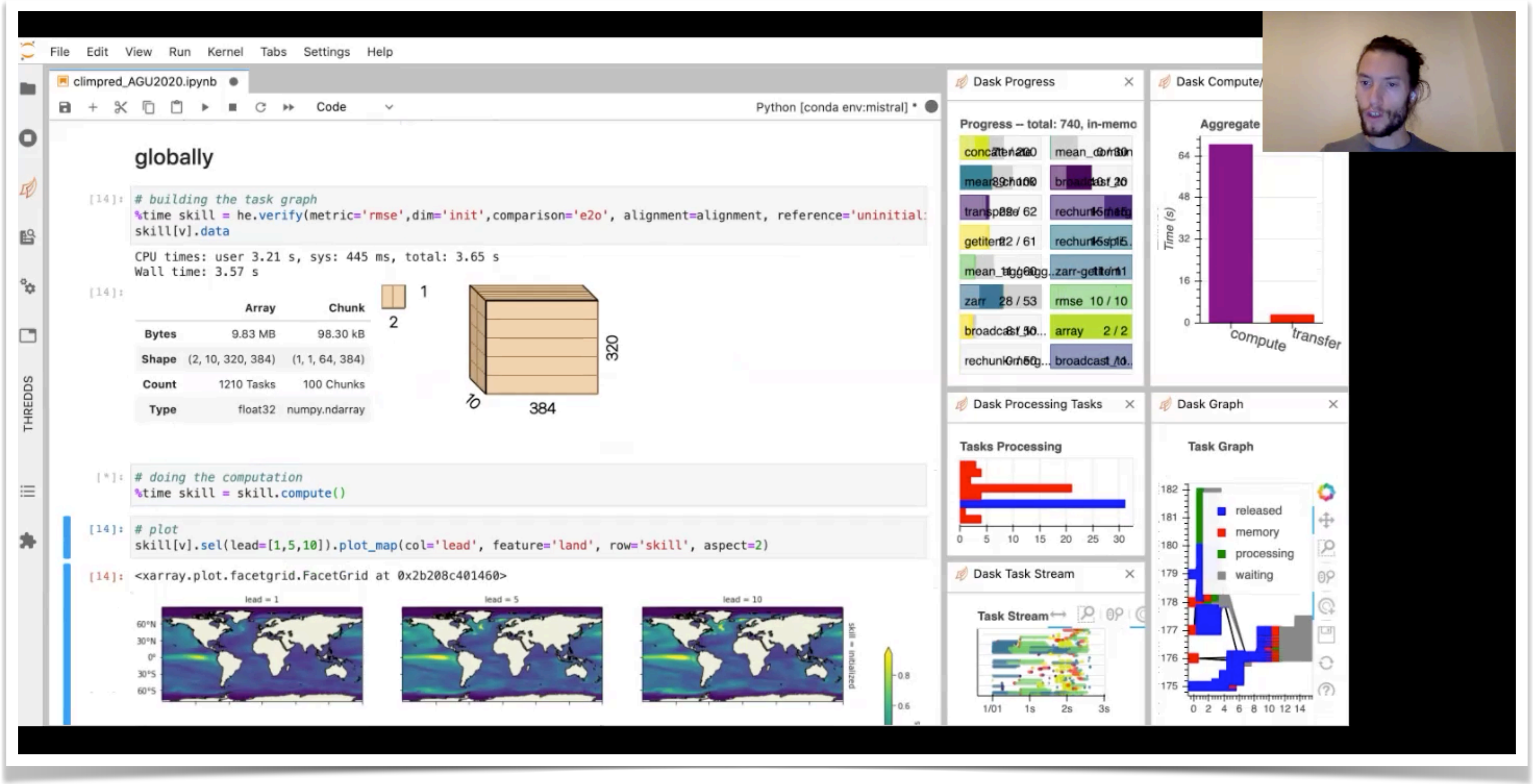


globally

```
[14]: # building the task graph
      %time skill = he.verify(metric='rmse',dim='init',comparison='e2o', alignment=alignment, reference='uninitial:
      skill[v].data

CPU times: user 3.21 s, sys: 445 ms, total: 3.65 s
Wall time: 3.57 s
```

[14]:

	Array	Chunk
Bytes	9.83 MB	98.30 kB
Shape	(2, 10, 320, 384)	(1, 1, 64, 384)
Count	1210 Tasks	100 Chunks
Type	float32	numpy.ndarray

```
[*]: # doing the computation
     %time skill = skill.compute()

[14]: # plot
      skill[v].sel(lead=[1,5,10]).plot_map(col='lead', feature='land', row='skill', aspect=2)

[14]: <xarray.plot.facetgrid.FacetGrid at 0x2b208c401460>
```

climpred

weather and climate prediction in python

Aaron Spring (MPI-M) and Riley Brady (CU)

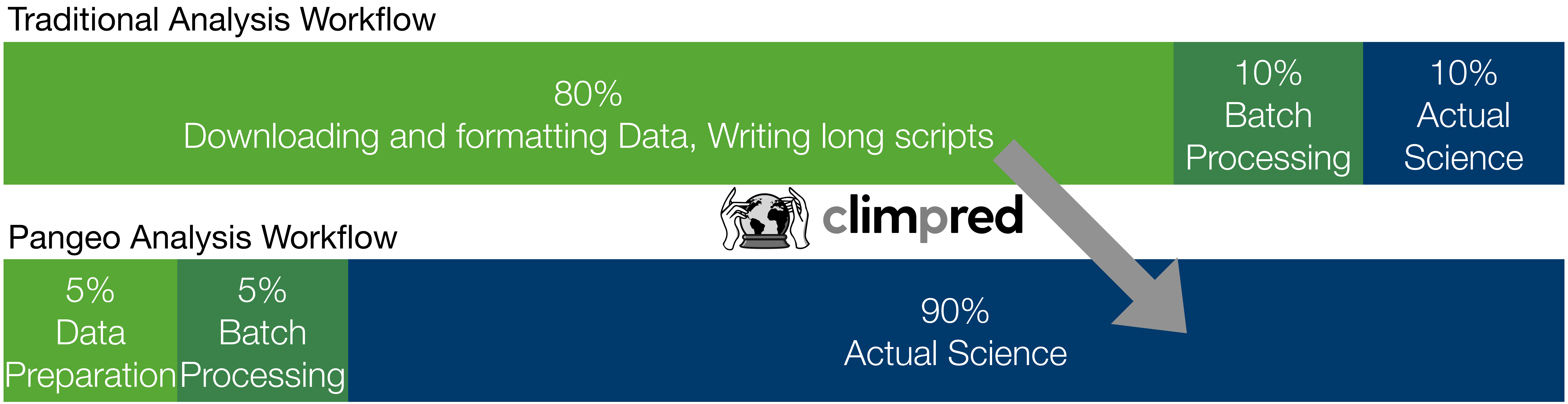
launch binder

Try climpred now live: https://github.com/aaronspring/climpred_egu22_demo

What impacts the velocity of science?

