

# Librería Xarray

Antonio Espín Herranz

# Contenidos

- Descripción y visión global de la librería
- Estructuras de datos (DataArray, Dataset, ...)
- Indexación y consulta de datos
- Interpolación y realización de cálculos
- Agrupaciones, combinaciones de datos
- Series temporales
- Interoperabilidad con Pandas
- Lectura/escritura de datos

# Descripción

- Xarray trabaja con matrices multidimensionales al estilo de la librería numpy.
- Introduce etiquetas para catalogar y trabajar con los datos de una forma más concisa.
- Las matrices son multidimensionales (N-Dimensionales, ND)
- La naturaleza N-dimensional de las estructuras de datos de xarray lo hace adecuado para tratar con datos científicos multidimensionales.

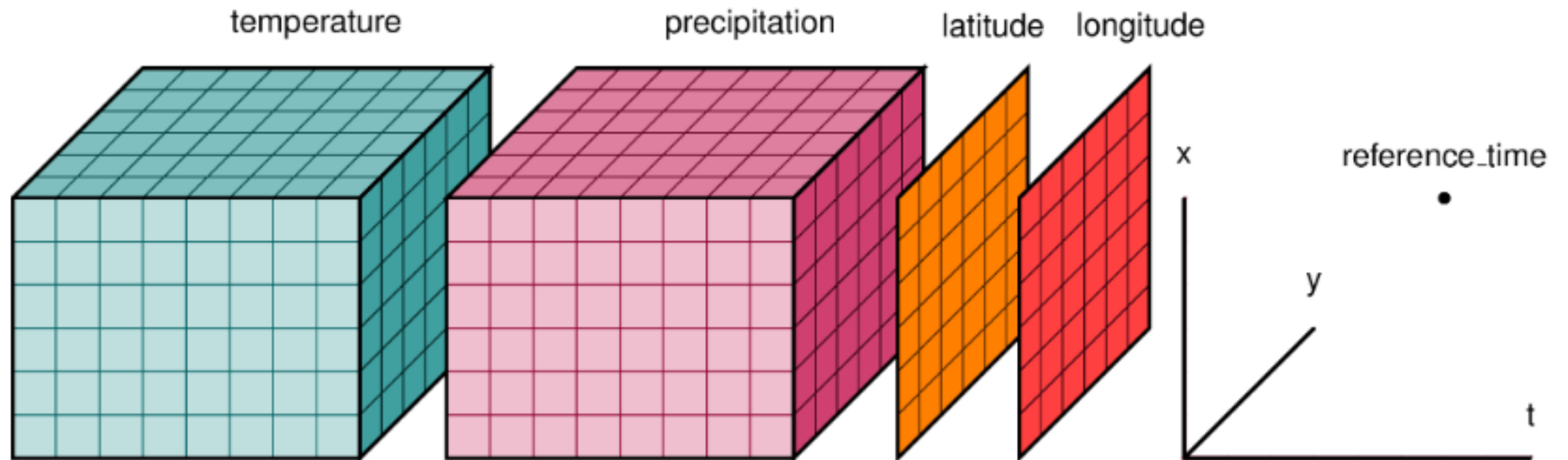
# Descripción

- Se pueden realizar operaciones por su nombre: **`x.sum('time')`**
- Seleccionar valores por etiqueta en lugar de ubicación entera:
  - **`x.loc['2014-01-01']`**
- Agrupar datos indicando las etiquetas:  
**`x.groupby('time.dayofyear').mean()`**

# Descripción

- En comparativa con pandas:
  - pandas se ajusta a datos tabulares con etiquetas.
  - No todos los conjuntos de datos se ajustan fácilmente al modelo “tabular” impuesto por pandas. A menudo tratamos con datos multidimensionales (N-dimensionales) . Datos muchas dimensiones o ejes independientes. Podríamos representar la temperatura de la superficie de la tierra como una variable tridimensional.  $T(x, y, t)$ 
    - X: longitud, Y: latitud, t: es la hora
  - El objetivo de xarray es proporcionar comodidad a nivel de pandas para trabajar con este tipo de datos.

