xarray

Working with netCDF files (N-dimensional arrays)

ESS 116 | Fall 2024

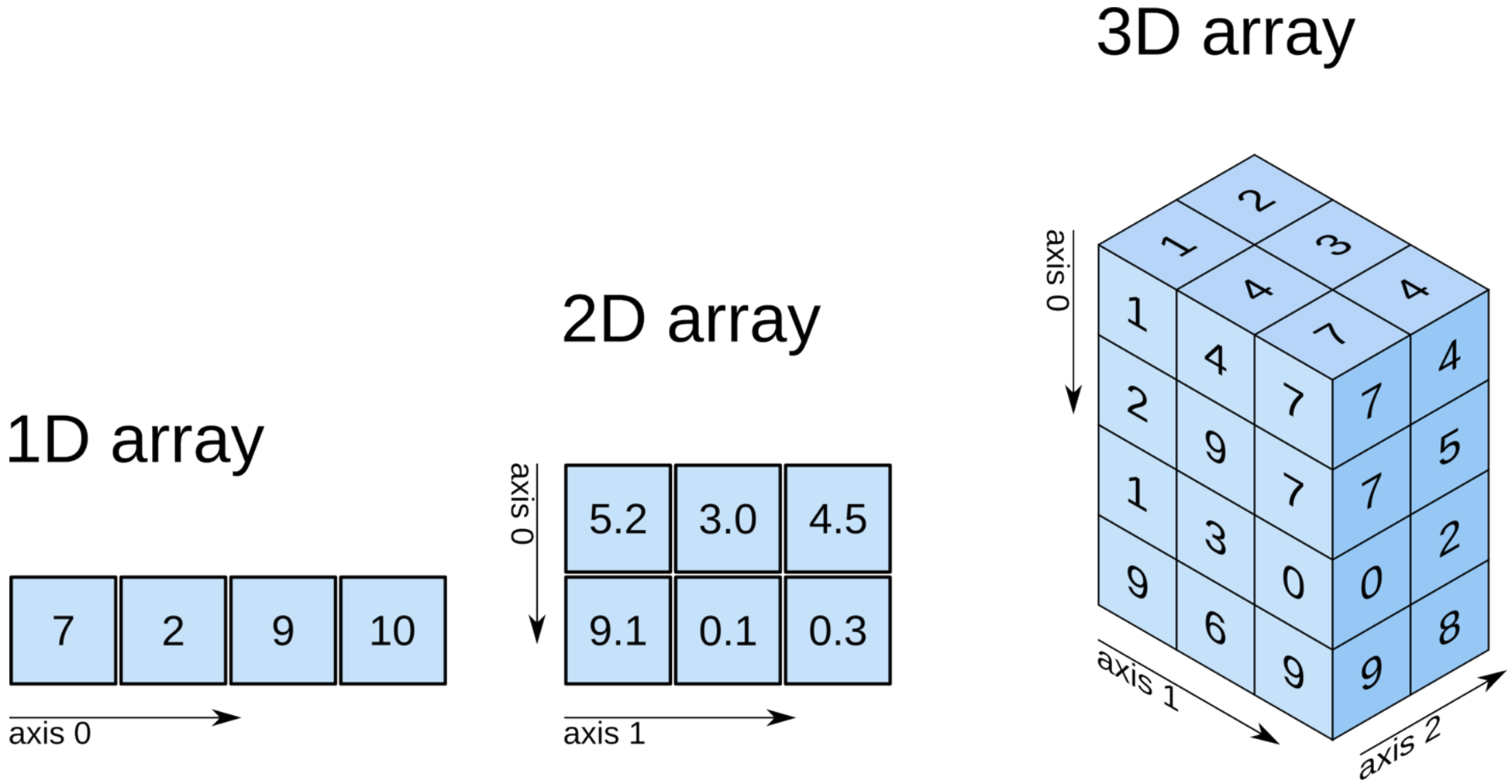
Prof. Henri Drake, Prof. Jane Baldwin, and Prof. Michael Pritchard

(Modified from Ethan Campbell and Katy Christensen's materials for UW's Ocean 215)

What we'll cover in this lesson

1. **`xarray`: `DataArray` and `Dataset` objects; netCDF files**
2. `xarray`: working with higher-dimensional data

