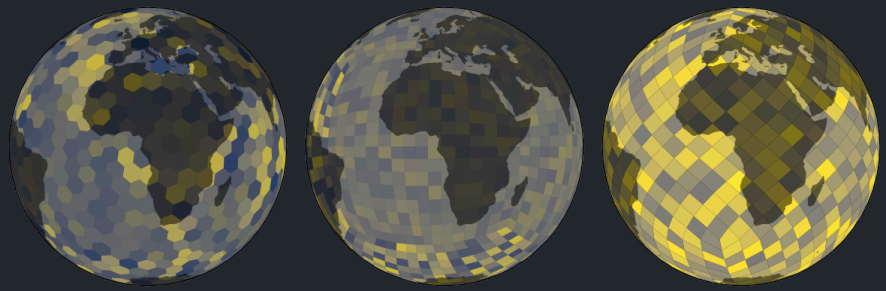
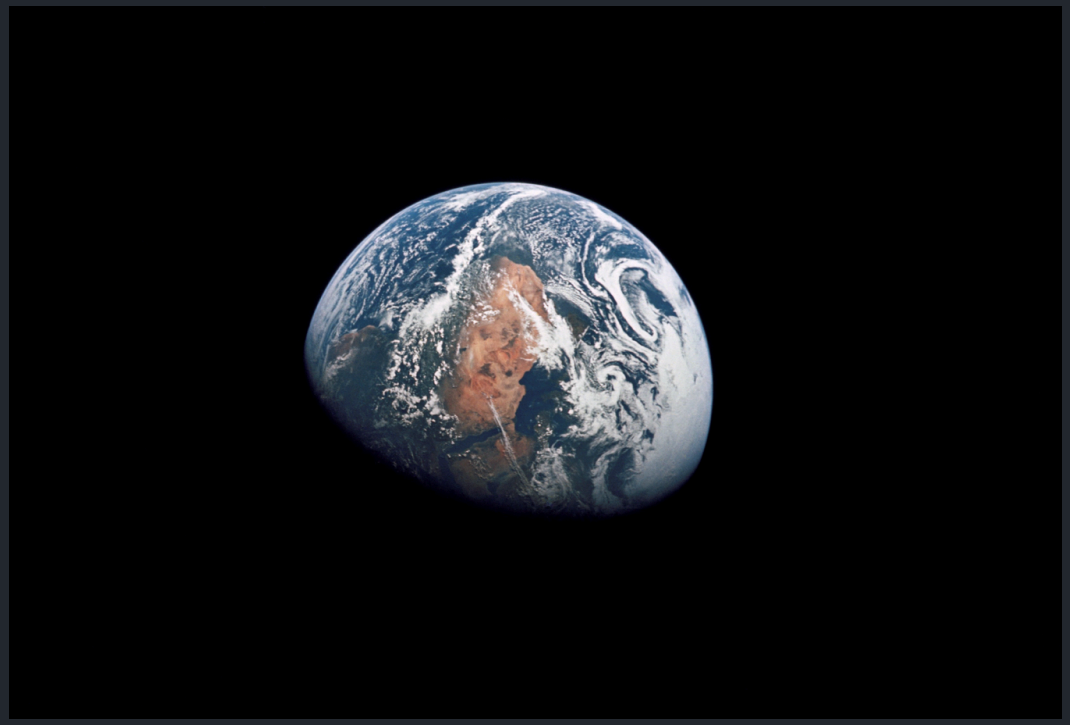
Spherical Geometry and Discrete Global Grid Systems (DGGS) Across Languages



SDSL - Prague, September 2024

The Earth is not flat



Vector data

Why not using projected coordinates?

