GEOS in the Python ecosystem

Joris Van den Bossche, FOSS4G Belgium, October 24, 2019

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

@jorisvdbossche

About me

Joris Van den Bossche

- Background: PhD bio-science engineer, air quality research
- Open source enthusiast: pandas core dev, geopandas maintainer, scikit-learn contributor
- Currently freelance open source developer and teacher, working part-time on Apache Arrow (at Ursa Labs)

https://github.com/jorisvandenbossche Twitter: @jorisvdbossche

pandas

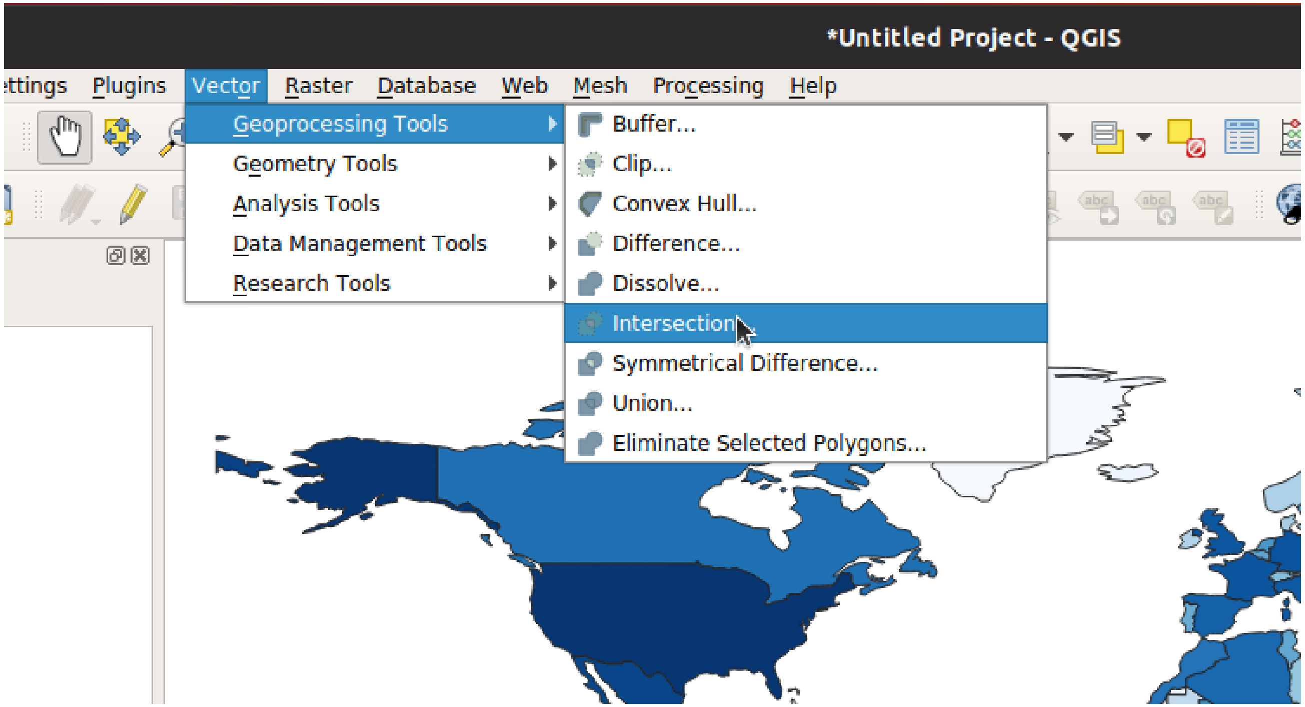
$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