Array values are identified by their coordinates along axes

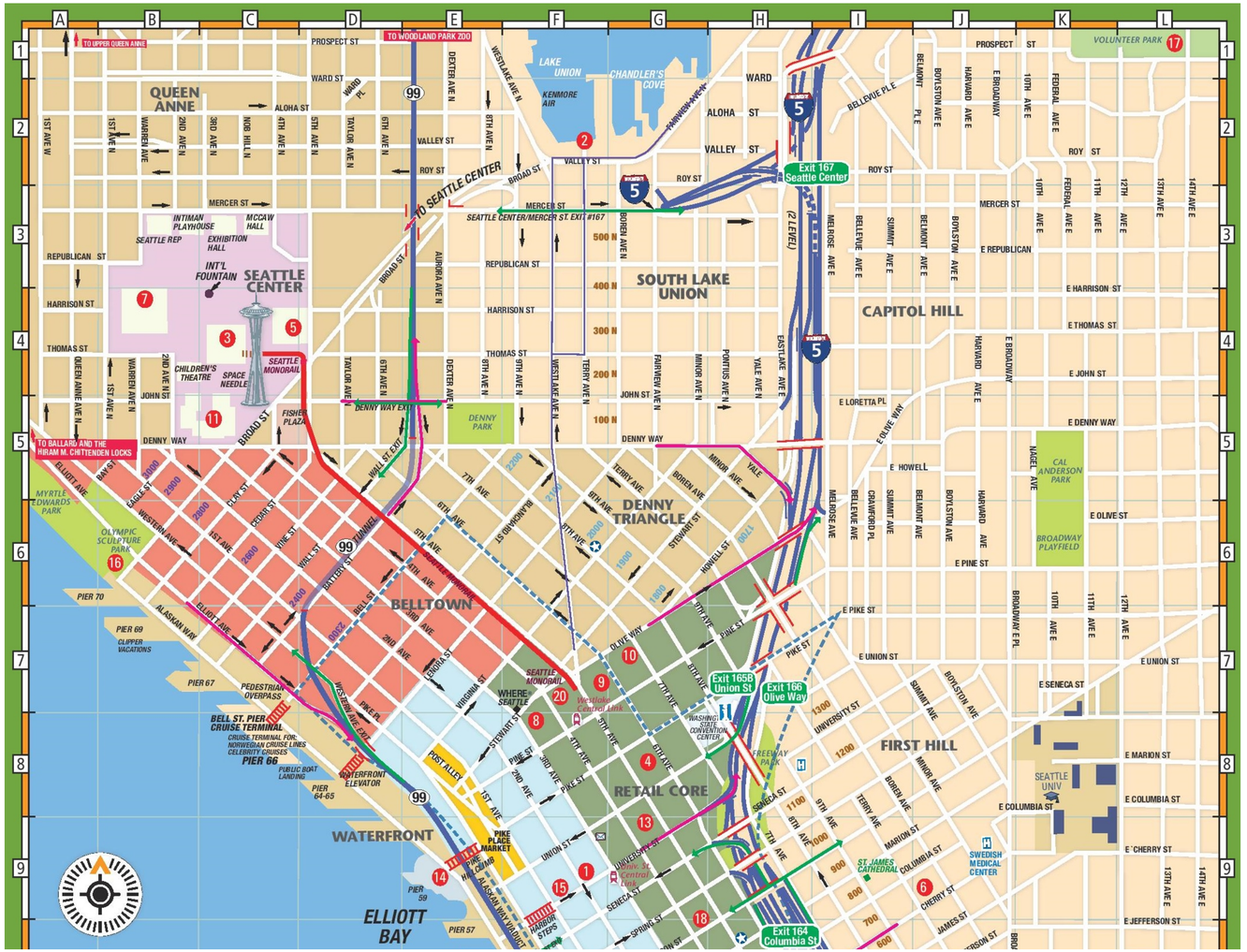


Source: O'Reilly

Coordinates along axes let us identify real-world locations

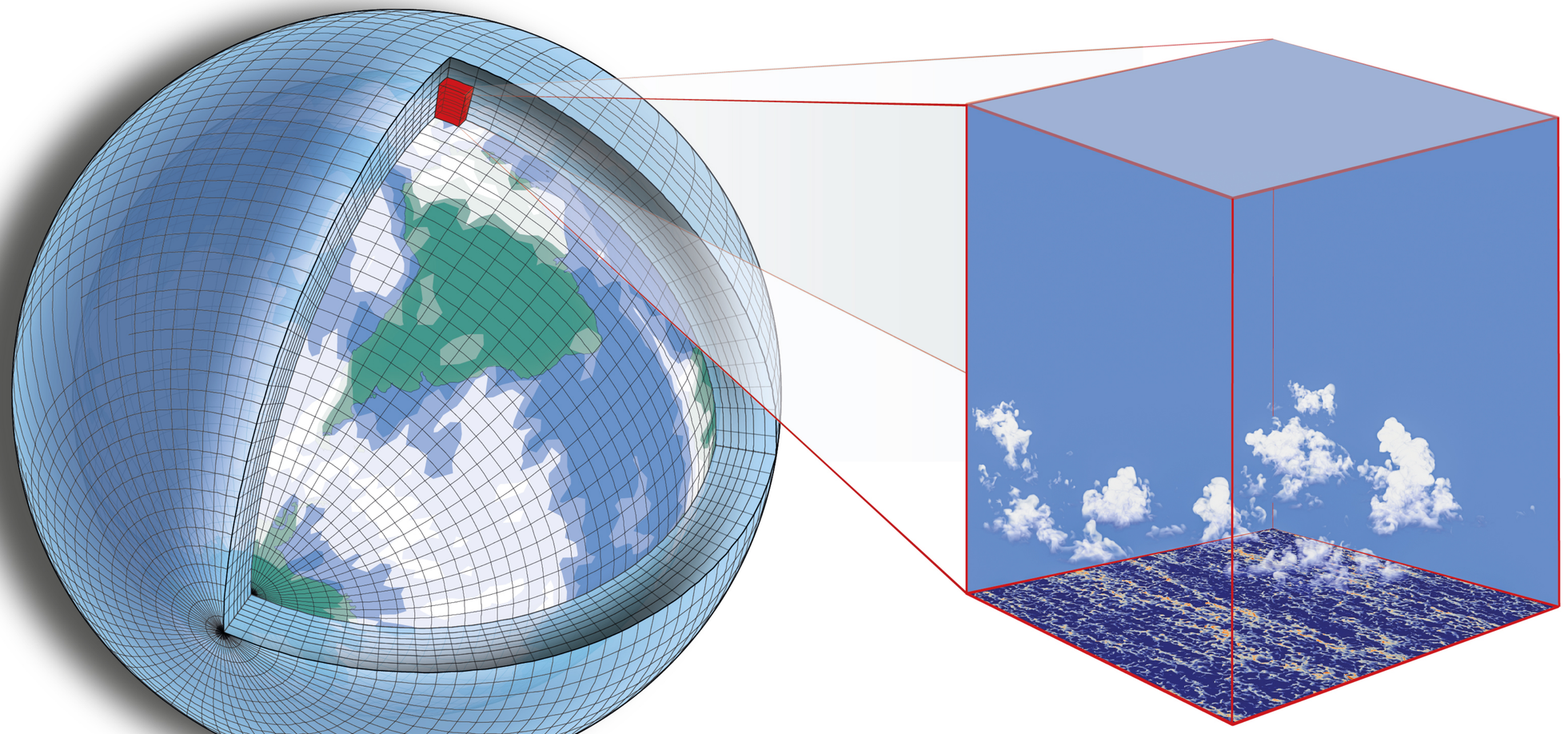
W ⟷ **E**

N ↕ **S**



Source: Where Maps

Models, satellites, and data providers divide the world into grid cells



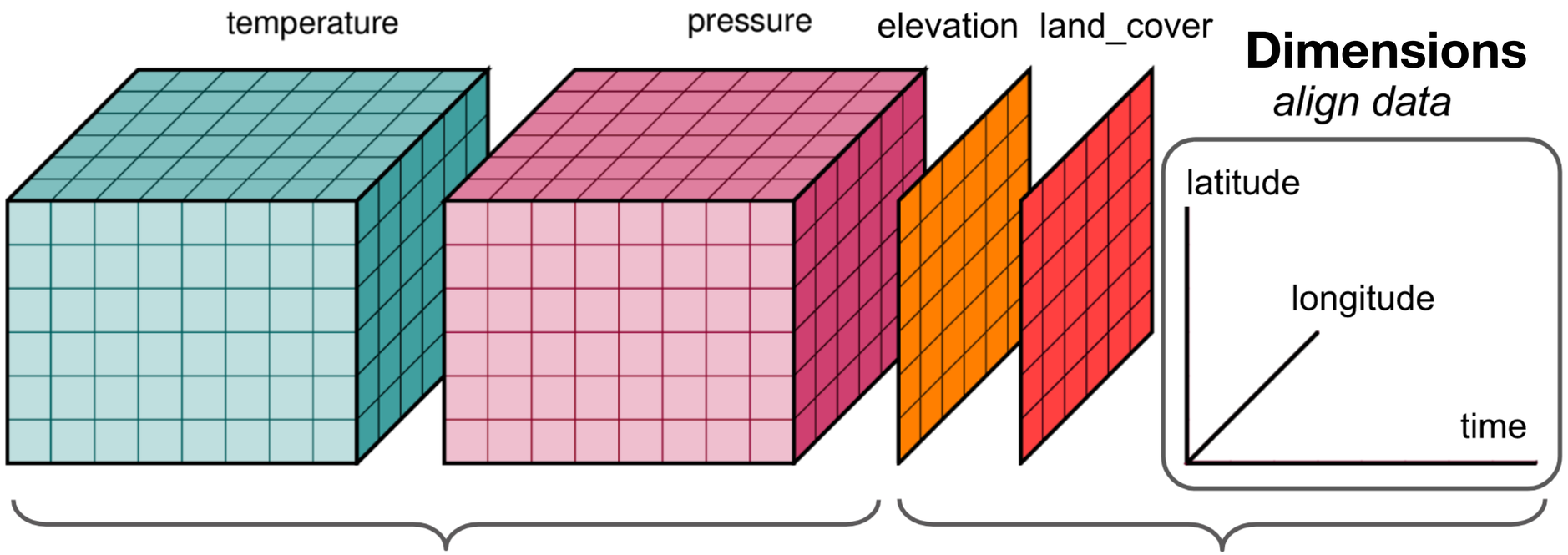
Source: Caltech

`xarray` lets us deal with gridded data...

... and gridded data is usually provided in a **netCDF file (.nc)**

`xarray` lets us deal with gridded data...

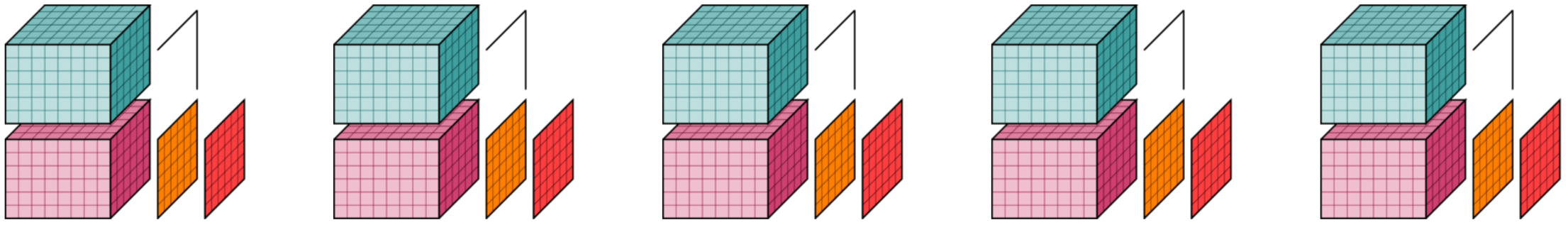
... and gridded data is usually provided in a **netCDF file (.nc)**

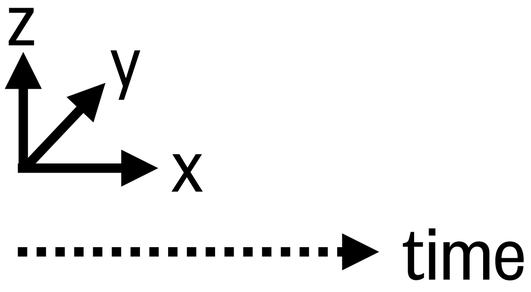


Source: Matthew Rocklin

4-D data is usually 3-D in space (x, y, z) + time

January **February** **March** **April** **May** **...**





Source: xarray

Importing `xarray` and loading the `netCDF4` library

```
import xarray as xr
```

```
