we-do-IT
Spatial Information Technology Consultants

Cartography with Python

Henry Walshaw
hwalshaw@we-do-it.com
henry@pythoncharmers.com
@om_henners everywhere else

we-do-IT
Spatial Information Technology Consultants

"Everything is related to everything else, but near things are more related than distant things."

Waldo Tolber (1970)

Get the code:
https://github.com/om-henners/pyconau2014_talk

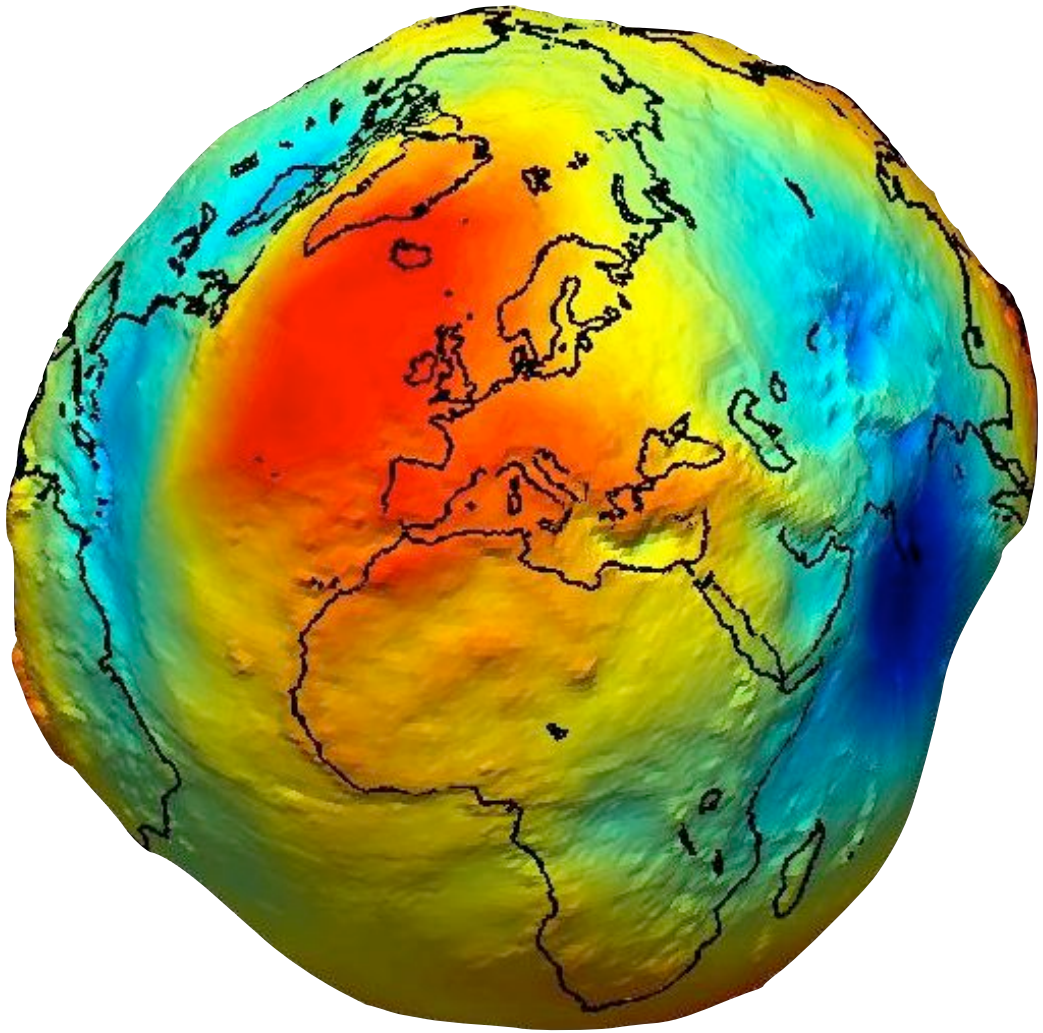

GIS
Police
Search/Rescue

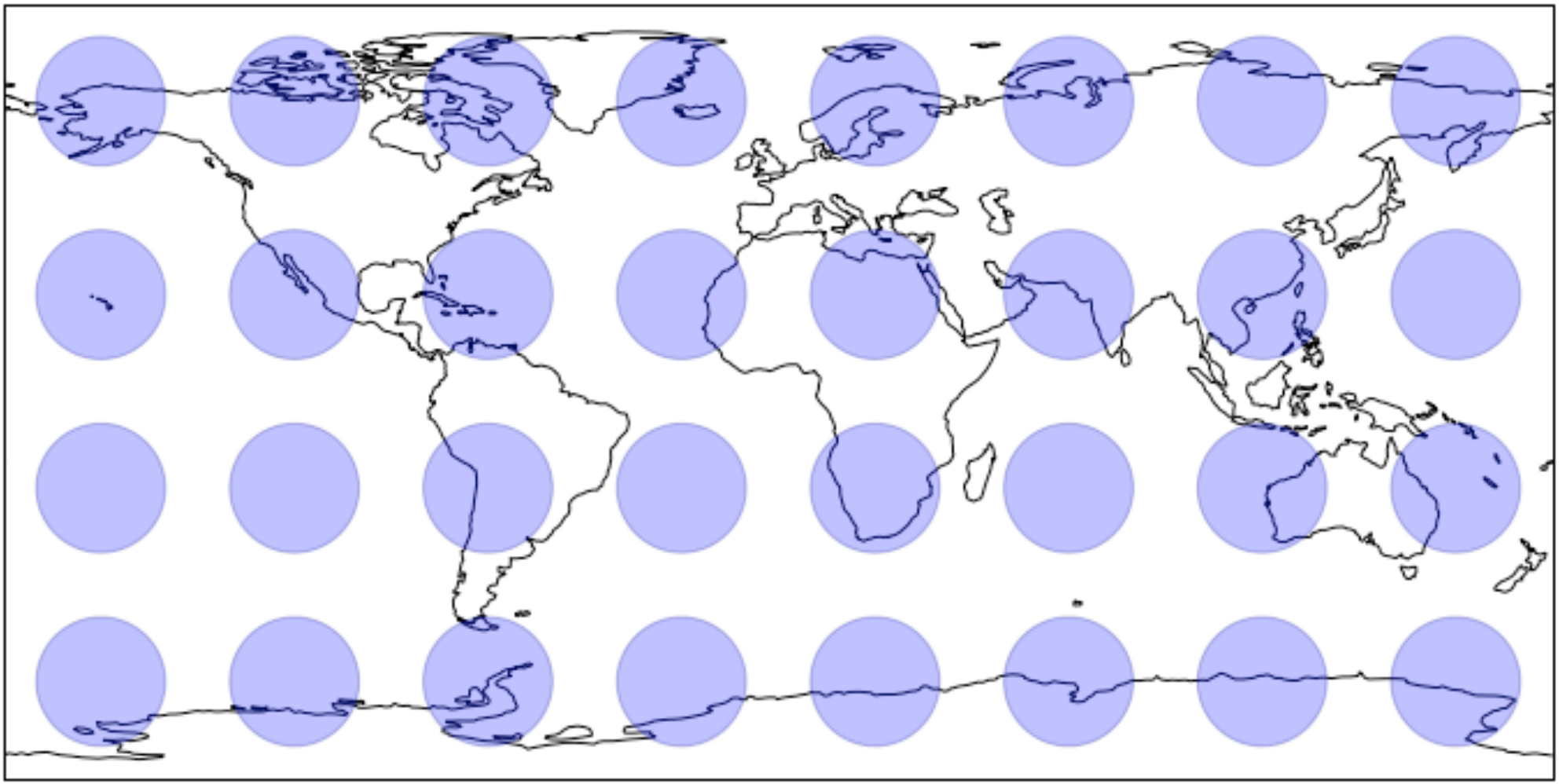
- Python is:
  - Primary scripting language for ESRI ArcGIS
  - Primary scripting language for QGIS
  - Used just about everywhere else