Data, Software and Computation

- Data: time to find, access, clean & format for analysis
- Software: easily available and combinable
- Computation: access and resources



Traditional Analysis Workflow

80%
Downloading and formatting Data, Writing long scripts

10%
Batch
Processing

10%
Actual
Science

climpred

Pangeo Analysis Workflow

5%
Data
Preparation

5%
Batch
Processing

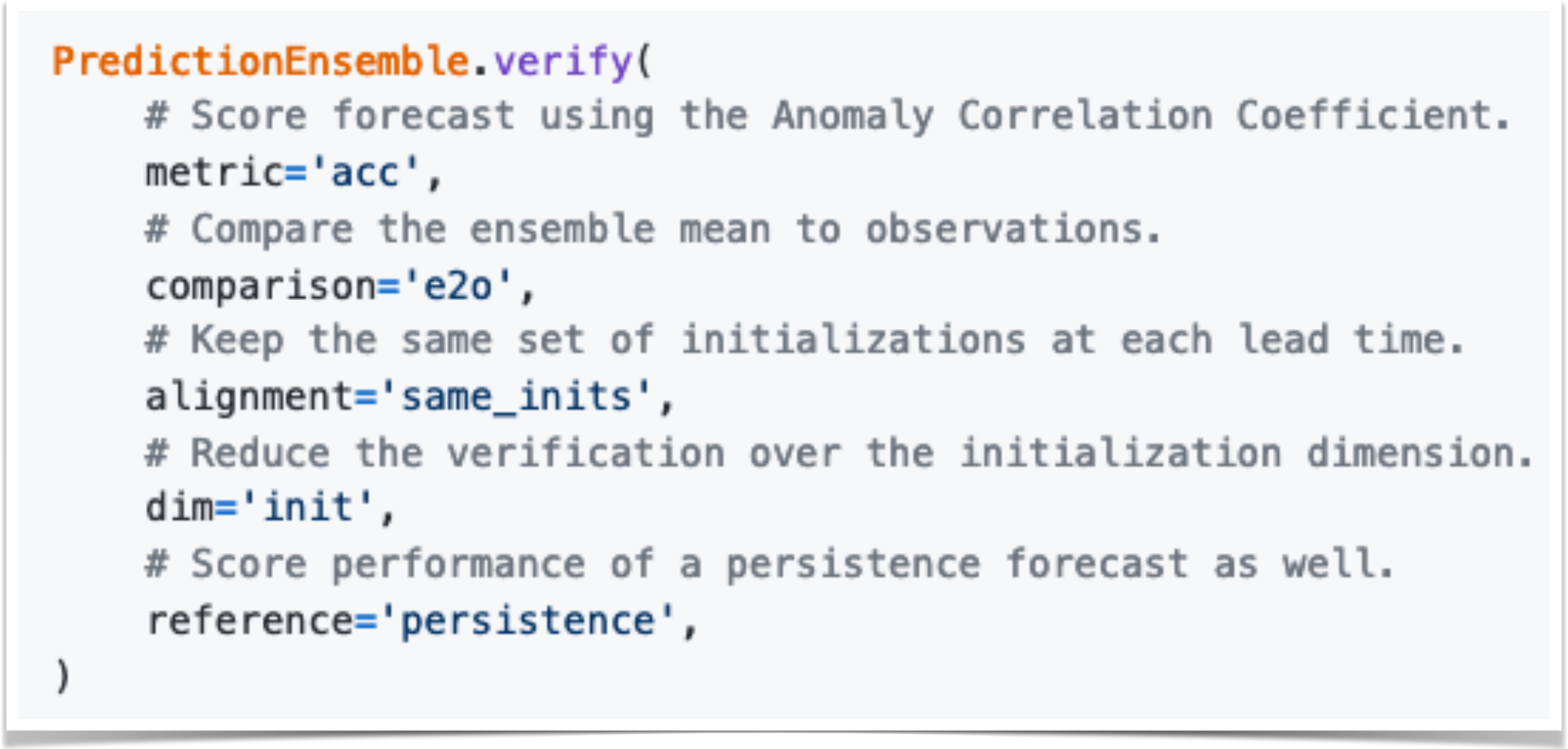
90%
Actual Science

Adopted from https://speakerdeck.com/cgentemann/empowering-transformational-science?slide=4

climpred

We believe forecast verification should be:

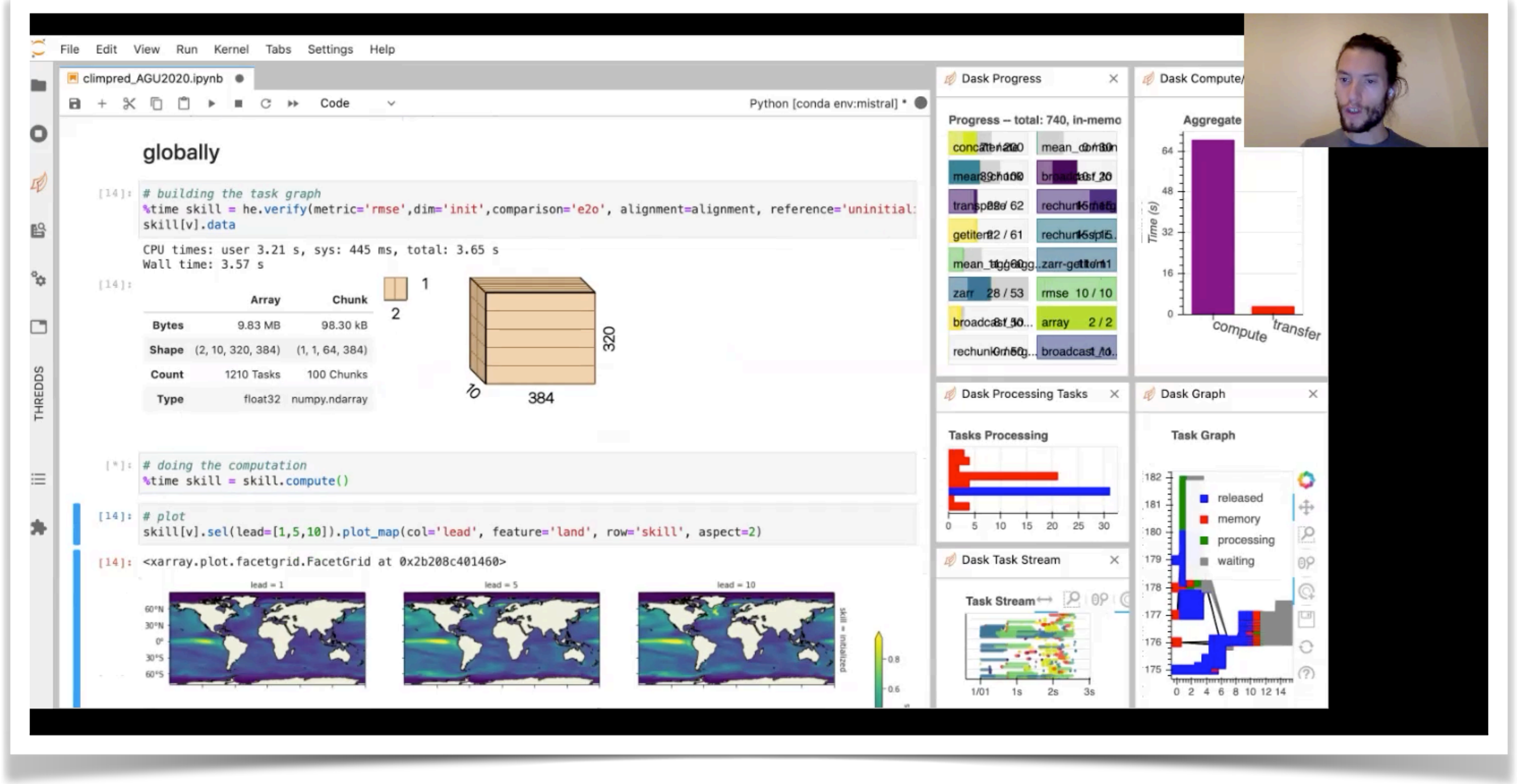
- interactive
- standardised
- reproducible
- simple to use



```
PredictionEnsemble.verify(
    # Score forecast using the Anomaly Correlation Coefficient.
    metric='acc',
    # Compare the ensemble mean to observations.
    comparison='e2o',
    # Keep the same set of initializations at each lead time.
    alignment='same_inits',
    # Reduce the verification over the initialization dimension.
    dim='init',
    # Score performance of a persistence forecast as well.
    reference='persistence',
)
```

What to not worry about

- metadata and time alignment: solved by xarray
- parallelisation and batch processing: solved by DASK



globally

```
[14]: # building the task graph
%time skill = he.verify(metric='rmse',dim='init',comparison='e2o', alignment=alignment, reference='uninitial
skill[v].data

CPU times: user 3.21 s, sys: 445 ms, total: 3.65 s
Wall time: 3.57 s
```

	Array	Chunk
Bytes	9.83 MB	98.30 kB
Shape	(2, 10, 320, 384)	(1, 1, 64, 384)
Count	1210 Tasks	100 Chunks
Type	float32	numpy.ndarray

```
[*]: # doing the computation
%time skill = skill.compute()

[14]: # plot
skill[v].sel(lead=[1,5,10]).plot_map(col='lead', feature='land', row='skill', aspect=2)

[14]: <xarray.plot.facetgrid.FacetGrid at 0x2b208c401460>
```

climpred API table

- `HindcastEnsemble.verify() and HindcastEnsemble.bootstrap()`
  - `metric`: `rmse, acc, roc, rank_histogram, reliability, crps`, …
  - `comparison`: `e2o`: ensemble mean against observations, `m2o`: member against observations
  - `dim`: dimension over which apply metric: `init`, `member`
  - `alignment`: which forecasts and verification to match: `same_inits, same_verifs, maximize`
  - `reference`: add reference forecast skill `persistence, climatology, uninitialized`
- `HindcastEnsemble.remove_bias()`
  - `how`: method: `additive_mean`, `gamma_mapping`, `EmpiricalQuantileMapping`, …
  - See `verify()`
  - `train_test_split`: which `inits` to use for training bias correction: `unfair, unfair-cv, fair`

Demo: Nino 3.4 NMME Hindcast vs. NOAA OISST Verification

- Bias correction `train_test_split` following Risbey et al. (2021). Standard assessments of climate forecast skill can be misleading. *Nature Comm.:*
  - `none`: no bias correction
  - `unfair`: train bias correction on same `inits` as verification
  - `unfair-cv`: as `unfair` but leave out given init
  - `fair`: train on different `inits` than used for `verify()`

Information of future verification data used in bias correction! ❌ Not possible for real-time forecasts ❌