!pip install netcdf4
```

You should only have to run this line once per Colab notebook.

Importing `xarray` and loading the `netCDF4` library

```
import xarray as xr
```

```
#!pip install netcdf4
```

You should only have to run this line once per Colab notebook.

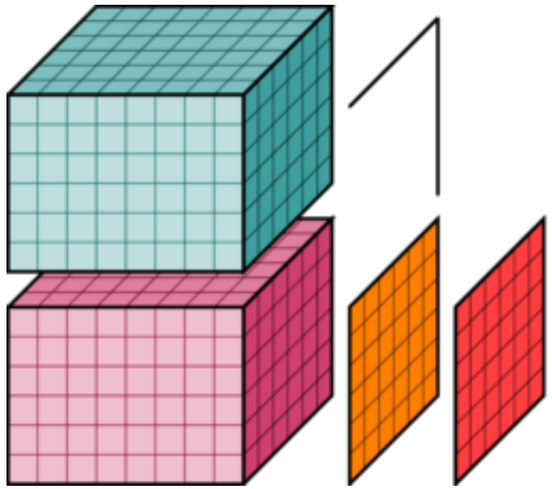
Loading a netCDF file as an `xarray Dataset`

Read a netCDF file as a `Dataset`:

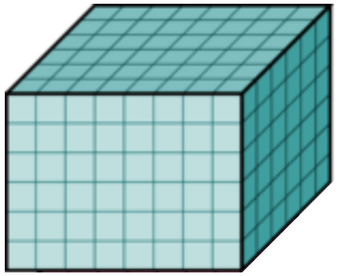
```
data = xr.open_dataset('filepath/including/filename.nc')
```

→ Documentation (API): http://xarray.pydata.org/en/stable/generated/xarray.open_dataset.html

`Dataset` object:



`DataArray` object:



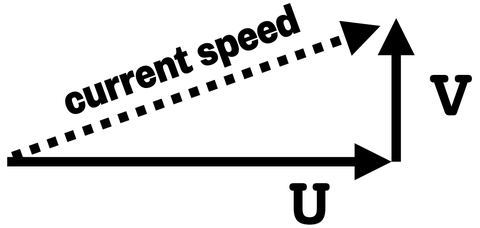
Demo: Southern Ocean current velocities from a climate model

File (~400 MB):
`bsose_monthly_velocities.nc`

Data source:
B-SOSE (Southern Ocean State Estimate) model output

Data resolution:
Time: monthly for 2012
Horizontal: 1/3° lat-lon grid
Vertical: 13 depth levels

