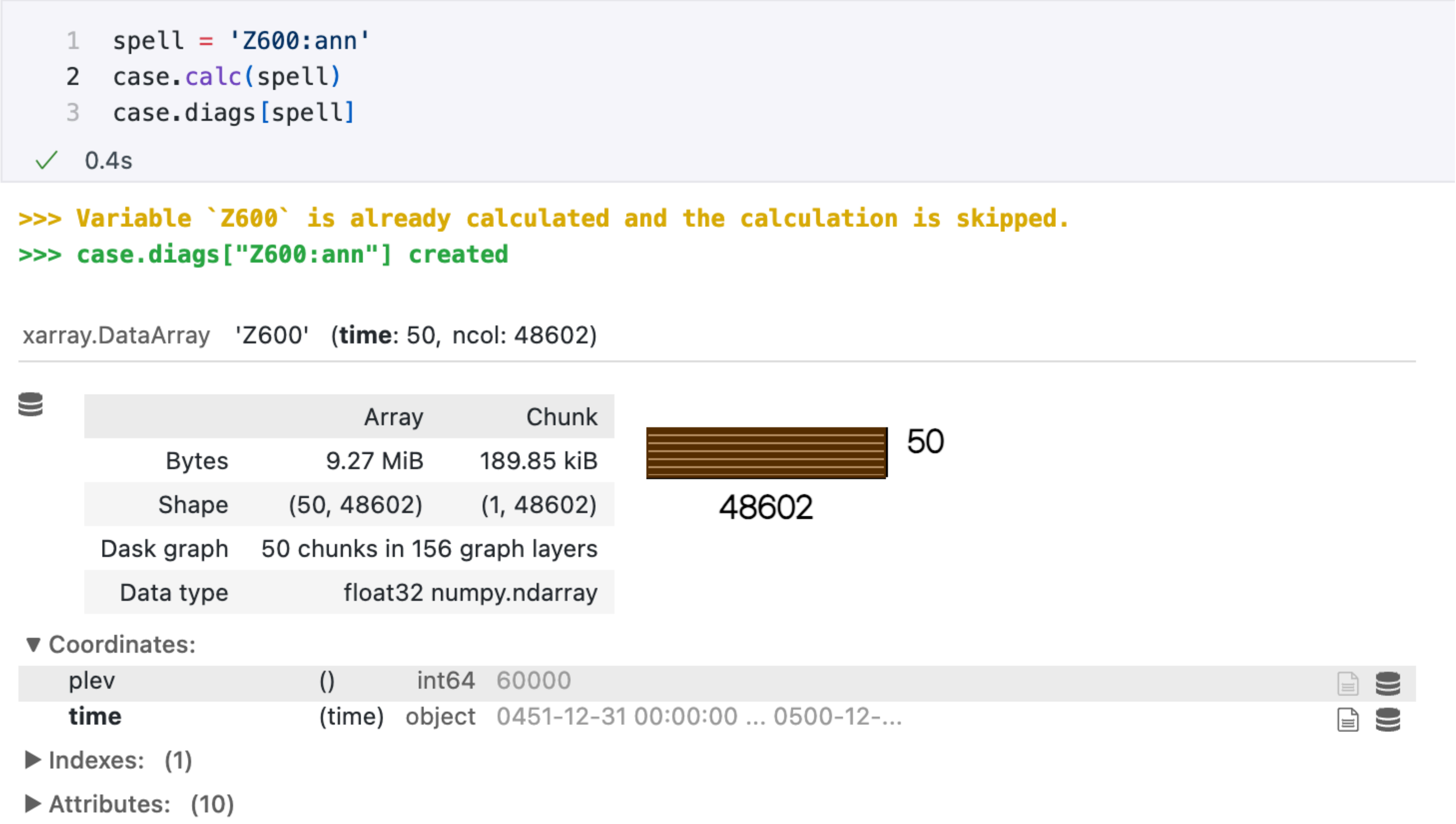
	Array	Chunk
Bytes	111.24 MiB	111.24 MiB
Shape	(600, 48602)	(600, 48602)
Dask graph	1 chunks in 5 graph layers	
Data type	float32 numpy.ndarray	

▼ Coordinates:

plev	()	int64	60000
time	(time)	object	0451-01-31 00:00:00 ... 0500-12-...