URSA LABS

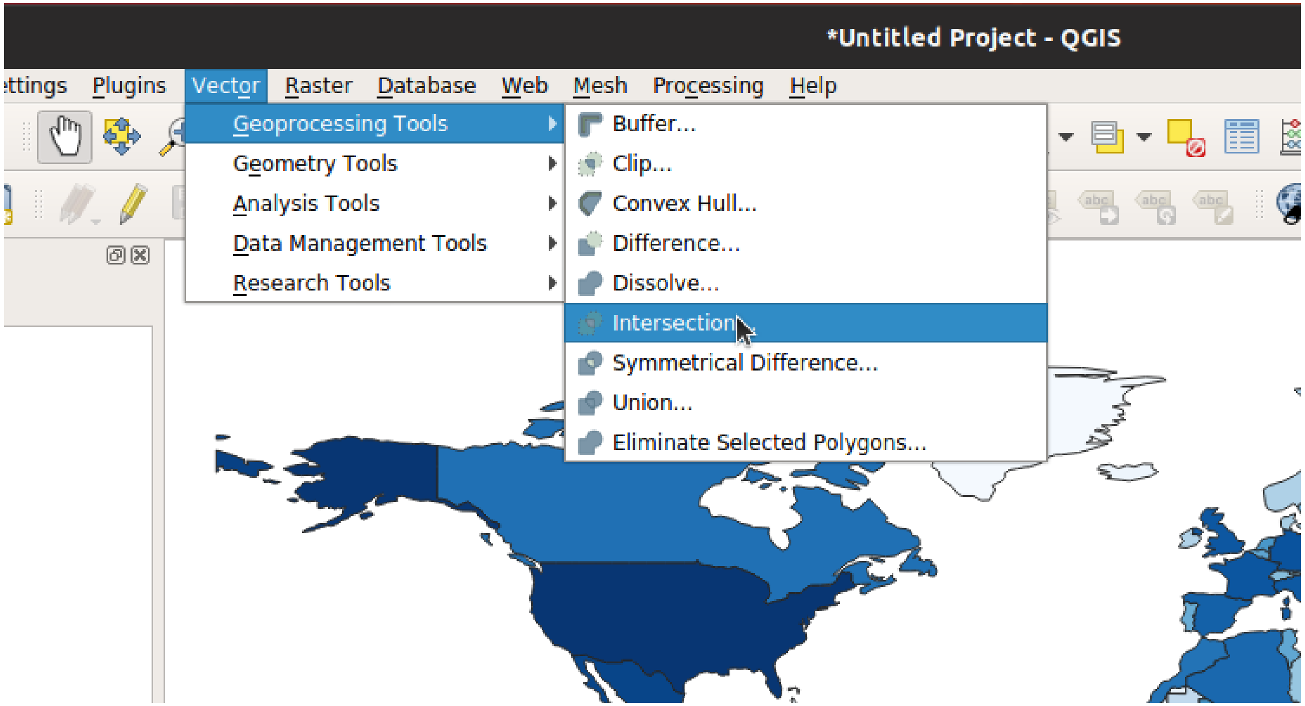
Innovation Lab for Data Science Tools

GEOS

Vector processing in QGIS



Vector processing in QGIS



➡ using the **GEOS** library under the hood.

Vector processing in Postgis

Example from https://postgis.net/workshops/postgis-intro/:

```sql
SELECT
  subways.name AS subway_name,
  neighborhoods.name AS neighborhood_name,
  neighborhoods.boroname AS borough
FROM nyc_neighborhoods AS neighborhoods
JOIN nyc_subway_stations AS subways
ON ST_Contains(neighborhoods.geom, subways.geom)
WHERE subways.name = 'Broad St';
```

Vector processing in Postgis

Example from https://postgis.net/workshops/postgis-intro/:

```
SELECT
  subways.name AS subway_name,
  neighborhoods.name AS neighborhood_name,
  neighborhoods.boroname AS borough
FROM nyc_neighborhoods AS neighborhoods
JOIN nyc_subway_stations AS subways
ON ST_Contains(neighborhoods.geom, subways.geom)
WHERE subways.name = 'Broad St';
```

➡ using the **GEOS** library under the hood.