Variables:
`U`: eastward velocity
`V`: northward velocity



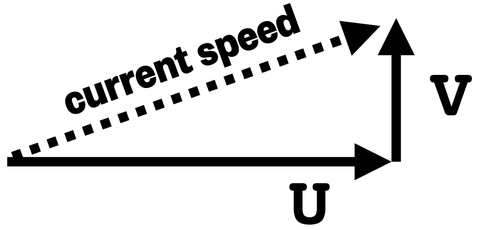
Demo: Southern Ocean current velocities from a climate model

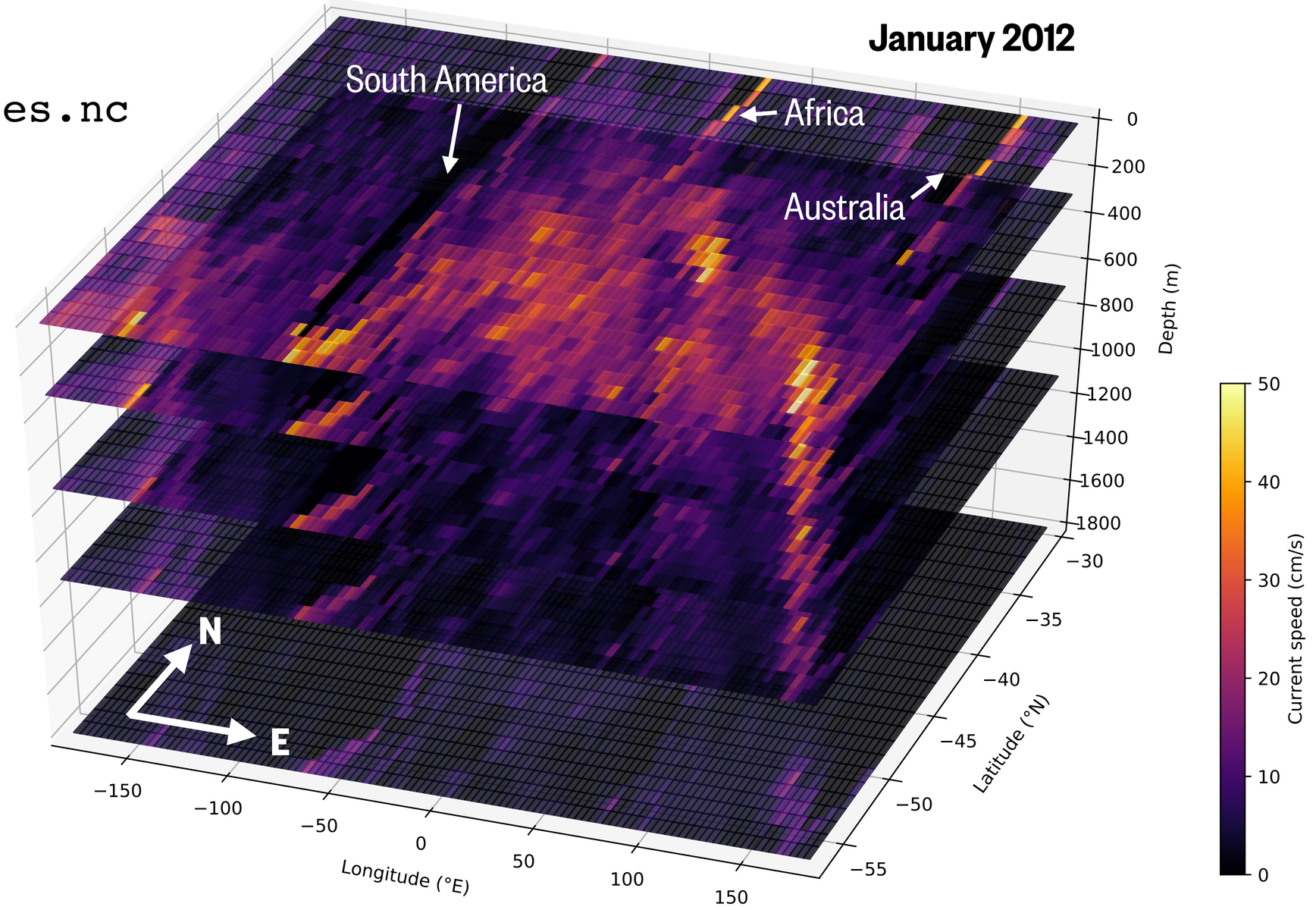
File (~400 MB):
`bsose_monthly_velocities.nc`

Data source:
B-SOSE (Southern Ocean State Estimate) model output

Data resolution:
Time: monthly for 2012
Horizontal: 1/3° lat-lon grid
Vertical: 13 depth levels

Variables:
U: eastward velocity
V: northward velocity





What we'll cover in this lesson

1. `xarray`: `DataArray` **and** `Dataset` **objects; netCDF files**

2. **`xarray`: working with higher-dimensional data**

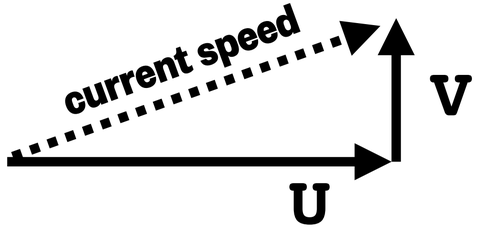
Demo: Southern Ocean current velocities from a climate model