▶ Indexes: (1)

▶ Attributes: (10)



```
spell = 'Z600:ann'
case.calc(spell)
case.diags[spell]
```

✓ 0.4s

```
>>> Variable `Z600` is already calculated and the calculation is skipped.
>>> case.diags["Z600:ann"] created
```

xarray.DataArray 'Z600' (**time**: 50, ncol: 48602)

	Array	Chunk
Bytes	9.27 MiB	189.85 kiB
Shape	(50, 48602)	(1, 48602)
Dask graph	50 chunks in 156 graph layers	
Data type	float32 numpy.ndarray	

▼ Coordinates:

plev	()	int64	60000
time	(time)	object	0451-12-31 00:00:00 ... 0500-12-...

▶ Indexes: (1)

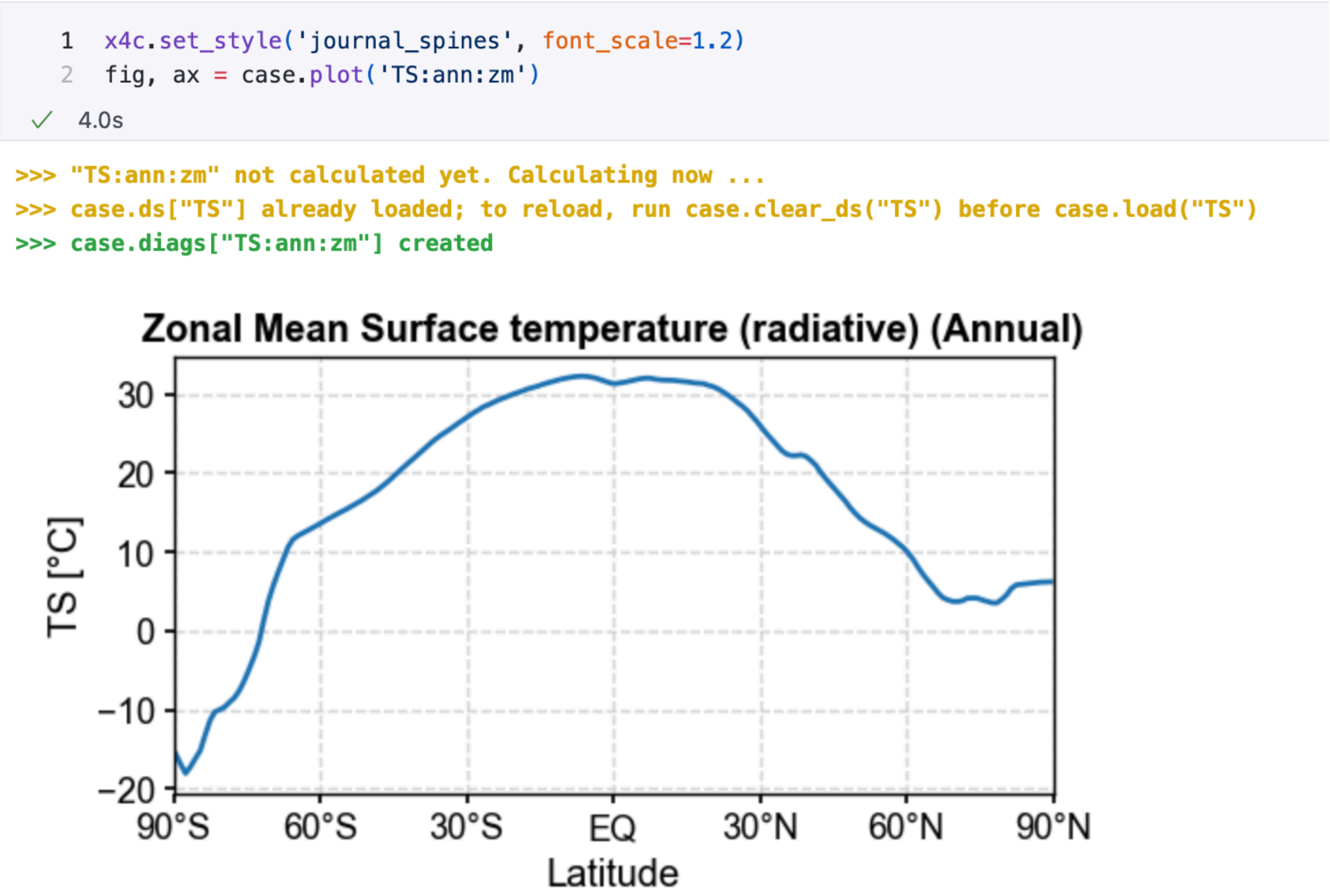
▶ Attributes: (10)