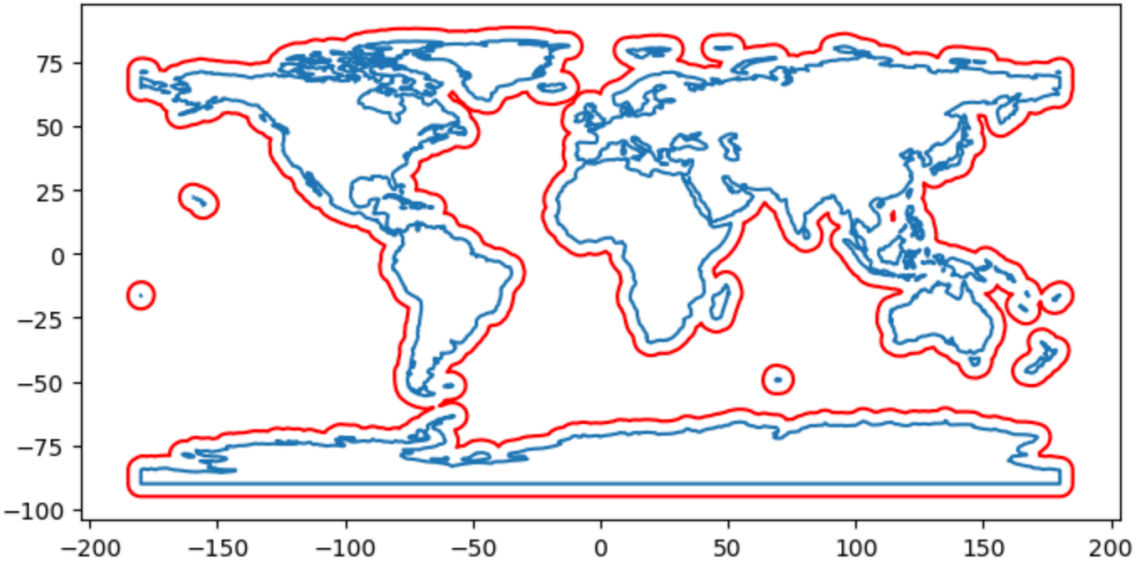
- my data is in latitude / longitude coordinates
- projection is not always trivial
  - choose a coordinate system depending on location, extent, etc.
  - may be computationally expensive
- boundary singularities, distortion.

https://github.com/jorisvandenbossche/geopandas-tutorial

Global analysis

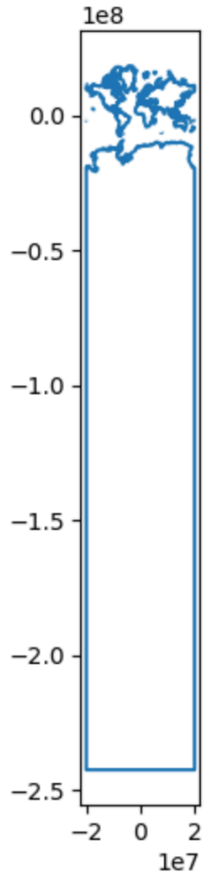
```
# geopandas (shapely, matplotlib)

fig, ax = plt.subplots(figsize=(8, 5))

world.boundary.plot(ax=ax);
world.buffer(5).boundary.plot(ax=ax, color="red");
```

```
/var/folders/xd/3ls911kd6_n2wphwwd74b1dc0000gn/T/ipykernel_53093/2457478289.py:6: UserWarning: Geometry is in a
geographic CRS. Results from 'buffer' are likely incorrect. Use 'GeoSeries.to_crs()' to re-project geometries to
a projected CRS before this operation.

  world.buffer(5).boundary.plot(ax=ax, color="red");
```



Projecting global datasets

```
fig, ax = plt.subplots(figsize=(4, 6))

# WGS 84 / Pseudo-Mercator
world.to_crs(3857).boundary.plot(ax=ax);
```



Spherical geometry across languages

Library	Language	Repository
S2	R	https://github.com/r-spatial/s2/
Spherely	Python	https://github.com/benbovy/spherely
GeometryOps.jl ? (*)	Julia	https://github.com/JuliaGeo/GeometryOps.jl

(*) https://github.com/JuliaGeo/GeometryOps.jl/issues/17

Library	Language	Repository
S2Geometry	C++	https://github.com/google/s2geometry
S2Geography	C++	https://github.com/paleolimbot/s2geography
Geo ?	Rust	https://github.com/georust/geo

Spherely Python bindings

Using Pybind11... Just a few lines of C++ code!

```
bool contains(PyObjectGeography a, PyObjectGeography b) {
    const auto& a_index = a.as_geog_ptr()->geog_index();
    const auto& b_index = b.as_geog_ptr()->geog_index();

    S2BooleanOperation::Options options;
    return s2geography::s2_contains(a_index, b_index, options);
}

void init_predicates(pybind11::module& m) {
    m.def("contains", pybind11::vectorize(&contains));
}
```

Integration with (geo) data-frame and data-cube libraries

- R's `sf`: global flag `sf_use_s2(TRUE)` + conversion methods
- Python's `geopandas`: `GeographyArray` ?
- Python's `xvec`: `GeographyIndex` ?