NMME Nino3.4 SST RMSE for different bias correction train_test splits

train_test_split = None

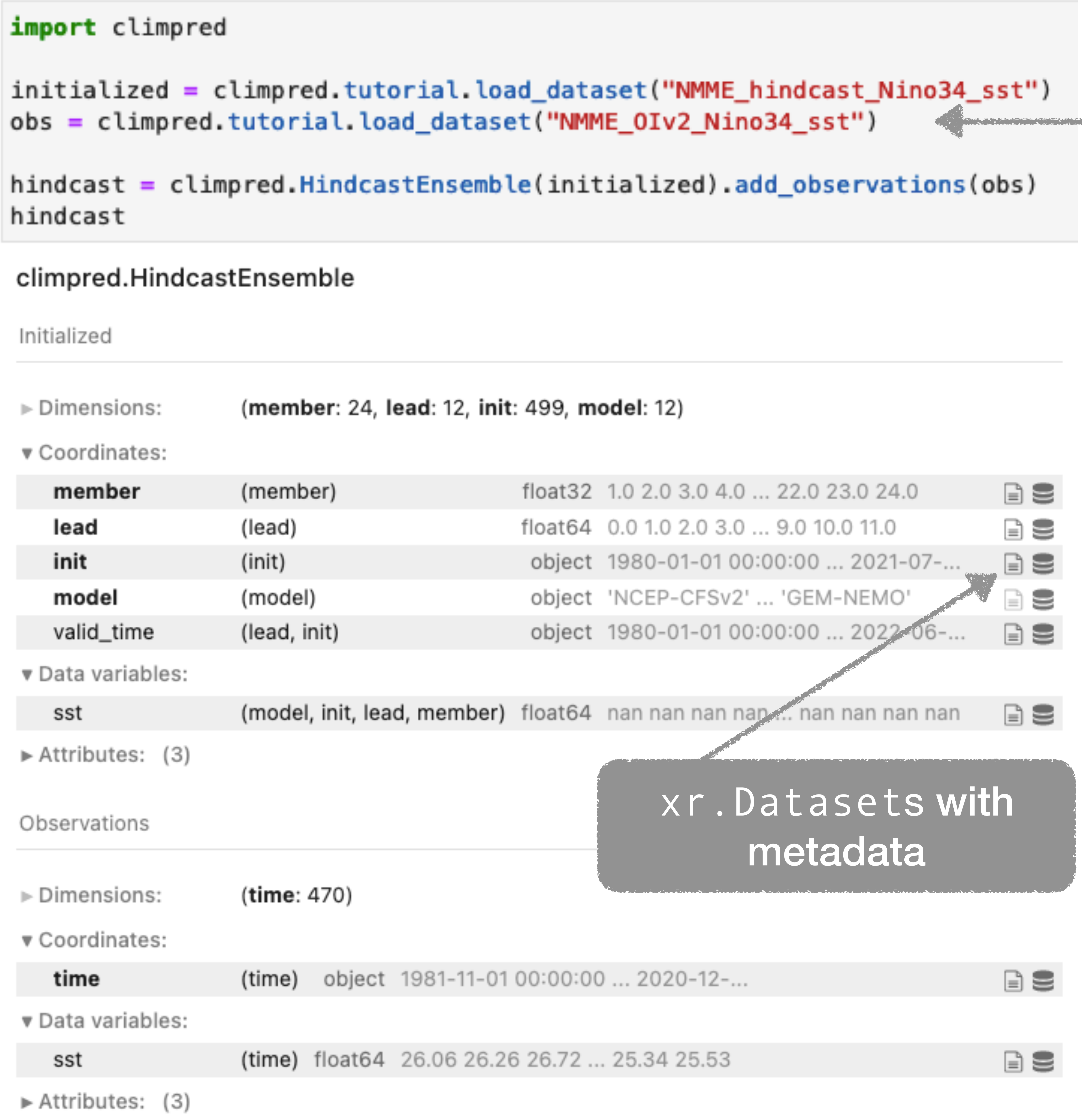
train_test_split = unfair

train_test_split = unfair-cv

train_test_split = fair

Sea Surface Temperature [C]
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
2.5
5.0
7.5
10.0
Lead [months]
model
NCEP-CFSv2
NCEP-CFSv1
NCAR-CESM1
NASA-GEOSS2S
CMC1-CanCM3
CanCM4i
NASA-GMAO
GFDL-CM2p5-FLOR-B01
GFDL-CM2p5-FLOR-A06
GFDL-CM2p1-aer04
COLA-RSMAS-CCSM4
GEM-NEMO

Demo: Data: Nino 3.4 NMME vs. NOAA OISST Verification



```python
import climpred

initialized = climpred.tutorial.load_dataset("NMME_hindcast_Nino34_sst")
obs = climpred.tutorial.load_dataset("NMME_OIv2_Nino34_sst")

hindcast = climpred.HindcastEnsemble(initialized).add_observations(obs)
hindcast
```

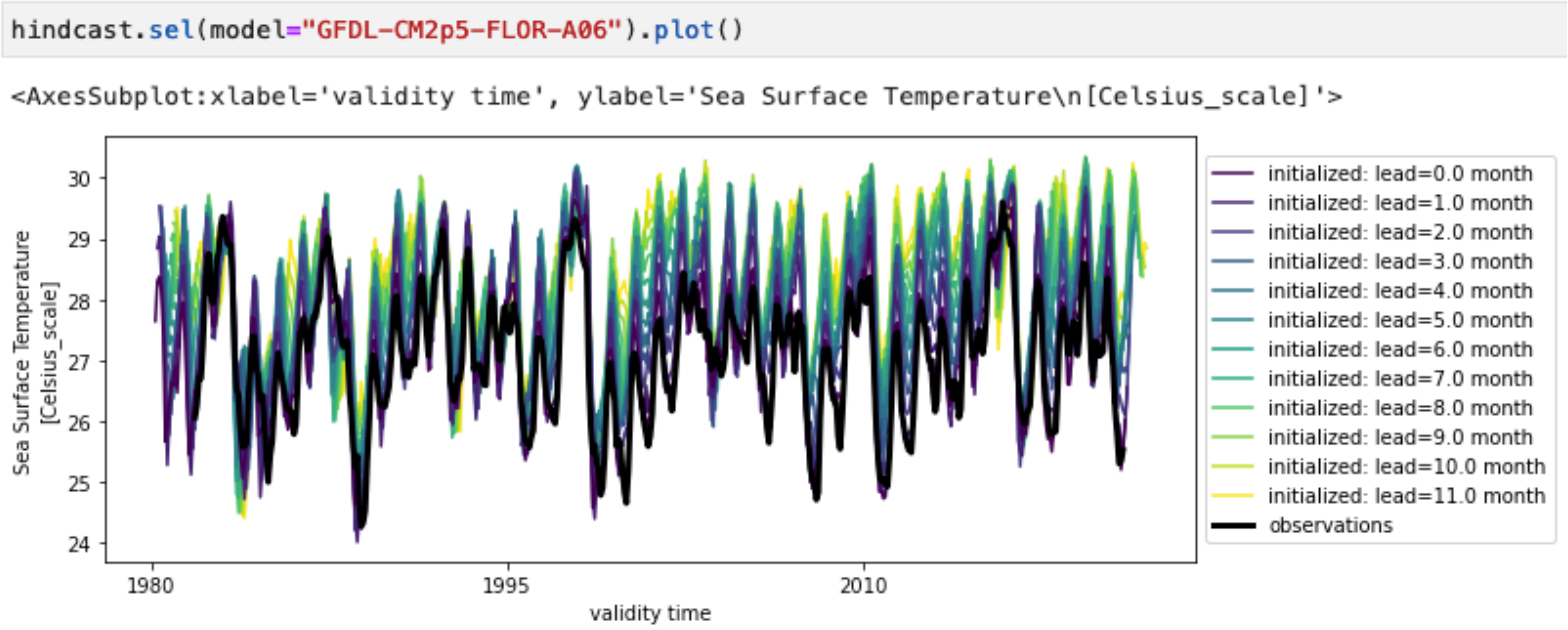
```
climpred.HindcastEnsemble

Initialized

▸ Dimensions:     (member: 24, lead: 12, init: 499, model: 12)
▾ Coordinates:
    member        (member)                    float32  1.0 2.0 3.0 4.0 ... 22.0 23.0 24.0
    lead          (lead)                      float64  0.0 1.0 2.0 3.0 ... 9.0 10.0 11.0
    init          (init)                      object   1980-01-01 00:00:00 ... 2021-07-...
    model         (model)                     object   'NCEP-CFSv2' ... 'GEM-NEMO'
    valid_time    (lead, init)                object   1980-01-01 00:00:00 ... 2022-06-...
▾ Data variables:
    sst           (model, init, lead, member) float64  nan nan nan nan ... nan nan nan nan
▸ Attributes:  (3)

Observations

▸ Dimensions:     (time: 470)
▾ Coordinates:
    time          (time)    object   1981-11-01 00:00:00 ... 2020-12-...
▾ Data variables:
    sst           (time)    float64  26.06 26.26 26.72 ... 25.34 25.53
▸ Attributes:  (3)
```