# Esquema



# Estructuras principales

- **DataArray:**

- Matriz N-dimensional etiquetada, es una generalización de **pandas.Series**
- Contiene una sola variable multidimensional y sus coordenadas.

- **Dataset:**

- Es una base de datos de matriz de multidimensional en memoria.
- Es un **contenedor de objetos DataArray** en forma de diccionario similar a **pandas.DataFrame**.
- Contiene múltiples variables que potencialmente comparten las mismas coordenadas.

# DataArray

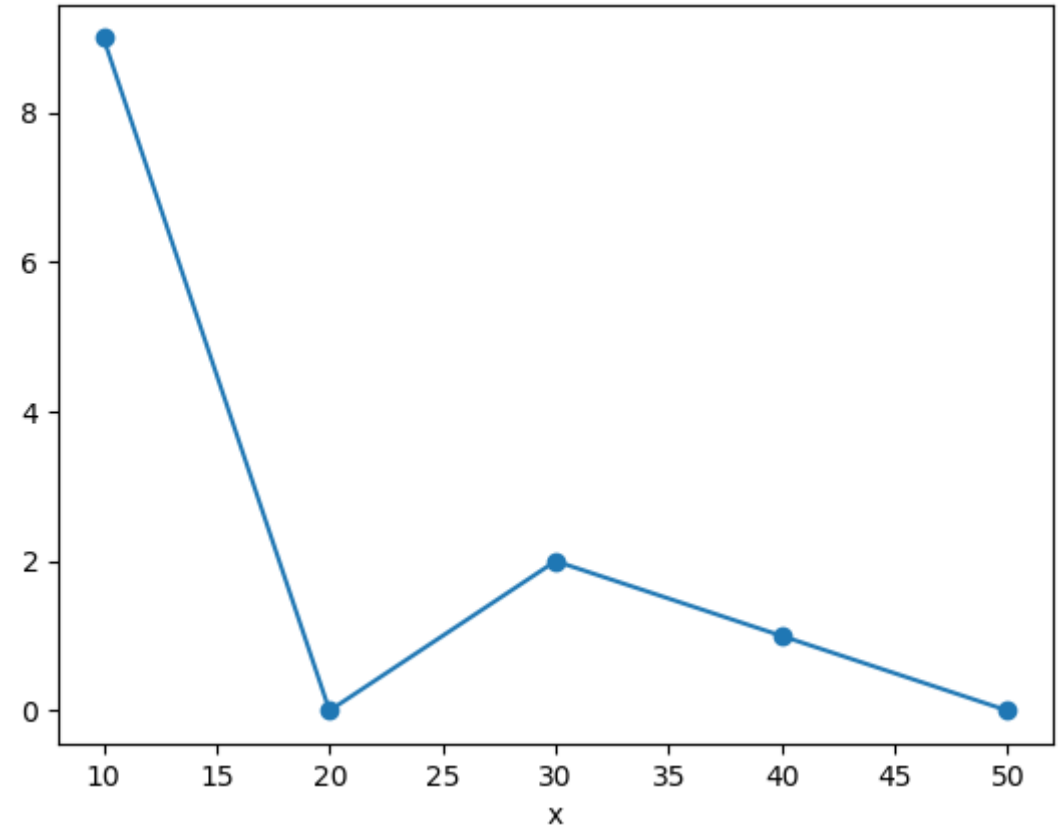
- Tiene 4 atributos principales:
  - **values**: los valores de la matriz basados en `numpy.ndarray`
  - **dims**: Nombres de la dimensión para cada eje: 'x', 'y', 'z'
  - **coords**: un contenedor de matrices (coordenadas).
  - **attrs**: un `OrderedDict` para mantener metadatos.
- Se puede crea un `DataArray` solo con los values, pero no tiene mucho sentido.
  - Lo más habitual es dar un dimensión y una coordenadas para esos valores.



