ANACONDA®

Python Geospatial Tools Webinar

Philipp Rudiger

September 15, 2021

GIS at Anaconda

- We’ve worked with Pangeo / NASA / NCAR / UCAR / USGS / MetOffice
- Tools we created, funded, and/or improved include geopandas, datashader, xarray-spatial, GeoViews, xarray, intake
- Note that we are presenting a very Python-centric view of the ecosystem

ANACONDA.

Challenges

Geospatial analysis is going through a major transformation (like many other fields)

- Data sizes have far outstripped single-core CPU speeds
  - Need scalable, distributed processing
- Data now too big to copy to a local machine
  - Need cloud-native tools
- Monolithic tools vs general/compositional tools
  - Compatibility & maintainability

PANGEO

An initiative to promote open, reproducible, and scalable science

Goals

- Foster collaboration around the open-source scientific Python ecosystem for ocean / atmosphere / land / climate science
- Support the development of domain-specific geoscience packages built on general-purpose computing tools
- Improve scalability of these tools to handle petabyte-scale datasets on HPC and cloud platforms

Old vs New approaches

Old

All-in-one solutions, e.g. ArcGIS, ENVI, MapInfo

Pros:

- Polished, all-in-one solutions

Cons:

- Monolithic
- Usually requires data and compute to be local and colocated
- Difficult to scale
- Difficult to extend

New

General purpose, compositional tools building open source ecosystems

Pros:

- Compositional & Combinatorial
- Built on general underlying libraries
- Scalable
- Easy to extend

Cons:

- Fragmented ecosystem
- Changing quickly

The Ecosystem

PySAL
Zarr
GeoPandas
movingpandas
Folium
Leaflet
GDAL
STAC
SpatioTemporal Asset Catalog
xarray
SpatialPandas
matplotlib
RAPIDS
R.
Iris
GeoViews
PYPROJ
Xarray Spatial
xarray
cartopy
INTAKE
DASK
Datashader
hvPlot

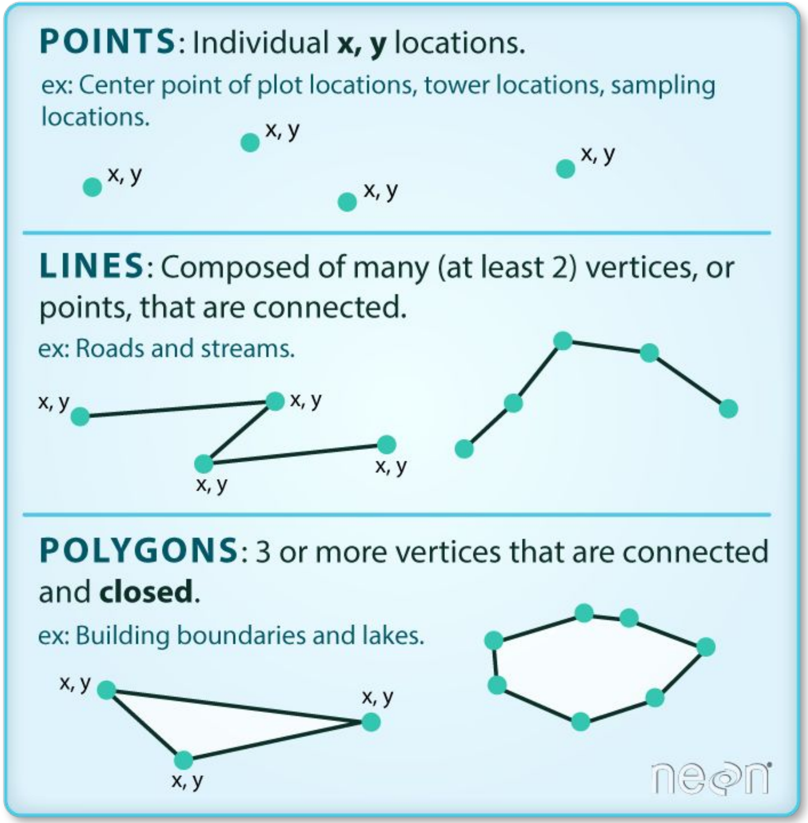
Scaling

To address the challenges of larger, cloud-native datasets developers are pushing to scale existing workflows along a number of axes:

- Horizontal scaling (run on many machines)
  - Dask
- Vertical scaling (faster on one machine)
  - Numba
  - CUDA
- Data Access scaling (indexing big collections)
  - STAC/Intake
  - Zarr and fsspec-reference-maker
- Vectorizing
  - Representing vector data in contiguous memory

Vector Data

Vector Data



POINTS: Individual x, y locations.
ex: Center point of plot locations, tower locations, sampling locations.
x, y
x, y
x, y
x, y
LINES: Composed of many (at least 2) vertices, or points, that are connected.
ex: Roads and streams.
x, y
x, y
x, y
x, y
POLYGONS: 3 or more vertices that are connected and closed.
ex: Building boundaries and lakes.
x, y
x, y
x, y
neon

Tabular Data (points)

Point data can easily and efficiently be represented as a DataFrame

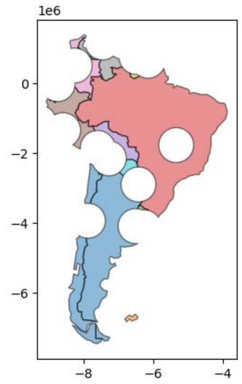
- Pandas: In-memory
- Dask: Out-of-memory/distributed
- RAPIDS cuDF: In GPU memory

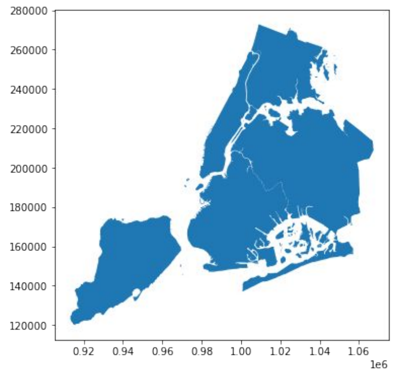
GeoPandas

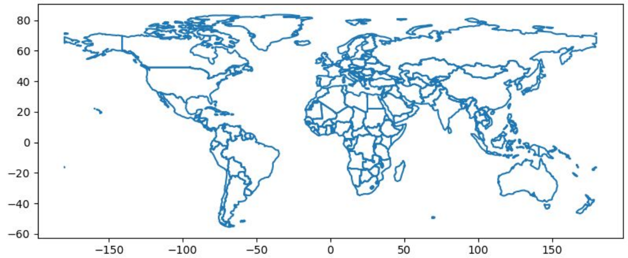
Extends the datatypes used by Pandas to allow spatial operations on geometric types (i.e. (Multi)Point, (Multi)LineString, (Multi)Polygon).

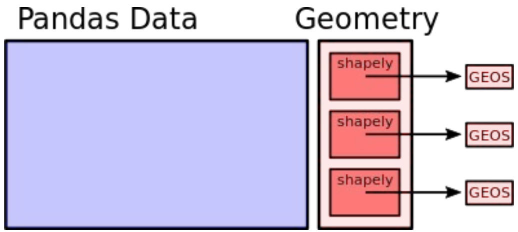
- Geometric operations performed by **Shapely**.
- File Access I/O performed by **Fiona**.
- Optionally stores objects using **PyGEOS**

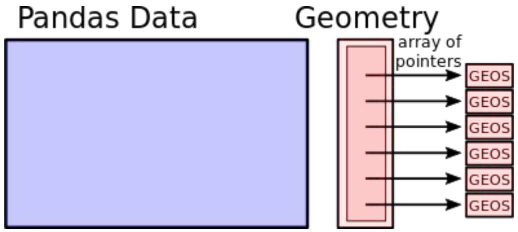
Anaconda funded work to store data as dense array of **PyGEOS** objects











	scalerank	featureclass	geometry
0	1	Country	POLYGON ((-59.57209 -80.04018, -59.86585 -80.5...
1	1	Country	POLYGON ((-159.20818 -79.49706, -161.12760 -79...
2	1	Country	POLYGON ((-45.15476 -78.04707, -43.92083 -78.4...
3	1	Country	POLYGON ((-121.21151 -73.50099, -119.91885 -73...
4	1	Country	POLYGON ((-125.55957 -73.48135, -124.03188 -73...