▣ Diagnostic Features (Timeseries)

A “Spell” System

Zonal mean of annual Surface Temperature

'TS:ann:zm'



```
x4c.set_style('journal_spines', font_scale=1.2)
fig, ax = case.plot('TS:ann:zm')
```

✓ 4.0s

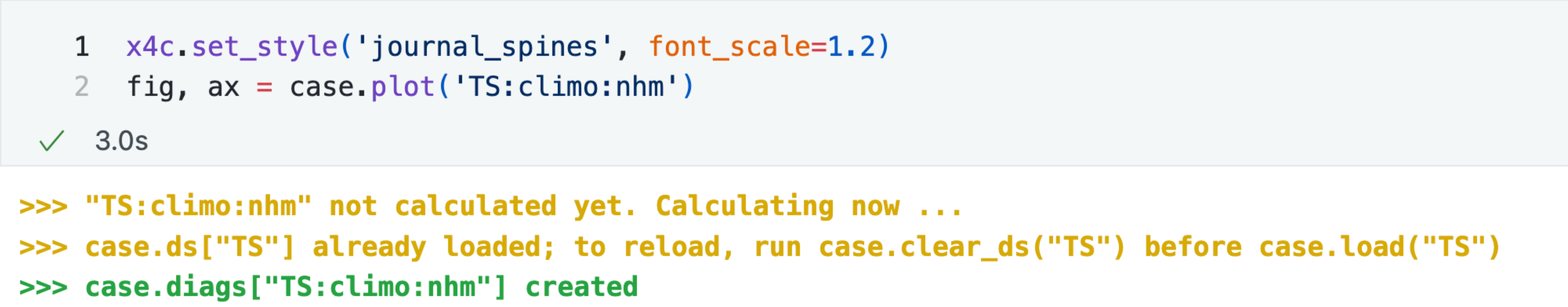
```
>>> "TS:ann:zm" not calculated yet. Calculating now ...
>>> case.ds["TS"] already loaded; to reload, run case.clear_ds("TS") before case.load("TS")
>>> case.diags["TS:ann:zm"] created
```

Diagnostic Features (Timeseries)

