Python in the Geosciences

Fall 2015 Seminar - University of Washington

Time: 1st Tuesday of the month, 4:00PM with Happy Hour to follow.

Location: 6th floor of the Physics Tower, Seminar Room C607

Python in the Geosciences

Other Workshops / Seminars / Resources:

Atmos-Python Workshop:

atmos-python@uw.edu - (Andre or Jeremy)

UW eScience Python Seminar: Fall 2015:

http://uwescience.github.io/python-seminar-2015/seminar/

AGU Fall Meeting 2015 Session:

Python Solutions for the Earth Sciences: IN041:

Data analysis tools for modern Python:

pandas, xray, and beyond

Joe Hamman
October 6, 2015
Python in the Geosciences
Fall 2015 Seminar - University of Washington

Department of Civil and Environmental Engineering

Audience Participation (familiarity with…)

python™

pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

netCDF

xray

We should stop rewriting the same things…

Working Assumption:

It is better to push repetitive tasks to well tested, optimized packages (e.g. pandas/xray/NumPy) python™

NumPy

IP[y]:
IPython

SciPy

matplotlib

SymPy

Cython

Mayavi

pandas

xray

SM StatsModels
Statistics in Python

scikits learn
machine learning in Python

scikits-image
image processing in python

PyTables

NetworkX

The core of scientific computing in Python

NumPy

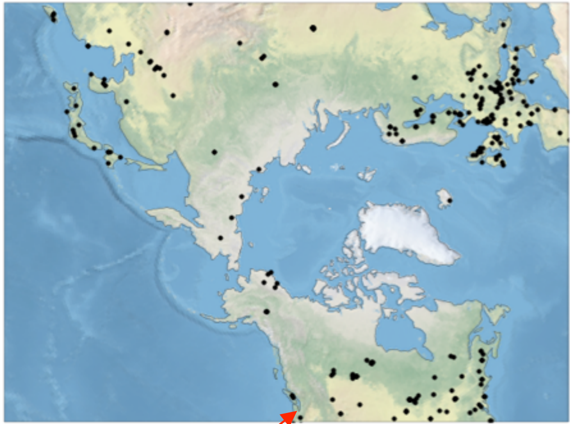
- N-dimensional array objects
- High level functions
- Linear algebra functions
- random number generation
- Foreign language (C/Fortran) integration

SciPy.org

- Depends on NumPy
- High level science / engineering modules
  - Fourier transforms
  - Integration
  - Interpolation
  - Optimization
  - Signal processing
  - Statistics