Vector processing in R (sf)

Snippets from presentation last year (https://pokyah.shinyapps.io/foss4GBXL2018):

```
library(sf)

belgium = sf::st_as_sf(
    rnaturalearth::ne_states(country = 'belgium'))
wallonia = belgium %>% dplyr::filter(region == "Walloon")
grid = sf::st_intersection(
    grid, sf::st_transform(wallonia, crs = 3812))
```

Vector processing in R (sf)

Snippets from presentation last year (https://pokyah.shinyapps.io/foss4GBXL2018):

```r
library(sf)

belgium = sf::st_as_sf(
    rnaturalearth::ne_states(country = 'belgium'))
wallonia = belgium %>% dplyr::filter(region == "Walloon")
grid = sf::st_intersection(
    grid, sf::st_transform(wallonia, crs = 3812))
```

➡ using the **GEOS** library under the hood.

Vector processing in Python

Using Shapely and GeoPandas:

```
import geopandas
import shapely.geometry

districts = geopandas.read_file("paris_districts.gpkg")
notre_dame = shapely.geometry.Point(452321, 5411311)

# filter districts that contain the point
districts[districts.contains(notre_dame)]
```

Vector processing in Python

Using Shapely and GeoPandas:

```python
import geopandas
import shapely.geometry

districts = geopandas.read_file("paris_districts.gpkg")
notre_dame = shapely.geometry.Point(452321, 5411311)

# filter districts that contain the point
districts[districts.contains(notre_dame)]
```

➡ using the **GEOS** library under the hood.

GEOS

Geometry Engine Open Source

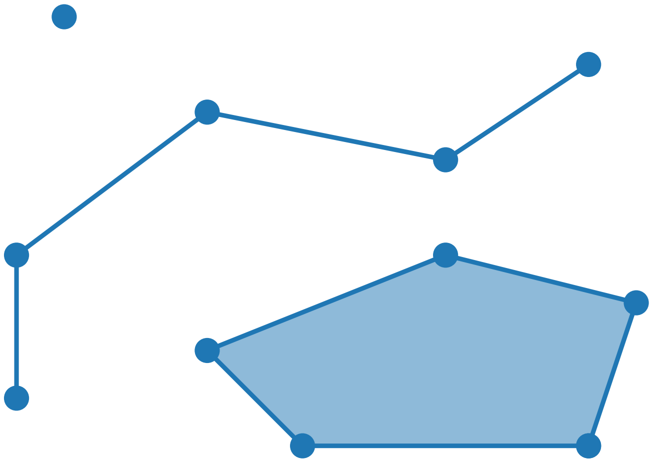
- C/C++ port of a subset of Java Topology Suite (JTS)
- Most widely used geospatial C++ geometry library
- Implements geometry objects (simple features), spatial predicate functions and spatial operations, prepared geometries, STR spatial index, WKT/WKB encoding and decoding

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

geos.osgeo.org

Simple features

Simple feature access - OGC / ISO standard:



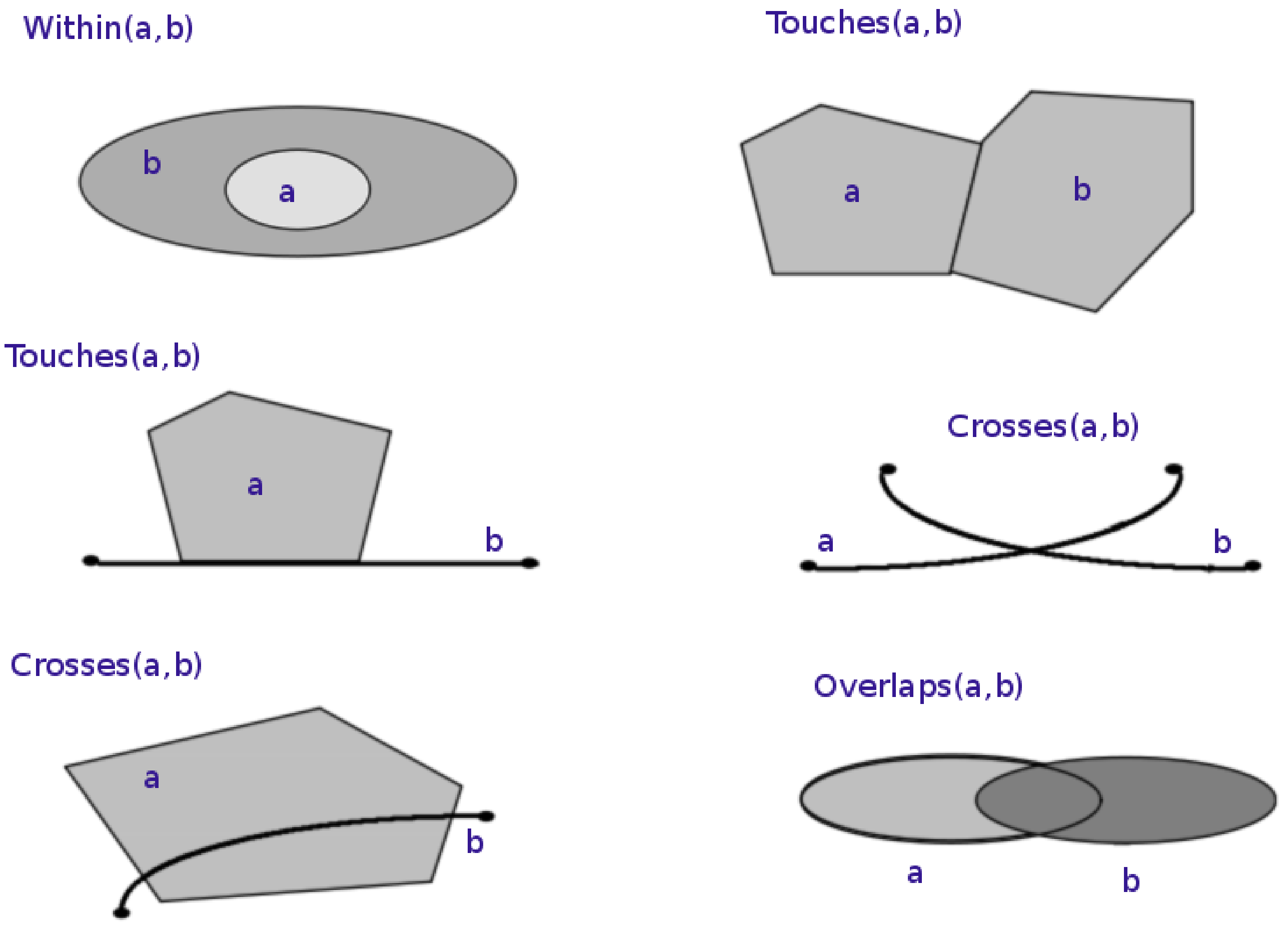
Point(2, 10)

LineString([(1, 2), (1, 5), ...])

