GeoPandas

Easy, fast and scalable geospatial analysis in Python

Joris Van den Bossche, FOSDEM, February 4, 2018

https://github.com/jorisvandenbossche/talks/

@jorisvdbossche

About me

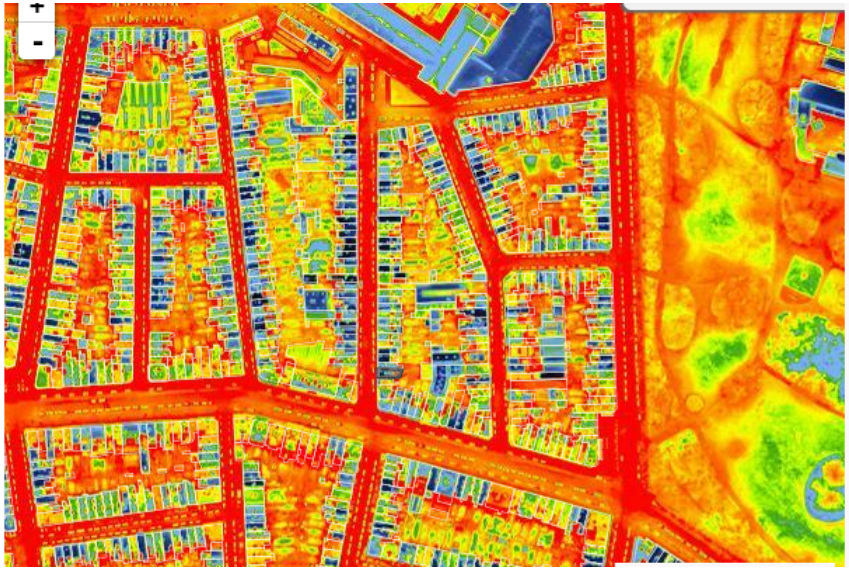
Joris Van den Bossche

- PhD bio-science engineer, air quality research
- pandas core dev, geopandas maintainer
- Currently working at the Université Paris-Saclay Center for Data Science (Inria)