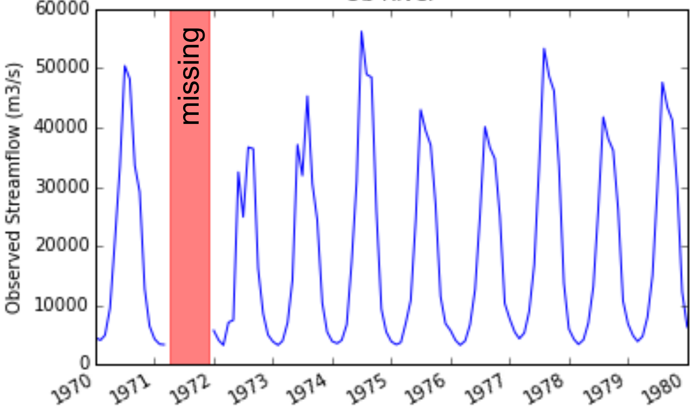
Geospatial Data

Global FluxNET Observations

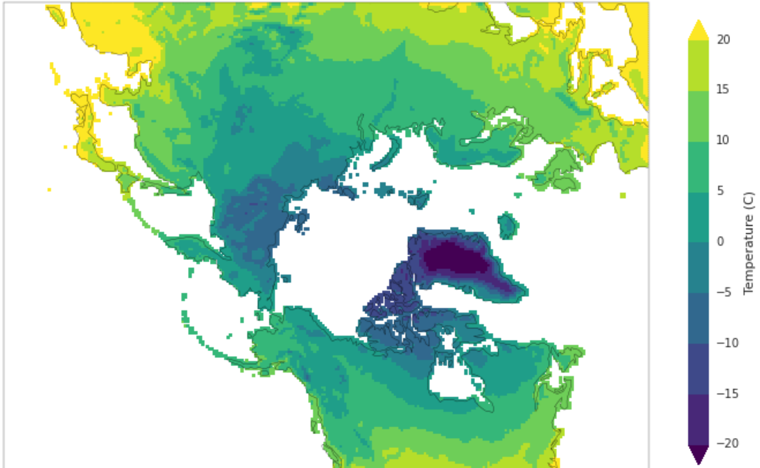


You are here

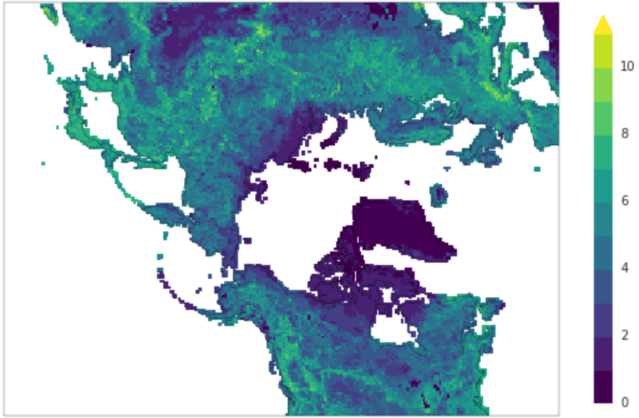
Ob River



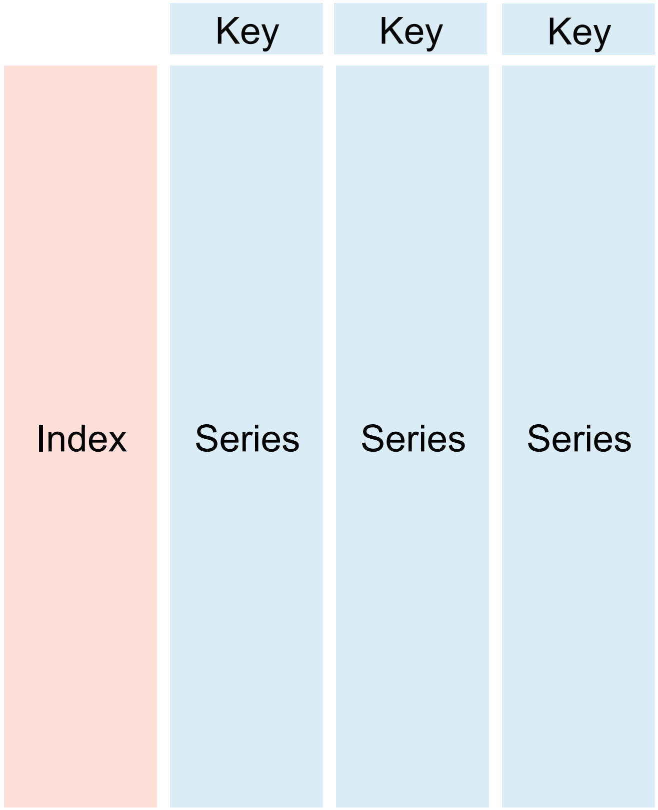
RASM Surface Air Temperature



UMD Vegetation Types Per Grid Cell



The pandas Data Model



Pandas Index:
The basic object storing axis labels for all pandas objects

Pandas Series:
One-dimensional ndarray with axis labels (including time series).

Pandas DataFrame:
Two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). Made up of one or more Series.

Pandas Panel:
Represents wide format panel data, stored as 3-dimensional array

The pandas DataFrame

the index

Missing data

station	Yenisey	Lena	Ob	Amur	Mackenzie	Yukon	Pechora	Nelson	Khatanga	Kolyma
time										
1948-08-01	9569	29675.396484	NaN	NaN	17100	NaN	2165.750000	NaN	NaN	NaN
1948-09-01	7826	36318.761719	NaN	NaN	9470	NaN	2209.843750	NaN	NaN	NaN
1948-10-01	8435	19653.771484	NaN	NaN	6100	NaN	2253.906250	NaN	NaN	NaN
1948-11-01	10248	10117.241211	NaN	NaN	3440	NaN	2254.562500	NaN	NaN	NaN
1948-12-01	10220	4085.000000	NaN	NaN	1870	NaN	1989.875000	NaN	NaN	NaN
1949-01-01	10628	3133.926758	3249	NaN	1390	NaN	1999.593750	NaN	NaN	NaN
1949-02-01	12876	2539.136963	3058	NaN	1820	NaN	1811.015625	NaN	NaN	NaN
1949-03-01	16504	1944.476685	2516	NaN	1740	NaN	1633.187500	NaN	NaN	NaN
1949-04-01	15826	2085.389160	2105	NaN	5070	NaN	1548.812500	NaN	NaN	NaN
1949-05-01	12234	3486.129883	2981	NaN	7270	NaN	1334.093750	NaN	NaN	NaN
1949-06-01	11142	5410.482422	5883	NaN	19800	NaN	926.875000	NaN	NaN	NaN
1949-07-01	11681	13076.666992	17810	NaN	29400	NaN	744.625000	NaN	NaN	NaN
1949-08-01	8569	28834.199219	43997	NaN	13000	NaN	880.875000	NaN	NaN	NaN
1949-09-01	7939	34319.695312	40283	NaN	6220	NaN	748.906250	NaN	NaN	NaN
1949-10-01	8087	23169.228516	18745	NaN	11800	NaN	1220.281250	NaN	NaN	NaN
1949-11-01	6949	10456.551758	9773	NaN	NaN	NaN	1697.531250	NaN	NaN	NaN

pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

Great

- Fast
- Flexible
- Missing data

Not so great

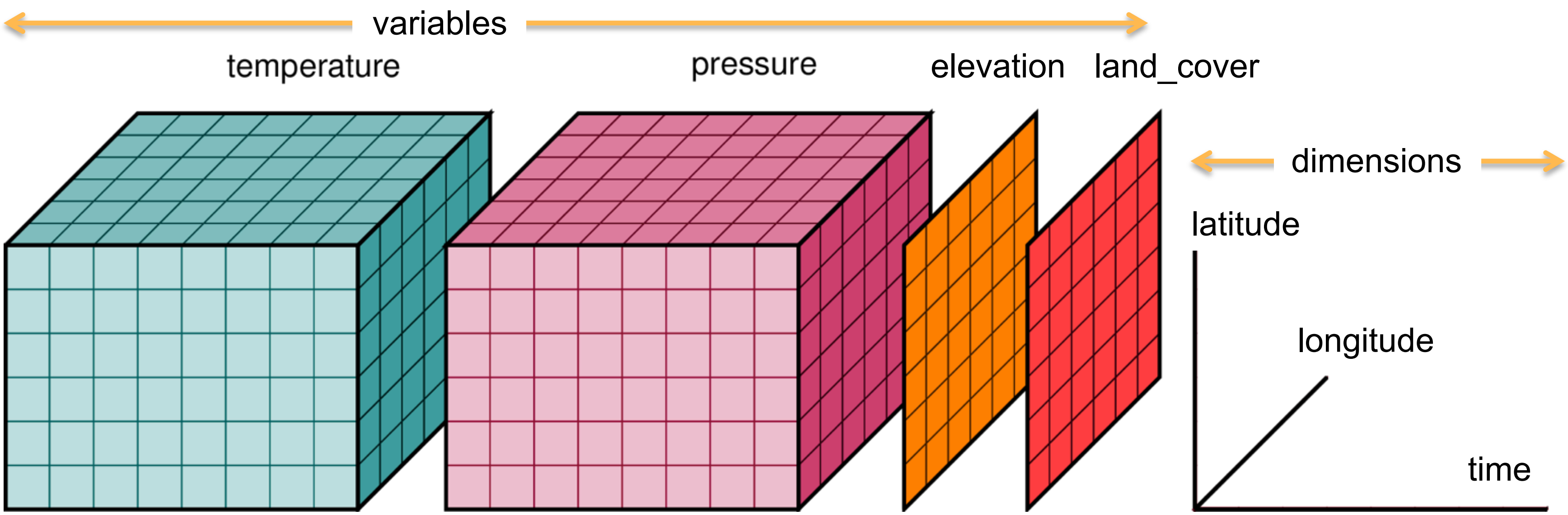
- N-Dimensional objects and operations
- Metadata

xray

- Fundamentally N-dimensional
- Leverages pandas for indexing
- Tries to copy pandas API
- netCDF (Data model, IO)
- Metadata
- Out of core computation