# Ejemplo

```
da3 = DataArray([9,0,2,1,0],  
                dims=['x'],  
                coords={'x': [10,20,30,40,50]})  
print(da3)  
da3.plot(marker='o')  
plt.show()
```

```
<xarray.DataArray (x: 5)>  
array([9, 0, 2, 1, 0])  
Coordinates:  
* x          (x) int32 10 20 30 40 50
```



# xarray vs pandas

- Si estamos tratando con datos de una dimensión. Pandas y Xarray tienen capacidades muy similares.
- El verdadero potencial de Xarray viene con datos multidimensionales.

# Ejemplo: DataArray

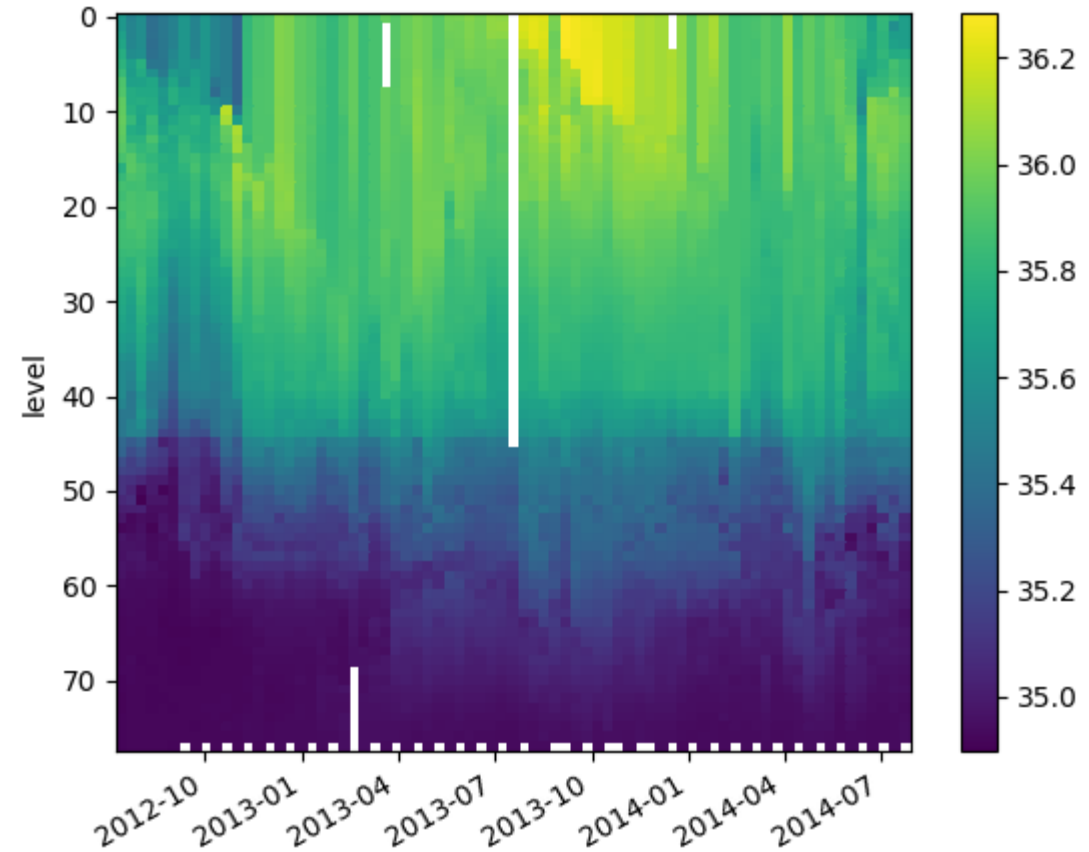
- curl -O [http://www.ideo.columbia.edu/~rpa/argo\\_float\\_4901412.npz](http://www.ideo.columbia.edu/~rpa/argo_float_4901412.npz)
- Cargar el fichero: **argo\_float\_4901412.npz**
- Con la librería numpy:
  - `argo_data = np.load('argo_float_4901412.npz')`
  - Disponemos de varias variables: salinidad, temperatura
  - Coordenadas: latitud, longitud
  - Registro de fechas.
- Consultar: `argo_data.keys()`
  - ['S', 'T', 'levels', 'lon', 'date', 'P', 'lat']
    - **S** → Salinidad
    - **T** → Temperatura
    - **P** → Presión
    - **Lat, lon**: Latitud, longitud
    - **date**: fecha
    - **levels**: nivel de 0 a 70 (una medida)

