GEOSPATIAL ANALYTICS IN PYTHON

October 2018

Christopher Prince

TOPICS

What we'll cover:

- Spatial data formats
- Feature creation
- Map projections
- Plotting
- Spatial operations and manipuation
- Statistical methods

What we won't cover:

- Interfaces to GIS packages (like QGIS or ArcGIS)

GEOPANDAS INSTALLATION

Installation is a *lot* easier than it used to be. Both `pip` and conda should take care of the dependencies you will need.

It's still not a 1.0 release, though, so it's a good idea to setup a seperate environment for it.

If you use the PUI kernels on ARDF, the installation is already taken care of!

WHAT IS GEOPANDAS?

Geopandas is the GIS-extension of pandas. Two new classes, `GeoSeries` and `GeoDataFrame` extend the functionality of the pandas data structures. It also builds on other python spatial packages to perform manipulation and analysis:

- fiona
- shapely
- pyproj
- rtree
- pysal

GEODATAFRAME

A special column of the dataframe contains the shape information associated with each geometry. Helpfully, this is accessed via `gdf.geometry`.

Additionally, a *coordinate reference system* may be stored as a property of the GeoDataFrame: `gdf.crs`.

USE WITH PANDAS!

Methods are defined on GeoDataFrames (and GeoSeries) to perform manipulations and calculations. Importantly, because GeoDataFrames are extensions of DataFrames, all of the same methods available to DataFrames can be used with GeoDataFrames.

FIONA

Fiona reads and writes GIS files in various formats. 99.9% of the time you will be working with either:

- *Shapefile*: Actually a set of database files, though it's common to refer to just the *.shp file. Other attributes and metadata are stored in (necessary) supplemental files.
- *GeoJSON*: A JSON file with additional geometry attributes, and possibly a metadata header.

GEOJSON EXAMPLE

```json
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [125.6, 10.1]
  },
  "properties": {
    "name": "Dinagat Islands"
  }
}
```

READING FILES

Geopandas uses Fiona as a dependency, and calls it directly, so you usually don't need to import it seperately.

```
import geopandas as gpd
gpd.read_file('path/to/shapefile.shp')
```

SHAPELY

Shapely is a python library for geometric operations using the GEOS library.

Shapely can perform:

- geometry validation
- geometry creation (e.g. collections)
- geometry operations

GEOMETRY CREATION

Shapely supports the creation of primitive geometry features. These include:

- Point
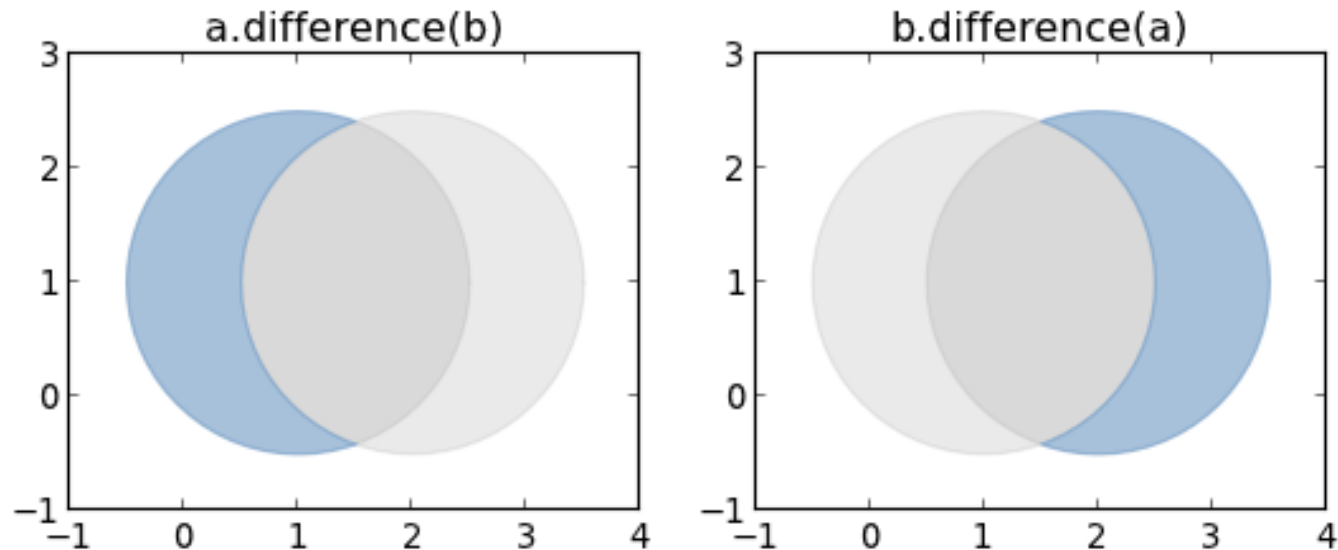- LineString
- LinearRing
- Polygon