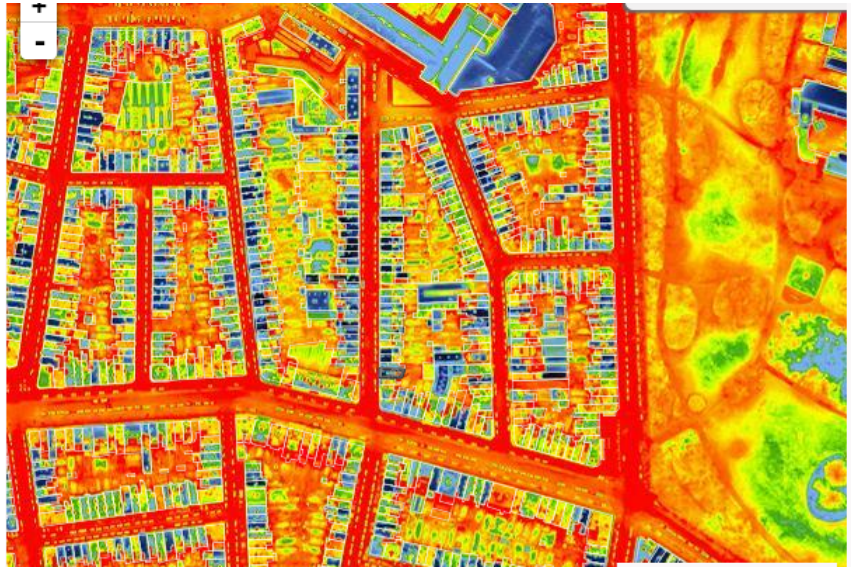
https://github.com/jorisvandenbossche

@jorisvdbossche

Raster vs vector data

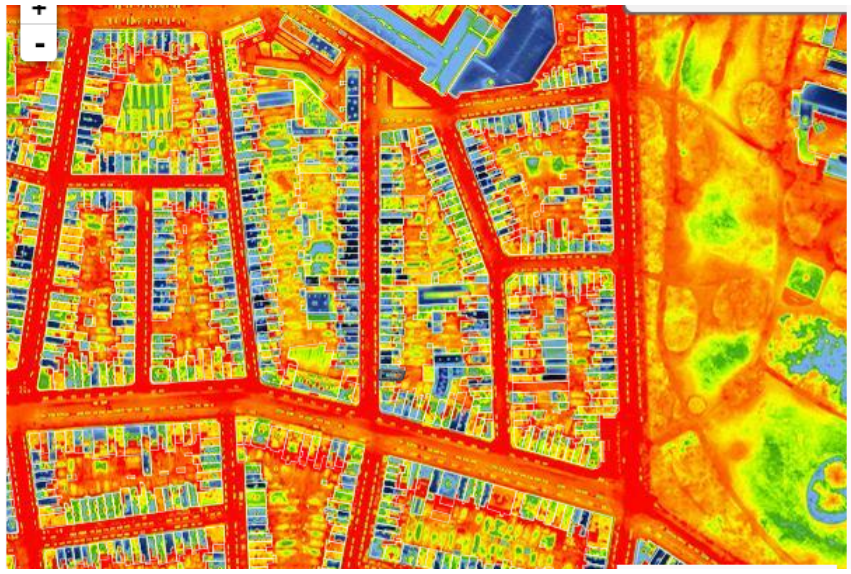


Raster vs vector data



-> in this talk: focus on vector data

Raster vs vector data



-> in this talk: focus on vector data

-> simple features (points, linestrings, polygons) with attributes

Open source geospatial software

GDAL / OGR

Geospatial Data Abstraction Library.

- The swiss army knife for geospatial.
- Read and write Raster (GDAL) and Vector (OGR) datasets
- More than 200 (mainly) geospatial formats and protocols.

Slide from "GDAL 2.2 What's new?" by Even Rouault (CC BY-SA)

GEOS

GEOS Geometry Engine Open Source

Geometry Engine Open Source

- C/C++ port of a subset of Java Topology Suite (JTS)
- Most widely used geospatial C++ geometry library
- Implements geometry objects (simple features), spatial predicate functions and spatial operations

Used under the hood by many applications (QGIS, PostGIS, MapServer, GRASS, GeoDjango, ...)

geos.osgeo.org

Python geospatial packages

Python geospatial packages

Interfaces to widely used libraries:

- Python bindings to GDAL/OGR (`from osgeo import gdal, ogr`)
- `pyproj`: python interface to PROJ.4.
- Pythonic binding to GDAL/OGR:
  - `rasterio` for GDAL
  - `fiona` for OGR
- `shapely`: python package based on GEOS.

Shapely

Python package for the manipulation and analysis of geometric objects

Pythonic interface to GEOS

Shapely

Python package for the manipulation and analysis of geometric objects

Pythonic interface to GEOS

```
>>> from shapely.geometry import Point, LineString, Polygon

>>> point = Point(1, 1)
>>> line = LineString([(0, 0), (1, 2), (2, 2)])
>>> poly = line.buffer(1)
```

```
>>> poly.contains(point)
True
```

Shapely

Python package for the manipulation and analysis of geometric objects

Pythonic interface to GEOS

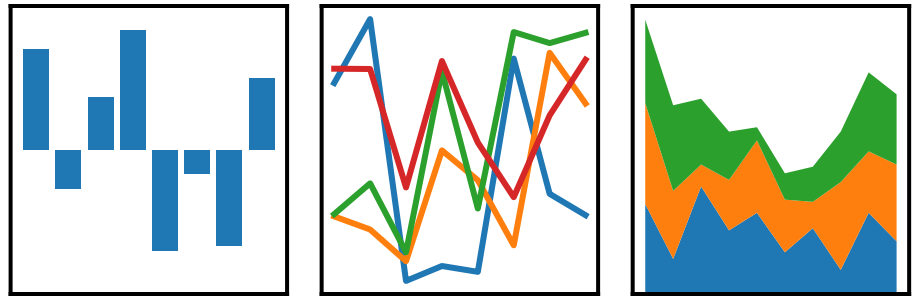
```
>>> from shapely.geometry import Point, LineString, Polygon

>>> point = Point(1, 1)
>>> line = LineString([(0, 0), (1, 2), (2, 2)])
>>> poly = line.buffer(1)
```

```
>>> poly.contains(point)
True
```

Nice interface to GEOS, but: single objects, no attributes

pandas

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



One of the packages driving the growing popularity of Python for data science, machine learning and academic research

- High-performance, easy-to-use data structures and tools
- Suited for tabular data (e.g. columnar data, spread-sheets, database tables)

```
import pandas as pd

df = pd.read_csv("myfile.csv")

subset = df[df['value'] > 0]
subset.groupby('key').mean()
```

GeoPandas

Easy, fast and scalable geospatial analysis in Python

GeoPandas

Make working with geospatial data in python easier

- Started by Kelsey Jordahl in 2013
- Extends the pandas data analysis library to work with geographic objects and spatial operations
- Combines the power of whole ecosystem of (geo) tools (pandas, geos, shapely, gdal, fiona, pyproj, rtree, ...)