# Ejemplo: DataArray II

- Podemos recoger los datos:
  - `S = argo_data.f.S` equivalente  $\rightarrow$  `S = argo_data['S']`
  - `T = argo_data.f.T`
  - `P = argo_data.f.P`
  - `levels = argo_data.f.levels`
  - `lon = argo_data.f.lon`
  - `lat = argo_data.f.lat`
  - `date = argo_data.f.date`
- **`argo_data.f`**  $\rightarrow$  un objeto intermedio de tipo: **BagObj**
- Los tipos de `S`, `T`, `P`, etc. son arrays de numpy  $\rightarrow$  **`np.ndarray`**

# Ejemplo: DataArray III

- A partir de los arrays de numpy se puede crear un DataArray con datos de la salinidad:
- `da_salinity = xr.DataArray(S,`
  - `dims=['level', 'date'],`
  - `coords={'level': levels, 'date': date},)`
- Después se puede representar en un gráfico:
  - `da_salinity.plot(yincrease=False)`



# Ejemplo: DataArray IV

- Se puede completar con attrs (metadatos)
  - `da_salinity.attrs['units'] = 'PSU'`
  - `da_salinity.attrs['standard_name'] = 'sea_water_salinity'`

# Dataset

- Un **Dataset**: mantiene muchos DataArrays que potencialmente pueden compartir coordenadas.
  - Igual que ocurre con pandas.DataFrame se compone de pandas.Series
  - xarray.Dataset se compone de xarray.DataArray
- El constructor de Dataset toma 3 argumentos:
  - **data\_vars**: (puede ser)
    - Una tupla de la forma: (dims, data [, attrs])
    - Un objeto pandas que se convierte el DataArray
    - Una matriz o lista (1 Dim) que se interpreta como una variable de coordenadas unidimensional.
  - **coords**: Debe ser un diccionario de la misma forma que data\_vars
  - **attrs**: Debería ser un diccionario

# Ejemplo: Dataset

- Crear un Dataset con 3 variables: Salinidad, Temperatura y Presión

```
ds = xr.Dataset(  
    data_vars={'salinidad': (('level', 'date'), S),  
    'temperatura': (('level', 'date'), T),  
    'presion': (('level', 'date'), P)},  
    coords={'level': levels, 'date': date})
```

```
Coordinates:  
* level      (level) int64 0 1 2 3 4 5 6 7 8 ... 69 70 71 72 73 74 75 76 77  
* date       (date) datetime64[ns] 2012-07-13T22:33:06.019200 ... 2014-07-24T03:02:33.014400  
Data variables:  
  salinidad  (level, date) float64 35.64 35.51 35.57 35.4 ... nan 34.94 nan  
  temperatura (level, date) float64 18.97 18.44 19.1 19.79 ... nan 3.714 nan  
  presion     (level, date) float64 6.8 6.1 6.5 5.0 ... 2e+03 nan 2e+03 nan
```



# Modificación Dataset

- Se pueden añadir coordenadas a posteriori, una vez creado el **Dataset**.
  - `ds.coords['lon']=lon`
- O también se pueden eliminar y añadirlas con una dimensión (date)
  - `del argo['lon']`
  - `argo.coords['lon'] = ('date', lon)`
  - `argo.coords['lat'] = ('date', lat)`