Multipart collections are also supported: MultiPoint, MultiLineString and MultiPolygon

GEOMETRY CREATION

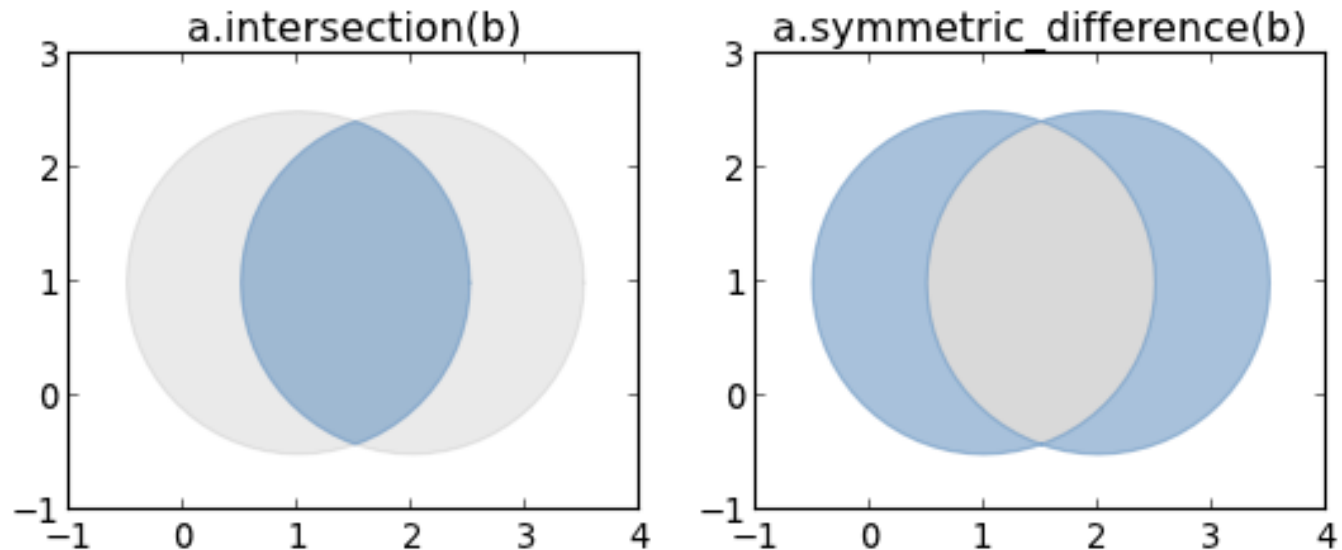A common use case is to create geometric points from a set of Latitudes and Longitudes:
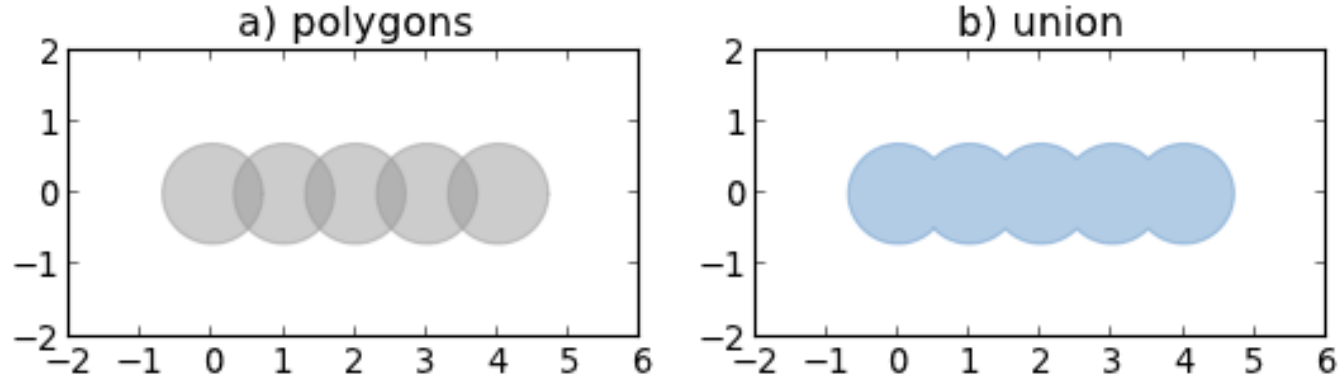
```python
from shapely.geometry import Point
from geopandas import GeoDataFrame

pointData = pd.read_csv('somefilewithlatlongs.csv')
pointData.lonlat = list(zip(pointData.lon, pointData.lat))
pointData['geometry'] = pointData.lonlat
                                    .applymap(lambda x: Point(x))
pointData = GeoDataFrame(pointData)     # Gives the geometry
                                        # column special powers!
```

