Introduction to the dataset

WORKING WITH GEOSPATIAL DATA IN PYTHON

Joris Van den Bossche
Open source software developer and teacher, GeoPandas maintainer

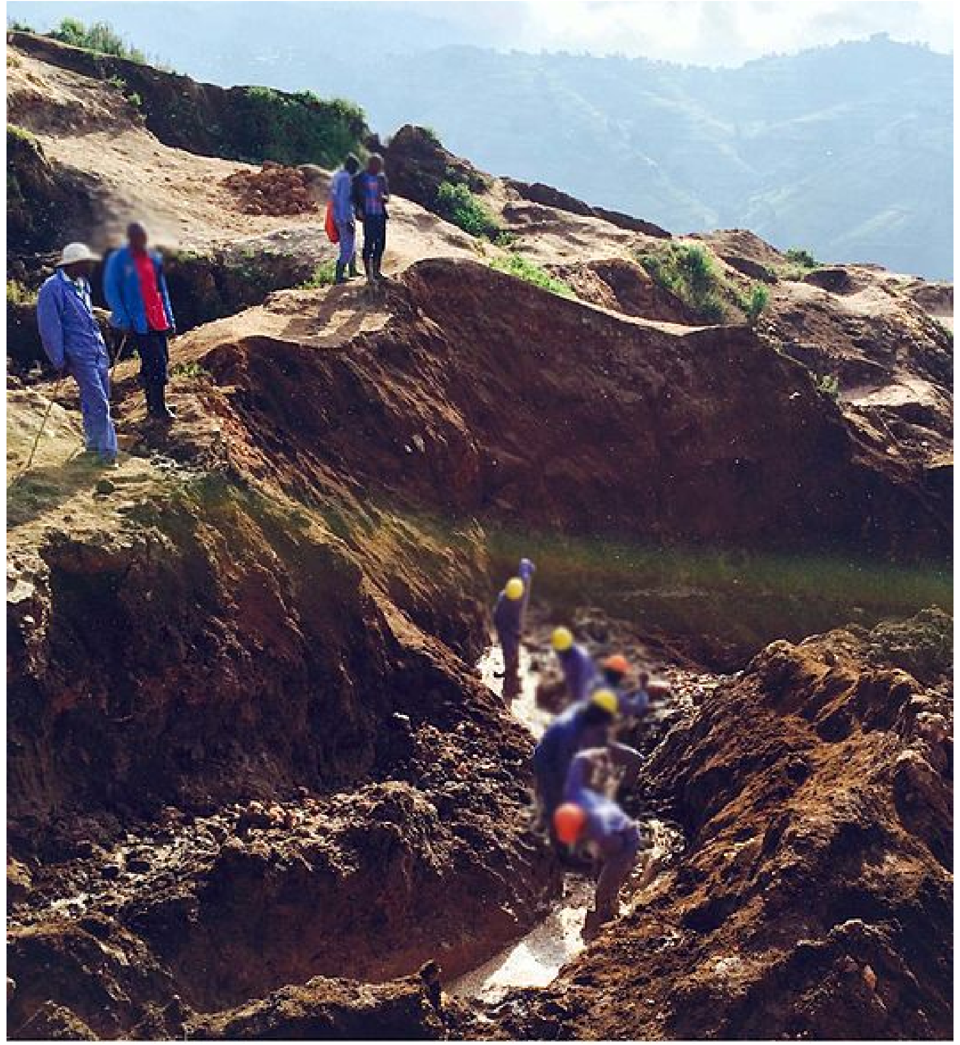
Artisanal mining site data from IPIS

IPIS: International Peace Information Service

Image: Connormah, CC BY-SA 3.0, from Wikimedia Commons

Artisanal mining site data from IPIS

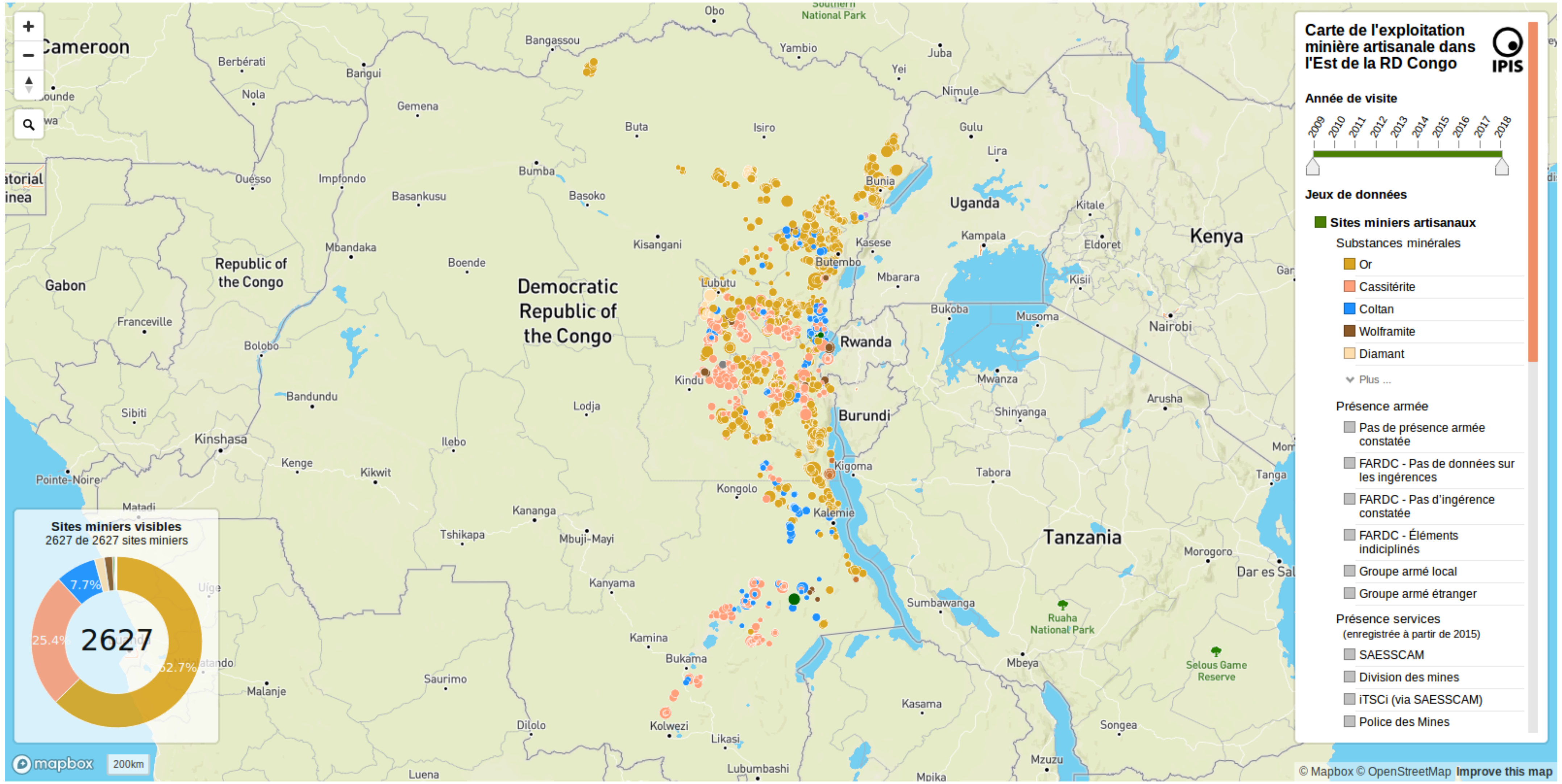
IPIS: International Peace Information Service