SpatialPandas

Pandas and Dask extensions for vectorized spatial and geometric operations.

- No dependencies on external geospatial libraries like GEOS
- Efficient memory layouts for geometry data (based on Apache Arrow)
- Fast Numba-based algorithms (e.g. spatial indexing and spatial joins)
- Not expected to be as complete as GeoPandas

Aiming to upstream these capabilities into GeoPandas (and related libraries) rather than forking the ecosystem

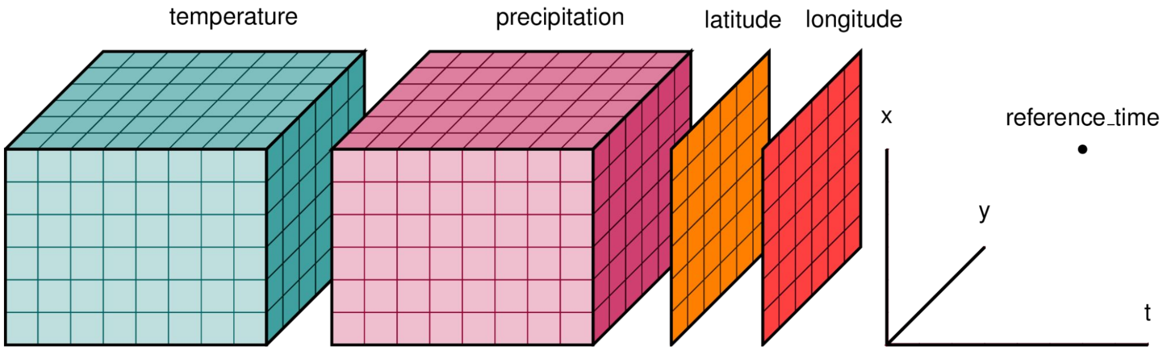
Raster Data

Makes working with labelled multi-dimensional arrays simple

Xarray introduces labels in the form of dimensions, coordinates and attributes on top of raw NumPy-like arrays and supports scaling computations with Dask.

Data ingestion:

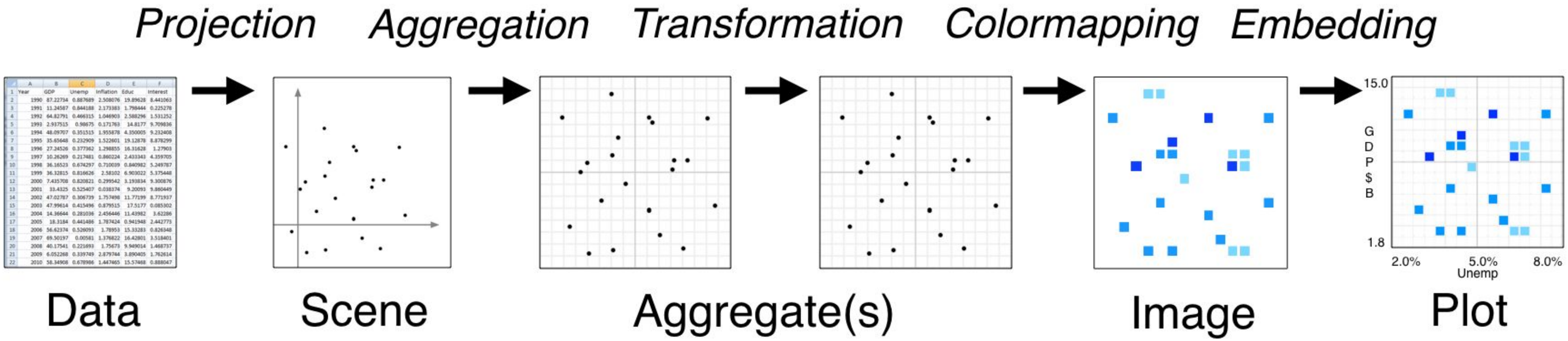
- GeoTiff (via rasterio)
- NetCDF (via netcdf4)
- Zarr
- OPeNDAP
- GRIB
- HDF4