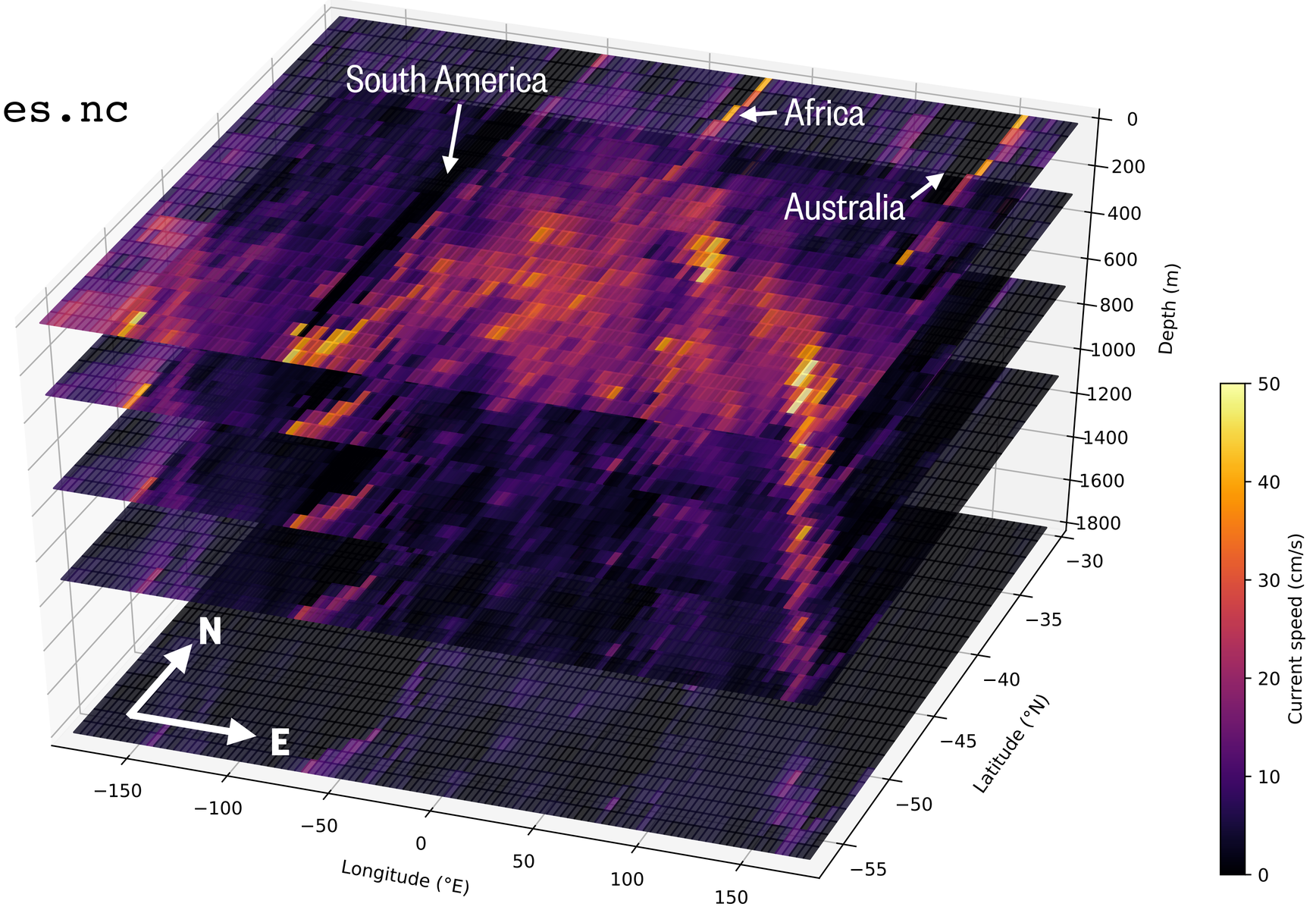
File (~400 MB):
`bsose_monthly_velocities.nc`

Data source:
B-SOSE (Southern Ocean State Estimate) model output

Data resolution:
Time: monthly for 2012
Horizontal: 1/3° lat-lon grid
Vertical: 13 depth levels

Variables:
U: eastward velocity
V: northward velocity





Getting information about a `Dataset`

`display(`<Dataset variable>**`)`**

xarray.Dataset

▶ Dimensions: (**depth**: 13, **lat**: 294, **lon**: 1080, **time**: 12)

▼ Coordinates:

time	(time)	datetime64[ns]	2012-01-30T20:00:00 ... 2012-12-30T1…
lat	(lat)	float32	-77.96525 -77.89555 ... -29.789328
lon	(lon)	float32	-179.66667 -179.33333 ... 180.0
depth	(depth)	float32	2.1 26.25 65.0 ... 3000.0 4600.0

▼ Data variables:

U	(time, depth, lat, lon)	float32	0.0 0.0 0.0 0.0 ... 0.0 0.0 0.0 0.0
V	(time, depth, lat, lon)	float32	0.0 0.0 0.0 0.0 ... 0.0 0.0 0.0 0.0

▶ Attributes: (0)

Extracting a single variable's `DataArray` from a `Dataset`

Syntax: <Dataset> **[** <variable name as a string> **]**

```
1 display(data['U'])
```

xarray.DataArray 'U' (**time**: 12, **depth**: 13, **lat**: 294, **lon**: 1080)

...

▼ Coordinates:

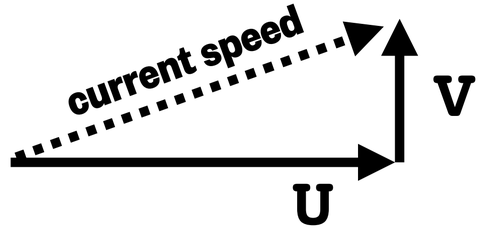
time	(time)	datetime64[ns]	2012-01-30T20:00:00 ... 2012-12-30T12:00:00	
lat	(lat)	float32	-77.96525 -77.89555 ... -29.789328	
lon	(lon)	float32	-179.66667 -179.33333 ... 180.0	
depth	(depth)	float32	2.1 26.25 65.0 ... 3000.0 4600.0	

▼ Attributes:

units :	m/s
long_name :	Zonal Component of Velocity (m/s)
standard_name :	UVEL
mate :	VVEL

Mathematical calculations with `xarray` objects works like NumPy

```
# Example: calculate current speed using Pythagorean theorem:
#          speed = sqrt(U^2 + V^2)
speed = (data['U']**2 + data['V']**2)**0.5
display(speed)
```



xarray.DataArray (**time**: 12, **depth**: 13, **lat**: 294, **lon**: 1080)

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 ... 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

▼ Coordinates:

time	(time)	datetime64[ns]	2012-01-30T20:00:00 ... 2012-12-30T12:00:00
lat	(lat)	float32	-77.96525 -77.89555 ... -29.789328
lon	(lon)	float32	-179.66667 -179.33333 ... 180.0
depth	(depth)	float32	2.1 26.25 65.0 ... 3000.0 4600.0

▶ Attributes: (0)

Accessing and changing attributes (metadata) of `xarray` objects

Syntax: <Dataset or DataArray>**`.attrs`** is a Python dictionary (a set of keys and values)

```
1 print(data['U'].attrs)
```

```
{'units': 'm/s', 'long_name': 'Zonal Component of Velocity (m/s)', 'standard_name': 'UVEL', 'mate': 'VVEL'}
```

Syntax: <Dataset or DataArray>**`.attrs[`**<attribute name as string>**`]`** gets the value of an attribute

```
1 print(data['U'].attrs['units'])
```

```
m/s
```

Syntax: <Dataset or DataArray>**`.attrs[`**<attribute name>**`]`** **`=`** <new value> changes its value

```
1 data['U'].attrs['units'] = 'meters/second'
```

Selecting data from `xarray` objects using `.isel()` (selection by integer index)

<DataArray or Dataset>`.isel(`<coordinate name>=<a single integer index>

OR <list or array of indices>

OR `slice(`<start>`,`<stop>`),...)`

Like Python/NumPy slicing, the end value is exclusive!