Source: GAO. | GAO-15-561

Image: G.A.O, public domain, from Wikimedia Commons

Artisanal mining site data from IPIS



More analysis (re. social & security)

Geospatial file formats

Reading files: `geopandas.read_file("path/to/file.geojson")`

Supported formats:

- ESRI Shapefile
  - One "file" consists of multiple files! (`.shp` , `.dbf` , `.shx` , `.prj` , ...)
- GeoJSON
- GeoPackage (`.gpkg`)
- ...

& PostGIS databases!

Writing to geospatial file formats

Writing a GeoDataFrame to a file with the `to_file()` method:

```
# Writing a Shapefile file
geodataframe.to_file("mydata.shp", driver='ESRI Shapefile')

# Writing a GeoJSON file
geodataframe.to_file("mydata.geojson", driver='GeoJSON')

# Writing a GeoPackage file
geodataframe.to_file("mydata.gpkg", driver='GPKG')
```

Let's practice!

WORKING WITH GEOSPATIAL DATA IN PYTHON

Additional spatial operations

WORKING WITH GEOSPATIAL DATA IN PYTHON

Joris Van den Bossche
Open source software developer and teacher, GeoPandas maintainer

Overview of spatial operations

Spatial **relationships**:

- `intersects`
- `within`
- `contains`
- ...

Join attributes based on spatial relation:

- `geopandas.sjoin`

Geometry operations:

- `intersection`
- `union`
- `difference`
- ...

Combine datasets based on geometry operation:

- `geopandas.overlay`

Unary union

Convert a series of geometries to a single union geometry

```
africa
```

Unary union

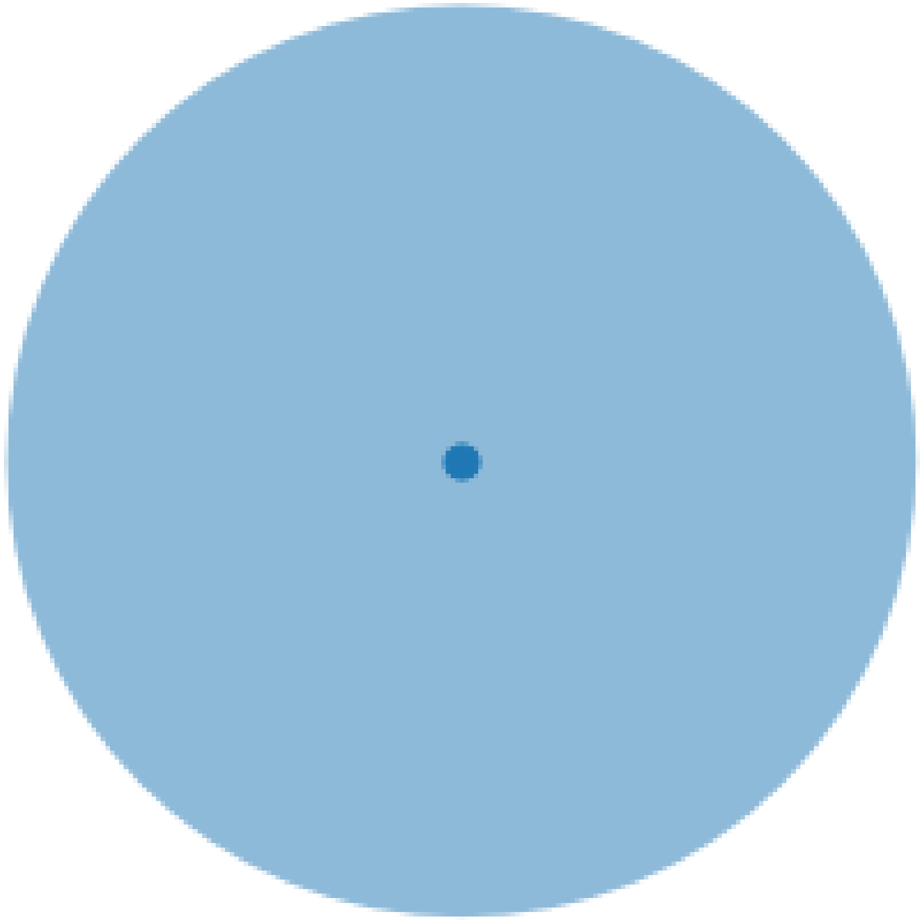
Convert a series of geometries to a single union geometry:

`africa`

`africa.unary_union`

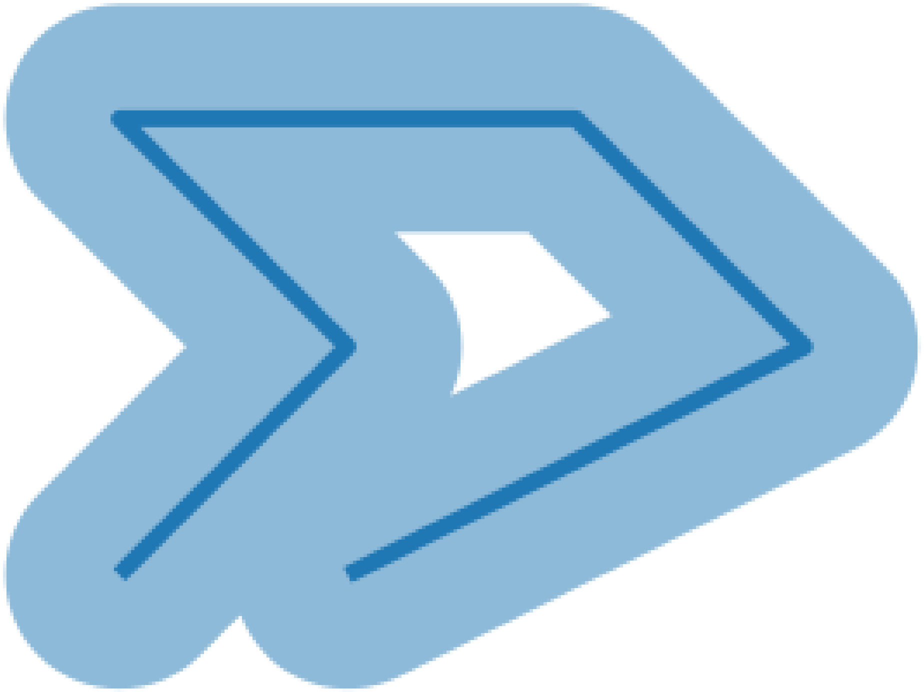
Buffer operation

Buffer operation

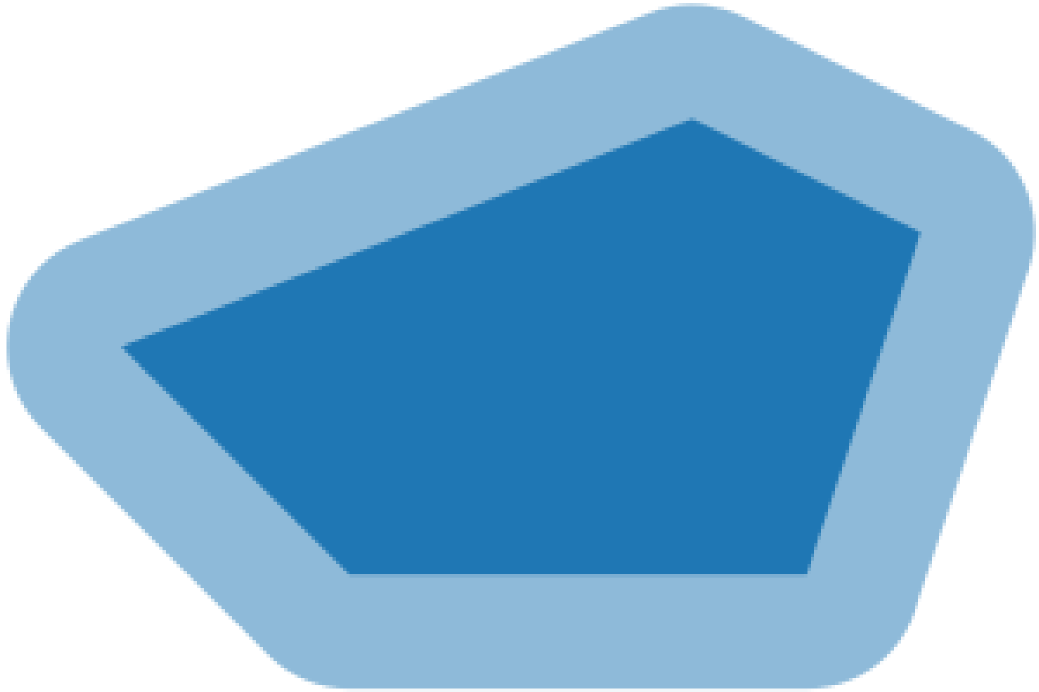


`point.buffer(distance)`

Buffer operation



`line.buffer(distance)`



`polygon.buffer(distance)`

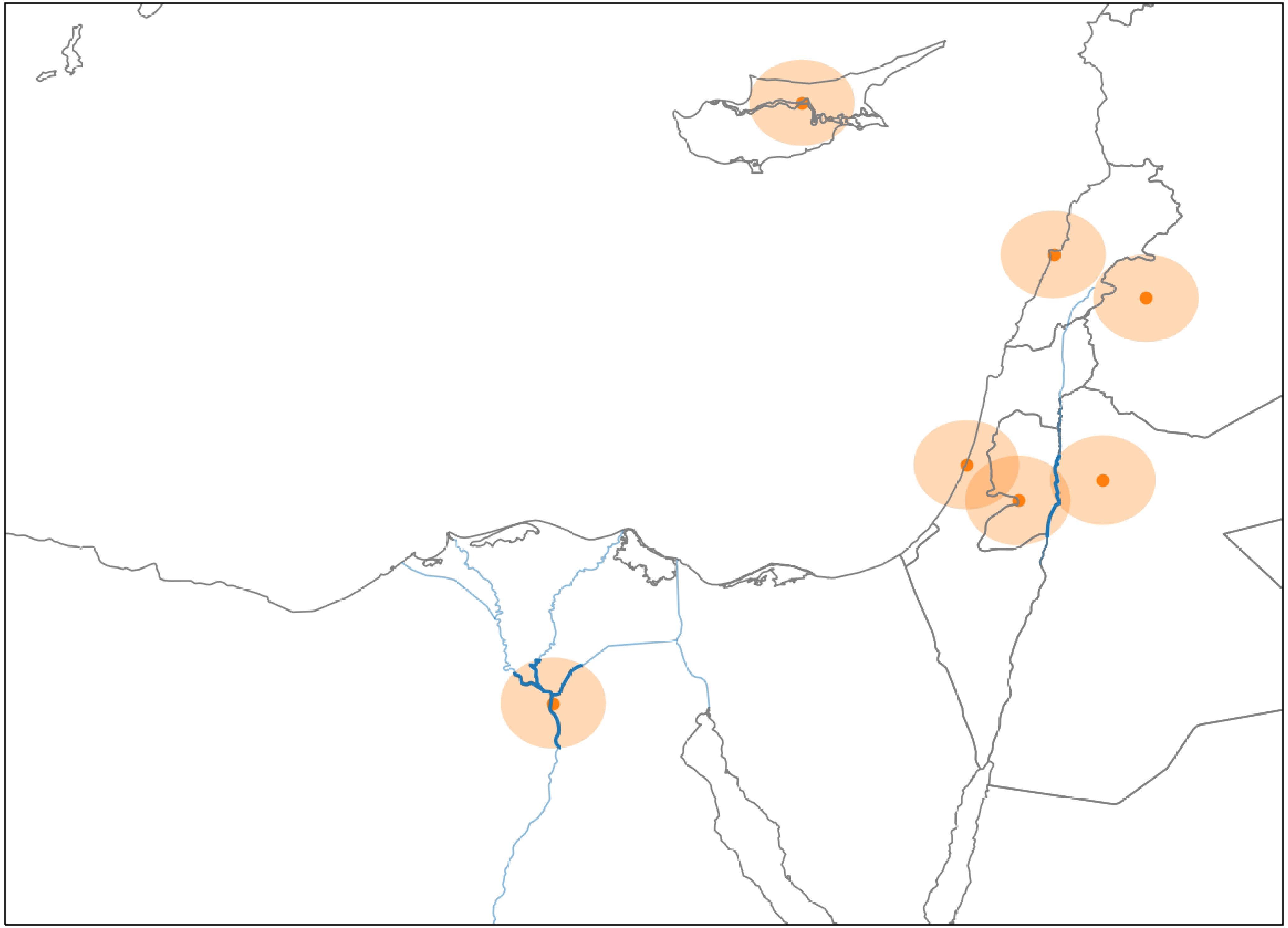
Let's practice!

WORKING WITH GEOSPATIAL DATA IN PYTHON

Applying custom spatial operations

WORKING WITH GEOSPATIAL DATA IN PYTHON

Joris Van den Bossche
Open source software developer and teacher, GeoPandas maintainer



Total river length within 50 km of each city?

For a single point (`cairo`):

```
area = cairo.buffer(50000)
rivers_within_area = rivers.intersection(area)
print(rivers_within_area.length.sum() / 1000)
```

```
186.397219642
```

The apply() method

`Series.apply()` : call a function on each of the values of the Series

> `Series.apply(function, **kwargs)`

- `function` : the function being called on each value; the value is passed as the first argument
- `**kwargs` : additional arguments passed to the function

For a `GeoSeries` , the function is called as `function(geom, **kwargs)` for each `geom` in the `GeoSeries`

Applying a custom spatial operation

The function to apply:

```
def river_length(geom, rivers):
    area = geom.buffer(50000)
    rivers_within_area = rivers.intersection(area)
    return rivers_within_area.length.sum() / 1000
```

Call function on the single geometry:

```
river_length(cairo, rivers=rivers)
```

```
186.3972196423455
```

Applying a custom spatial operation

Applying on all cities:

```
cities.geometry.apply(river_length, rivers=rivers)
```

```
0        0.000000
1        0.000000
2      106.072198
             ...
```

Applying a custom spatial operation

Applying on all cities and assigning result to new column:

```
cities['river_length'] = cities.geometry.apply(river_length, rivers=rivers)
cities.head()
```

```
              name                       geometry  river_length
0     Vatican City  POINT (1386304.6 5146502.5)      0.000000
1       San Marino  POINT (1385011.5 5455558.1)      0.000000
2            Vaduz  POINT (1059390.7 5963928.5)    106.072198
..             ...                          ...           ...
```

Let's practice!

Working with raster data

WORKING WITH GEOSPATIAL DATA IN PYTHON

Joris Van den Bossche
Open source software developer and teacher, GeoPandas maintainer