Documentation: http://geopandas.readthedocs.io/

Demo time!

See static version

Summary

- Read and write variety of formats (fiona, GDAL/OGR)
- Familiar manipulation of the attributes (pandas dataframe)
- Element-wise spatial predicates (intersects, within, ...) and operations (intersection, union, difference, ..) (shapely)
- Re-project your data (pyproj)
- Quickly visualize the geometries (matplotlib, descartes)
- More advanced spatial operations: spatial joins and overlays (rtree)

Summary

- Read and write variety of formats (fiona, GDAL/OGR)
- Familiar manipulation of the attributes (pandas dataframe)
- Element-wise spatial predicates (intersects, within, ...) and operations (intersection, union, difference, ..) (shapely)
- Re-project your data (pyproj)
- Quickly visualize the geometries (matplotlib, descartes)
- More advanced spatial operations: spatial joins and overlays (rtree)

-> Interactive exploration and analysis of geospatial data

Ecosystem

geoplot (high-level geospatial visualization), cartopy (projection aware cartographic library)

folium (Leaflet.js maps)

OSMnx (python for street networks)

PySAL (Python Spatial Analysis Library)

rasterio (working with geospatial raster data)

...

GeoPandas

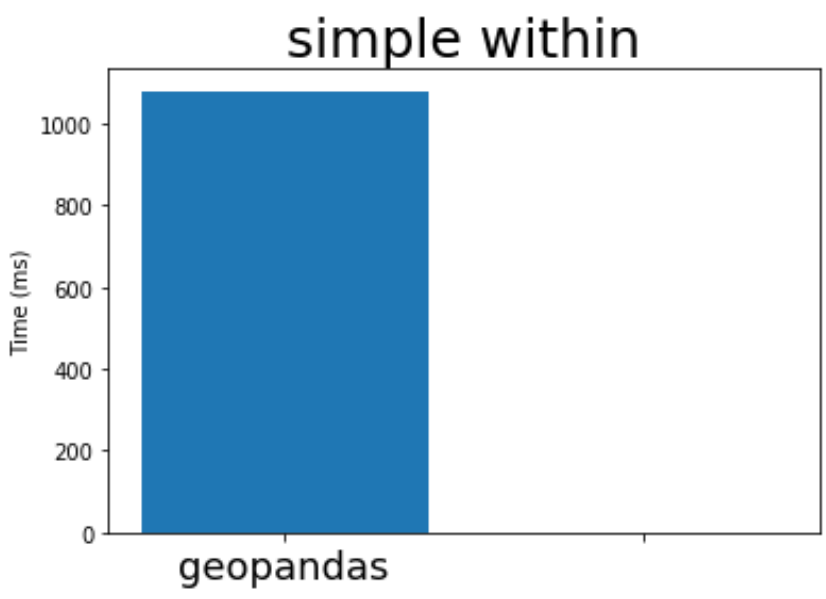
Easy, fast and scalable geospatial analysis in Python

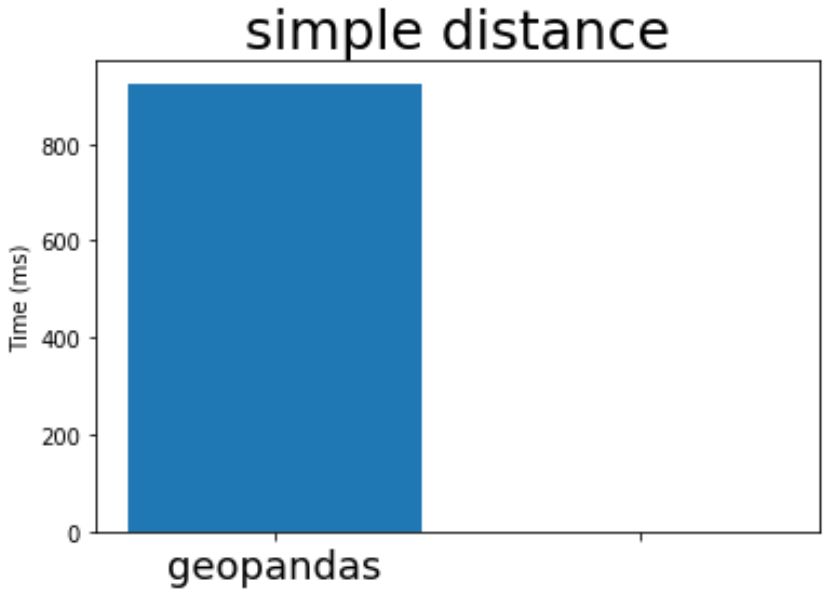
However ...