Example:

```
1 data['U'].isel(time=0,lat=200,lon=500,depth=0)
```

xarray.DataArray 'U'

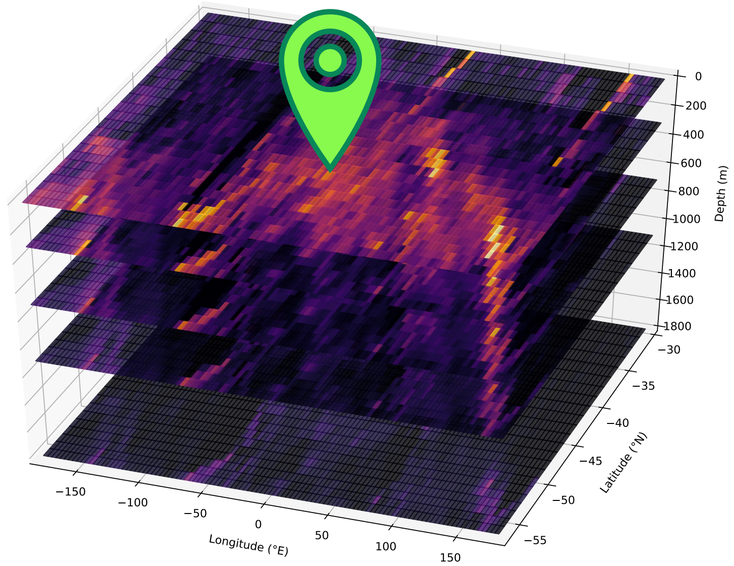
0.12588988

▼ Coordinates:

time	()	datetime64[ns]	2012-01-30T20:00:00
lat	()	float32	-52.70605
lon	()	float32	-13.0
depth	()	float32	2.1

▼ Attributes:

units :	m/s
long_name :	Zonal Component of Velocity (m/s)
standard_name :	UVEL
mate :	VVEL



Convert a single-value `Dataset` to a number using `float()` or `item()`

<DataArray or Dataset>**`.isel(...).item()`**

OR...

`float(`<DataArray or Dataset>**`.isel(...))`**

Example:

```
1 data['U'].isel(time=0,lat=200,lon=500,depth=0).item()
```

```
0.1258898824453354
```

```
1 float(data['U'].isel(time=0,lat=200,lon=500,depth=0))
```

```
0.1258898824453354
```

Selecting data from `xarray` objects using `.isel()` (selection by integer index)

<DataArray or Dataset>`.isel(`<coordinate name>=<a single integer index>

OR <list or array of indices>

OR `slice(`<start>`,`<stop>`),...)`

Example:

```
1 data['U'].isel(time=0,lat=200,lon=500,depth=[0,1,2,3,4])
2 data['U'].isel(time=0,lat=200,lon=500,depth=slice(0,5))
```

xarray.DataArray 'U' (**depth**: 5)

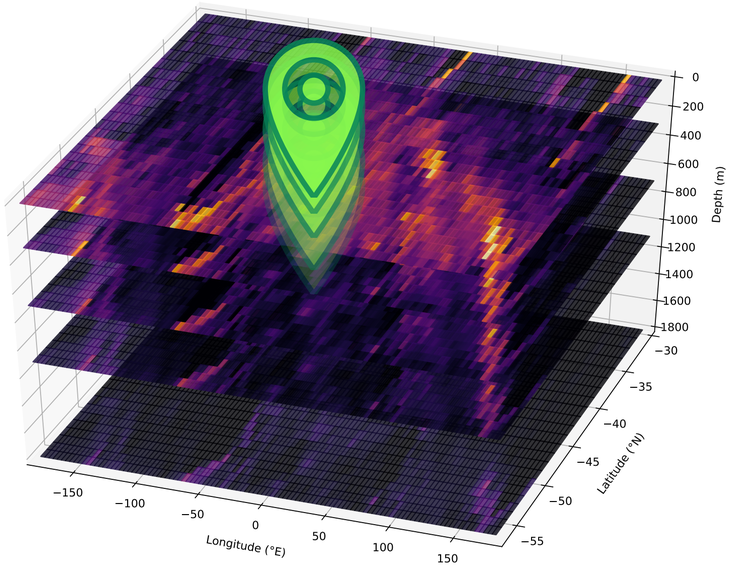
0.12588988 0.050398406 0.057173315 0.061554562 0.057381995

▼ Coordinates:

time	()	datetime64[ns]	2012-01-30T20:00:00
lat	()	float32	-52.70605
lon	()	float32	-13.0
depth	(depth)	float32	2.1 26.25 65.0 105.0 146.5

▼ Attributes:

units :	m/s
long_name :	Zonal Component of Velocity (m/s)
standard_name :	UVEL
mate :	VVEL



Convert a `Dataset` with multiple values to a NumPy array using `.values`

<DataArray or Dataset>**`.values`**

<DataArray or Dataset>**`.isel(…).values`**

Example:

```
1 data['U'].isel(time=0,lat=200,lon=500,depth=slice(0,5)).values
```

```
array([0.12588988, 0.05039841, 0.05717332, 0.06155456, 0.057382  ],
      dtype=float32)
```

Selecting data from `xarray` objects using `.sel()` (selection by coordinate value)

<DataArray or Dataset>`.sel(`<coordinate name>=<a single coordinate value>

OR <list or array of coordinate values>

OR `slice(`<start>`,`<stop>`),...)`

Unlike Python/NumPy slicing, the end value is inclusive!