SHAPELY GEOMETRIC OPERATIONS

SHAPELY GEOMETRIC OPERATIONS

source: http://kjordahl.github.io/SciPy-Tutorial-2015/

SHAPELY GEOMETRIC OPERATIONS

source: http://kjordahl.github.io/SciPy-Tutorial-2015/
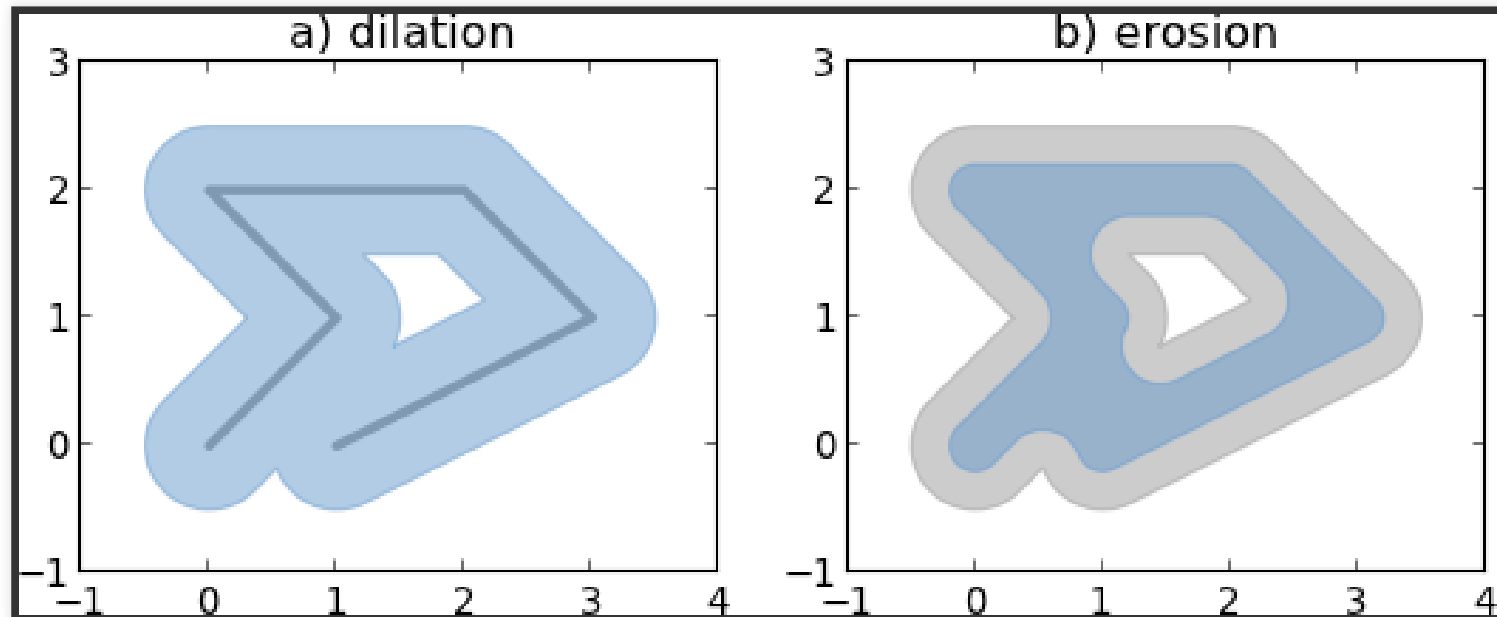
SHAPELY GEOMETRIC OPERATIONS