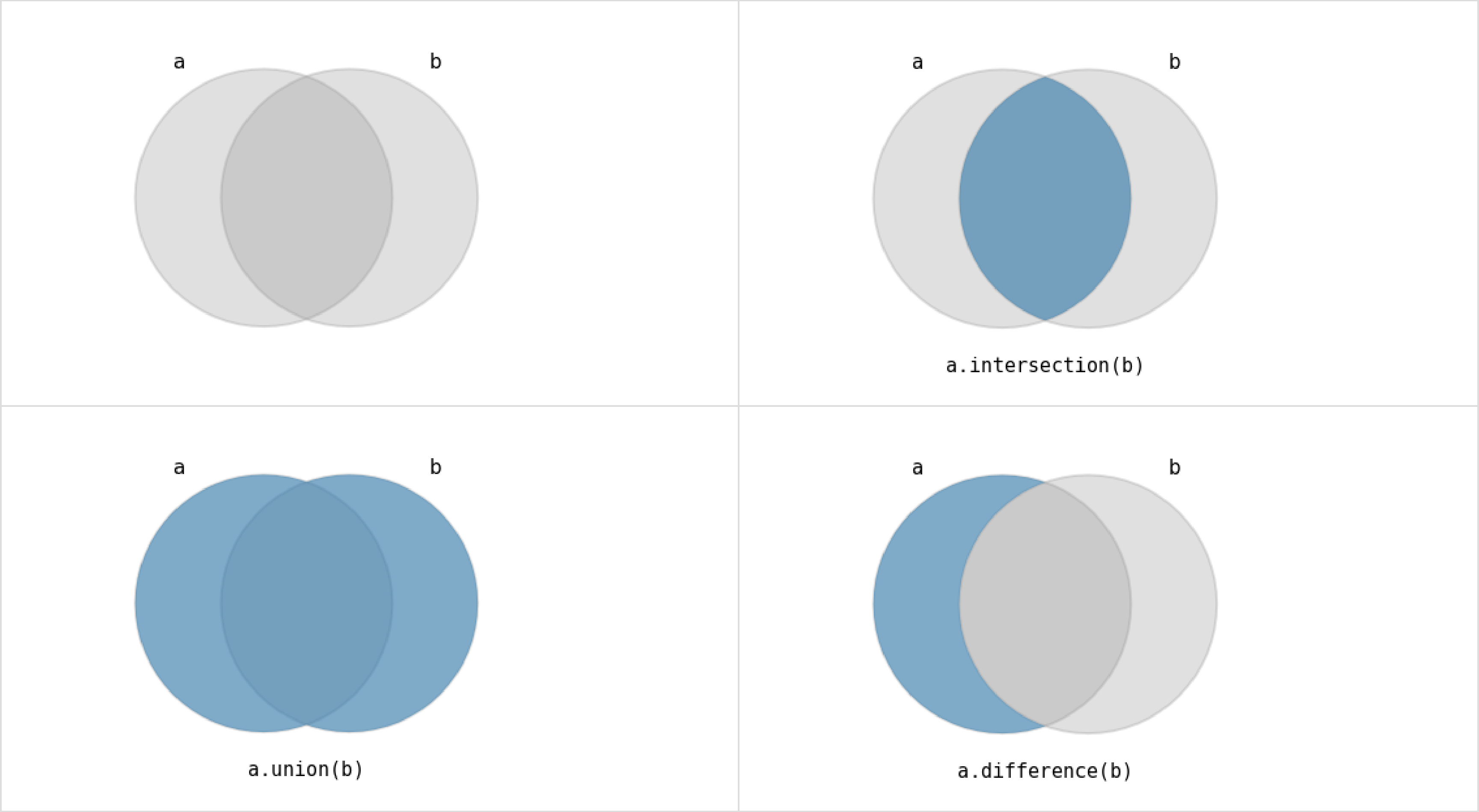
Polygon([(13, 1), (14, 4), ...])