Here postprocessed to download or your data

`xr.Datasets` with metadata

- NMME: North American Multi-Model Ensemble
- OISST: NOAA Optimum Interpolation (OI) Sea Surface Temperature
- Nino34: SST area averaged 5N-5S;170W-120W

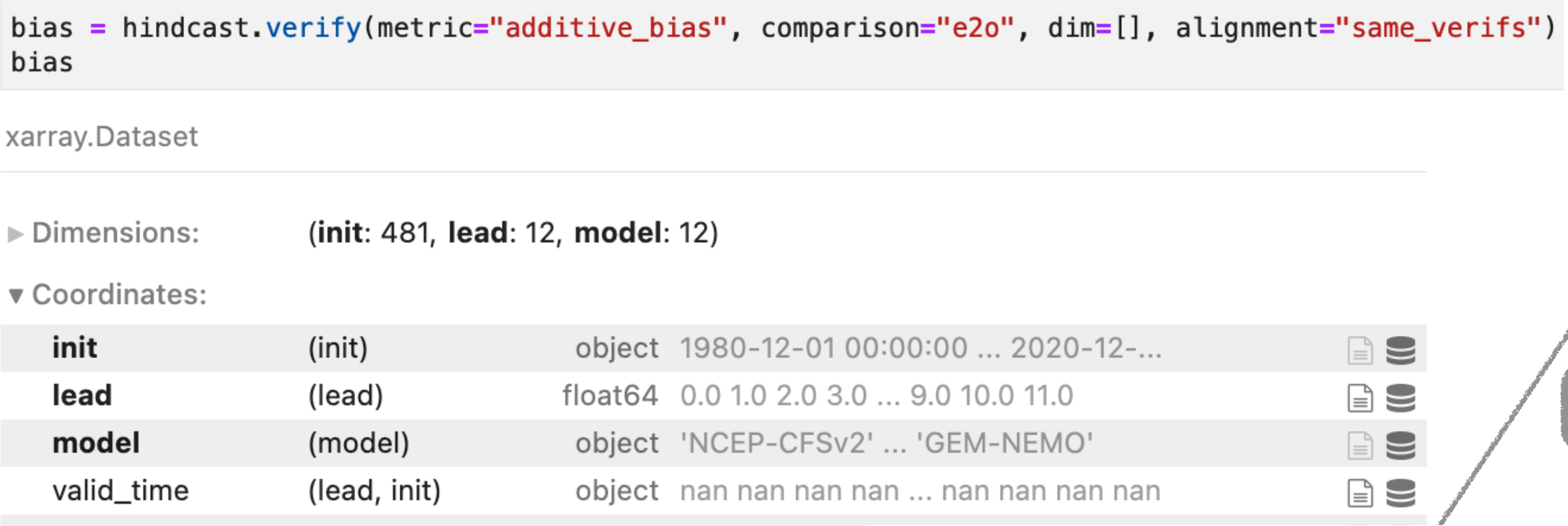
Easily plotting: time = init + lead



```python
hindcast.sel(model="GFDL-CM2p5-FLOR-A06").plot()
```

```
<AxesSubplot:xlabel='validity time', ylabel='Sea Surface Temperature\n[Celsius_scale]'>
```