El \* en **level** y **date** son coordenadas de dimensión  
Y **lon** y **lat** no son coordenadas no dimensionales.

```
<xarray.Dataset>
Dimensions:      (date: 75, level: 78)
Coordinates:
  * level        (level) int64 0 1 2 3 4 5 6 7 8 ... 69 70 71 72 73 74 75 76 77
  * date         (date) datetime64[ns] 2012-07-13T22:33:06.019200 ... 2014-07-24T03:02:33.014400
  lon           (date) float64 -39.13 -37.28 -36.9 ... -33.83 -34.11 -34.38
  lat           (date) float64 47.19 46.72 46.45 46.23 ... 42.6 42.46 42.38
Data variables:
  salinidad     (level, date) float64 35.64 35.51 35.57 35.4 ... nan 34.94 nan
  temperatura   (level, date) float64 18.97 18.44 19.1 19.79 ... nan 3.714 nan
  presion       (level, date) float64 6.8 6.1 6.5 5.0 ... 2e+03 nan 2e+03 nan
```

# Coordenadas vs Datos variables

- Las variables de datos: T, P, S se pueden modificar y las coordenadas se mantienen.
- Las modificaciones se pueden realizar a través de operaciones y otras funciones.
- `print(ds * 10000)`
  - Modifica los datos de las variables no de las coordenadas

# Indexación y selección de datos

- Se puede utilizar indexación o slicing tipo numpy
- **Seleccionar** (el nivel 2 de salinidad)
  - `ds.salinidad[2].plot()` → Selecciona el nivel 2 y lo pinta en una gráfica
  - `ds.salinidad.sel(level=2).plot()`
- **Seleccionar** por fecha:
  - `ds.salinidad.sel(date='2012-10-22').plot(y='level', yincrease=False)`
- **Slicing**:
  - `ds.salinidad[:, 10].plot()`
- **Slice** (seleccionar un rango de fechas)
  - `ds.salinidad.sel(date=slice('2012-10-01', '2012-12-01')).plot()`

# Computación con los datos

- Podemos aplicar operaciones aritméticas y funciones numpy
  - `temp_kevin = ds.temperatura + 237.15`
  - Se aplica a todos los valores de la variable temperatura.
- Se pueden combinar varias variables:
  - `g = 9.8 # Gravedad`
  - `flotabilidad = g * (2e-4 * ds.temperatura - 7e-4 * ds.salinidad)`
  - `flotabilidad.plot(yincrease=False)`

# Operaciones estadísticas


- `ds_media = ds.temperatura.mean(dim='date')`
- `ds_std = ds.std(dim='date')`
- `ds.sum()`

# Agrupar datos


```
import pandas as pd
import xarray as xr
import numpy as np
```

```
ds = xr.Dataset(
    {"foo": (("x", "y"), np.random.rand(4, 3))},
    coords={"x": [10, 20, 30, 40], "letters": ("x",
list("abba"))}
)
```

```
print(ds)
print(ds.groupby("letters"))
print(ds.groupby("letters").groups)
print(list(ds.groupby("letters").groups))
```



```
Dimensions:  (x: 4, y: 3)
* x          (x) int32 10 20 30 40
  letters    (x) <U1 'a' 'b' 'b' 'a'
Dimensions without coordinates: y
Data variables:
  foo        (x, y) float64 0.4874 0.005209 0.1367 ... 0.07725 0.8152 0.395
```



```
DatasetGroupBy, grouped over 'letters'
2 groups with labels 'a', 'b'.
-----
{'a': [0, 3], 'b': [1, 2]}
-----
['a', 'b']
```