A “Spell” System

Seasonal cycle of Northern Hemisphere Surface Temperature

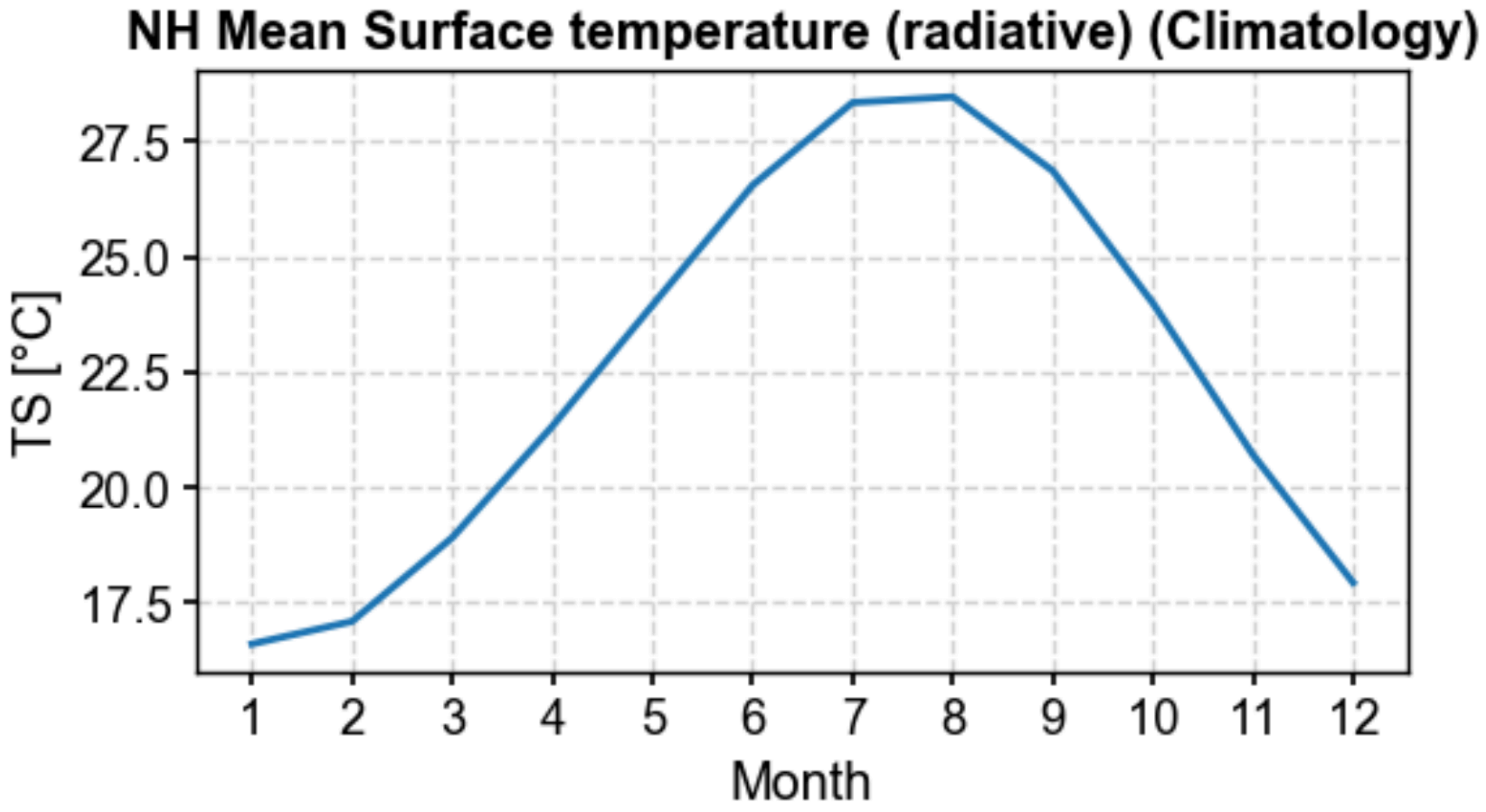
'TS:climo:nhm'



```
x4c.set_style('journal_spines', font_scale=1.2)
fig, ax = case.plot('TS:climo:nhm')
```