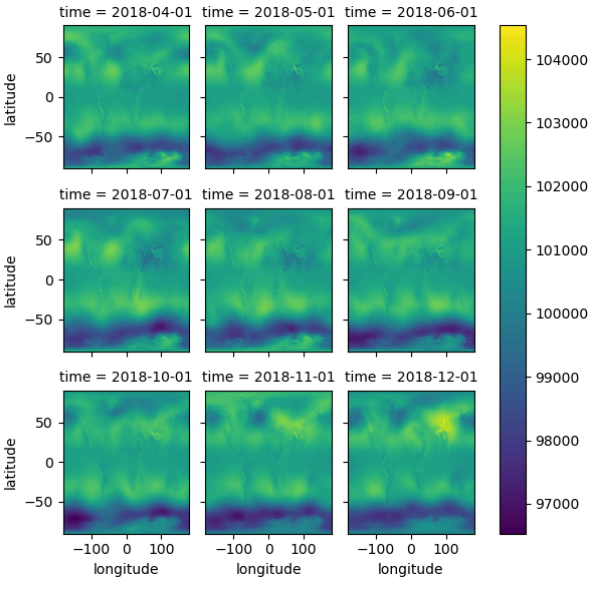
Gridded data

Gridded data examples

ERA5 / global / unprojected

xarray.Dataset

▸ Dimensions:	(**longitude**: 480, **latitude**: 241, **time**: 480)		
▾ Coordinates:			
longitude	(longitude)	float32	-179.6 -178.9 ... 178.9 179.6
latitude	(latitude)	float32	90.0 89.25 88.5 ... -89.25 -90.0
time	(time)	datetime64[ns]	1979-01-01 ... 2018-12-01
▾ Data variables:			
u10	(time, latitude, longitude)	float64	...
v10	(time, latitude, longitude)	float64	...
msl	(time, latitude, longitude)	float64	...
▸ Indexes: (3)			
▸ Attributes: (2)			

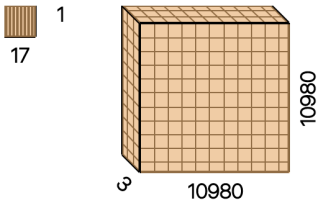


Gridded data examples

Sentinel-2 / regional / UTM-projected

xarray.DataArray 'stackstac-482803e0573469fa5dd78309fe0fd9aa'
(**time**: 17, **band**: 3, **y**: 10980, **x**: 10980)

	Array	Chunk
Bytes	45.81 GiB	8.00 MiB
Shape	(17, 3, 10980, 10980)	(1, 1, 1024, 1024)
Dask graph	6171 chunks in 4 graph layers	
Data type	float64 numpy.ndarray	

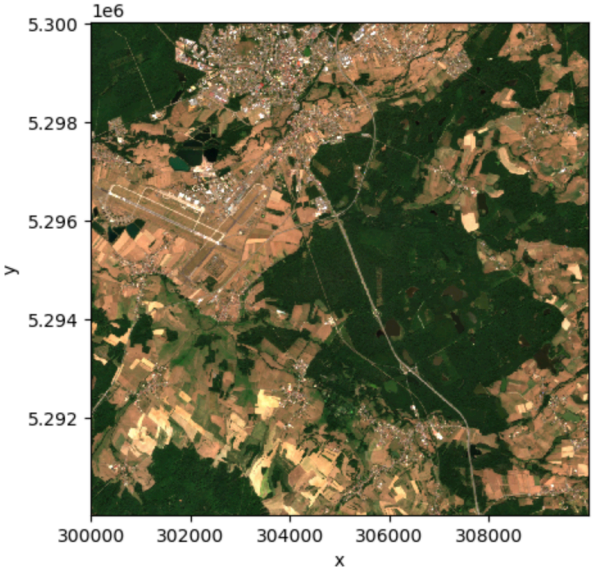


▸ Coordinates: (52)

▸ Indexes: (4)