Spatial predicates

https://en.wikipedia.org/wiki/DE-9IM



Spatial operations

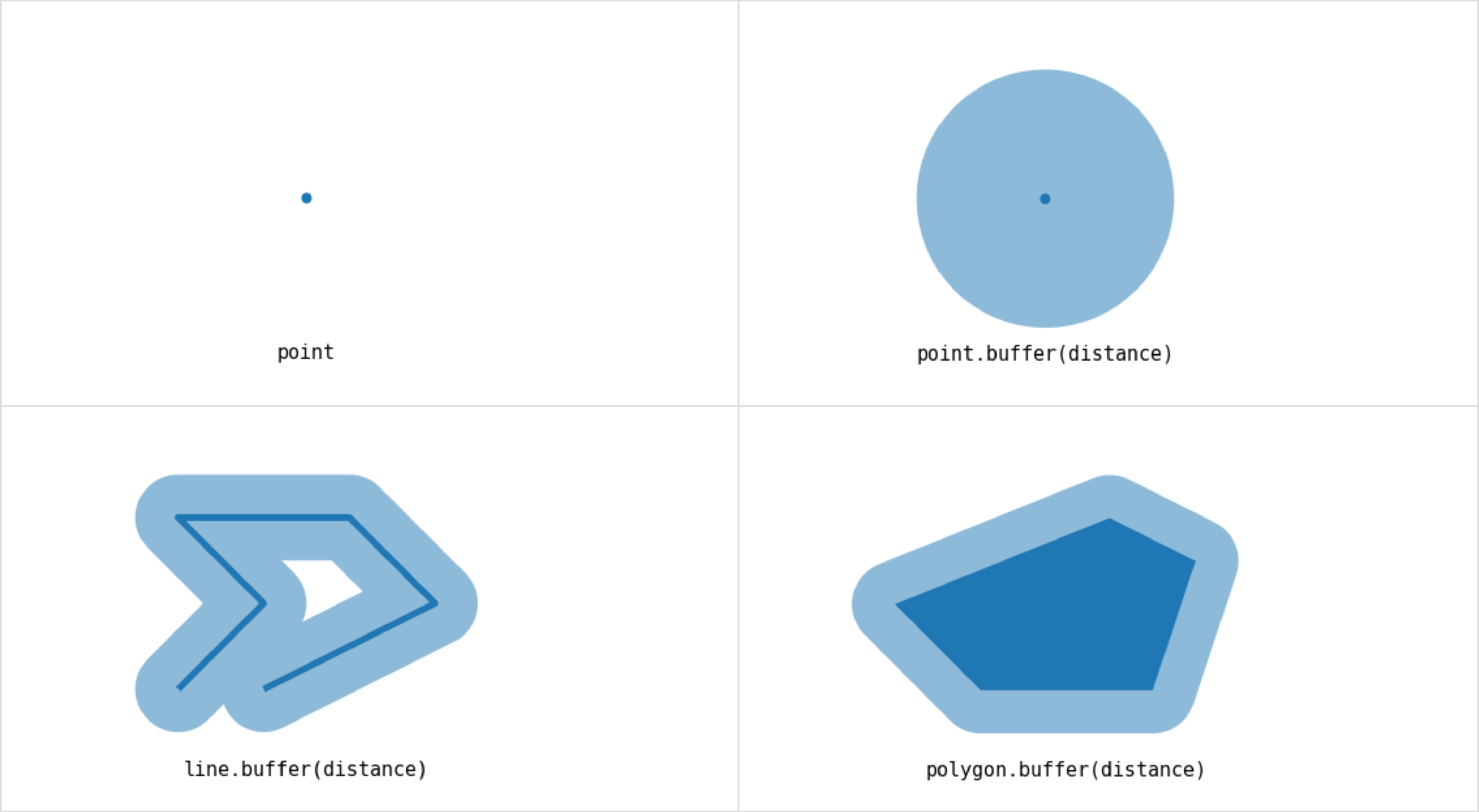


a.intersection(b)

a.union(b)

a.difference(b)

Spatial operations



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, prepared geometries, STR spatial index, WKT/WKB encoding and decoding