Datashader

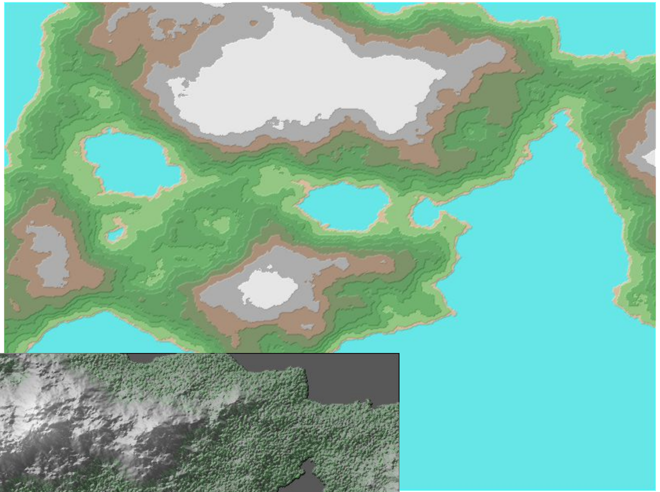
Quickly and accurately rasterizes vector data (or regrids existing raster data)

- Accelerated with Numba (and optionally CUDA)
- Scalable with Dask
- Generates Xarray output



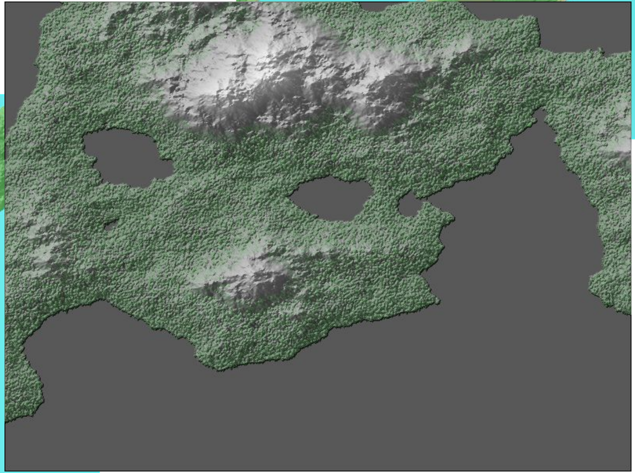
Xarray Spatial

- Fast, accurate Python library for raster operations
- Extensible with Numba
- Scalable with Dask and CUDA
- Free of GDAL / GEOS Dependencies
- General-purpose spatial processing focused on GIS



Utility functions tailored for GIS professionals:

- Surface tools
- Zonal Statistics
- Classification
- Multispectral analysis



Data Access & Catalogs

STAC
SpatioTemporal Asset Catalog

INTAKE

Catalogs provide standardized ways to describe data, publish, and access data, which is crucial for working with large collections of data.

SpatioTemporal Asset Catalog (STAC) provides a common metadata specification, API, and catalog format to describe geospatial assets across languages.

Intake provides a standardized Python way to catalog data to simplify loading and sharing data in data science projects. Plugins allow loading a variety of data, whether NumPy, Pandas, Dask, or Xarray, and interface with everything from STACs, SQL, Zarr files and more.