Example:

```
1 data['U'].sel(time=datetime(2012,1,30,20),lat=-52.70605,lon=-13.0,depth=2.1)
```

xarray.DataArray 'U'

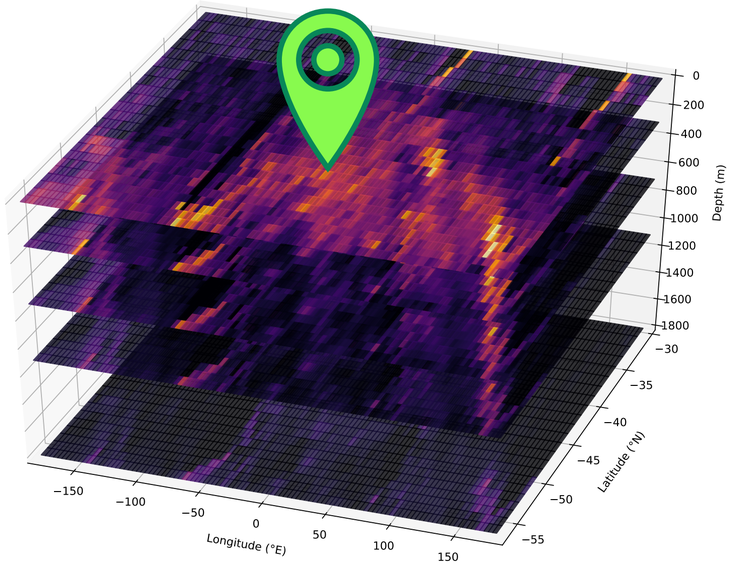
0.12588988

▼ Coordinates:

time	()	datetime64[ns]	2012-01-30T20:00:00
lat	()	float32	-52.70605
lon	()	float32	-13.0
depth	()	float32	2.1

▼ Attributes:

units :	meters/second
long_name :	Zonal Component of Velocity (m/s)
standard_name :	UVEL
mate :	VVEL



Selecting data from `xarray` objects using `.sel()` (selection by coordinate value)

<DataArray or Dataset>**`.sel(`**<coordinate name>=<a single coordinate value>

OR <list or array of coordinate values>

OR `slice(`<start>`,`<stop>`),...)`

Example:

```
1 data['U'].sel(time=datetime(2012,1,30,20,0,0),lat=-52.70605,lon=-13.0,depth=slice(2,147))
```

xarray.DataArray 'U' (**depth**: 5)

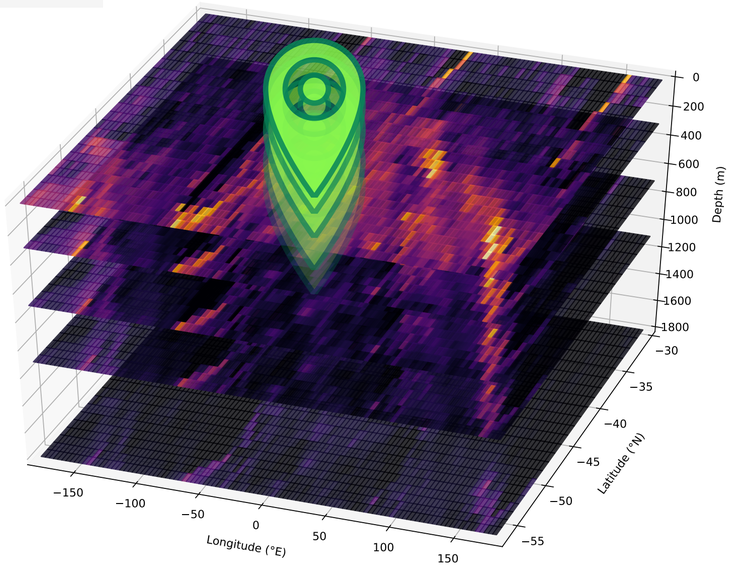
0.12588988 0.050398406 0.057173315 0.061554562 0.057381995

▼Coordinates:

time	()	datetime64[ns]	2012-01-30T20:00:00	
lat	()	float32	-52.70605	
lon	()	float32	-13.0	
depth	(depth)	float32	2.1 26.25 65.0 105.0 146.5	

▼Attributes:

units :	meters/second
long_name :	Zonal Component of Velocity (m/s)
standard_name :	UVEL
mate :	VVEL



Selecting data from `xarray` objects using `.sel()` (selection by coordinate value)

Use method='nearest' when you don't know the exact coordinate values...

<DataArray or Dataset>`.sel(`<coordinate name>=<a single coordinate value>`, ..., method='nearest')`

Example:

```
1 data['U'].sel(time=datetime(2012,1,30),lat=-53,lon=-13,depth=2,method='nearest')
```

xarray.DataArray 'U'

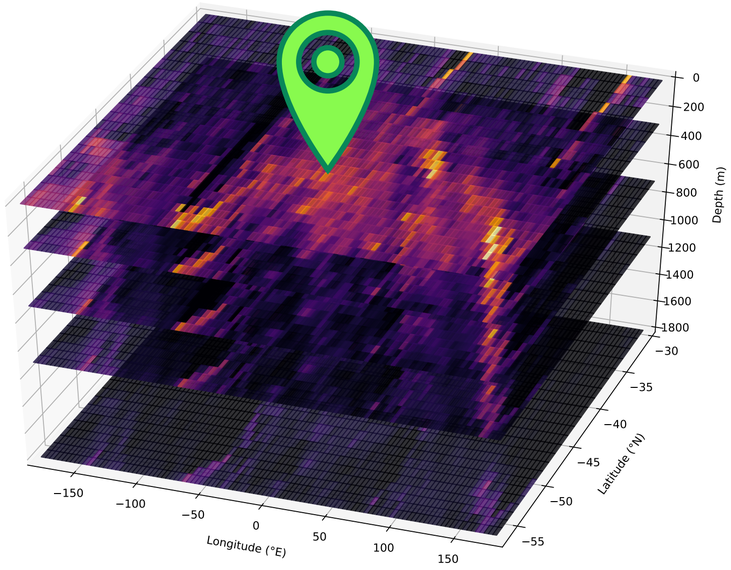
0.12865335

▼ Coordinates:

time	()	datetime64[ns]	2012-01-30T20:00:00
lat	()	float32	-52.90755
lon	()	float32	-13.0
depth	()	float32	2.1