Demo: `verify()` for `lead`-time dependent bias

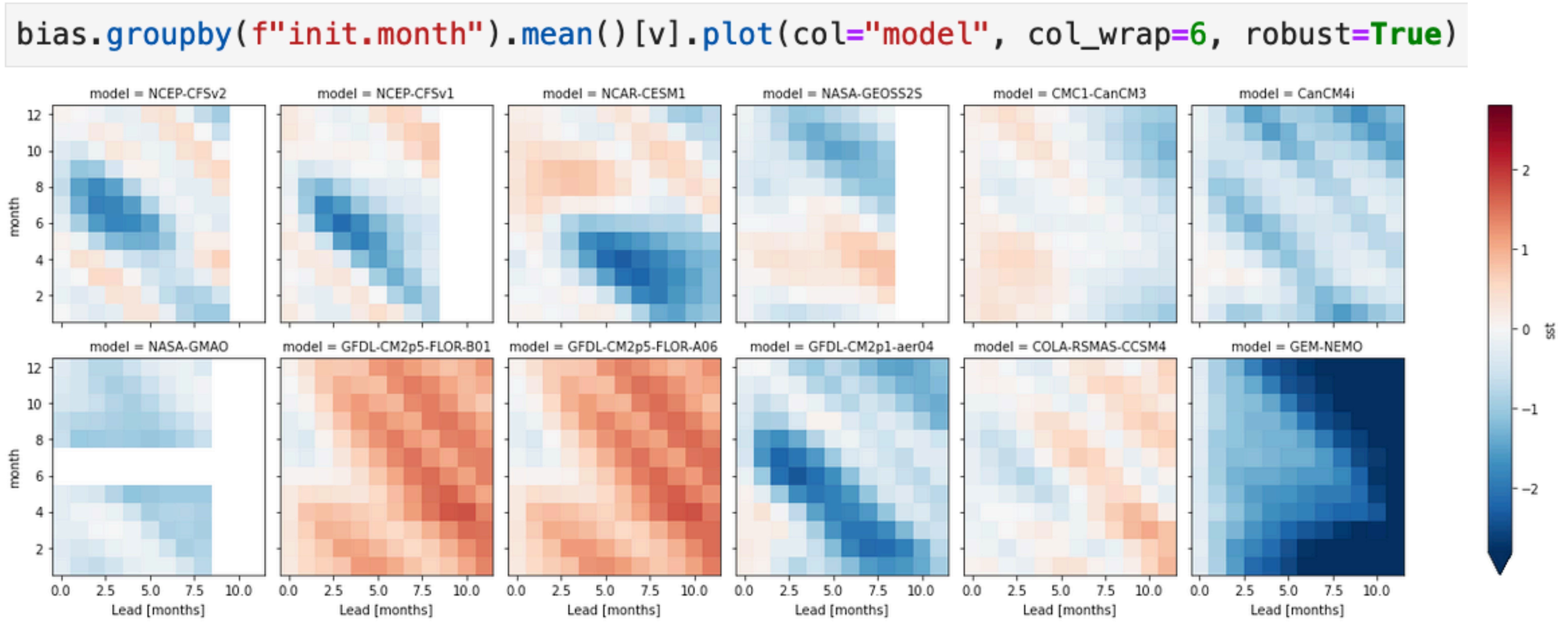


```
bias = hindcast.verify(metric="additive_bias", comparison="e2o", dim=[], alignment="same_verifs")
bias
```

```
xarray.Dataset

▶ Dimensions:     (init: 481, lead: 12, model: 12)

▼ Coordinates:
   init           (init)                object   1980-12-01 00:00:00 ... 2020-12-...
   lead           (lead)                float64  0.0 1.0 2.0 3.0 ... 9.0 10.0 11.0
   model          (model)               object   'NCEP-CFSv2' ... 'GEM-NEMO'
   valid_time     (lead, init)          object   nan nan nan nan ... nan nan nan nan
   skill          ()                    <U11     'initialized'
▼ Data variables:
   sst            (lead, model, init)   float64  nan nan nan n
```

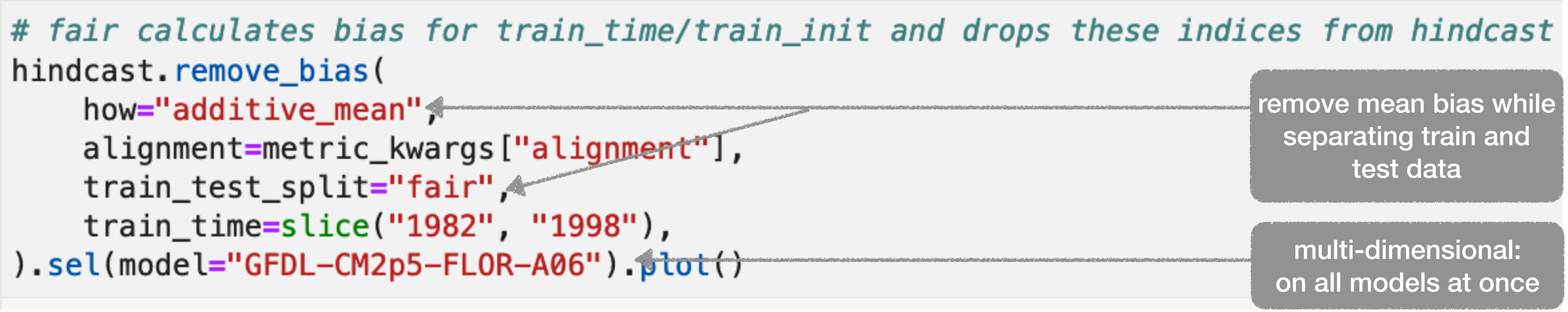
`xr.Dataset` with metadata

`xr.DataArray.plot()`



```
bias.groupby(f"init.month").mean()[v].plot(col="model", col_wrap=6, robust=True)
```

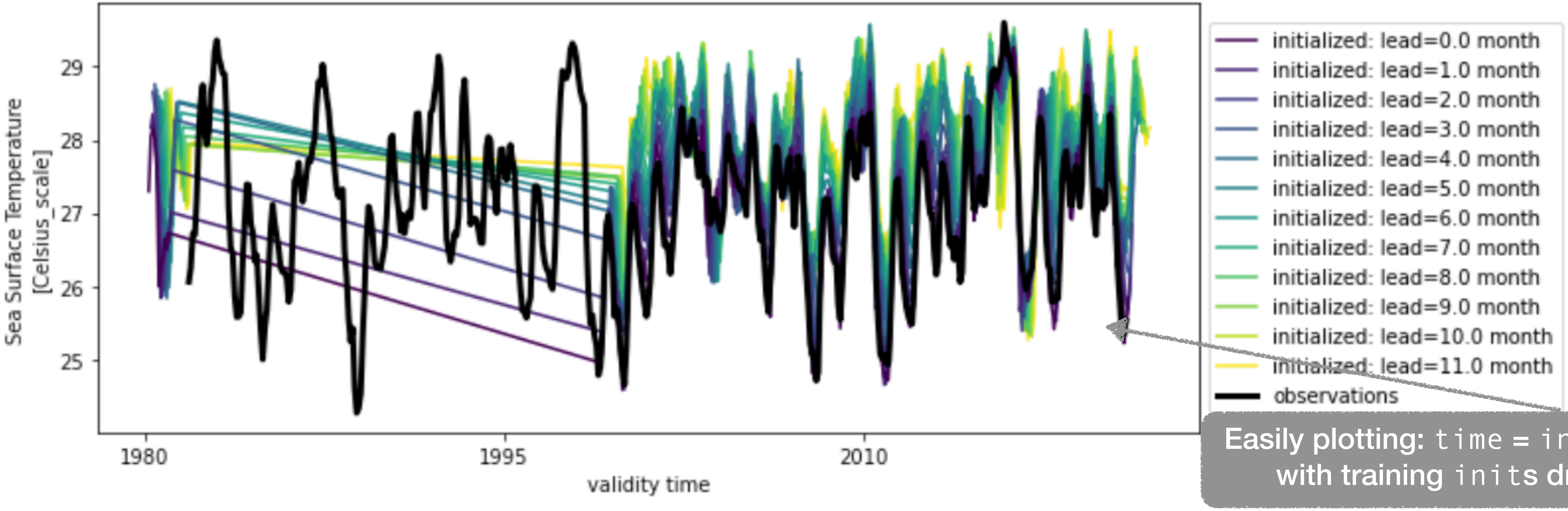
Demo: lead time-dependent bias removal



```
# fair calculates bias for train_time/train_init and drops these indices from hindcast
hindcast.remove_bias(
    how="additive_mean",
    alignment=metric_kwargs["alignment"],
    train_test_split="fair",
    train_time=slice("1982", "1998"),
).sel(model="GFDL-CM2p5-FLOR-A06").plot()
```