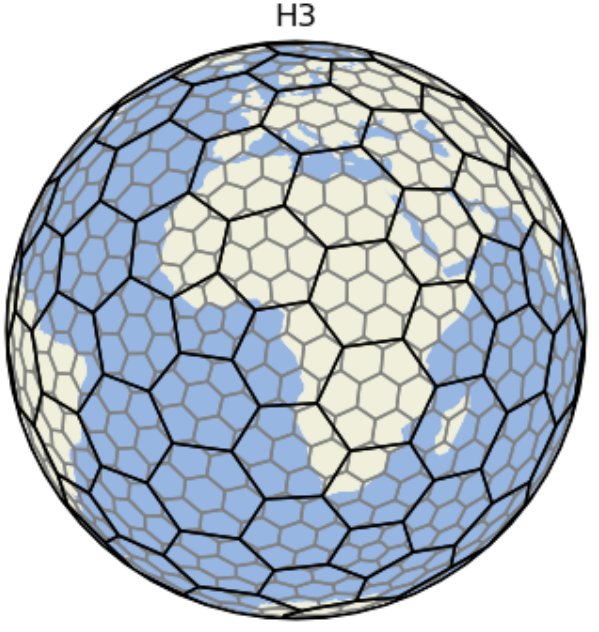
▾ Attributes:

spec :	RasterSpec(epsg=32632, bounds=(300000.0, 5190240.0, 409800.0, 5300040.0), resolutions_xy=(10.0, 10.0))
crs :	epsg:32632
transform :	| 10.00, 0.00, 300000.00| | 0.00,-10.00, 5300040.00| | 0.00, 0.00, 1.00|
resolution :	10.0

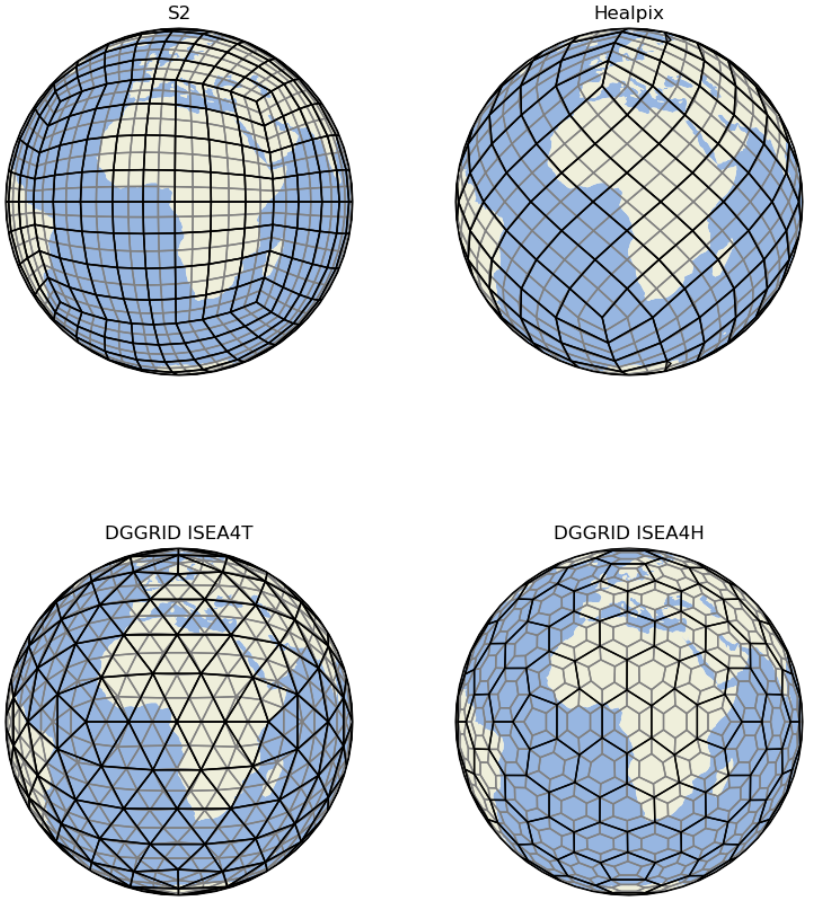


Discrete Global Grid System (DGGS)

Tesselation of the entire Earth surface (sphere) evenly into a hierarchy of grid cells.