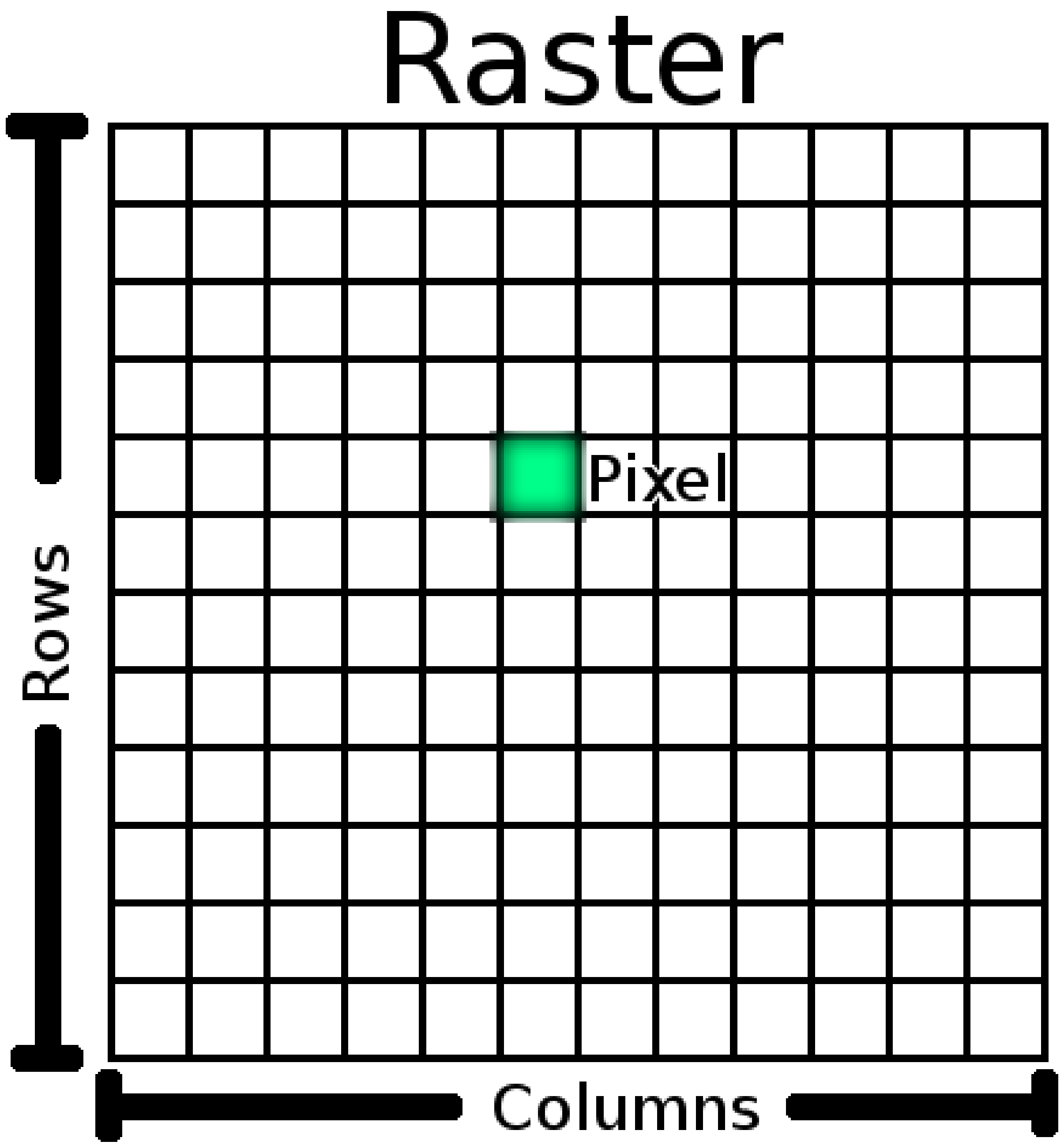
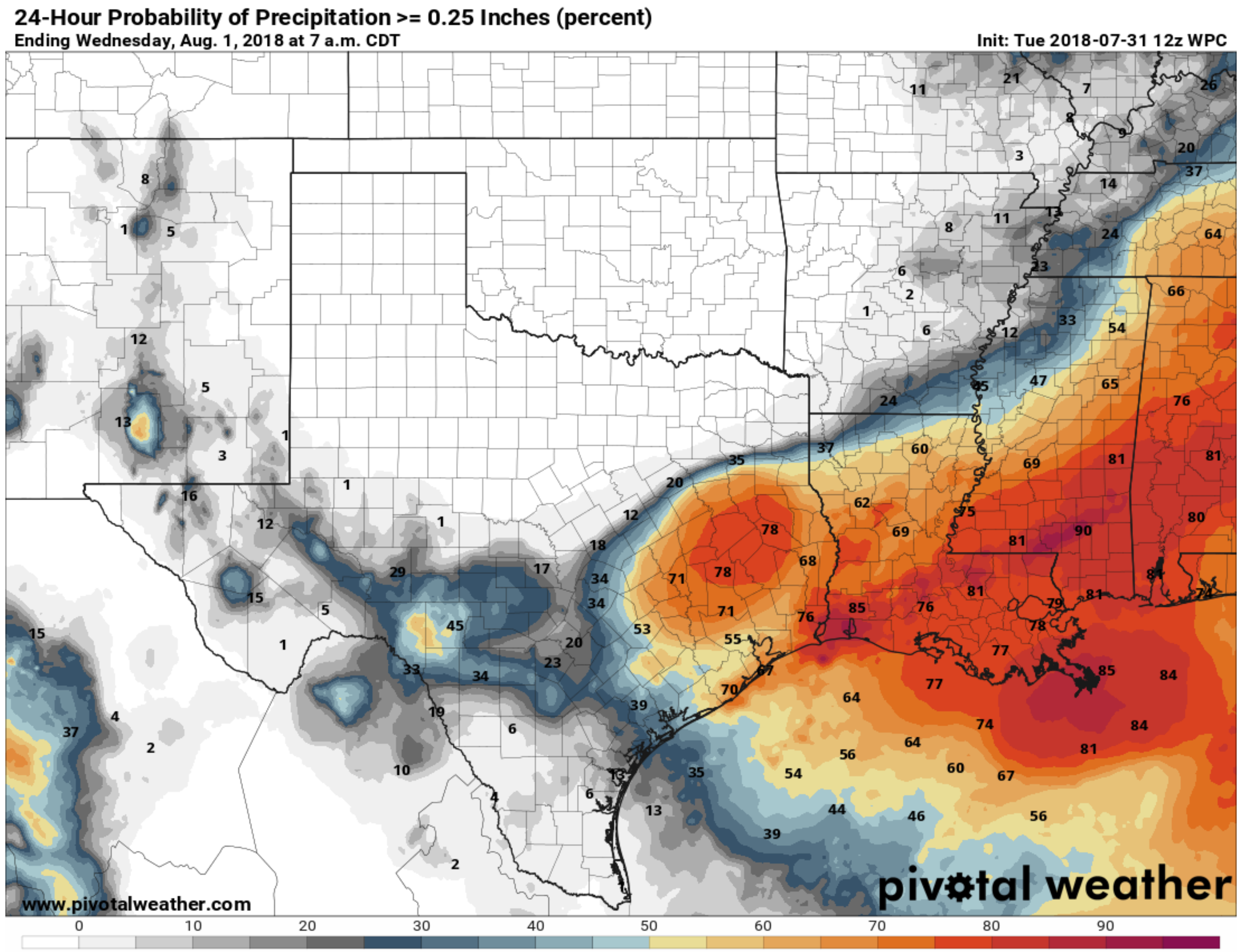