✓ 3.0s

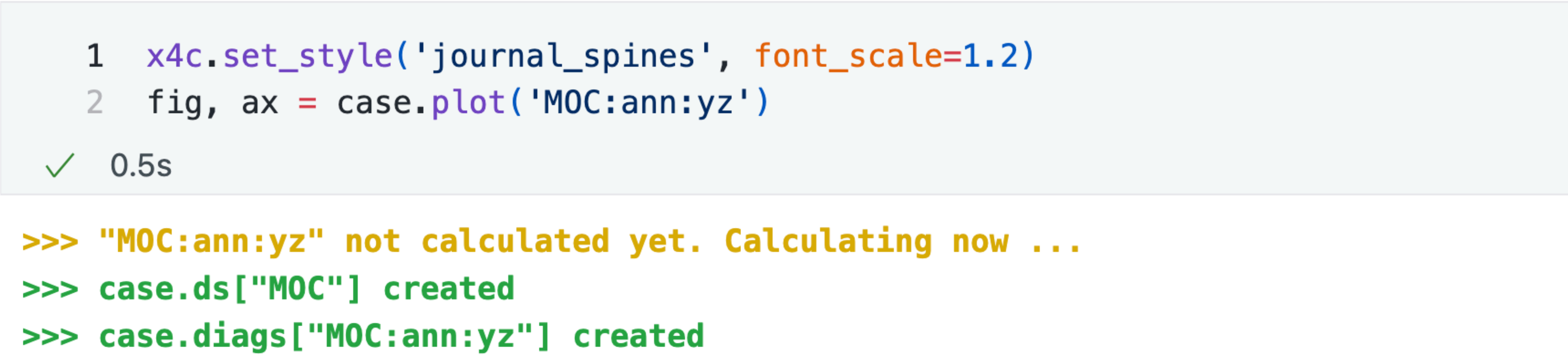
```
>>> "TS:climo:nhm" not calculated yet. Calculating now ...
>>> case.ds["TS"] already loaded; to reload, run case.clear_ds("TS") before case.load("TS")
>>> case.diags["TS:climo:nhm"] created
```



Diagnostic Features (Timeseries)

Zonal mean of annual Meridional Ocean Circulation (MOC)

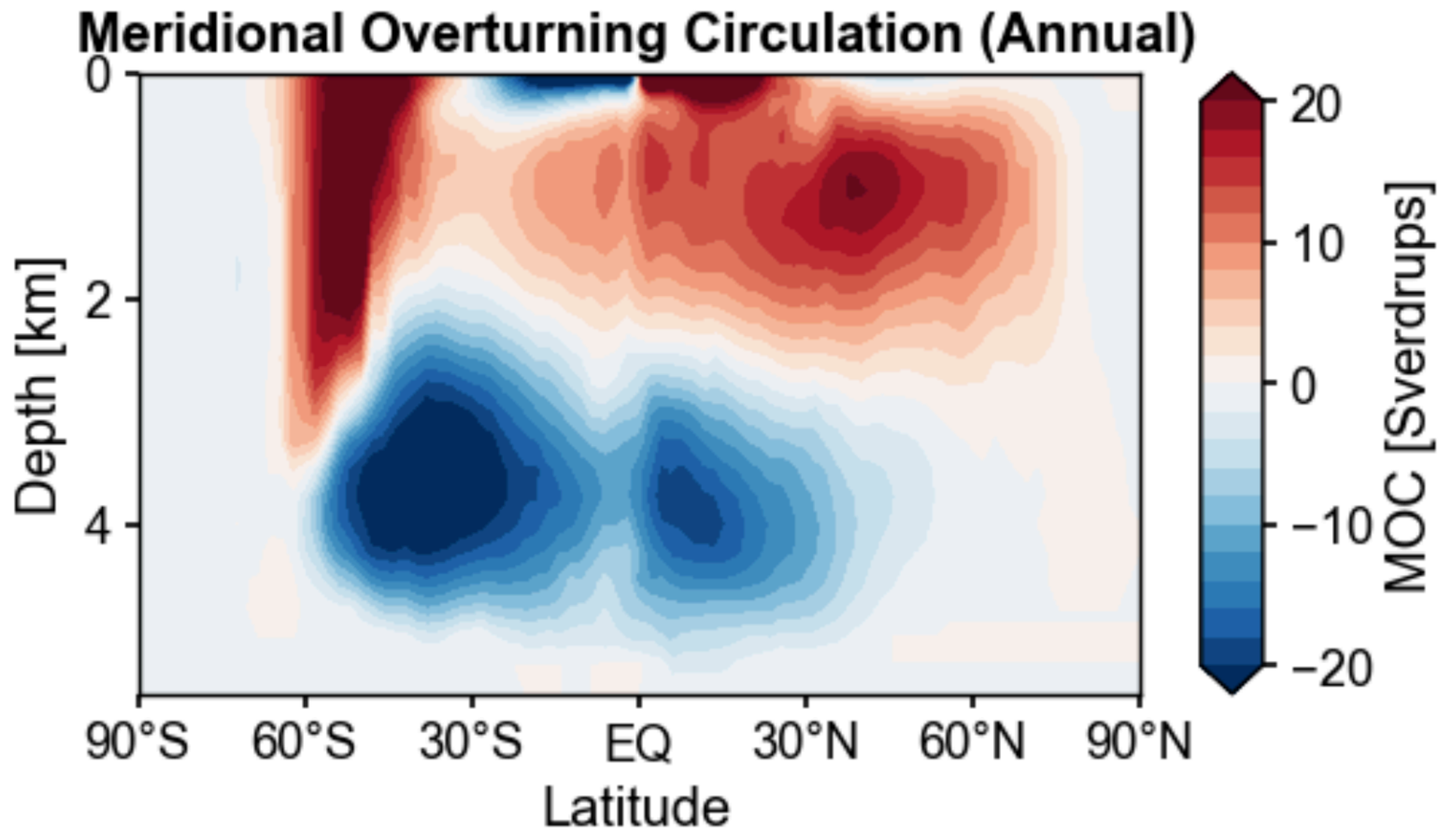
'MOC:ann:yz'