source: http://kjordahl.github.io/SciPy-Tutorial-2015/
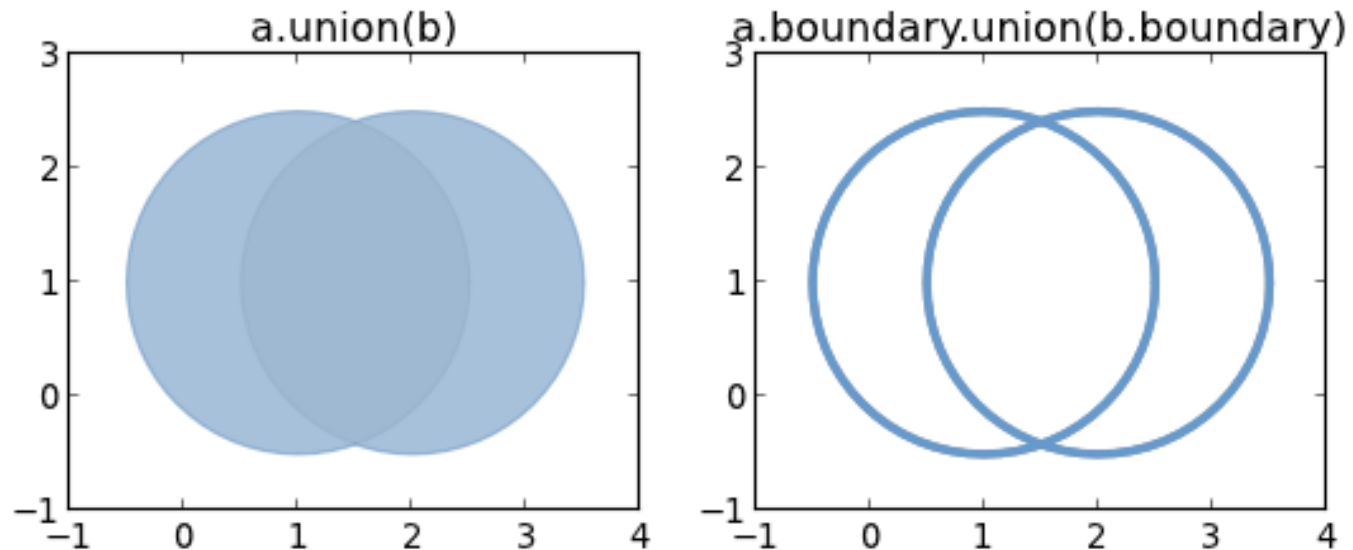
SHAPELY GEOMETRIC OPERATIONS

```
from shapely.geometry import LineString
line = LineString([(0, 0), (1, 1), (0, 2), (2, 2), (3, 1), (1, 0)
dilated = line.buffer(0.5)
eroded = dilated.buffer(-0.3)
```