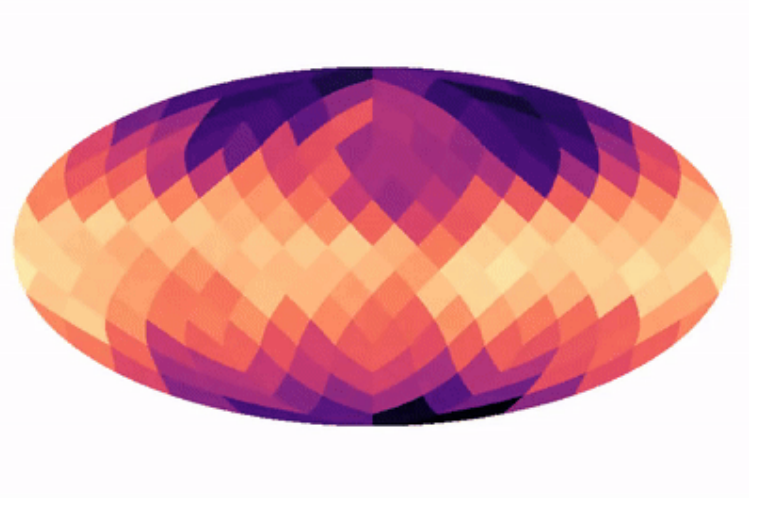
Many different DGGSs



Properties: cell shapes and distribution, aperture, congruent parent-child cells...

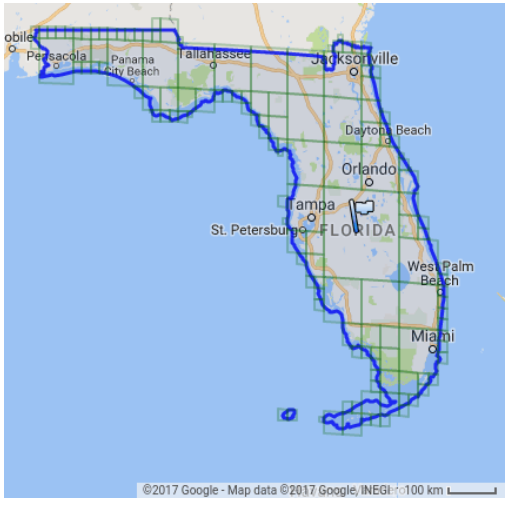
Properties common to all DGGSs

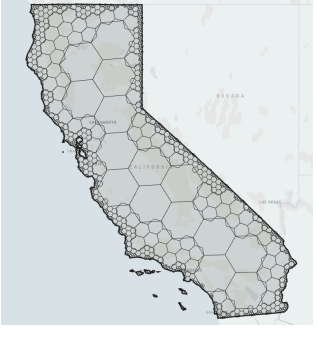
- hierarchical structure: multi-resolution friendly
- unique indexing system: spatial index friendly
- ISO / OGC API standards: work in progress (https://ogcapi.ogc.org/dggs/)

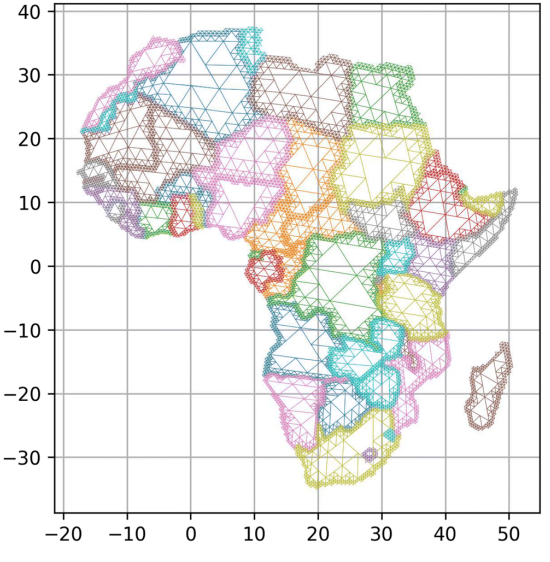


(M. Veerman (WUR) - https://nextgems-h2020.eu)

Multi-resolution







Overview of DGGS Libraries

Library	Language (implementation)	Repository
H3	C	https://github.com/uber/h3
H3o	Rust	https://github.com/HydroniumLabs/h3o
S2	C++	https://github.com/google/s2geometry
Healpix	C - C++ - Fortran	https://healpix.sourceforge.io
DGGRID	C - C++	https://github.com/sahrk/DGGRID
OpenEAGGR	C++	https://github.com/riskaware-ltd/open-eaggr/
STARE	C - C++	https://github.com/SpatioTemporal/STARE

DGGS in Julia

DGGS.jl: https://github.com/danlooo/DGGS.jl

DGGS in R

?