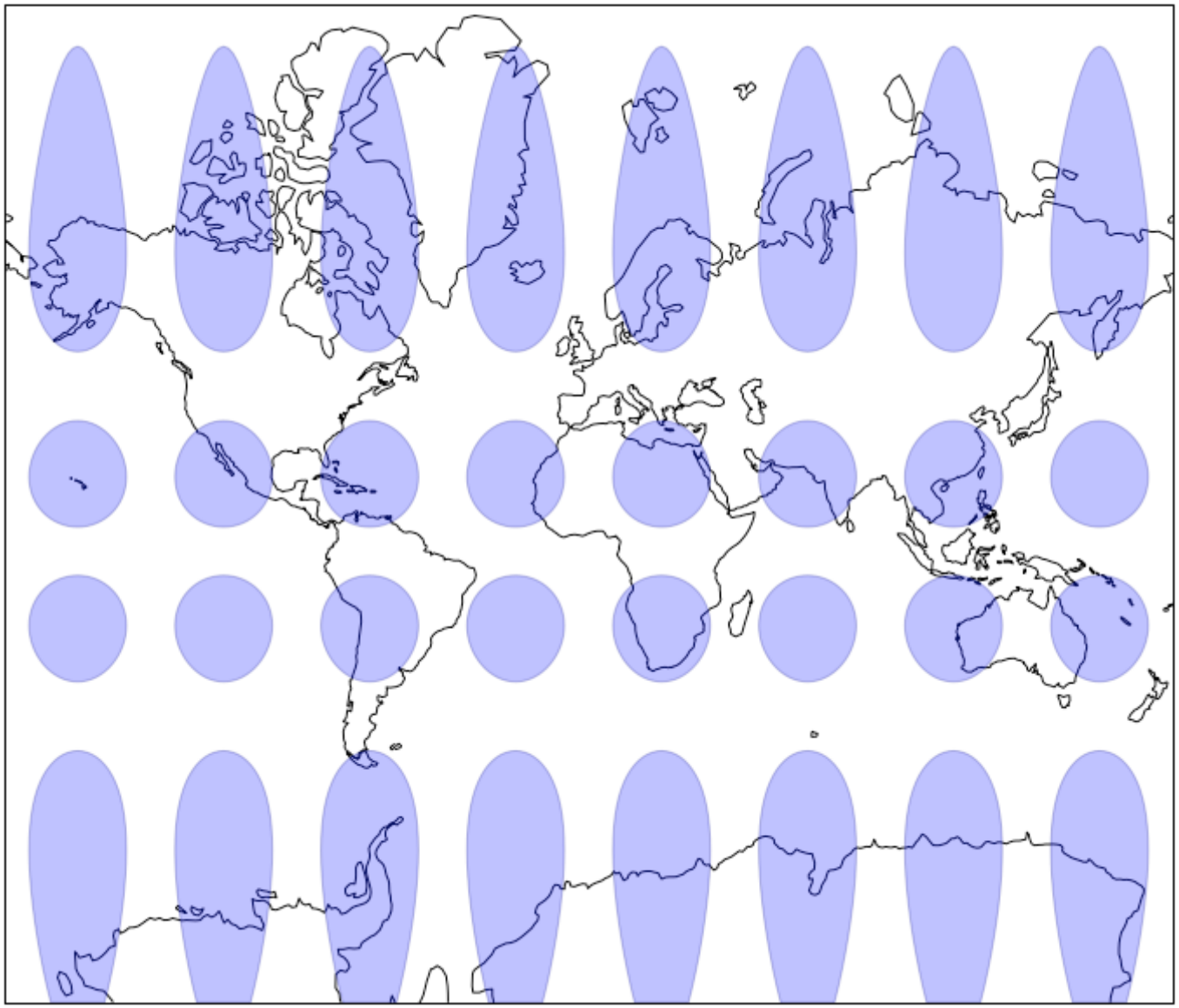
- Two basic spatial data types
  - Vector geometry (points, lines, polygons, etc.)
  - Rasters (continuous data)
  - There are others as well (think networks and temporal data)
- Features are some data with a geometry attached

- Geometries have a projection
- Projections give us a reference in space to compare geometries and to work with them in a two dimensional space (a map)
- Based on your projection you can preserve direction, distance, area, shape, bearing or scale
  - Never all of them