netCDF data model

N-Dimensional Homogeneous (type) arrays



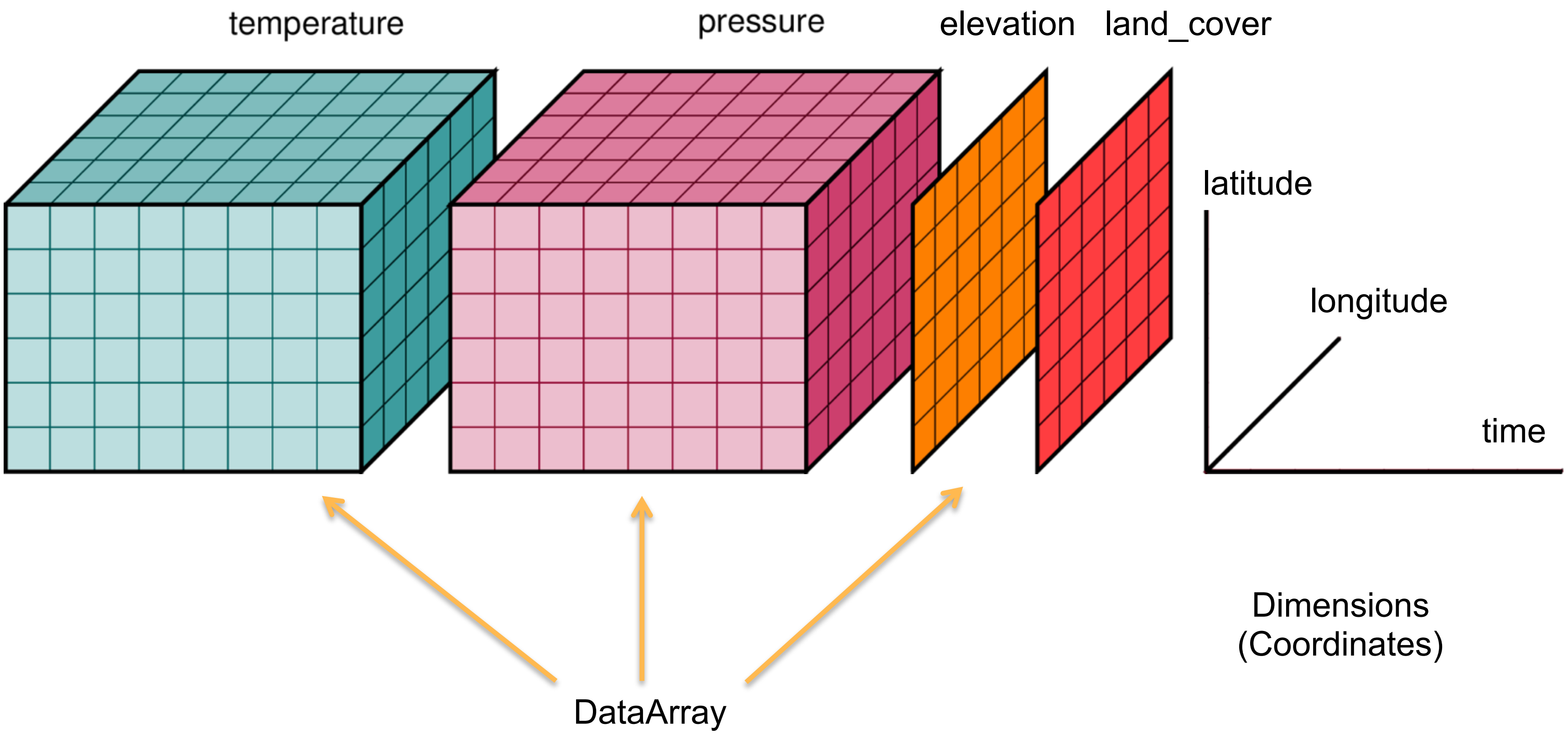
attributes:
- units
- description
- fill_value
- coordinates
- etc.

The xray Dataset

```
In [5]: print(ds)

<xray.Dataset>
Dimensions:  (nv4: 4, time: 366, x: 275, y: 205, z: 1)
Coordinates:
  * time     (time) datetime64[ns] 2000-01-01 2000-01-02 2000-01-03 ...
    xc       (y, x) float64 189.2 189.4 189.6 189.7 189.9 190.1 190.2 190.4 ...
    yc       (y, x) float64 16.53 16.78 17.02 17.27 17.51 17.76 18.0 18.25 ...
  * z        (z) float32 0.0
  * nv4      (nv4) int64 0 1 2 3
  * x        (x) int64 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 ...
  * y        (y) int64 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 ...
Data variables:
    xc_bnds  (y, x, nv4) float64 189.3 189.4 189.2 189.0 189.4 189.6 189.3 ...
    yc_bnds  (y, x, nv4) float64 16.33 16.58 16.74 16.49 16.58 16.82 16.98 ...
    prcp     (time, z, y, x) float64 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 ...
    tas      (time, z, y, x) float64 291.6 291.4 291.5 291.5 290.8 290.7 ...
Attributes:
    CDI: Climate Data Interface version 1.5.4 (http://code.zmaw.de/projects/cdi)
    Conventions: CF-1.4
```

xray’s Dataset is a N-dimensional DataFrame



Serialization and IO

Because nobody likes parsing messy data

pandas	**xray**
`{read, to}_csv()`	`open_dataset(), to_netcdf()`
`{read, to}_excel`	Backends:
`{read, to}_hdf`	`netCDF4`
`{read, to}_sql`	`scipy`
`{read, to}_json`	`h5netcdf`
`{read, to}_html`	`pydap`

Leveraging the index and labels

Selecting, slicing, and subsetting

```
# Xray Style
In [6]: ds.sel(time=slice('2000-01-15', '2000-03-15')).mean(dim=‘station’)

# Numpy Style
In [7]: array[15:60, :, :].mean(axis=2)
```

Label based methods are:

- More reliable
- Easier to maintain
- Easier to read

Leveraging the index

Resampling: frequency conversion and resampling of regular time-series data.

```
# Resample Dataset from 6-hourly to monthly means
In [8]: ds.resample('MS', dim='time', how='mean’)

# Resample Dataset from 6-hourly to daily sums
In [9]: ds.resample('D', dim='time', how=’sum’)
```