However ... it can be slow

Timings for basic `within` and `distance` operation on 100 000 points:

```
s.within(polygon)
s.distance(polygon)
```





Comparison with PostGIS

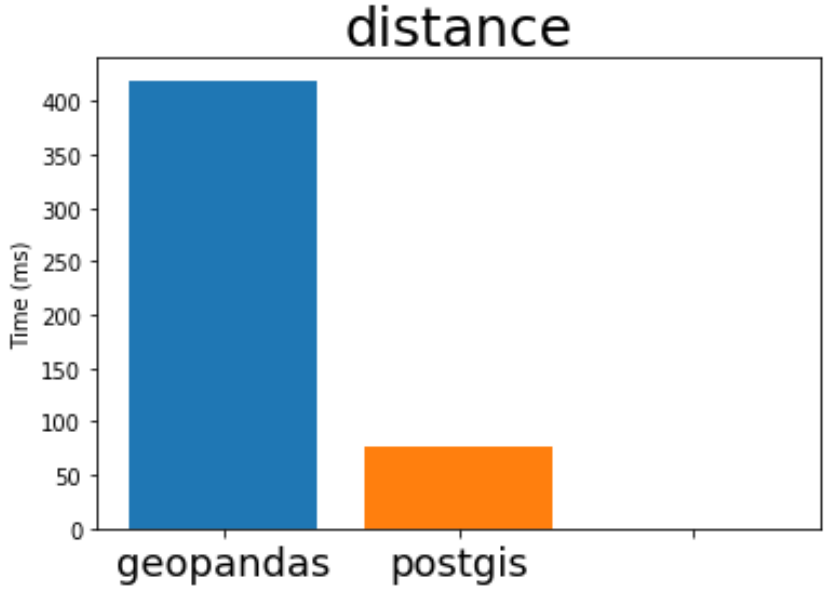
```sql
-- What is the population and racial make-up of the neighborhoods of Manhattan?
SELECT
  neighborhoods.name AS neighborhood_name, Sum(census.popn_total) AS population,
  100.0 * Sum(census.popn_white) / NULLIF(Sum(census.popn_total),0) AS white_pct,
  100.0 * Sum(census.popn_black) / NULLIF(Sum(census.popn_total),0) AS black_pct
FROM nyc_neighborhoods AS neighborhoods
JOIN nyc_census_blocks AS census
ON ST_Intersects(neighborhoods.geom, census.geom)
GROUP BY neighborhoods.name
ORDER BY white_pct DESC;
```

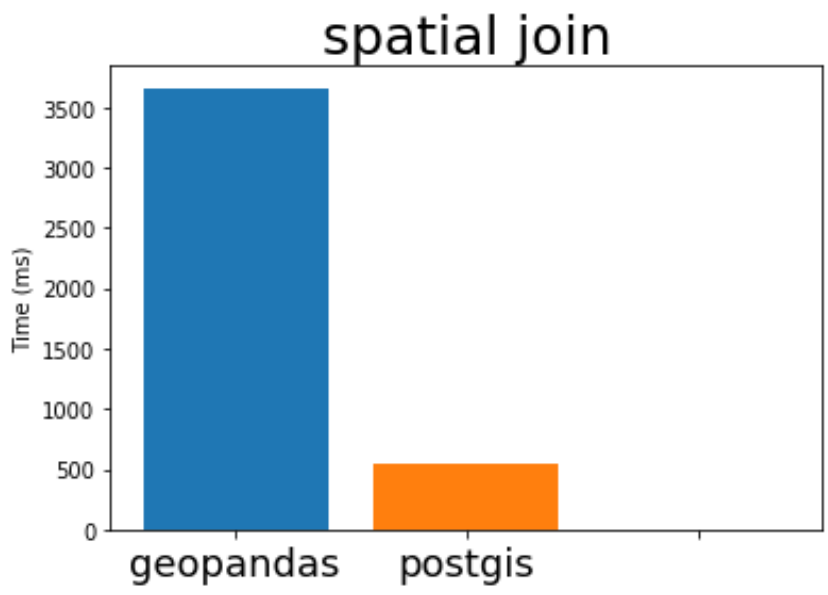
```python
res = geopandas.sjoin(nyc_neighborhoods, nyc_census_blocks, op='intersects')
res = res.groupby('NAME')[['POPN_TOTAL', 'POPN_WHITE', 'POPN_BLACK']].sum()
res['POPN_BLACK'] = res['POPN_BLACK'] / res['POPN_TOTAL'] * 100
res['POPN_WHITE'] = res['POPN_WHITE'] / res['POPN_TOTAL'] * 100
res.sort_values('POPN_WHITE', ascending=False)
```

Disclaimer: dummy benchmark, and I am not a PostGIS expert!