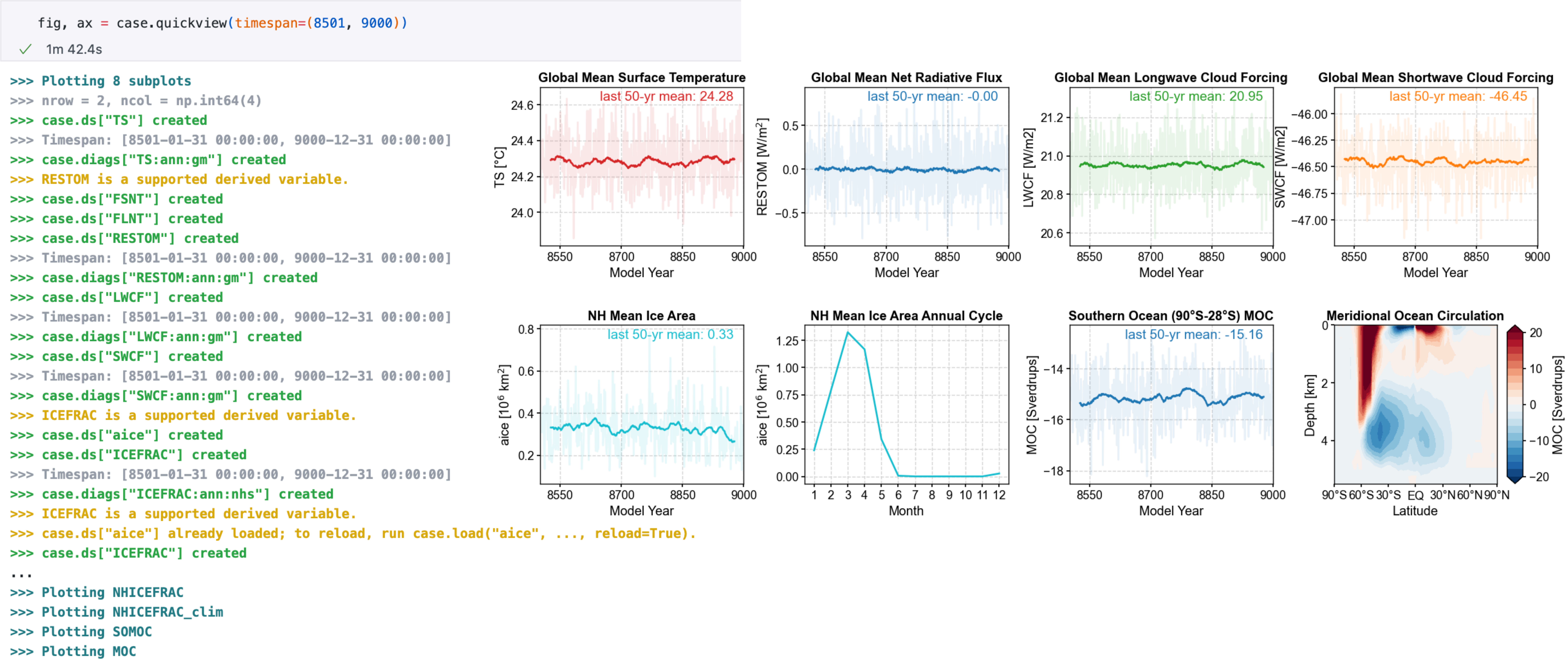
```
x4c.set_style('journal_spines', font_scale=1.2)
fig, ax = case.plot('MOC:ann:yz')
```