Leveraging the index

Grouping: split – apply – combine

- **Split** your data into multiple independent groups.
- **Apply** some function to each group.
- **Combine** your groups back into a single data object.

```
# Get Monthly Climatology
In [10]: ds.groupby('time.month').mean(dim='time')

# Or Seasonal Climatology
In [11]: ds.groupby('time.season').mean(dim='time')
```

Other stuff

```
# Fancy Indexing:
In [12]: ds.isel_points(lon=[1, 3], lat=[2, 2])

# Vectorized math and brodcasting (based on dim name)
In [13]: ds.x * ds.y + ds.z

# Alignment:
In [14]: x, y = xray.align(x, y, join='outer’)
```

xray and pandas make working with(out) missing data much easier

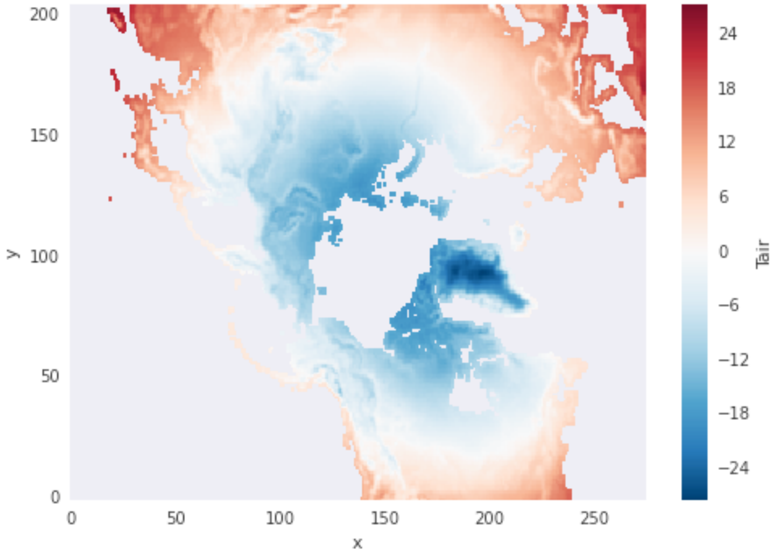
```
# Select it:
In [15]: df.isnull()

# Skip it:
In [16]: df.mean()

# Drop it:
In [17]: df.dropna()

# Fill it:
In [18]: df.fillna()
```

```
# For Example:
In [19]: ds.Tair.\
mean(dim='time’).plot ()
```



Interoperability with Numpy/Scipy

```
# Calculate the 5th percentile for each month
In [20]: percents = ds.groupby('time.month').reduce(np.percentile, dim='time', q=5)

In [21]: percents
Out[21]:
<xray.Dataset>
Dimensions:  (lat: 25, lon: 53, month: 12)
Coordinates:
  * lat      (lat) float32 75.0 72.5 70.0 67.5 65.0 62.5 60.0 57.5 55.0 52.5 …
  * lon      (lon) float32 200.0 202.5 205.0 207.5 210.0 212.5 215.0 217.5 …
  * month    (month) int64 1 2 3 4 5 6 7 8 9 10 11 12
Data variables:
    air      (month, lat, lon) float64 237.5 237.8 237.6 237.4 236.7 235.8 …
```