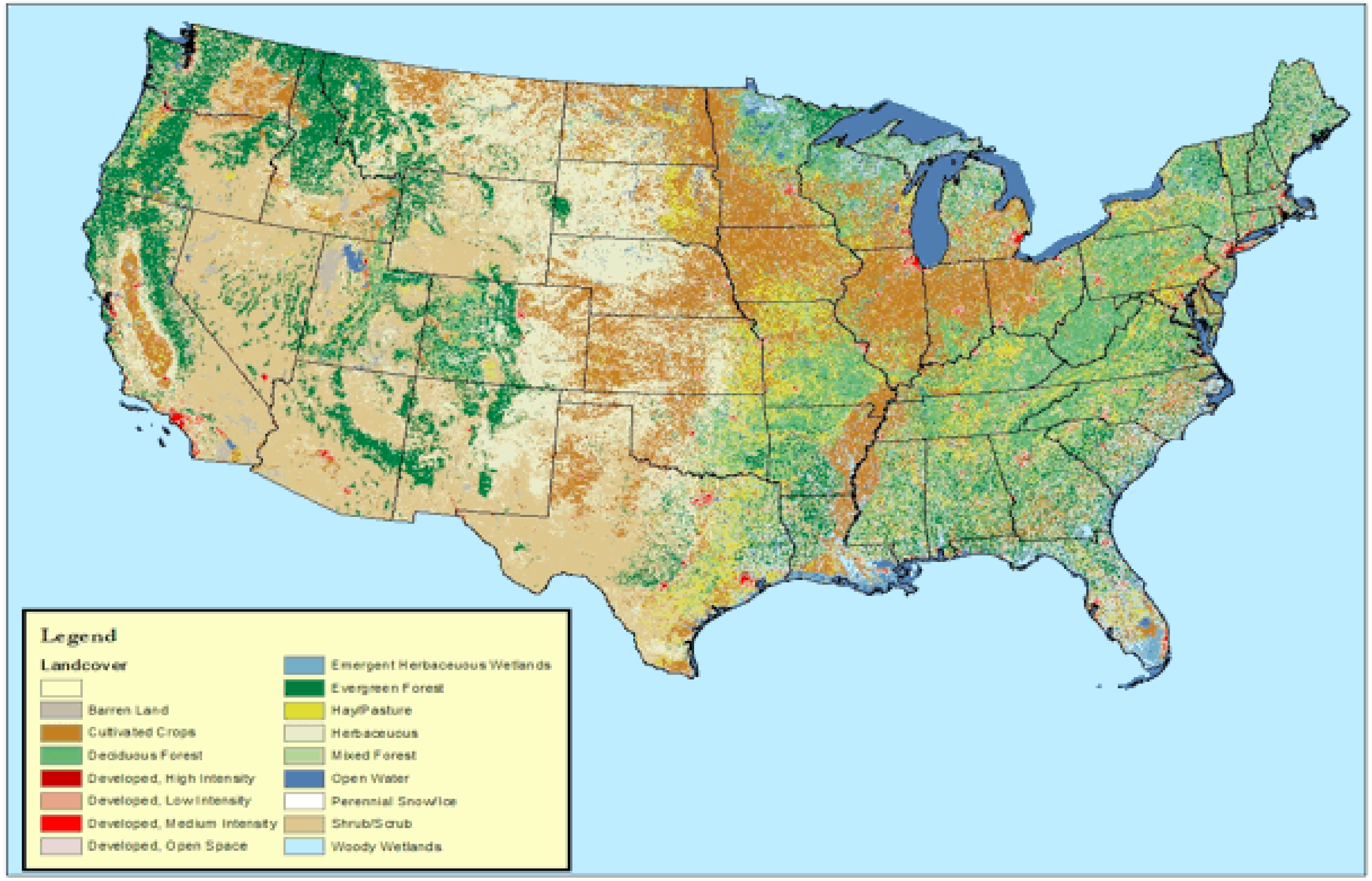