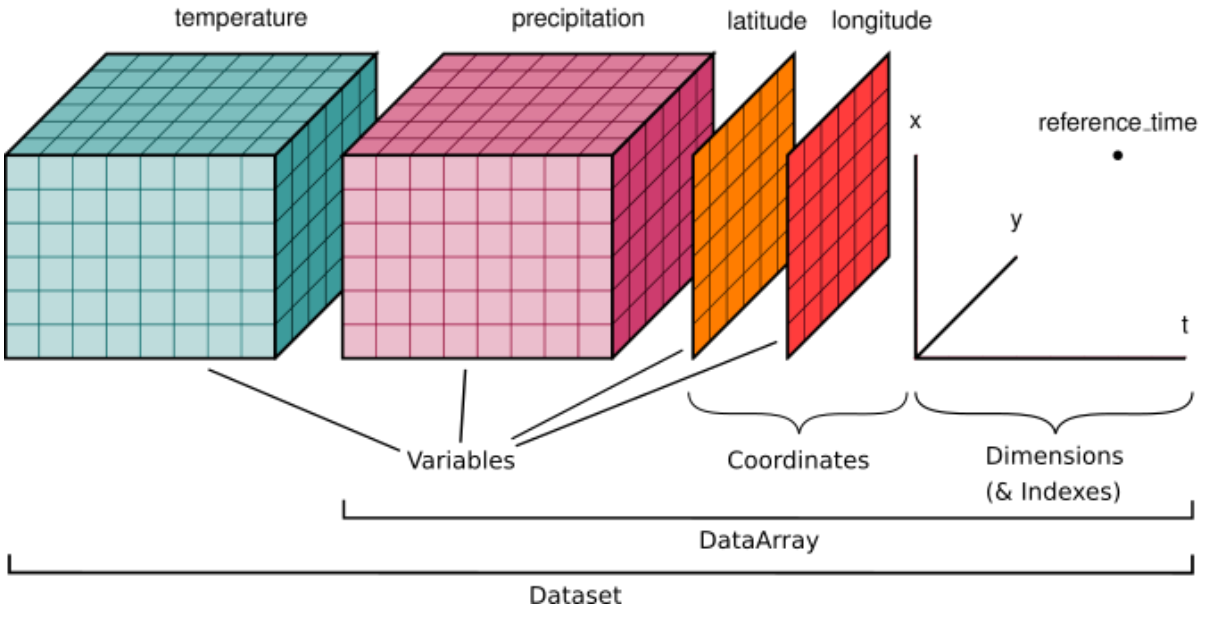
DGGS in Python

Library	Actively maintained?	Vectorized?	Comments
h3py	yes	partially	bindings of H3 (C / Cython)
h3ronpy	yes	yes	built on top of H3o, pandas / arrow integration
S2	no?	no	bindings of S2 (C++ / SWIG)
s2phere	no	no	pure-Python implementation of S2
healpy	yes	yes	
dggrid4py	yes	?	wrapper around DGGRID command-line interface

Towards a unified (Python) API?

Xarray

- labeled Numpy arrays
- multi-dimensional Pandas data frames



Xarray custom indexes

```
da = xr.DataArray(counties.PO60, coords=[counties.geometry], dims=["county"])
da
```

xarray.DataArray 'PO60' (**county**: 3085)

4304 3889 17884 25520 6914 5809 ... 46235 39085 21583 50164 39260

▼ Coordinates:

county	(county)	object	POLYGON ((-95.34258270263672 48....

▼ Indexes:

county	PandasIndex

▶ Attributes: (0)

xvec: https://xvec.readthedocs.io

Xarray custom indexes

```
da = da.drop_indexes("county").set_xindex("county", xvec.GeometryIndex, crs=counties.crs)
da
```

xarray.DataArray 'PO60' (**county**: 3085)

4304 3889 17884 25520 6914 5809 ... 46235 39085 21583 50164 39260

▼ Coordinates:

county	(county)	object	POLYGON ((-95.34258270263672 48....