Example from Boundless tutorial (CC BY SA)

Comparison with PostGIS



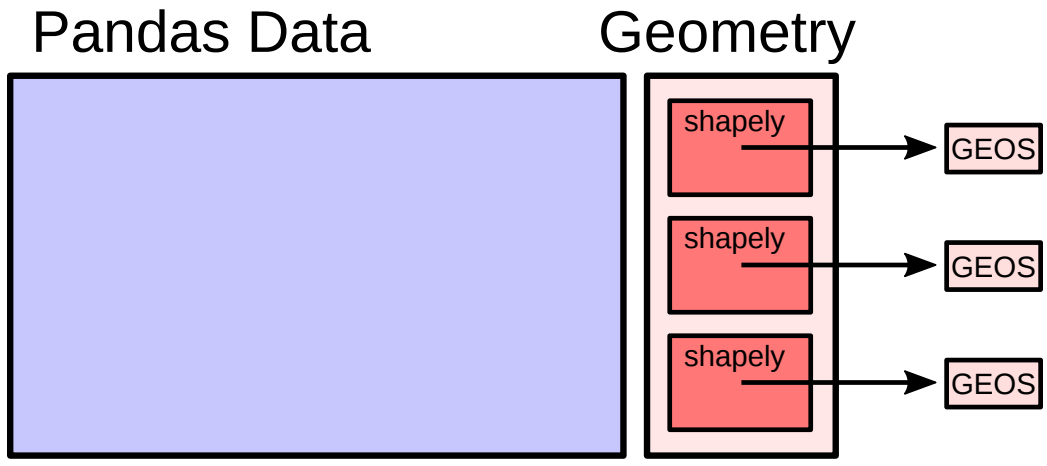


Disclaimer: dummy benchmark, and I am not a PostGIS expert!

Example from Boundless tutorial (CC BY SA)

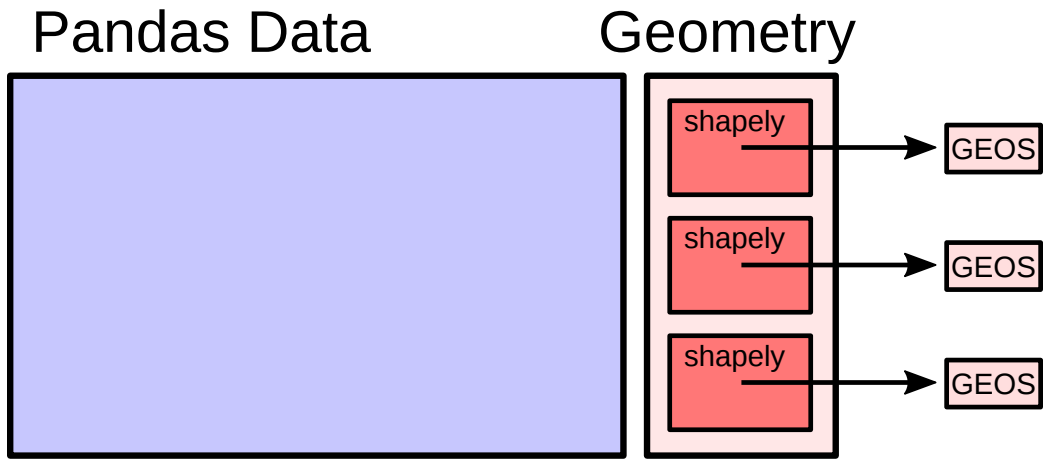
Why is GeoPandas slower?

- GeoPandas stores custom Python objects in arrays
- For operations, it iterates through those objects
- Those Python objects each call the GEOS C operation