remove mean bias while separating train and test data

multi-dimensional: on all models at once



Easily plotting: `time` = `init` + `lead` with training `init`s dropped

Demo: Calculate skill for different `train_test_split` strategies

```python
metric_kwargs = dict(metric="rmse", alignment="same_verifs", dim="init", comparison="e2o", skipna=True)
```

`verify()` API

```python
train_test_split = ["unfair", "unfair-cv", "fair"]  # different train_test_split methods to compare
verify_init = slice("1999", None)
skill_train_test_split = [hindcast.sel(init=verify_init).verify(**metric_kwargs)]
skill_train_test_split.append(
    hindcast.remove_bias(
        how="additive_mean",
        alignment=metric_kwargs["alignment"],
        train_test_split="unfair",
    ).sel(init=verify_init).verify(**metric_kwargs)
)
skill_train_test_split.append(
    hindcast.remove_bias(
        how="additive_mean",
        alignment=metric_kwargs["alignment"],
        train_test_split="unfair-cv",
        cv="LOO",  # leave-one-out
    ).sel(init=verify_init).verify(**metric_kwargs)
)
skill_train_test_split.append(
    hindcast.remove_bias(
        how="additive_mean",
        alignment=metric_kwargs["alignment"],
        train_test_split="fair",
        train_time=slice("1982", "1998"),
    ).sel(init=verify_init).verify(**metric_kwargs)
)

skill_train_test_split = xr.concat(skill_train_test_split, "train_test_split")[v].assign_coords(train_test_split=["None"] + train_test_split)
```

First bias removal, then skill

use same `inits` for bias as for `verify()`

as above, but leave out given `init`

use different `inits` for bias correction than for `verify()`

`xr.DataArray.plot()`

```python
skill_train_test_split.plot(hue="model", col="train_test_split", x="lead")
plt.ylim([0, 1.4])
plt.suptitle(f"NMME Nino3.4 {metric_kwargs['metric'].upper()} for different train_test_split", y=1.0)
```

Demo II: Hindcast data NMME Nino 3.4 vs. Verification NOAA OISST

- Bias correction trained during init 1982-1998, verification for remaining inits
- +/- 2 month predictable signal depending on bias correction with `fair` splits

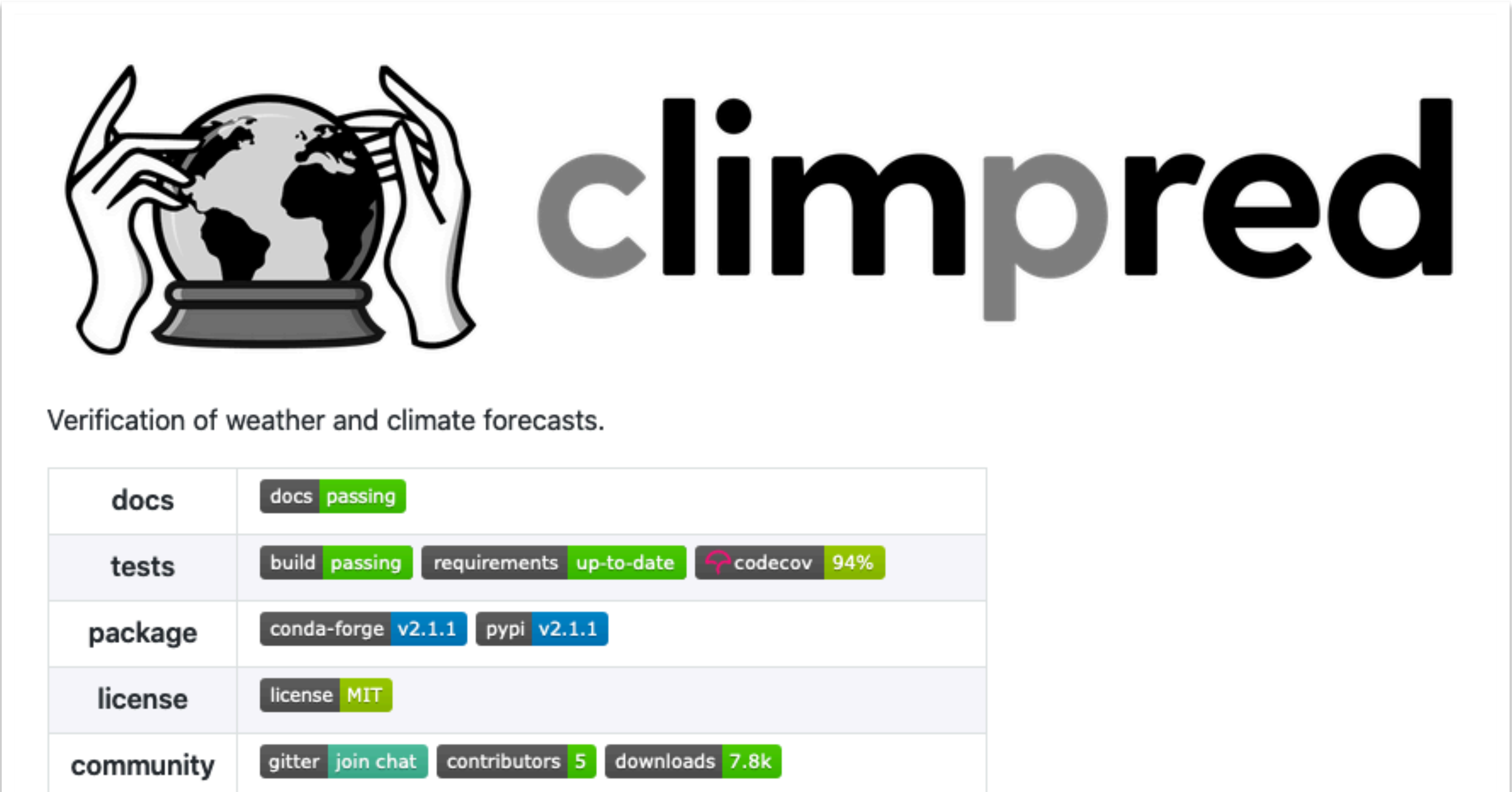
NMME Nino3.4 SST RMSE for different bias reduction methods
bias_correction = None
bias_correction = additive_...
bias_correction = Detrended...
bias_correction = Empirical...
bias_correction = QuantileD...
bias_correction = Scaling
bias_correction = modified_...
bias_correction = basic_qua...
Sea Surface Temperature [C]
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
Lead [months]
0.0
2.5
5.0
7.5
10.0
model
NCEP-CFSv2
NCEP-CFSv1
NCAR-CESM1
NASA-GEOSS2S
CMC1-CanCM3
CanCM4i
NASA-GMAO
GFDL-CM2p5-FLOR-B01
GFDL-CM2p5-FLOR-A06
GFDL-CM2p1-aer04
COLA-RSMAS-CCSM4
GEM-NEMO