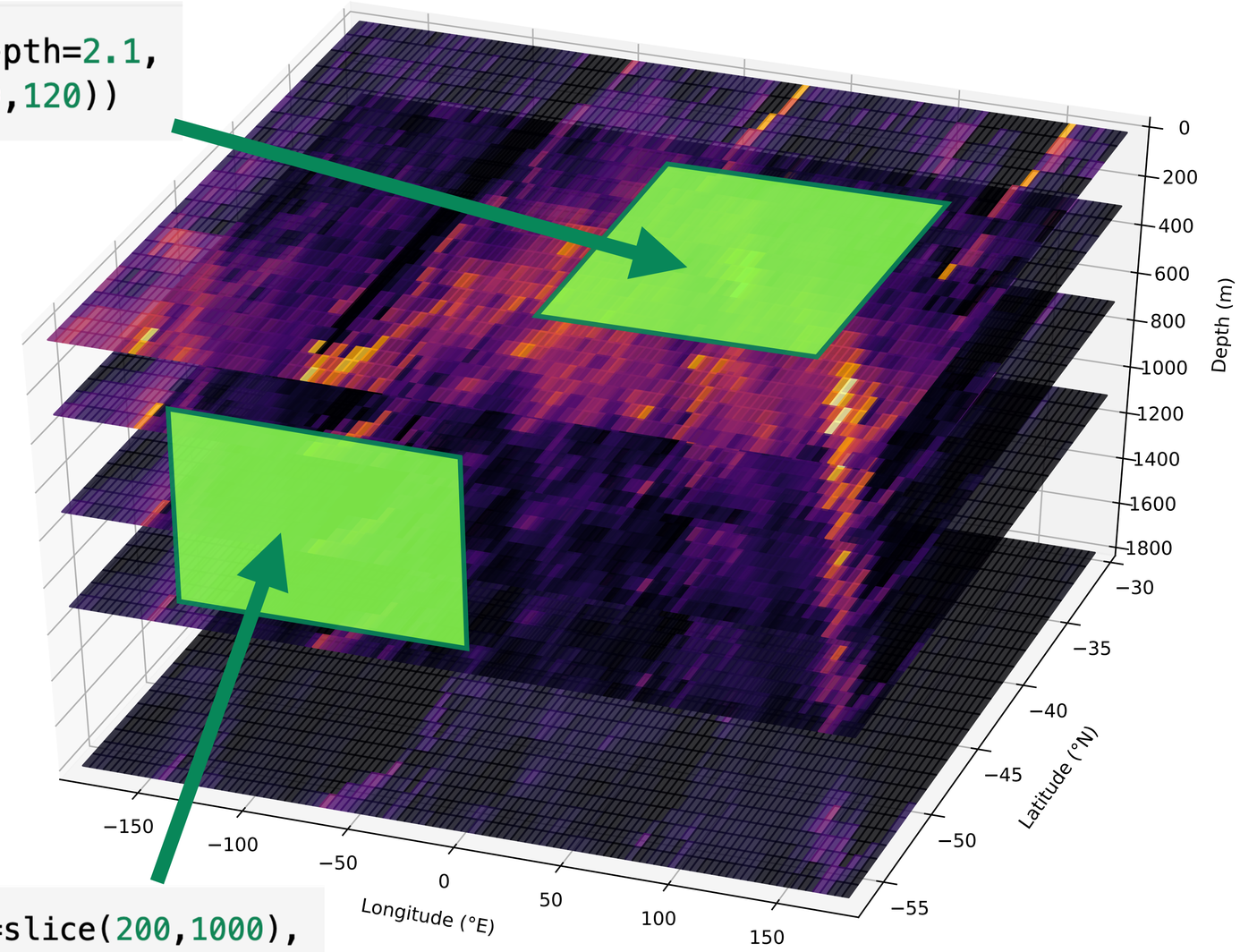
▼ Attributes:

units :	meters/second
long_name :	Zonal Component of Velocity (m/s)
standard_name :	UVEL
mate :	VVEL



Selecting data from `xarray` objects using `.sel()` (selection by coordinate value)

```
data['U'].sel(time=datetime(2012,1,30,20),depth=2.1,
              lat=slice(-50,-40),lon=slice(0,120))
```

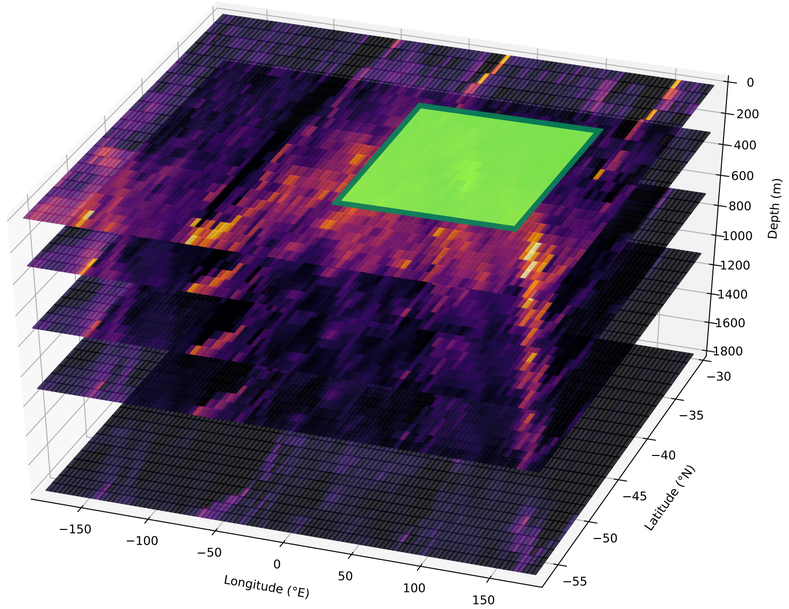



```
data['U'].sel(time=datetime(2012,1,30,20),depth=slice(200,1000),
              lon=slice(-120,0)).sel(lat=-57,method='nearest')
```

Applying NumPy functions to an `xarray` object (or selection from an object)

Take the average across ALL the dimensions:

<DataArray or Dataset>`.mean()`



Example:

```
1 data['U'].sel(time=datetime(2012,1,30,20),depth=2.1,
2                lat=slice(-50,-40),lon=slice(0,120)).mean().item()
```

```
0.16497819125652313
```

Applying NumPy functions to an `xarray` object (or selection from an object)

Take the average across certain dimension(s):

<DataArray or Dataset>`.mean(dim=`<dimension name(s) as string or list of strings>`)`

Example:

```
1 display(data['U'].sel(time=datetime(2012,1,30,20),depth=2.1,
2                       lat=slice(-50,-40),lon=slice(0,120)).mean(dim='lon'))
```

xarray.DataArray 'U' (**lat**: 42)

0.19636832 0.19726074 0.19570175 ... 0.112251155 0.108821966

▼Coordinates:

time	()	datetime64[ns]	2012-01-30T20:00:00	
lat	(lat)	float32	-49.78614 -49.570454 ... -40.151318	
depth	()	float32	2.1	

Applying NumPy functions to an `xarray` object (or selection from an object)