✓ 0.5s

```
>>> "MOC:ann:yz" not calculated yet. Calculating now ...
>>> case.ds["MOC"] created
>>> case.diags["MOC:ann:yz"] created
```



Diagnostic Features (Timeseries)



```
fig, ax = case.quickview(timespan=(8501, 9000))
```

✓ 1m 42.4s

```
>>> Plotting 8 subplots
>>> nrow = 2, ncol = np.int64(4)
>>> case.ds["TS"] created
>>> Timespan: [8501-01-31 00:00:00, 9000-12-31 00:00:00]
>>> case.diags["TS:ann:gm"] created
>>> RESTOM is a supported derived variable.
>>> case.ds["FSNT"] created
>>> case.ds["FLNT"] created
>>> case.ds["RESTOM"] created
>>> Timespan: [8501-01-31 00:00:00, 9000-12-31 00:00:00]
>>> case.diags["RESTOM:ann:gm"] created
>>> case.ds["LWCF"] created
>>> Timespan: [8501-01-31 00:00:00, 9000-12-31 00:00:00]
>>> case.diags["LWCF:ann:gm"] created
>>> case.ds["SWCF"] created
>>> Timespan: [8501-01-31 00:00:00, 9000-12-31 00:00:00]
>>> case.diags["SWCF:ann:gm"] created
>>> ICEFRAC is a supported derived variable.
>>> case.ds["aice"] created
>>> case.ds["ICEFRAC"] created
>>> Timespan: [8501-01-31 00:00:00, 9000-12-31 00:00:00]
>>> case.diags["ICEFRAC:ann:nhs"] created
>>> ICEFRAC is a supported derived variable.
>>> case.ds["aice"] already loaded; to reload, run case.load("aice", ..., reload=True).
>>> case.ds["ICEFRAC"] created
...
>>> Plotting NHICEFRAC
>>> Plotting NHICEFRAC_clim
>>> Plotting SOMOC
>>> Plotting MOC
```

Summary

- x4c aims to liberate scientists from technical details, facilitating scientific thinking.
- x4c enables concise gridded data analysis and publication-ready visualization with the .x command.
- x4c.History() makes CESM postprocessing (hist2ts) flexible and transparent.
- x4c.Timeseries(), along with the “spell” system, makes CESM diagnostic and visualization intuitive and efficient.
- x4c could serve as a foundational framework for diagnostic systems such as CUPiD.
- **Next Step:** *AI-enhanced Data Analysis and Visualization*

Porting LLMs to the “spell” system.

<u>Welcome funding support to expand this work!</u>

x4c

— Xarray for CESM —

Installation

- `conda install -c conda-forge xesmf cartopy geocat-comp pop-tools`
- `pip install x4c`

Documentation

- https://ncar.github.io/x4c

Thank you!

(fengzhu@ucar.edu)

Feng Zhu

Paleoclimate Software Engineer II

PPC, CGD, NSF NCAR

Nov 10, 2025

This material is based upon work supported by the National Center for Atmospheric Research, which is a major facility sponsored by the National Science Foundation under Cooperative Agreement No. 1852977.