launch binder

climpred is open-source.

If you waited for long to switch from NCL to python, climpred might be what you have been waiting for.

- Call for switching to python: The `xarray` community is awesome.
- Call for new users and developers
  - NWP (lead minutes to hours)
  - S2S (lead days to weeks)
  - S2D (lead months to decades)
- Call for feedback.
- Call for new contributors.



climpred

Verification of weather and climate forecasts.

docs	docs passing
tests	build passing requirements up-to-date codecov 94%
package	conda-forge v2.1.1 pypi v2.1.1
license	license MIT
community	gitter join chat contributors 5 downloads 7.8k
tutorials	climpred example gallery climpred workshop climpred cloud demo

Installation

You can install the latest release of `climpred` using `pip` or `conda`:

```
pip install climpred
```

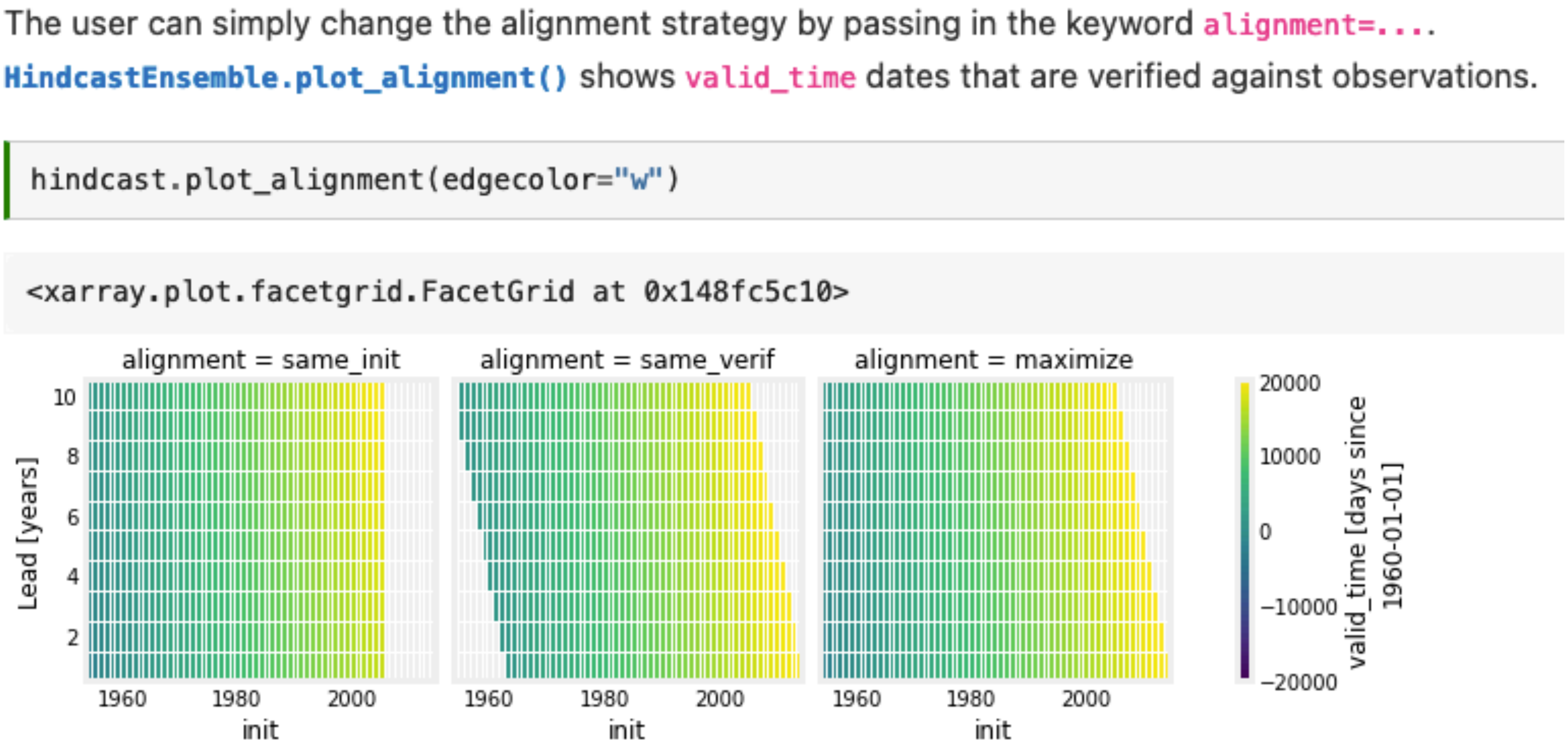
```
conda install -c conda-forge climpred
```

Documentation: https://climpred.readthedocs.io/en/latest/

Github repository: https://github.com/pangeo-data/climpred

Backup slides

The `alignment` keyword in `verify()` and `bootstrap()`



The user can simply change the alignment strategy by passing in the keyword `alignment=...`.

`HindcastEnsemble.plot_alignment()` shows `valid_time` dates that are verified against observations.

```
hindcast.plot_alignment(edgecolor="w")
```

```
<xarray.plot.facetgrid.FacetGrid at 0x148fc5c10>
```