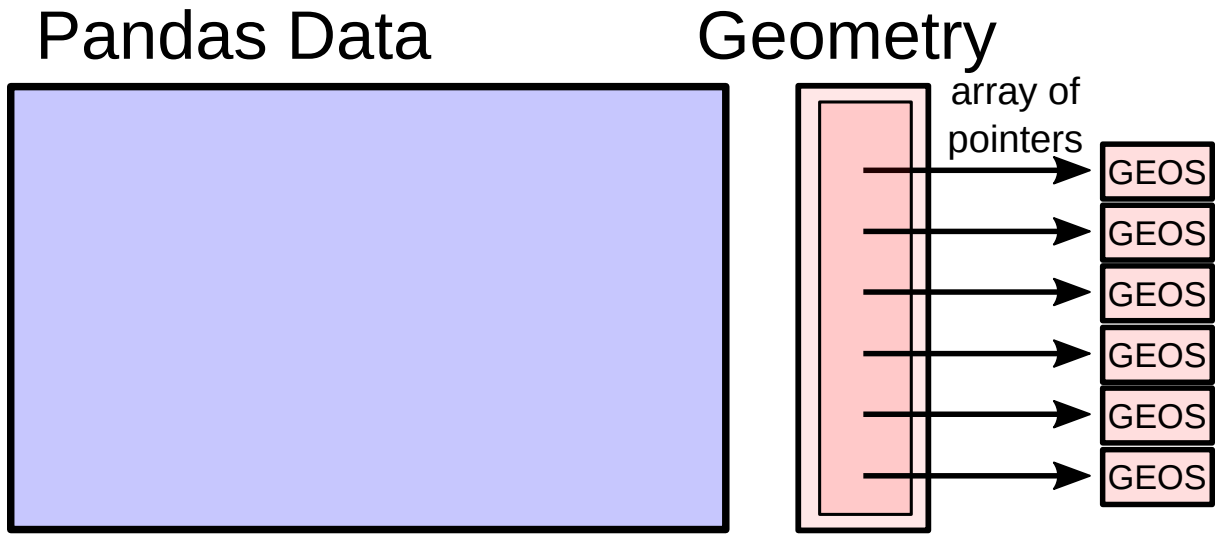


Why is GeoPandas slower?

- GeoPandas stores custom Python objects in arrays
- For operations, it iterates through those objects
- Those Python objects each call the GEOS C operation



New version in development

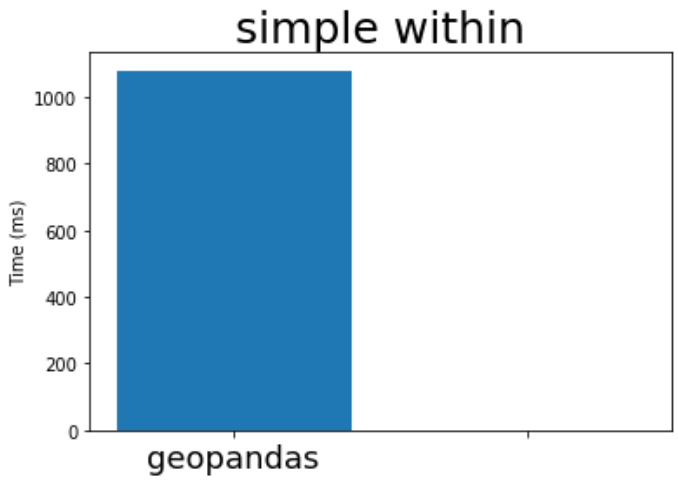


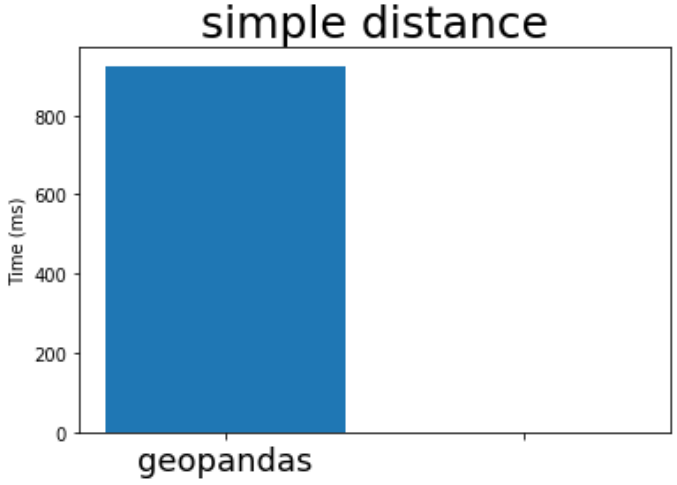
Remove python overhead by only storing pointers to C GEOS objects and iterating in C

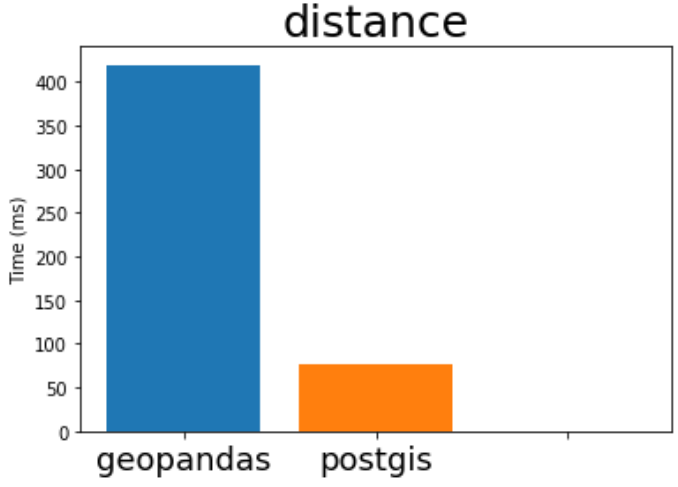
TL;DR: same API, but better performance and less memory use