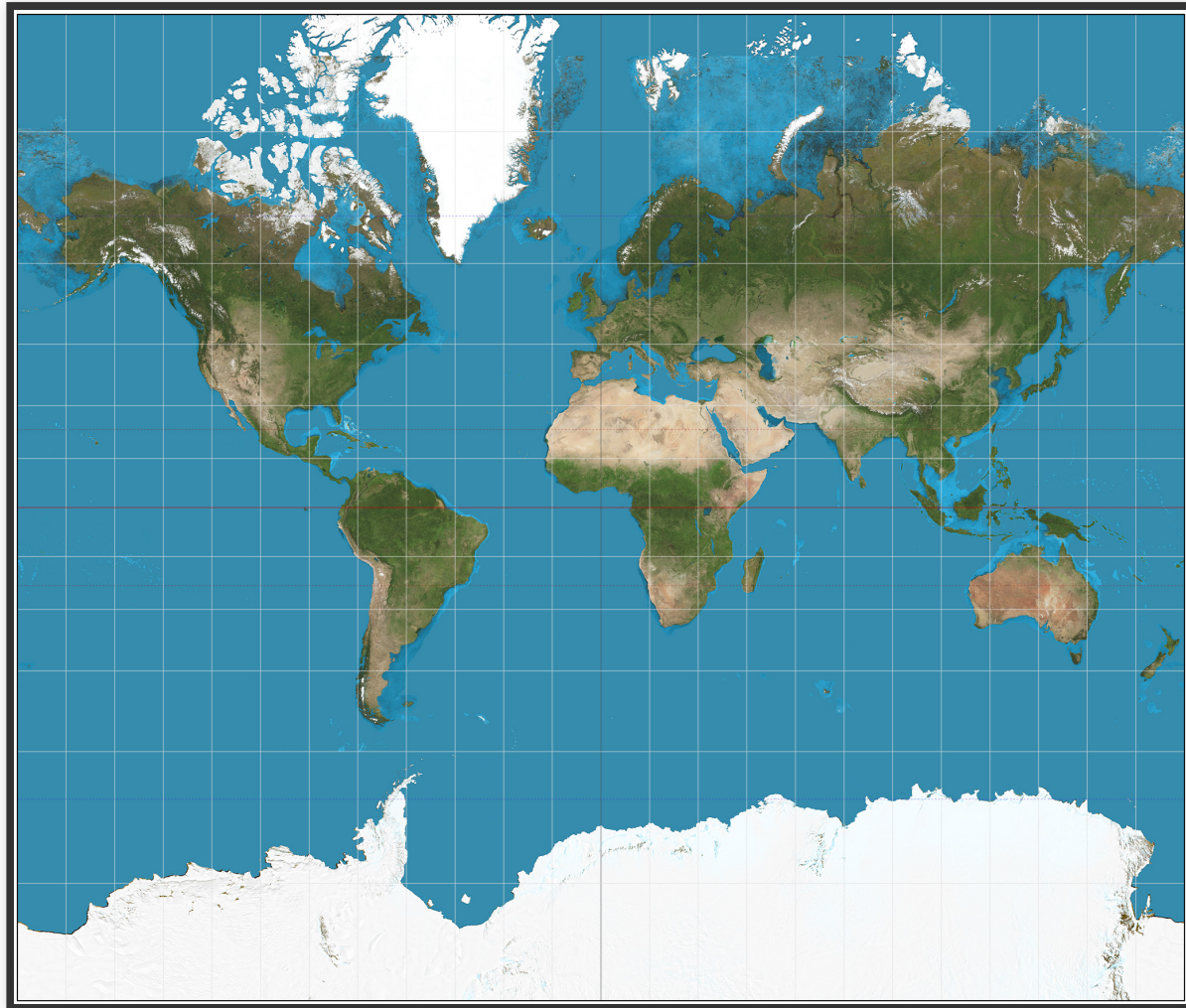source: http://kjordahl.github.io/SciPy-Tutorial-2015/