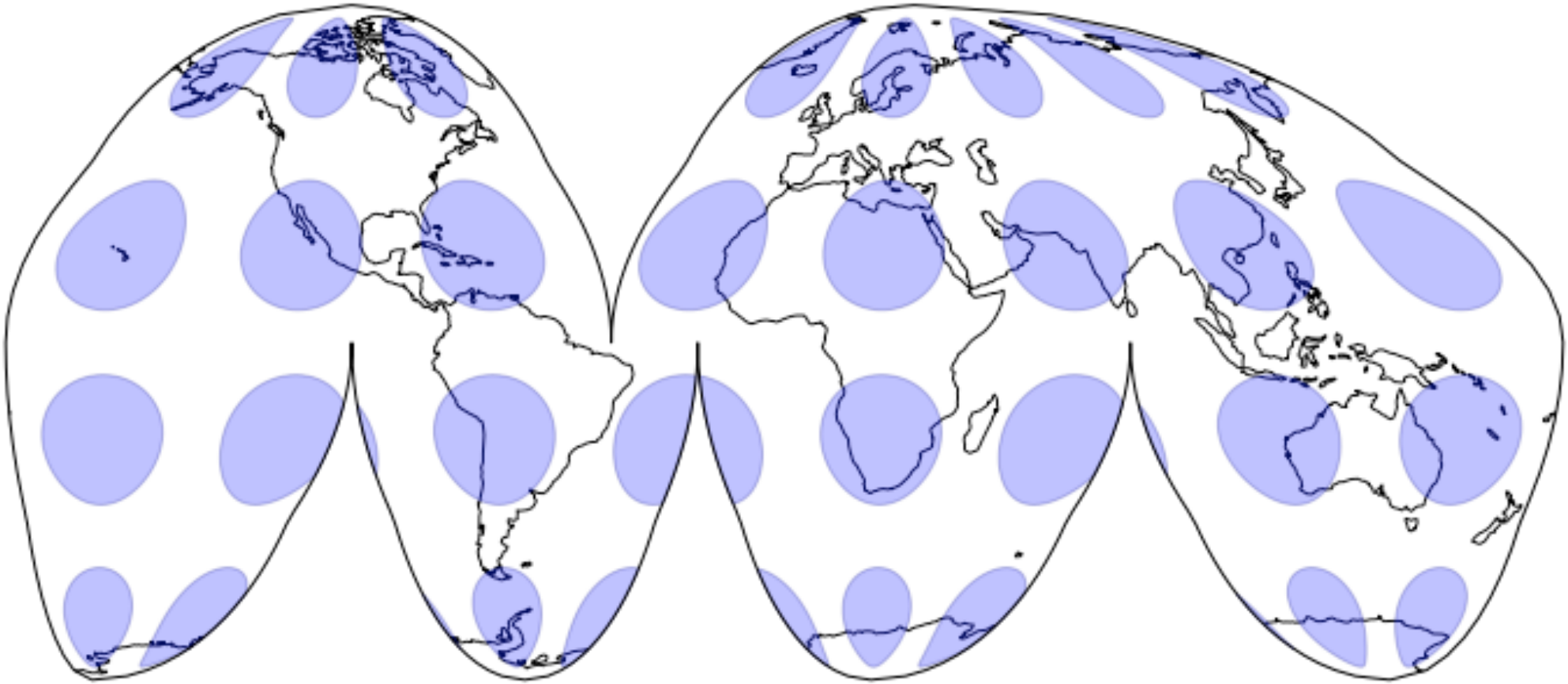
we-do-IT
Spatial Information Technology Consultants







we-do-IT
Spatial Information Technology Consultants



we-do-IT
Spatial Information Technology Consultants

- I've got some geolocated tweets and some satellite imagery. I'd like to make a map showing these tweets and any cluster of three or more tweets within 100 metres of each other.

- Fiona (http://toblerity.org/fiona/index.html) is a Pythonic replacement for the OGR python bindings

```
import fiona
with fiona.open("./data/tweets.shp") as source:
    print source[0]
```

VS

```
from osgeo import ogr
dset = ogr.Open("./data/tweets.shp")
lyr = dset[0]
record = lyr[0]
print record.ExportToJson()
del dset
```