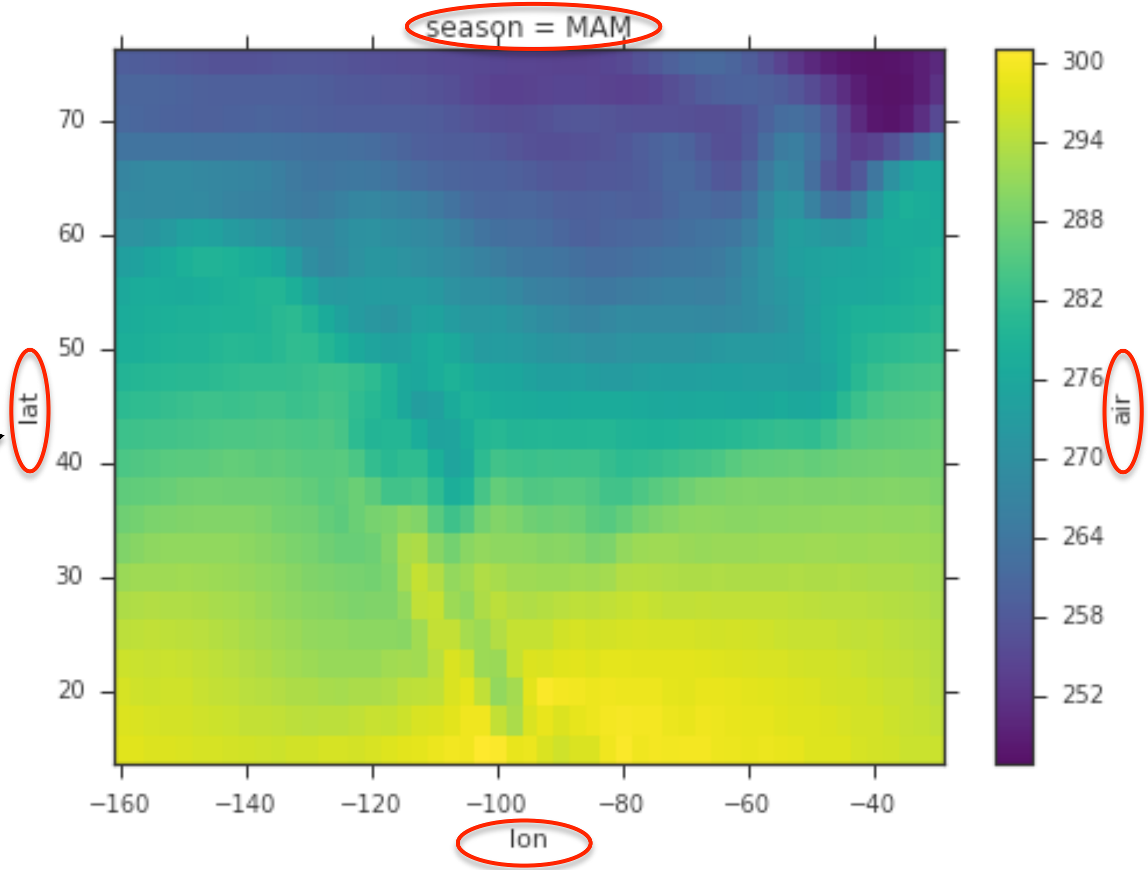
Xray and Pandas Interoperability

```
# Get a pandas Dataframe from an xray Dataset
In [22]: df = percents.sel(lat=70., lon=202.5).to_dataframe()
In [23]: print(df)
              air   lat     lon
month
1       240.2350    70   202.5
2       239.3000    70   202.5
…
# Get a xray Dataset from a Pandas DataFrame
In [24]: xray.Dataset.from_dataframe(df)
Out[25]:
<xray.Dataset>
Dimensions:  (month: 12)
Coordinates:
  * month    (month) int64 1 2 3 4 5 6 7 8 9 10 11 12
Data variables:
    air      (month) float64 240.2 239.3 243.6 247.5 257.2 270.5 274.8 272.6 ...
    lat      (month) float32 70.0 70.0 70.0 70.0 70.0 70.0 70.0 70.0 70.0 ...
    lon      (month) float32 202.5 202.5 202.5 202.5 202.5 202.5 202.5 202.5 …
```

Labeled data can plot itself

```
# For Example:
In [25]: da.plot()
```

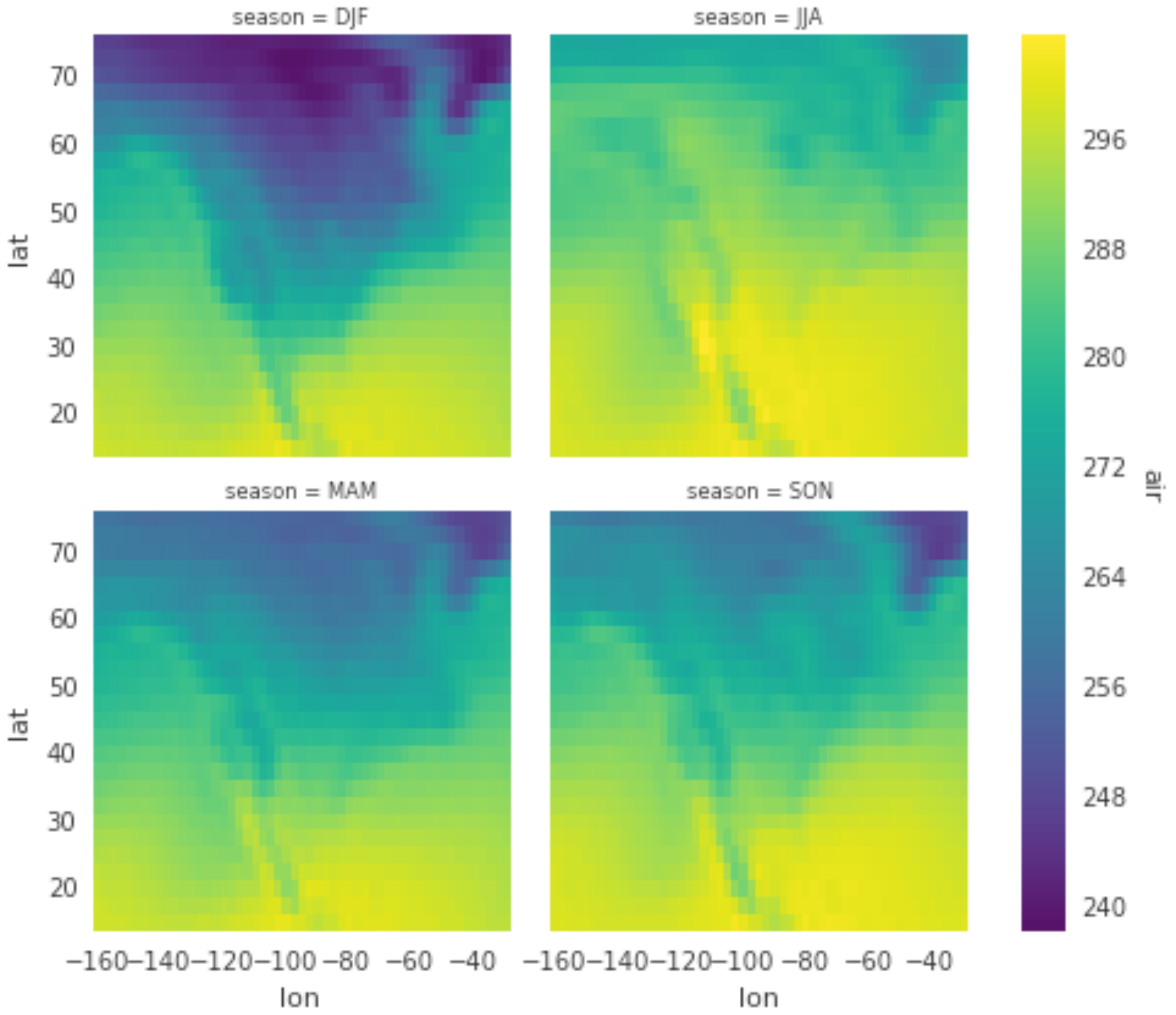
People need incentives to preserve metadata



