an xarray wrapper for analysis of ensemble forecast models for climate prediction.

Riley X. Brady and Aaron Spring

https://climpred.readthedocs.io
Why do we need a package like this?

The current convention is for scientists to write their own code snippets in their language of choice: e.g., NCL, MATLAB, GrADS, FORTRAN. This means that scientists spend a considerable portion of their research time manually aligning forecast and verification dates and writing their own metrics.
Why do we need a package like this?

The current convention is for scientists to write their own code snippets in their language of choice: e.g., NCL, MATLAB, GrADS, FORTRAN. This means that scientists spend a considerable portion of their research time manually aligning forecast and verification dates and writing their own metrics.

With databases like the CESM-DPLE, the Decadal Climate Prediction Protocol (DCPP), SubX, NMME, we have an unprecedented amount of prediction experiments to analyze.
Why do we need a package like this?

The current convention is for scientists to write their own code snippets in their language of choice: e.g., NCL, MATLAB, GrADS, FORTRAN. This means that scientists spend a considerable portion of their research time manually aligning forecast and verification dates and writing their own metrics.

With databases like the CESM-DPLE, the Decadal Climate Prediction Protocol (DCPP), SubX, NMME, we have an unprecedented amount of prediction experiments to analyze.

If the community can unite around an open-source framework, we can spend more time answering questions about predictability and less time writing (and re-writing) code.
Mission Statement

climpred is a high-level package that leverages the scientific python ecosystem to provide an *interactive* experience for analyzing initialized prediction systems, from file input to publication-ready visualizations.
climpred and the pangeo ecosystem
climpred and the pangeo ecosystem

xarray is the core driver of climpred and allows for easy analysis of labeled multi-dimensional arrays of data.
climpred and the pangeo ecosystem

**xarray** is the core driver of climpred and allows for easy analysis of *labeled* multi-dimensional arrays of data.

**dask** allows for out-of-memory and parallel computations. It makes running interactive computations on many nodes easy so you can work with massive datasets.
**PredictionEnsemble Objects**

**HindcastEnsemble** – A system initialized from an observation-like product.

- Compare ensemble mean to observations or each member individually to observations.
- Select from different methods of alignment to pair initialization dates with verification dates.

**PerfectModelEnsemble** – A system initialized from a control run.

- Multiple ways to compare ensemble members (e.g. all members to each other, ensemble mean to control run, ...)
- No need to select alignment, based on how the system is configured.
**PredictionEnsemble Objects**

**init** – Initialization dates and times.
- [1954, 1955, 1956, …]
  OR

**lead** – Time since initialization.
- [1, 2, 3, 4, …]
- `units` attribute: [‘years’, ‘seasons’, ‘months’, ‘weeks’, ‘pentads’, ‘days’]

**member** – Ensemble member (optional, but needed for most probabilistic metrics).

Any arbitrary number of additional dimensions (depth, height, x, y, …)
Workflow: Choosing Forecast Alignment

same_verifs – Use a common verification window across all leads.

same_inits – Use a common set of initializations that verify across all leads.

maximize – Use all available initializations at each lead that verify against the observations provided.
Workflow: Choosing Forecast Alignment

**same_verifs** – Use a common verification window across all leads.

**same_inits** – Use a common set of initializations that verify across all leads.

---

**Verification with Different Alignment Methods**

- **SST with trend**
- **Detrended SST**

![Graph showing anomaly correlation coefficient for different methods over lead years.](image)
Workflow: Creating a Prediction System Object

```python
from climpred import HindcastEnsemble

# 'ds' is NCAR DCPP output with properly renamed dimensions.
hindcast = HindcastEnsemble(ds)

# 'fosi' is the forced POP run that initialized CESM-DPLE.
hindcast = hindcast.add_observations(fosi, name='Forced Ocean-Sea Ice Reconstruction')
```

CESM-DPLE output from DCPP

Reconstruction POP product for validation
Workflow: Creating a Prediction System Object

```python
from climpred

# `ds` is N
hindcast = |
# `fosi` is |
hindcast = |
```

```
climpred.HindcastEnsemble

Initialized Ensemble

- Dimensions: (init: 18, lat: 180, lead: 10, lon: 360, member: 10)
- Coordinates:
  - lon (lon) float64 0.5 1.5 2.5 ... 357.5 358.5 359.5
  - lead (lead) int64 1 2 3 4 5 6 7 8 9 10
  - lat (lat) float64 -89.5 -88.5 -87.5 ... 88.5 89.5
  - init (init) object 2000-01-01 00:00:00 ... 2017-01-01 ...
- Data variables:
  - `tos` (init, lead, member, lat, lon) float32 dask.array<chunksize=(1, 1, 10, 180, 3...>
- Attributes: (0)

Verification Data Forced Ocean–Sea Ice Reconstruction

- Dimensions: (lat: 180, lon: 360, time: 70)
- Coordinates:
  - z_t () float32 500.0
  - time (time) object 1948-01-01 00:00:00 ... 2017-01-01 00:00:00
  - lon (lon) float64 0.5 1.5 2.5 ... 357.5 358.5 359.5
  - lat (lat) float64 -89.5 -88.5 -87.5 ... 88.5 89.5
- Data variables:
  - `tos` (time, lat, lon) float64 nan nan nan ... -1.352 -1.352 nan
- Attributes: (0)
Workflow: Verifying a Forecast

```python
hindcast.verify(metric='acc', alignment='maximize', comparison='e2o', reference=['persistence']).compute()
```