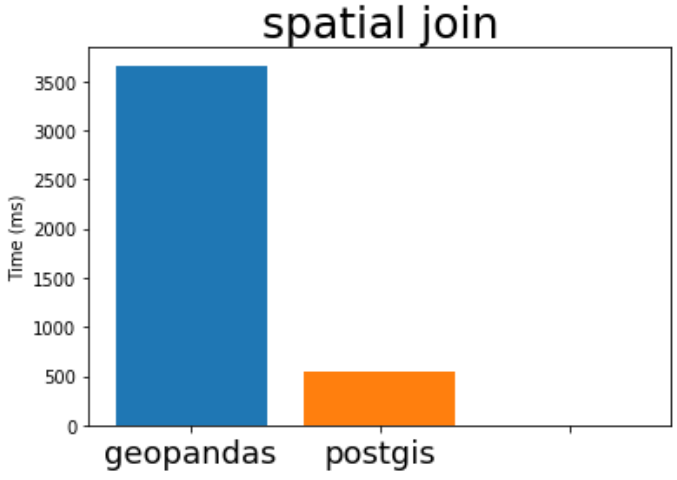
Many thanks to Matthew Rocklin (Anaconda, Inc.) for his work!

New timings

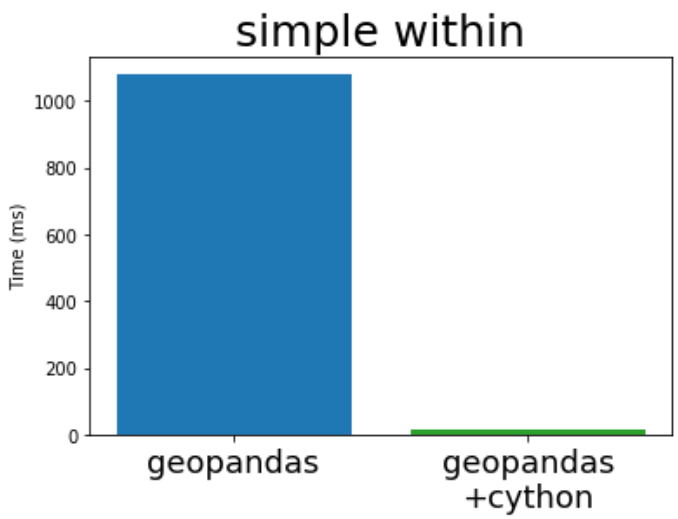

simple within
Time (ms)
0
200
400
600
800
1000
geopandas

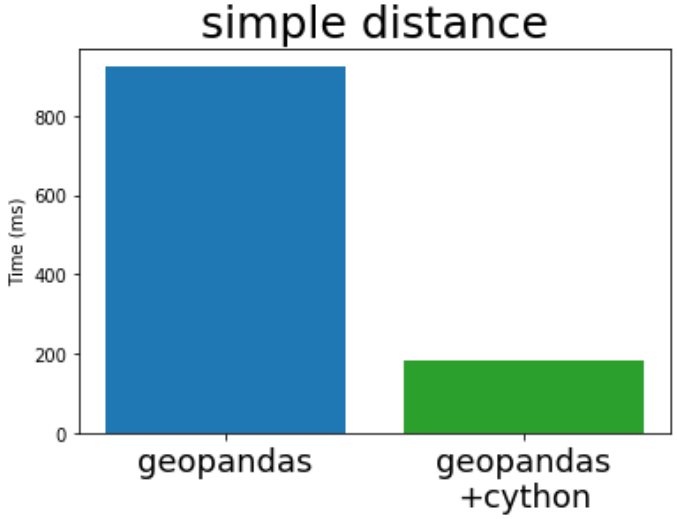

simple distance
Time (ms)
0
200
400
600
800
geopandas

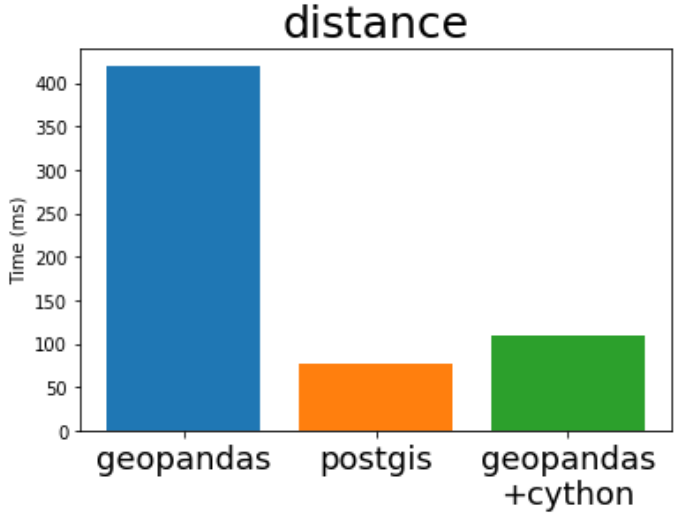

distance
Time (ms)
0
50
100
150
200
250
300
350
400
geopandas
postgis

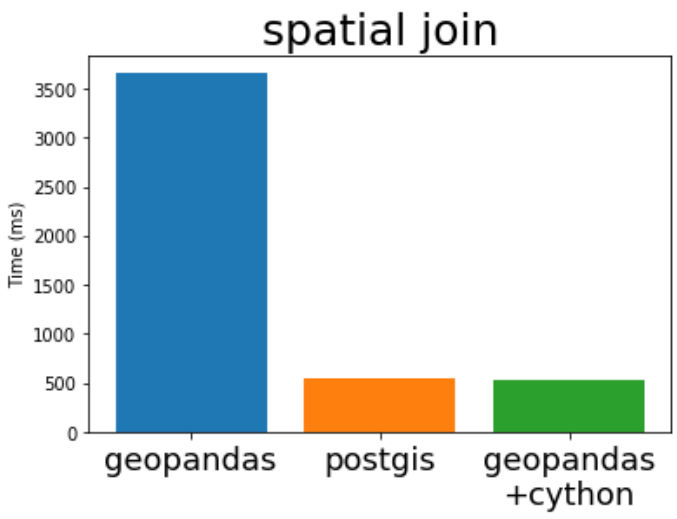

spatial join
Time (ms)
0
500
1000
1500
2000
2500
3000
3500
geopandas
postgis

New timings


simple within
Time (ms)
0
200
400
600
800
1000
geopandas
geopandas
+cython


simple distance
Time (ms)
0
200
400
600
800
geopandas
geopandas
+cython


distance
Time (ms)
0
50
100
150
200
250
300
350
400
geopandas
postgis
geopandas
+cython


spatial join
Time (ms)
0
500
1000
1500
2000
2500
3000
3500
geopandas
postgis
geopandas
+cython

Sounds interesting?

Blogpost of me and Matthew with more background:

- http://matthewrocklin.com/blog/work/2017/09/21/accelerating-geopandas-1
- https://jorisvandenbossche.github.io/blog/2017/09/19/geopandas-cython/

Try out development version (binary builds):

```
conda install --channel conda-forge/label/dev geopandas
```

GeoPandas

Easy, fast and scalable geospatial analysis in Python

A flexible library for parallelism

A flexible library for parallelism

- A parallel computing framework, written in pure Python
- Lets you work on larger-than-memory datasets