FacetGrid

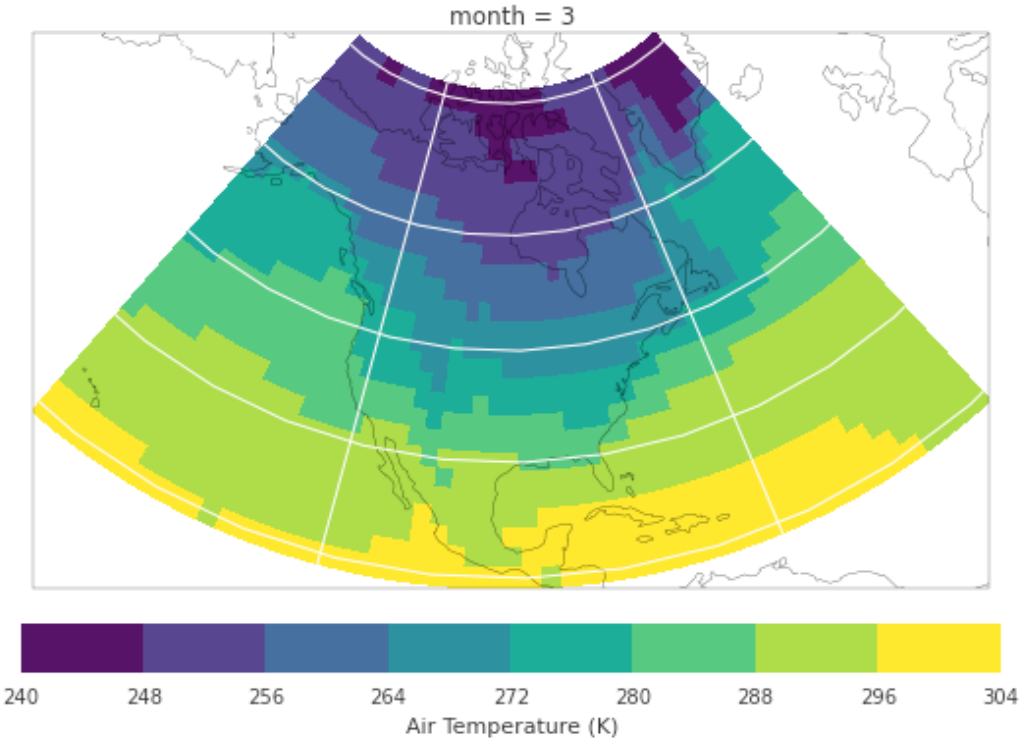
Borrowed from Seaborn

```
# For Example:
In [26]:
da.plot(col='season'
, col_wrap=2)
```

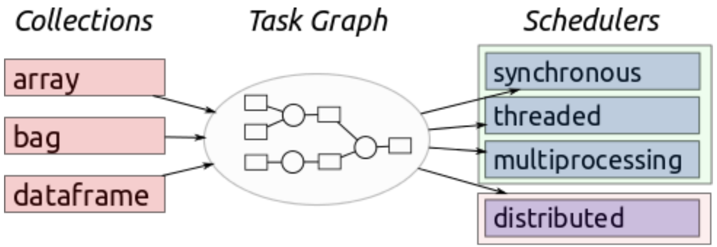


Plotting with Cartopy

```
ax = plt.axes(projection=ccrs.LambertConformal())
mappable = da.plot(levels=10, transform=ccrs.PlateCarree())
```