we-do-IT
Spatial Information Technology Consultants

- Features in fiona are read as GeoJSON (http://geojson.org/) objects

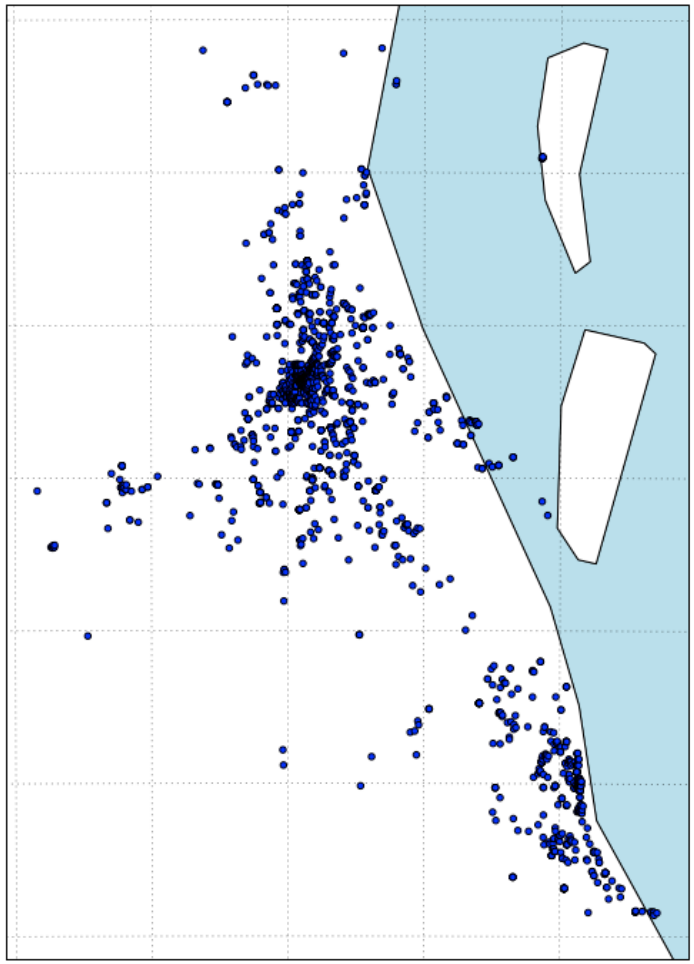
```
{
  "geometry": {
    "type": "Point",
    "coordinates": [
      501960.4667412634,
      6960829.027516253
    ]
  },
  "type": "Feature",
  "id": "910",
  "properties": {
    "created_at": "2014-08-01T17:36:52+10:00",
    "pyconau": 1,
    "message": "#pyconau data miniconf was awesome. Now back to my
room to finish my talk on carto in Python using Cartopy #gis",
    "user": "om_henners"
  }}
```

- I can read all the tweets in with Fiona but a lot of them are outside our area interest
- So we can filter them

```
records = source.filter(bbox=(xmin,
ymin, xmax, ymax))
```

- And then put them in a numpy array

```
all_points = np.array(
    [r["geometry"]["coordinates"] for r
in records]
)
```



- I'm using the scipy.cluster.hierarchy module to build my clusters
  - I could use scikit-learn clustering
- My cluster criterion will be "distance", and my cluster method will be "single"

```
clusters = scipy.cluster.hierarchy.fclusterdata(all_points, 100,
criterion="distance", method="single")
```