- That leverages the excellent Python ecosystem
- Using blocked algorithms and task scheduling

http://dask.pydata.org/

An experiment with taxi data

Ravi Shekhar published a blogpost Geospatial Operations at Scale with Dask and GeoPandas in which he counted the number of rides originating from each of the official taxi zones of New York City

Matthew Rocklin re-ran the experiment with the in-development version: 3h -> 8min (see his blogpost)

dask-geopandas: experimental library with parallelized geospatial operations and joins

An experiment with taxi data

Ravi Shekhar published a blogpost Geospatial Operations at Scale with Dask and GeoPandas in which he counted the number of rides originating from each of the official taxi zones of New York City

Matthew Rocklin re-ran the experiment with the in-development version: 3h -> 8min (see his blogpost)

dask-geopandas: experimental library with parallelized geospatial operations and joins

Demo time!

Thanks for listening!

Thanks to all contributors!

Those slides:

- https://github.com/jorisvandenbossche/talks/
- jorisvandenbossche.github.io/talks/2018_FOSDEM_geopandas

http://geopandas.readthedocs.io

About me

Joris Van den Bossche

- PhD bio-science engineer, air quality research
- pandas core dev, geopandas maintainer
- Currently working at the Université Paris-Saclay Center for Data Science (Inria)

https://github.com/jorisvandenbossche

@jorisvdbossche