

◀ (/nDarrays/08-out-of-core-computation/)

## Multidimensional Arrays (/nDarrays/)

# masking

➤ (/nDarrays/10-wrap-up/)

### 🔍 Overview

**Teaching:** 10 min

**Exercises:** 5 min

#### Questions

- What is masking and how can it be used to analyze portions of a dataset

#### Objectives

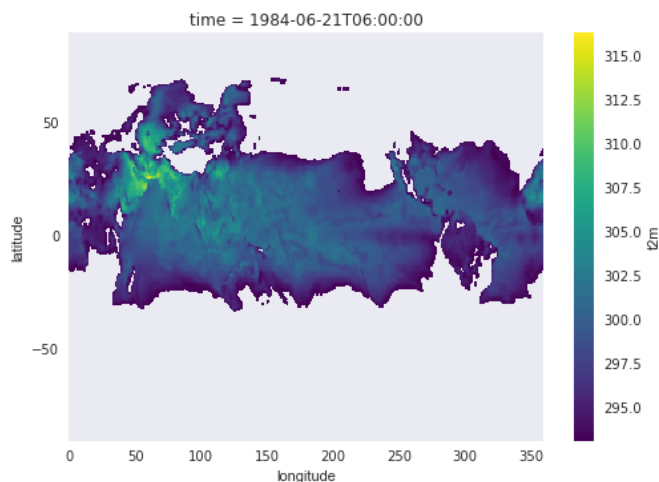
- Learn the concepts of masking with xarray.

## Masking with where:

So far we have used indexing to return subsets of the original. The subset array shape will be different from the original. However, we often want to retain the array shape and mask out some observations. There are applications here in remote sensing, land cover modeling, etc.

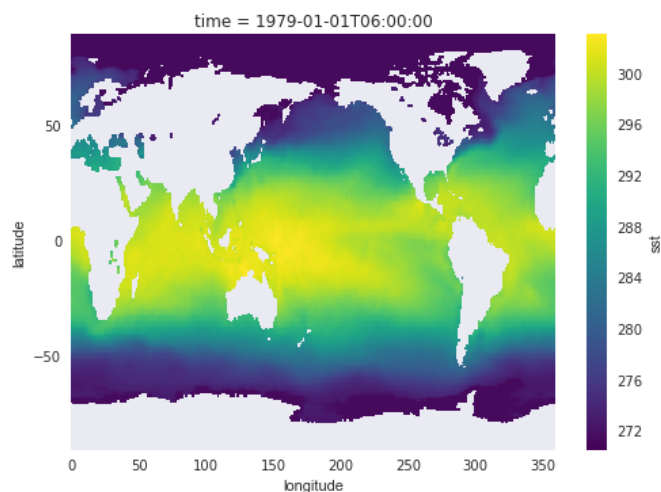
Suppose we need to determine which grid cells had temperatures > 20 deg C on June 21, 1984? We will use `where()` (<http://xarray.pydata.org/en/stable/indexing.html#masking-with-where>) for this selection:

```
ds.sel(time="1984-06-21")['t2m'].where(ds.t2m > 293.15).plot()
```



Another common Earth science application is to create land cover masks. Let's use the sea surface temperature field (sst) to build a land and ocean mask. We'll assign land a value of 1, and ocean a value of 2 (arbitrary). Note that the sst field currently has NaN for all land surfaces:

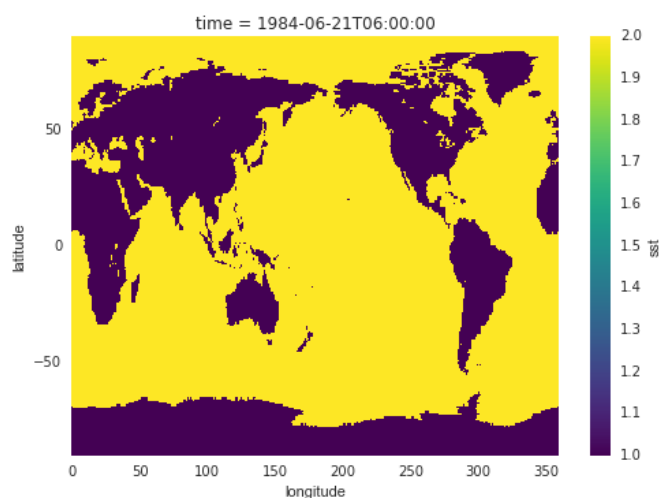
```
ds.sst.isel(time=0).plot()
```



## Buliding the mask:

Here we'll use some lower-level numpy commands to build the mask (and we'll need to import the numpy library). The mask number depends on whether the cells are finite or NaN:

```
import numpy as np
mask_ocean = 2 * np.ones((ds.dims['latitude'], ds.dims['longitude'])) * np.isfinite(ds.sst.isel(time=0))
mask_land = 1 * np.ones((ds.dims['latitude'], ds.dims['longitude'])) * np.isnan(ds.sst.isel(time=0))
mask_array = mask_ocean + mask_land
mask_array.plot()
```



## Mask as Coordinates

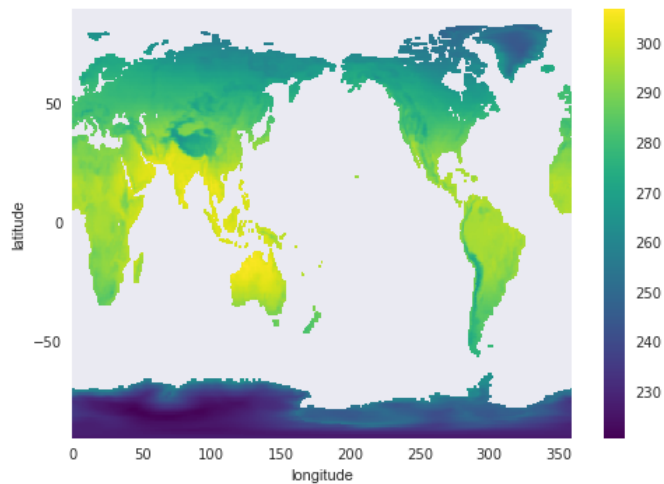
We can keep the mask as a separate array entity, or, if we are using it routinely, there are advantages to adding it (<http://xarray.pydata.org/en/stable/data-structures.html#dataarray-coordinates>) as a coordinate to the DataArray:


```
ds.coords['mask'] = (('latitude', 'longitude'), mask_array)
ds
```

```
<xarray.Dataset>
Dimensions:  (latitude: 241, longitude: 480, time: 13361)
Coordinates:
  * longitude (longitude) float32 0.0 0.75 1.5 2.25 3.0 3.75 4.5 5.25 6.0 ...
  * latitude (latitude) float32 90.0 89.25 88.5 87.75 87.0 86.25 85.5 ...
  * time (time) datetime64[ns] 1979-01-01T06:00:00 1979-01-02T06:00:00 ...
  mask (latitude, longitude) float64 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 ...
Data variables:
  v10 (time, latitude, longitude) float64 -7.053 -7.077 -7.1 -7.123 ...
  sst (time, latitude, longitude) float64 271.5 271.5 271.5 271.5 ...
  t2m (time, latitude, longitude) float64 242.0 242.0 242.0 242.0 ...
  u10 (time, latitude, longitude) float64 2.418 2.325 2.233 2.142 ...
Attributes:
  history: 2015-10-29 17:20:47 GMT by grib_to_netcdf-1.13.1: grib_to_netcdf /data/data04/scratch/netcdf-atls00-a562
cefd8a29a7288fa0b8b7f9413f7-eG3SK6.target -o /data/data04/scratch/netcdf-atls00-a562cefd8a29a7288fa0b8b7f9413f7-udf
dnE.nc -utime
Conventions: CF-1.0
```

Now that the mask is integrated into the coordinates, we can easily apply the mask using `where()`. We can integrate this with statistical functions operating on the array:

```
with ProgressBar():  
    ds['t2m'].mean('time').where(ds.mask == 1).plot()
```



✎ Calculating a climate index 

### ❗ Key Points

- xarray provides tools for creating and analyzing masked data.

Copyright © 2016 Geohackweek (<https://geohackweek.github.io>)

Source (<https://github.com/geohackweek/nDarrays/>) / Contributing (<https://github.com/geohackweek/nDarrays/blob/gh-pages/CONTRIBUTING.md>) / Contact (<mailto:arendta@uw.edu>)