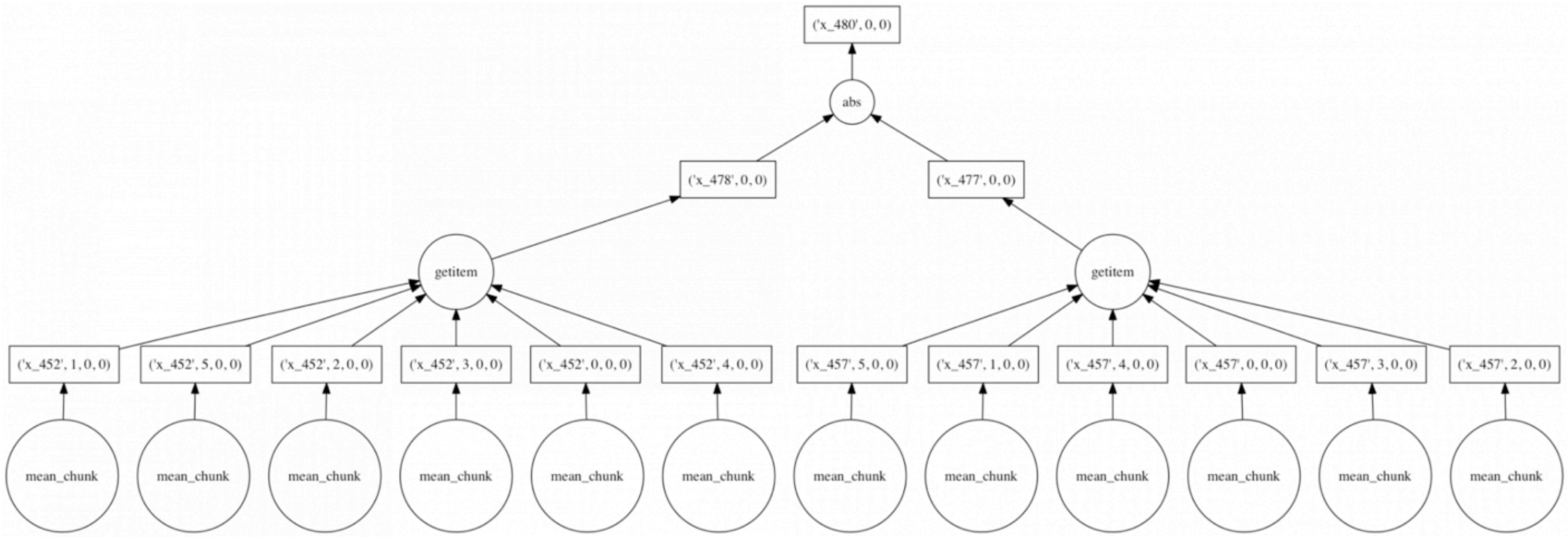


Getting out of core: Dask



Data processing tasks are often embarrassingly parallel.

Lazy Computation



Who's in the game

Blaze
Dask

Dask enables parallel computing through task scheduling and blocked algorithms.

What is medium data?

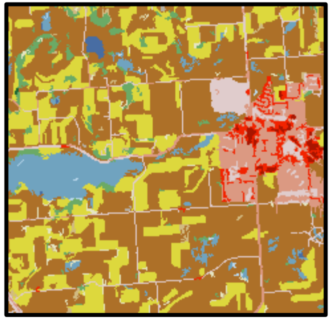
Cartopy is a Python package designed to make drawing maps for data analysis and visualization as easy as possible.

Replacement? to Matplotlib Basemap.

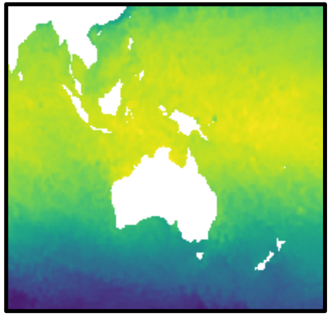
Scientific Python’s Horizons

We should be able to write new dtypes in Python

Categorical



Missing data



Dates & times

Physical Units

$$52.8\,\mathrm{ft/s}$$
$$= 36\,\mathrm{mi/h}$$

We should stop rewriting the same things…

Working Assumption:

It is better to push repetitive tasks to well tested, optimized packages (e.g. pandas/xray/NumPy)

So…

Don’t hide your code

Contribute to open source projects

Build on the success of others

Questions?

- Joe Hamman
- Computational Hydrology, CEE
- Email: jhamman1@uw.edu
- Web: joehamman.com
- Github: jhamman