Image source: QGIS documentation

Raster data

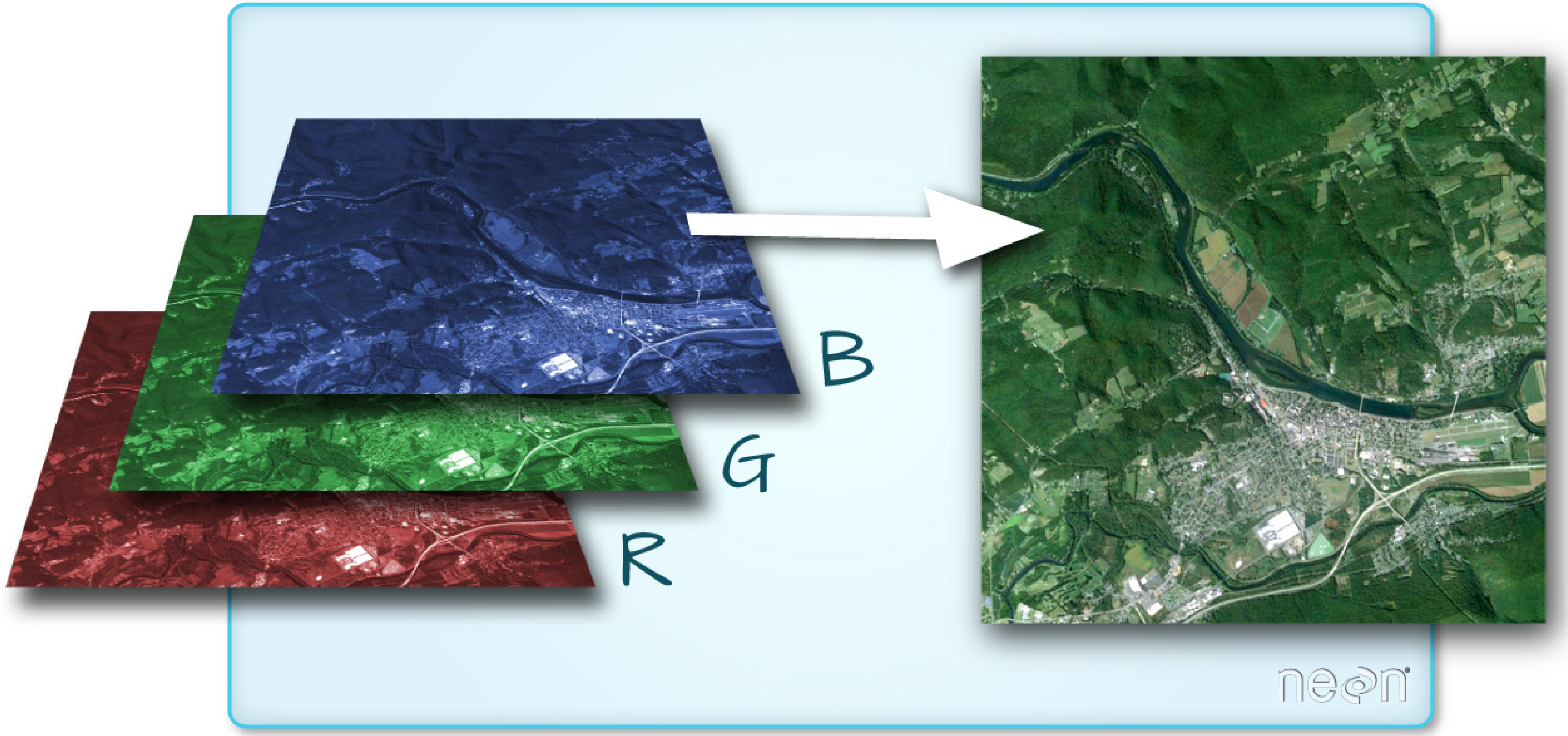


24-Hour Probability of Precipitation >= 0.25 Inches (percent)
Ending Wednesday, Aug. 1, 2018 at 7 a.m. CDT
Init: Tue 2018-07-31 12z WPC
www.pivotalweather.com
pivotal weather
0 10 20 30 40 50 60 70 80 90



Legend
Landcover
Barren Land
Cultivated Crops
Deciduous Forest
Developed, High Intensity
Developed, Low Intensity
Developed, Medium Intensity
Developed, Open Space
Emergent Herbaceuous Wetlands
Evergreen Forest
Hay/Pasture
Herbaceuous
Mixed Forest
Open Water
Perennial Snow/Ice
Shrub/Scrub
Woody Wetlands

Raster data with multiple bands



The rasterio package

```python
import rasterio
```

- "Pythonic" bindings to GDAL
- **Reading** and **writing** raster files
- **Processing** tools (masking, reprojection, resampling, ..)

https://rasterio.readthedocs.io/en/latest/

Opening a raster file

```
import rasterio

src = rasterio.open("DEM_world.tif")
```

Metadata:

```
src.count
```

```
1
```

```
src.width, src.height
```

```
(4320, 2160)
```

Raster data = numpy array

```
array = src.read()
```

Standard `numpy` array:

```
array
```

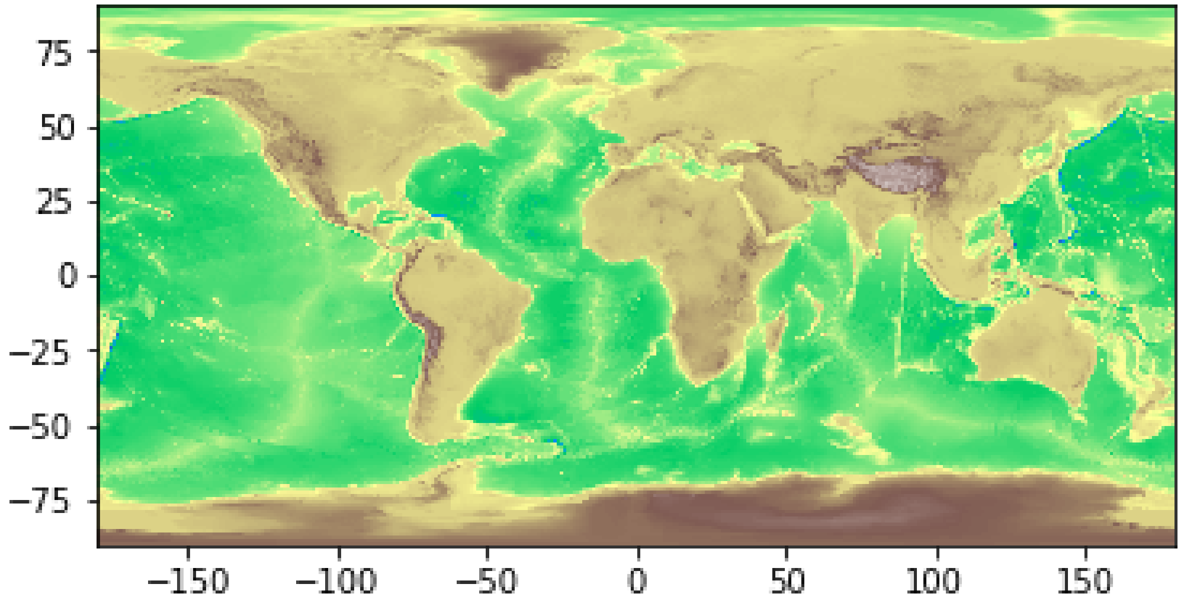
```
array([[[-4290, -4290, -4290, ..., -4290, -4290, -4290],
        [-4278, -4278, -4278, ..., -4278, -4278, -4278],
        [-4269, -4269, -4269, ..., -4269, -4269, -4269],
        ...,
        [ 2804,  2804,  2804, ...,  2804,  2804,  2804],
        [ 2804,  2804,  2804, ...,  2804,  2804,  2804],
        [ 2804,  2804,  2804, ...,  2804,  2804,  2804]]], dtype=int16)
```