▼ Indexes:

county	GeometryIndex (crs=EPSG:4326)

▶ Attributes: (0)

xvec: https://xvec.readthedocs.io

Xarray custom indexes

```
da.sel(county=shapely.box(-100, 40, -90, 45), method="intersects")
```

xarray.DataArray 'PO60' (**county**: 313)

5356 13903 11620 16473 7226 9502 ... 4926 16320 13330 9913 11706 20220

▼ Coordinates:

county	(county)	object	POLYGON ((-96.24031066894531 39....

▼ Indexes:

county	GeometryIndex (crs=EPSG:4326)

▶ Attributes: (0)

xvec: https://xvec.readthedocs.io

Xdggs

Example with H3:

```
ds = xr.open_dataset("data/h3_example.nc")
ds
```

xarray.Dataset

▶ Dimensions: (**time**: 2920, **cell**: 5305)

▼ Coordinates:

time	(time)	datetime64[ns]	2013-01-01 ... 2014-12-31T18:00:00	
cell	(cell)	int64	590733444024107007 ... 590995677...	
grid_name :	h3			
resolution :	3			

▼ Data variables:

air	(time, cell)	float64	...	

▼ Indexes:

time	PandasIndex	
cell	PandasIndex	

▶ Attributes: (5)

Xdggs

Example with H3:

```
ds_idx = ds.drop_indexes("cell").set_xindex("cell", xdggs.DGGSIndex)

ds_idx
```

xarray.Dataset

▶ Dimensions: (**time**: 2920, **cell**: 5305)

▼ Coordinates:

time	(time)	datetime64[ns]	2013-01-01 ... 2014-12-31T18:00:00	
cell	(cell)	int64	590733444024107007 ... 590995677...	

▼ Data variables:

air	(time, cell)	float64	...	

▼ Indexes:

time	PandasIndex	
cell	H3Index(resolution=3)	

▶ Attributes: (5)

Xdggs Demo

Community effort!

Alexander Kmoch, Benoît Bovy, Daniel Loos, Evelyn Uuemaa, Wai Tik Chan, Ryan Abernathey, Justus Magin, Alejandro Coca-Castro, Peter Strobl, Anne Fouilloux, Jean-Marc Delouis, Tina Odaka

Code repository: https://github.com/xarray-contrib/xdggs

Xdggs development roadmap

Conversion to/from DGGS

```
# convert from lat/lon grid
ds.dggs.from_latlon_grid( ... )

# convert from raster
ds.dggs.from_raster( ... )

# convert from point data (with aggregation using Xarray API)
ds.dggs.from_points( ... ).groupby( ... ).mean()

# convert to lat/lon grid
ds.dggs.to_latlon_grid( ... )

# convert to raster
ds.dggs.to_raster( ... )

# convert to points (cell centroids)
ds.dggs.to_points( ... )

# convert to polygons (cell boundaries)
ds.dggs.to_polygons( ... )
```

Xdggs development roadmap

Extracting DGGS Cell Geometries (shapely or spherely)

```
# return a DataArray of DGGS cell centroids as shapely.POINT objs
ds.dggs.cell_centroids()

# return a DataArray of DGGS cell boundaries as shapely.POLYGON objs
ds.dggs.cell_boundaries()

# return a DataArray of DGGS cell envelopes as shapely.POLYGON objs
ds.dggs.cell_envelopes()
```

Xdggs development roadmap

Advanced spatial queries

- support it directly via `ds.dggs.query()` if this is supported in backend DGGS libraries
- alternatively, convert cells to vector geometries and use `xvec` (non-optimal)

Hierarchical operations

- change grid resolution (i.e., DGGS-aware `.reindex()`)
- aggregation using Xarray's `.groupby()`

Xdggs development roadmap

Data cube alignment

- DGGS-aware comparison od Datasets and DataArrays

Plotting

- via conversion to a gridded or raster data cube
- via conversion to a vector data cube (`xvec`)
- plotting it directly via `lonboard` (H3, S2)

Conclusion (open questions)

- Use cases
  - Do we really need spherical geometry / DGGS?
  - Which features do we need (roadmap)?
- Leveraging low-level libraries in high-level languages
  - Fragmented ecosystem
- Integration with geo data-frame and data-cube libraries
- Sphere vs. ellipsoid