Visualization

matplotlib

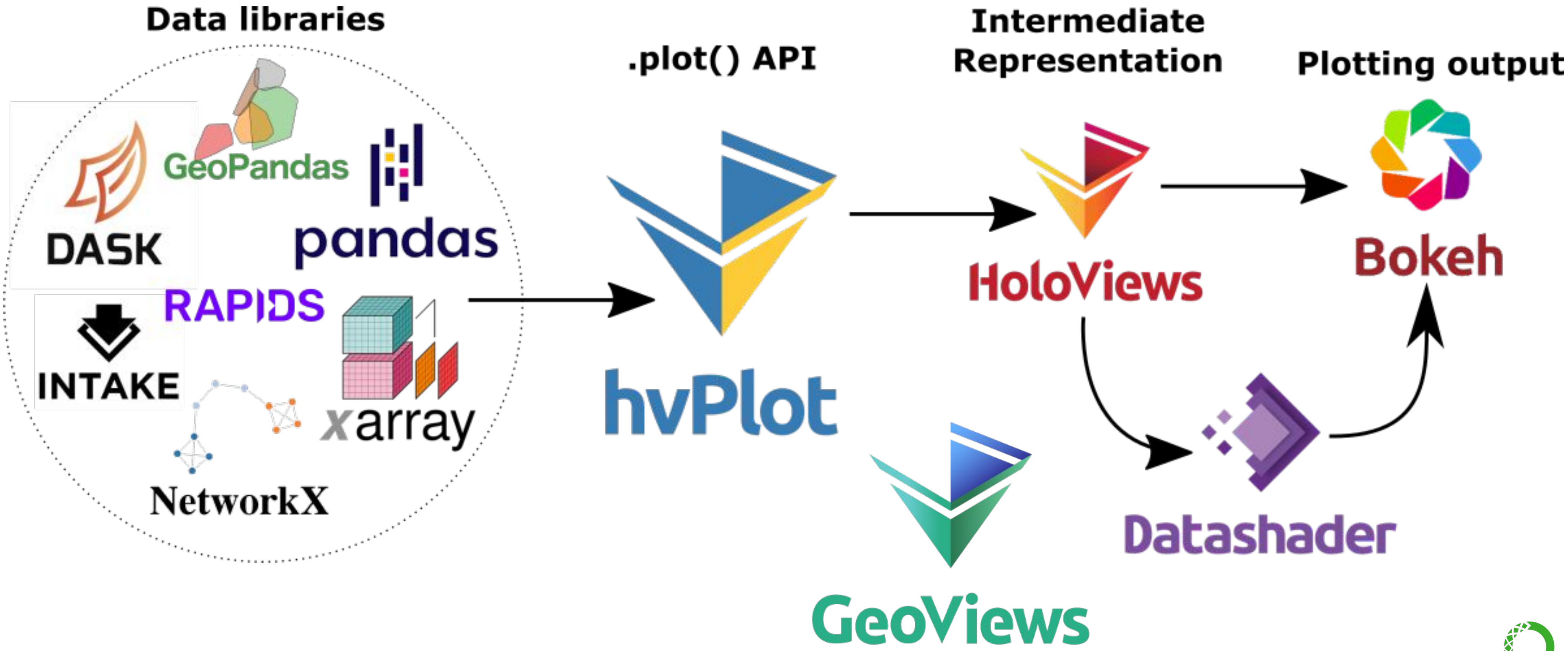
The de-facto standard for plotting in Python.

Integrates with many of the tools mentioned, including:

- xarray
- pandas
- GeoPandas

Many geo-specific extensions, e.g. GeoPlot, Cartopy
(DO NOT USE Basemap any more)

hvPlot


Data libraries
GeoPandas
DASK
pandas
RAPIDS
INTAKE
xarray
NetworkX
.plot() API
hvPlot
Intermediate Representation
HoloViews
GeoViews
Datashader
Plotting output
Bokeh

Summary

Major initiatives underway to:

- Scale geospatial workflows in Python
- Bring compute to your data sitting in the cloud
- Speed up geospatial algorithms
- Improve the Python geospatial ecosystem and foster collaboration

If you have struggled with any of these challenges, try these tools.

Join the community to make the ecosystem grow and prosper!

Thank You!

Philipp Rudiger
prudiger@anaconda.com

ANACONDA®