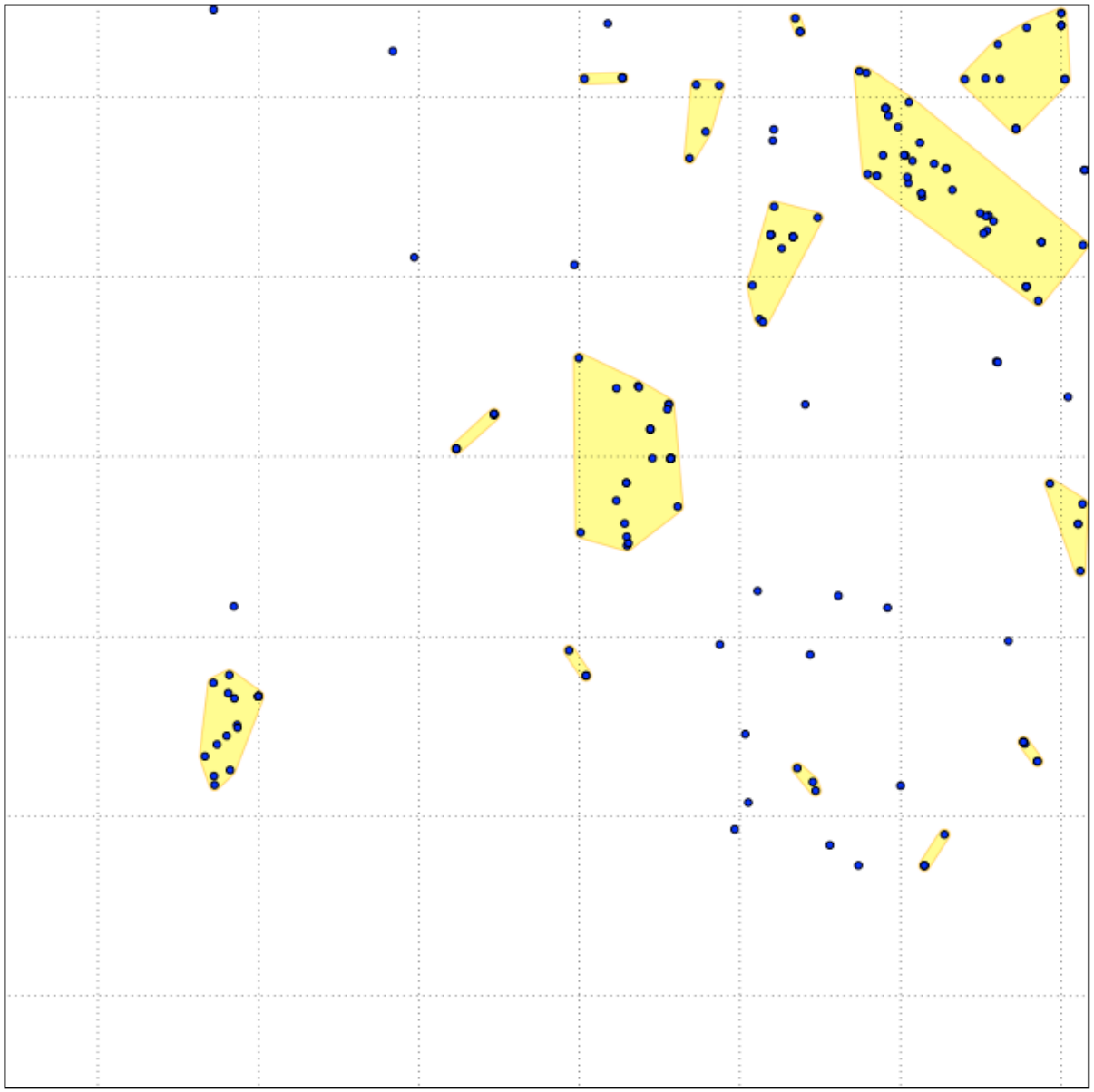
- Shapely (http://toblerity.org/shapely/manual.html) is a vector manipulation and analysis library based on the GEOS package
- Shapely can convert a numpy array to a vector geometry

```
import shapely.geometry
multipoint = shapely.geometry.asMultiPoint(np.array([[0,0],
[1,1]])
```

- I'll use one of the Shapely to build a convex hull around the clusters

```
multipoints = []
print np.bincount(clusters)

for i, count in enumerate(np.bincount(clusters)):
    if count < 3:
        continue
    coords = all_points[clusters == i]
    multipoints.append(
  shapely.geometry.asMultiPoint(coords).convex_hull.buffer(10))
```



- Rasterio (https://github.com/mapbox/rasterio) is a better Python GDAL wrapper

```
import rasterio

with rasterio.open(imagery_path) as source:
    image = source.read_band(1)
```

- vs

```
from osgeo import gdal

source = gdal.Open(imagery_path)
band = source.GetRasterBand(1)
image = band.ReadAsArray()

del source
```

- **WARNING**: Raster band indexes start with 1 (not 0)
- This is to stay consistent with the Landsat band numbering convention

- Raster data is read as a numpy array
- I can use imshow to view the data

- Cartopy (http://scitools.org.uk/cartopy/docs/latest/index.html) is a library from the British Meteorological Office
- It's a great replacement for matplotlib.basemap
- Every map in this presentation was made using Cartopy

- In cartopy every map object (including the canvas itself) has a projection
- The easiest way to define this in cartopy is with the “epsg” method.

```
proj = cartopy.crs.epsg(32756)
```

- **PROTIP** If you can possibly avoid it don’t reproject rasters

- Then it's simply a matter of setting the projection of the map canvas

```
ax = plt.axes(projection=proj)
ax.gridlines()
```

- And adding data

```
ax.imshow(image, extent=img_bounds, transform=proj,
origin='upper', cmap='gray')
ax.scatter(all_points[:,0], all_points[:, 1], transform=proj,
zorder=2)
```

- Cartopy supports adding collections of Shapely geometries

```
ax.add_geometries(multipoints, proj, facecolor="yellow",
edgecolor="orange", alpha=0.3)
```