Plotting a raster dataset

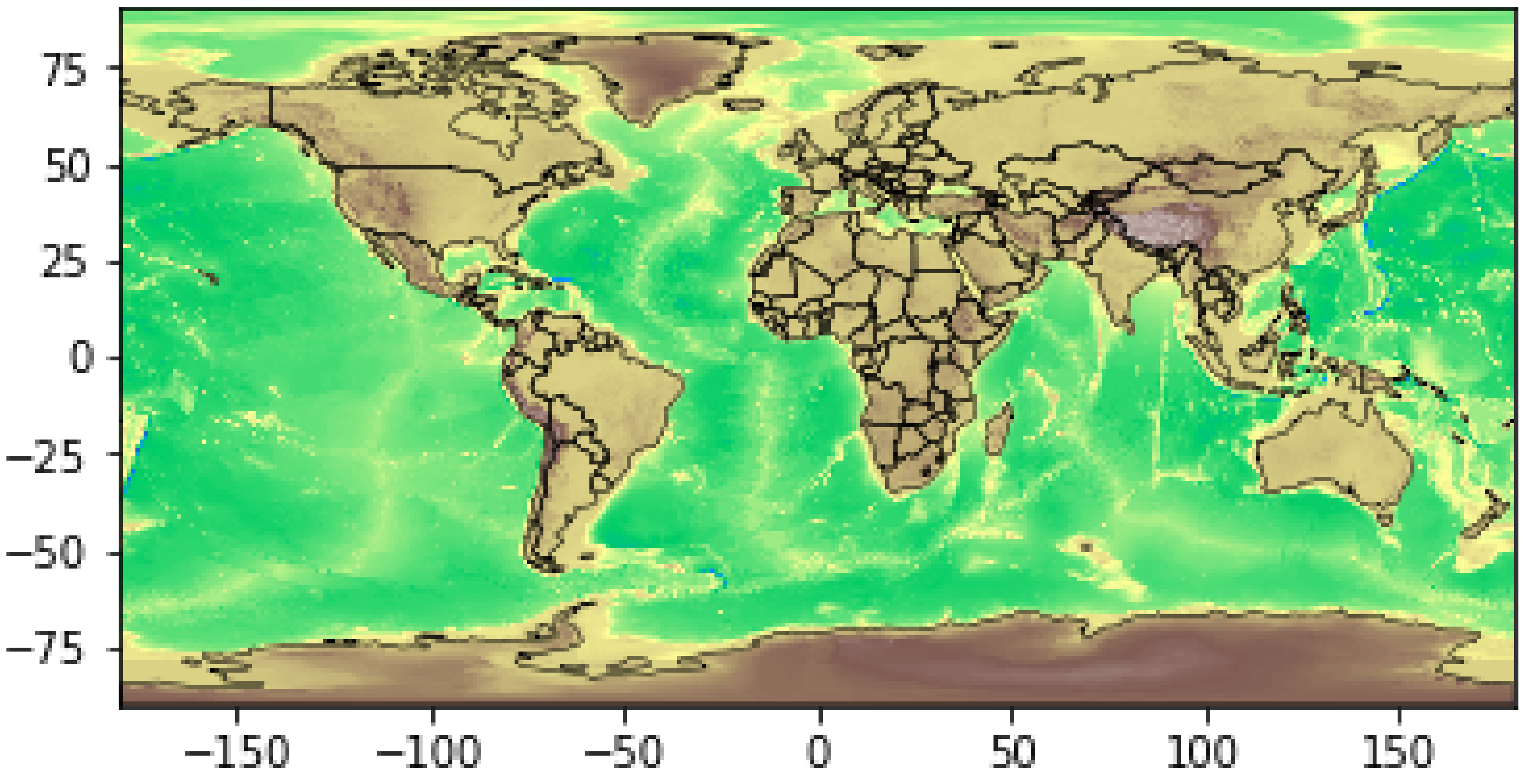
Using the `rasterio.plot.show()` method:

```
import rasterio.plot


rasterio.plot.show(src, cmap='terrain')
```



Extracting information based on vector data



`rasterstats` : Summary statistics of geospatial raster datasets based on vector geometries (https://github.com/perrygeo/python-rasterstats)

Extract raster values with rasterstats

- For point vectors:

```
rasterstats.point_query(geometries, "path/to/raster",
                        interpolation='nearest'|'bilinear')
```

- For polygon vectors:

```
rasterstats.zonal_stats(geometries, "path/to/raster",
                        stats=['min', 'mean', 'max'])
```

Extract raster values with rasterstats

```
result = rasterstats.zonal_stats(countries.geometry, "DEM_gworld.tif",
                                 stats=['mean'])

countries['mean_elevation'] = pd.DataFrame(result)

countries.sort_values('mean_elevation', ascending=False).head()
```

```
            name   continent                          geometry  mean_elevation
157   Tajikistan        Asia  POLYGON ((74.98 37.41, ...          3103.231105
85    Kyrgyzstan        Asia  POLYGON ((80.25 42.34, ...          2867.717142
24        Bhutan        Asia  POLYGON ((91.69 27.77, ...          2573.559846
119        Nepal        Asia  POLYGON ((81.11 30.18, ...          2408.907816
6     Antarctica  Antarctica  (POLYGON ((-59.57 -80.04...         2374.075028
..           ...         ...                          ...             ...
```

Let's practice!

WORKING WITH GEOSPATIAL DATA IN PYTHON

The end

WORKING WITH GEOSPATIAL DATA IN PYTHON

Instructors
Joris Van den Bossche & Dani Arribas-Bel

Taking the next steps ...

More on GeoPandas:

- GeoPandas docs and example gallery: **https://geopandas.readthedocs.io/**
- Other online sources, e.g.: **https://automating-gis-processes.github.io/2018/**

Looking for spatial statistics? Check **PySAL**

Working with multi-dimensional gridded data? Check **xarray**

Want to create interactive web maps? Check **folium**, **ipyleaflet** or **geoviews**

Make matplotlib plots with projection support? Check **cartopy**

Good luck!

WORKING WITH GEOSPATIAL DATA IN PYTHON