BINARY PREDICATES

```
object.almost_equals(other[, decimal=6])
object.contains(other)
object.crosses(other)
object.disjoint(other)
object.equals(other)
object.intersects(other)
object.touches(other)
object.within(other)
```
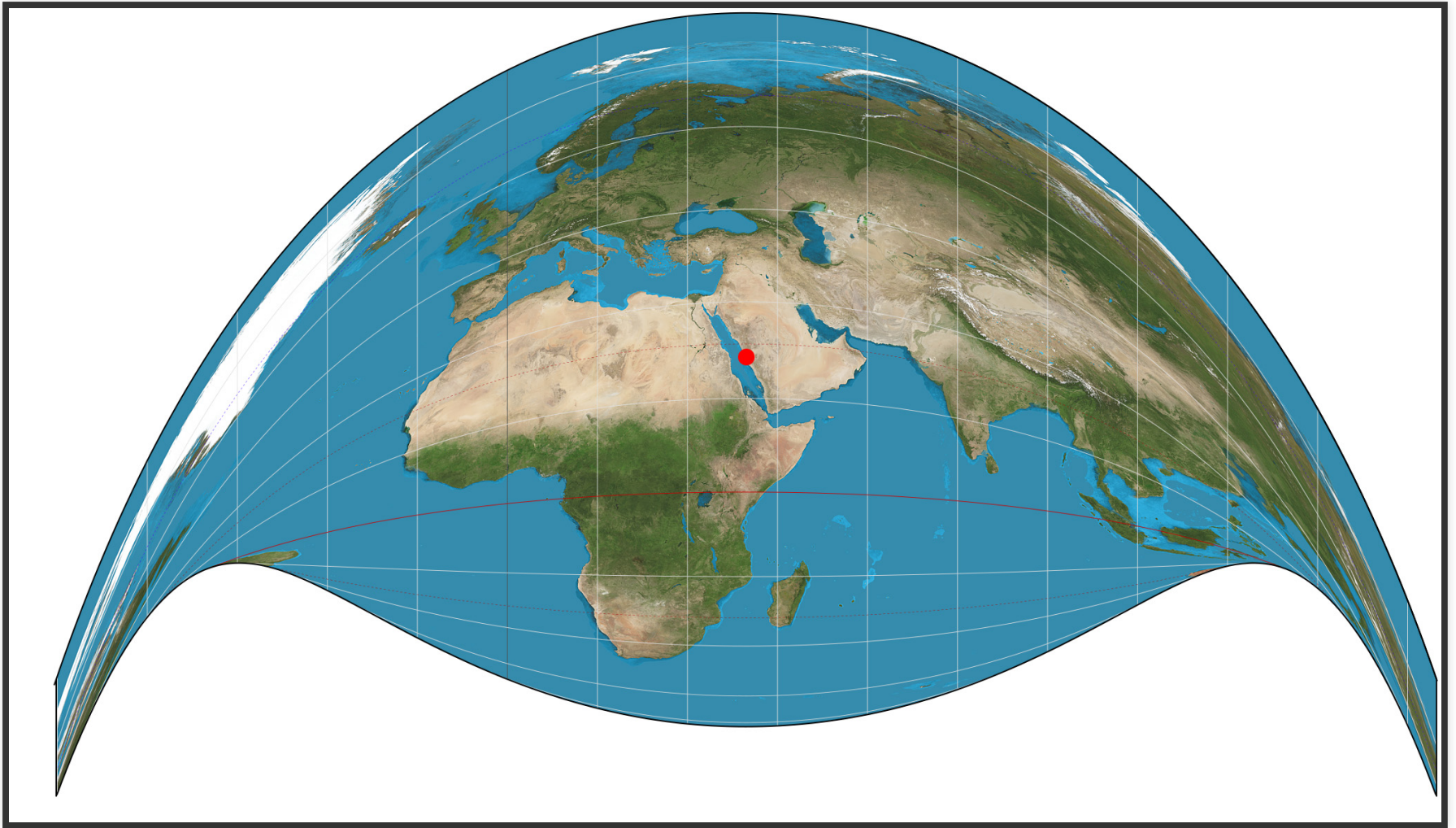
details

LET'S TRY IT!