```py
xarray.Dataset
```

- **Dimensions:**
  - (lat: 180, lead: 10, lon: 360, skill: 2)

- **Coordinates:**
  - **z_t**: () -> float32 500.0
  - **lon**: (lon) -> float64 0.5 1.5 2.5 ... 357.5 358.5 359.5
  - **lat**: (lat) -> float64 -89.5 -88.5 -87.5 ... 88.5 89.5
  - **lead**: (lead) -> int64 1 2 3 4 5 6 7 8 9 10
  - **skill**: (skill) -> <U11 'init' 'persistence'

- **Data variables:**
  - tos: (skill, lead, lat, lon) -> float64 nan nan nan ... 0.5133 0.5123 nan

- **Attributes:** (0)

```
CPU times: user 561 ms, sys: 79.9 ms, total: 641 ms
Wall time: 2.76 s
```
Workflow: Verifying a Forecast

```
hindcast.verify(metric='acc', alignment='maximize', comparison='e2o', reference=['persistence']).compute()
```
Workflow: Verifying a Forecast

```python
hindcast.verify(metric='nmae', alignment='maximize', comparison='e2o', reference=None).compute()
```

We currently have 28 metrics available, but you can pass a user-defined one in (or submit a PR to have it hard-coded into the package!).

CPU times: user 973 ms, sys: 100 ms, total: 1.07 s
Wall time: 2.87 s
Workflow: Bootstrapping and Significance Testing

1. Temporal p-value and field significance through metric API
Workflow: Bootstrapping and Significance Testing

1. Temporal p-value and field significance through `metric` API

2. Work in progress: scalable bootstrapping with `dask`

```python
bootstrap_hindcast(hind.chunk(), fosi.chunk(), 500).load()
```

CPU times: user 5.75 s, sys: 194 ms, total: 5.94 s
Wall time: 11.5 s

dataset

- **Dimensions:** (lat: 36, lead: 10, lon: 72, quantile: 2)

- **Coordinates:**
  - `lead`: int64 1 2 3 4 5 6 7 8 9 10
  - `lat`: float64 -87.5 -82.5 -77.5 ... 82.5 87.5
  - `lon`: float64 2.5 7.5 12.5 ... 347.5 352.5 357.5
  - `quantile`: float64 0.05 0.95

- **Data variables:**
  - `tos`: float64 nan nan nan ... -0.00111 -0.0694

- **Attributes:** (0)
Development Roadmap

• Simpler bootstrap API with better scaling (summer/fall 2020)
• Improvements for S2S time scales (summer 2020)
• Refactor codebase to use more inheritance for easier development (~August 2020)
• Journal of Open Source Software (JOSS) publication (fall 2020)
How to get involved

Read the documentation, install, and start using! (https://climpred.readthedocs.io)

Post issues or open pull requests. (www.github.com/bradyrx/climpred)

Email for support, questions, ideas (riley.brady@colorado.edu) (aaron.spring@mpimet.mpg.de)
Additional Slides
climpred and the pangeo ecosystem

**xarray** is the core driver of climpred and allows for easy analysis of *labeled* multi-dimensional arrays of data.

```python
dataset = xarray.load_dataset('./data/nmc_air_temperature.nc',
                             use_cftime=True)
print(dataset)

<xarray.Dataset>
Dimensions: (lat: 25, lon: 53, time: 2920)
Coordinates:
  * lat  (lat) float32 75.0 72.5 70.0 67.5 65.0 ... 25.0 22.5 20.0 17.5 15.0
  * lon  (lon) float32 200.0 202.5 205.0 207.5 ... 322.5 325.0 327.5 330.0
  * time (time) object 2013-01-01 00:00:00 ... 2014-12-31 18:00:00
Data variables:
  air     (time, lat, lon) float32 241.2 242.5 243.5 ... 296.49 296.19 295.69
Attributes:
  Conventions: COARDS
description:  Data is from NMC initialized reanalysis\n(4x/day). These a...
  platform:  Model
  references:  http://www.esrl.noaa.gov/psd/data/gridded/data.ncep.reanalysis...
title:  4x daily NMC reanalysis (1948)
```
climpred and the pangeo ecosystem

xarray is the core driver of climpred and allows for easy analysis of labeled multi-dimensional arrays of data.

```python
dataset['time'].head()

<xarray.DataArray 'time' (time: 5)>
array([cftime.DatetimeGregorian(2013-01-01 00:00:00),
cftime.DatetimeGregorian(2013-01-01 06:00:00),
cftime.DatetimeGregorian(2013-01-01 12:00:00),
cftime.DatetimeGregorian(2013-01-01 18:00:00),
cftime.DatetimeGregorian(2013-01-02 00:00:00)], dtype=object)

Coordinates:
* time  (time) object 2013-01-01 00:00:00 ... 2013-01-02 00:00:00
```
climpred and the pangeo ecosystem

xarray is the core driver of climpred and allows for easy analysis of labeled multi-dimensional arrays of data.

```python
ds_slice = dataset.sel(time=slice('2013-06-05', '2013-06-06'),
                      lat=slice(50, 30),
                      lon=210)

ds_slice.std('lat')
```

<xarray.Dataset>
Dimensions: (time: 8)
Coordinates:
  lon   float32 210.0
  * time (time) object 2013-06-05 00:00:00 ... 2013-06-06 18:00:00
Data variables:
  air   (time) float32 4.285068 4.431042 4.223673 ... 4.2307315 4.196668
```
climpred and the pangeo ecosystem

ds_mean = dataset.mean(['lat', 'lon'])
# Group off into each point's corresponding day of the year
# and take the average over that group.
anual_cycle = ds_mean.groupby('time.dayofyear').mean('time')

annual_cycle['air'].plot()

[<matplotlib.lines.Line2D at 0x10fab75f8>]

26
Same Initializations

initialized forecasting system

lead year

initialized forecasting system

verification data

1. union with observations  2. verifies at all leads
Maximize

![Diagram showing lead and time with verification data]

1. union with observations
2. lead-dependent verification