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

geos.osgeo.org

GEOS in the Python ecosystem

Shapely

Python package for the manipulation and analysis of geometric objects

Pythonic interface to GEOS

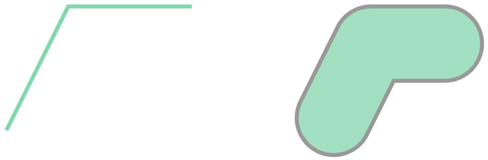
Shapely

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
```

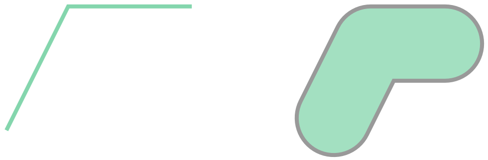
Shapely

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

GeoPandas

Make working with tabular geospatial data in python easier by combining Shapely and pandas

- 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, ...)
- Bridge between geospatial packages and the scientific / data science stack

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

GeoPandas

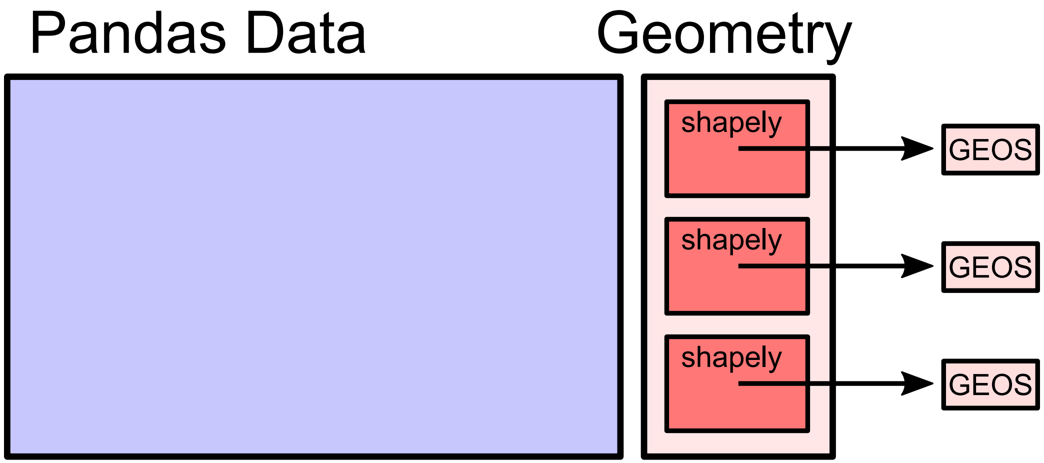
Make working with tabular geospatial data in python easier by combining Shapely and pandas

```
>>> df = geopandas.read_file("ne_110m_admin_0_countries.shp")
>>> df
       pop_est      continent       name iso_a3  gdp_md_est                       geometry
0       920938        Oceania       Fiji    FJI      8374.0  MULTIPOLYGON (((180.00000 ...
1     53950935         Africa   Tanzania    TZA    150600.0  POLYGON ((33.90371 -0.9500...
2       603253         Africa  W. Sahara    ESH       906.5  POLYGON ((-8.66559 27.6564...
3     35623680  North America     Canada    CAN   1674000.0  MULTIPOLYGON (((-122.84000...
..         ...            ...        ...    ...         ...                            ...

>>> df = df.to_crs(epsg=3857)
>>> df.geometry.area / 1e9
0         21.283337
1        952.255175
2        117.102338
3      52166.480440
..              ...
```

Why is GeoPandas slow?

- 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 slow?

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

```
class GeoSeries:
    ...

    def distance(self, other):
        result = [geom.distance(other) for geom in self.geometry]
        return pd.Series(result)
```

Introducing PyGEOS

Introducing PyGEOS

New library that exposes geospatial operations from GEOS into Python:

- array-based
- fast

Introducing PyGEOS

New library that exposes geospatial operations from GEOS into Python:

- array-based
- fast

Started by Casper van der Wel:
https://caspervdw.github.io/Introducing-Pygeos/

GitHub repo:
https://github.com/pygeos/pygeos/

Array-based

Instead of (using Shapely)

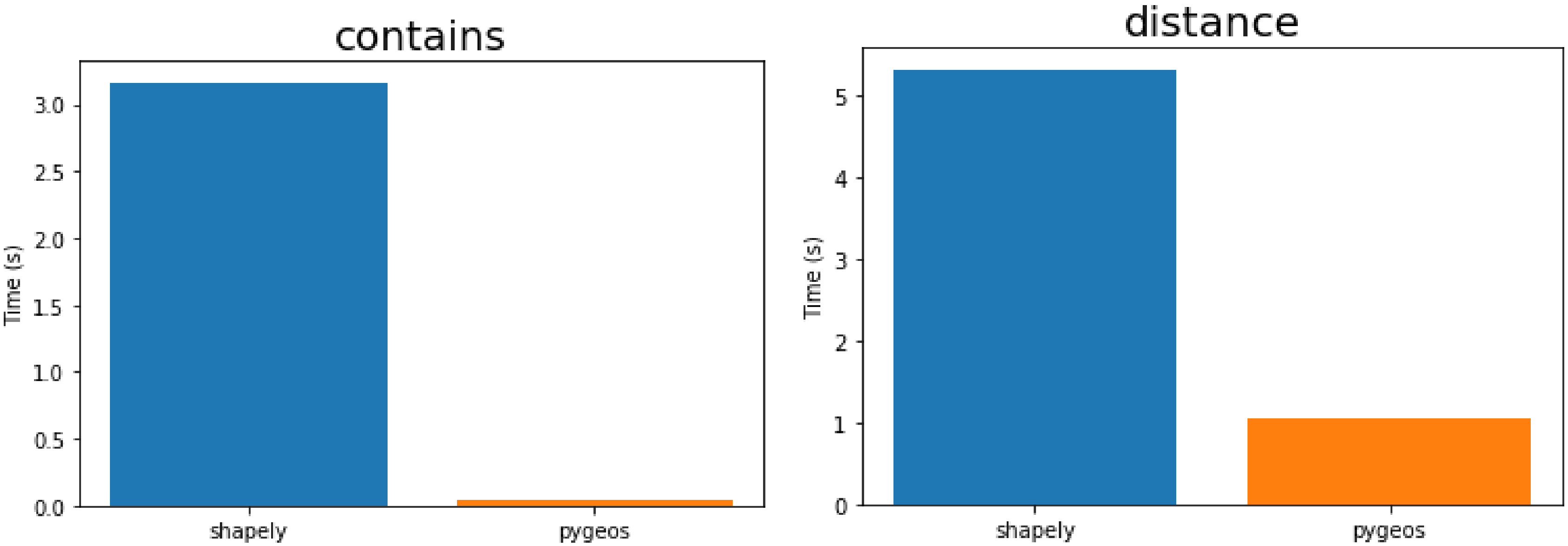
```
[poly.contains(point) for point in points]
```

you can do

