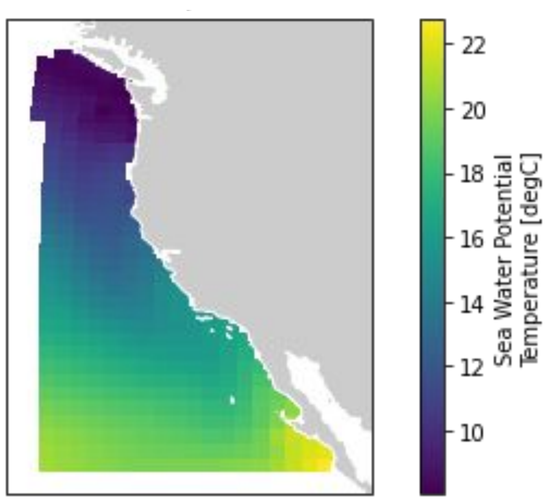
The xgcm Python package makes ocean model processing easier, better, faster, and shorter

Dianne Deauna[1], Anderson Banihirwe[2], Julius Busecke[3], and Deepak Cherian[2]
[1]University of Hawai'i at Mānoa, [2]National Center for Atmospheric Research, [3]Columbia University

NCAR

Interactive Jupyter notebook:
bit.ly/xgcm_demo_siparcs2021

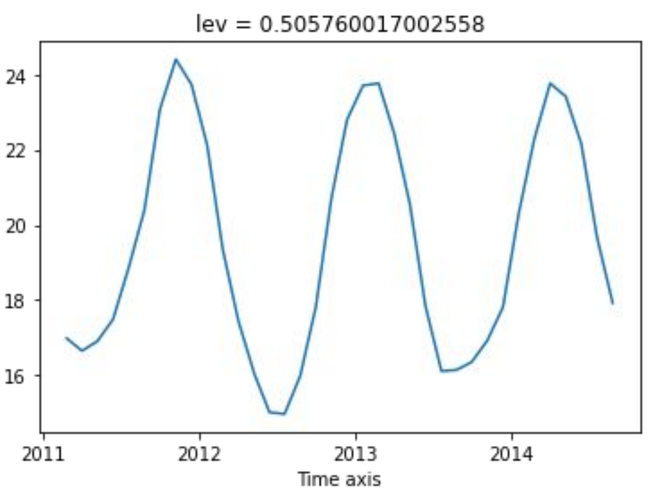
Input: model surface temperature

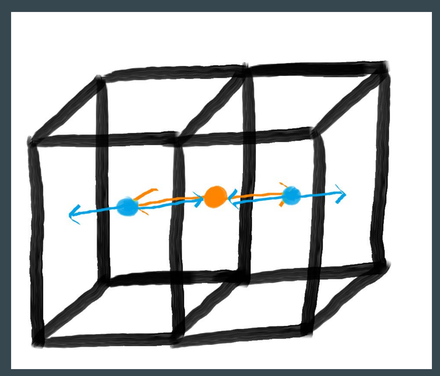



```
grid.average(temperature, ['X', 'Y'])
```

Ocean modeling is mostly representing the ocean as cubes, where vector and scalar quantities are computed at different positions within them. The **xgcm** Python package can account for these model geometries (***metrics***) and do operations such as area-weighted average with minimal, intuitive code.

Output: time series of area-weighted temperature





Model ***metrics*** refer to the length, width, area, and volume of grid cells within circulation models. For example, the distance along the x-axis between temperature points (blue), differs from the distance between u-velocity points (yellow).

The xgcm package relies on the knowledge of model metrics to run operations such as area-weighted average temperature. Over the summer, the following features were updated to improve xgcm's ability to handle metrics:

set_metrics()	• Enables overwriting of previously assigned metrics, and allows for assigning multiple metrics on the same axis with different dimensions
interp_like()	• Allows for the interpolation of a data array to the positions of another data array (e.g., given the distance along x-axis for temperature, we can interpolate this to the distance along the x-axis for u-velocity)
get_metric()	• Selects for the metric required for a data variable along a specified axis for grid-aware operations and allows for automatic interpolation of missing metrics from available metric values on surrounding positions