# Agrupar por otros intervalos

- A veces nos interesa agrupar los datos por medio de otros intervalos:
- Para esto xarray proporciona **groupby\_bins()**
  - `x_bins = [0, 25, 50]`
  - `print(ds.groupby_bins("x", x_bins).groups)`

```
{Interval(0, 25, closed='right'): [0, 1], Interval(25, 50, closed='right'): [2, 3]}
```

- Los grupos se pueden etiquetar:
  - `x_bin_labels = [12.5, 37.5]`
  - `print(ds.groupby_bins("x", x_bins, labels=x_bin_labels).groups)`

```
{12.5: [0, 1], 37.5: [2, 3]}
```

# Agrupar datos: aplicar funciones

- Se puede aplicar una función a cada grupo:
- Se puede utilizar el método **map()**.
  - Es la misma idea de map de las listas, se aplica a todos los elementos en este caso del dataset.

```
import numpy as np
import xarray as xr
from xarray import Dataset, DataArray

da = DataArray(np.random.randn(2,3))
ds = Dataset({"var1":da, "var2":("x", [-1,2])})

print(ds)
ds_abs = ds.map(np.fabs)
print()
print(ds_abs)
```



# Series temporales

- <http://xarray.pydata.org/en/stable/time-series.html>
- Se suele hacer con la librería pandas y la función
  - pandas.**date\_range()** o con pandas.**to\_datetime()**

# Interacción con pandas

- Se puede crear un **Dataset** a partir de un **DataFrame**
- `import pandas as pd`
- `import xarray as xr`
  
- `df = pd.read_csv("IRMA.csv",sep=';')`
  
- **# Cargar el dataset a partir de un DataFrame**
- `ds = xr.Dataset.from_dataframe(df)`
- `print(ds)`
  
- **# Convertir a un DataFrame de pandas:**
- `df = ds.to_dataframe()`

# Lectura / Escritura de ficheros

- Soporta directamente la serialización, ficheros serializados con el módulo **pickle** y otros más flexibles como son: **netCDF** (recomendado).
  - <https://www.unidata.ucar.edu/software/netcdf/docs/faq.html#What-Is-netCDF>
  - Ejemplo: **gistemp1200\_ERSSTv5.nc**
- Para escribir un fichero en formato: **netCDF**
- A partir de un Dataset (ds)
  - **ds.to\_netCDF("fichero.nc")**
- Cargar un Dataset a partir de un fichero en formato: netCDF
  - `import xarray as xr`
  - **ds = xr.open\_dataset("fichero.nc")**

# Fichero: gistemp1200\_ERSSTv5.nc

```
Dimensions:      (lat: 90, lon: 180, nv: 2, time: 1675)
Coordinates:
  * lat          (lat) float32 -89.0 -87.0 -85.0 -83.0 ... 83.0 85.0 87.0 89.0
  * lon          (lon) float32 -179.0 -177.0 -175.0 -173.0 ... 175.0 177.0 179.0
  * time         (time) datetime64[ns] 1880-01-15 1880-02-15 ... 2019-07-15
Dimensions without coordinates: nv
Data variables:
  time_bnds      (time, nv) datetime64[ns] ...
  tempanomaly    (time, lat, lon) float32 ...
Attributes:
  title:         GISTEMP Surface Temperature Analysis
  institution:   NASA Goddard Institute for Space Studies
  source:        http://data.giss.nasa.gov/gistemp/
  Conventions:   CF-1.6
  history:       Created 2019-08-09 14:28:29 by SBBX_to_nc 2.0 - ILAND=1200,...
```

# Enlaces

- <http://xarray.pydata.org/en/stable/faq.html>
- <https://openresearchsoftware.metajnl.com/articles/10.5334/jors.148/>
- [https://ravernat.github.io/research\\_computing\\_2018/xarray.html](https://ravernat.github.io/research_computing_2018/xarray.html)
- [http://meteo.unican.es/work/xarray\\_seminar/xArray\\_seminar.html](http://meteo.unican.es/work/xarray_seminar/xArray_seminar.html)