```
pygeos.contains(poly, points)
```

Fast

Benchmark for 1M points: contained in or distance to a polygon



Significant performance increase: 80x (contains) to 5x (distance) for this example

Numpy "universal functions"

Numpy universal functions (ufuncs) are vectorized functions that work on arrays element-by-element supporting numpy features such as broadcasting

Demo!

Running in parallel (WIP)

Possibility to run in parallel (releasing the GIL)

Combination with Dask (https://dask.org/):

```
# with pygeos, single core
res1 = pygeos.distance(points, poly)
```

```
# chunked using dask, multi-threaded
points_chunked = dask.array.from_array(points, chunks=100_000)
res2 = points_chunked.map_blocks(pygeos.distance, poly, dtype=float)
```

Running in parallel (WIP)

Possibility to run in parallel (releasing the GIL)

Combination with Dask (https://dask.org/):

```
# with pygeos, single core
res1 = pygeos.distance(points, poly)
```

```
# chunked using dask, multi-threaded
points_chunked = dask.array.from_array(points, chunks=100_000)
res2 = points_chunked.map_blocks(pygeos.distance, poly, dtype=float)
```

-> 3x speed-up on my 4 core laptop

PyGEOS implementation ?

- `pygeos.Geometry` Python C extension type holding pointer to GEOS Geometry object
- Extension type ensures garbage collection on the Python level, but the pointer is accessible from C without overhead
- The ufuncs are implemented in C using the numpy C API

Further work

- Speed-up GeoPandas by leveraging PyGEOS

Further work

- Speed-up GeoPandas by leveraging PyGEOS
- Integration with Shapely?

Further work

- Speed-up GeoPandas by leveraging PyGEOS
- Integration with Shapely?
- Spatial index (STRTree), spatial join
- Prepared geometries
- More coverage of GEOS functions
- ...

https://github.com/pygeos/pygeos/issues

Want to try out? Contribute?

Docs: https://pygeos.readthedocs.io

Install using conda:

```
$ conda install --channel conda-forge pygeos
```

Contribute: https://github.com/pygeos/pygeos/

Want to try out? Contribute?

Docs: https://pygeos.readthedocs.io

Install using conda:

```
$ conda install --channel conda-forge pygeos
```

Contribute: https://github.com/pygeos/pygeos/

Feedback and contributions very welcome!

Thanks for listening! Questions?

Thanks to Casper Van der Wel for the collaboration

Those slides:

- https://github.com/jorisvandenbossche/talks/
- jorisvandenbossche.github.io/talks/2019_FOSS4GBE_pygeos

http://pygeos.readthedocs.io