GEOGRAPHIC PROJECTION
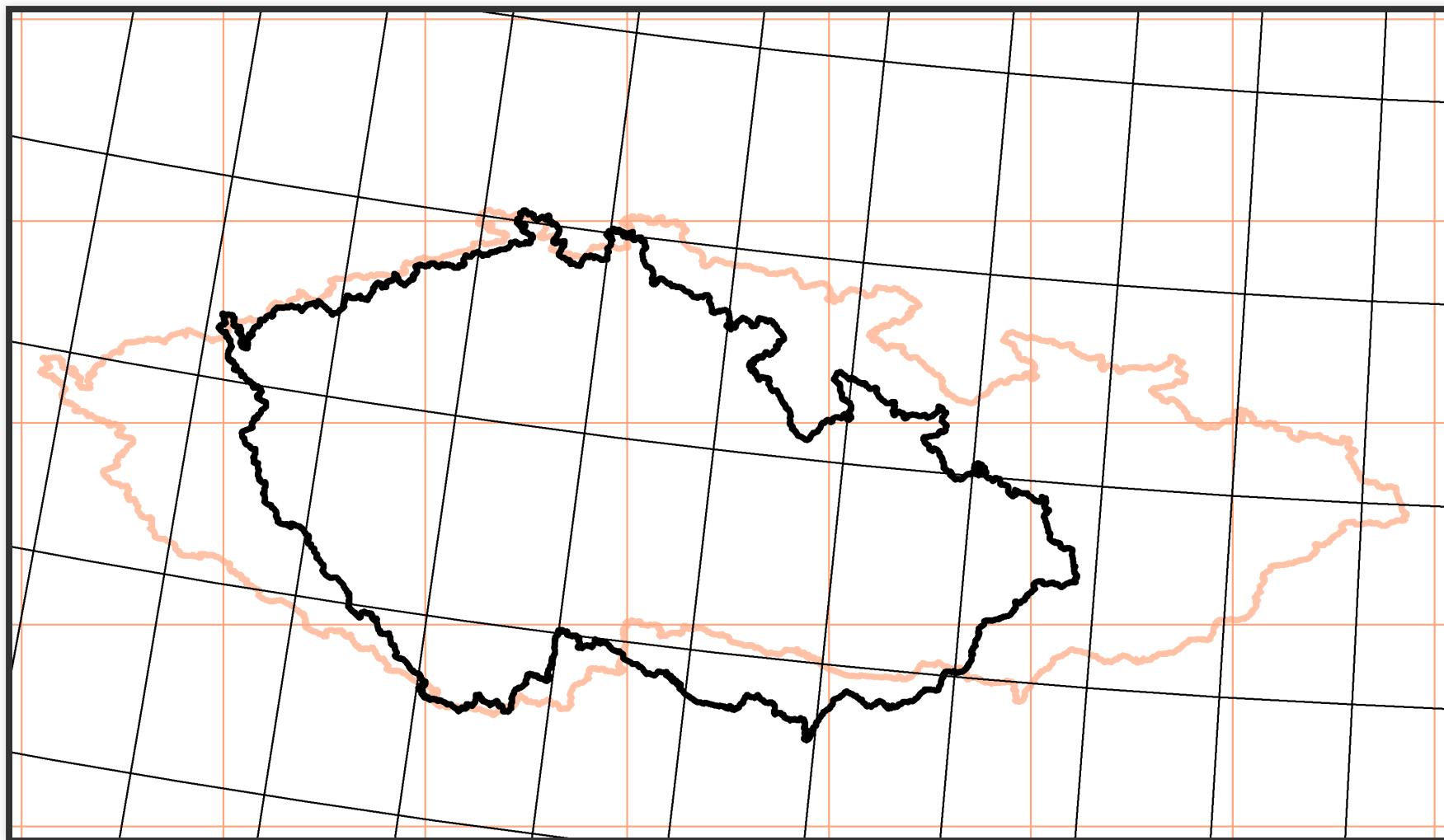
https://youtu.be/vVX-PrBRtTY?t=46s

Mercator projection (1569)

Craig retroazimuthal projection (1909)

Křovák's projection (black) versus WGS84 projection (orange)

PYPROJ

Pyproj provides an interface to the PROJ.4 library which performs the transformations between coordinate reference systems (CRS). This can be done explicitly by specifying datums, geodetic ellipse, origin, and other parameters in a "PROJ4 string".

Many common projections are indexed with a numerical code from the European Petroleum Survey Group, and using these is generally much simpler with fiona.

PROJECTION EXAMPLE

PROJ.4 string (North American Equidistant Conic):

```
naec = '+proj=eqdc +lat_0=40 +lon_0=-96
+lat_1=20 +lat_2=60 +x_0=0 +y_0=0
+datum=NAD83 +units=m +no_defs'
```

Fiona EPSG 4326 (WGS84 lat/long):

```
from fiona.crs import from_epsg
states.crs = from_epsg(4326)
```

Transforming from one to the other:

```
states.to_crs(naec)
```

LET'S MAKE A MAP!

USEFUL LINKS

Looking up projections: http://spatialreference.org/
Paper on R-trees for spatial joins:
http://www.cs.ucr.edu/~ravi/CS236Papers/r-tree-join.pdf
Notebooks for JavaScript (and d3!):
https://observablehq.com

// More info about config & dependencies: // - https://github.com/hakimel/reveal.js#configuration // - https://github.com/hakimel/reveal.js#dependencies