Mastering Geospatial Analysis with Python

Eric van Rees
Technical writer

Betabeers

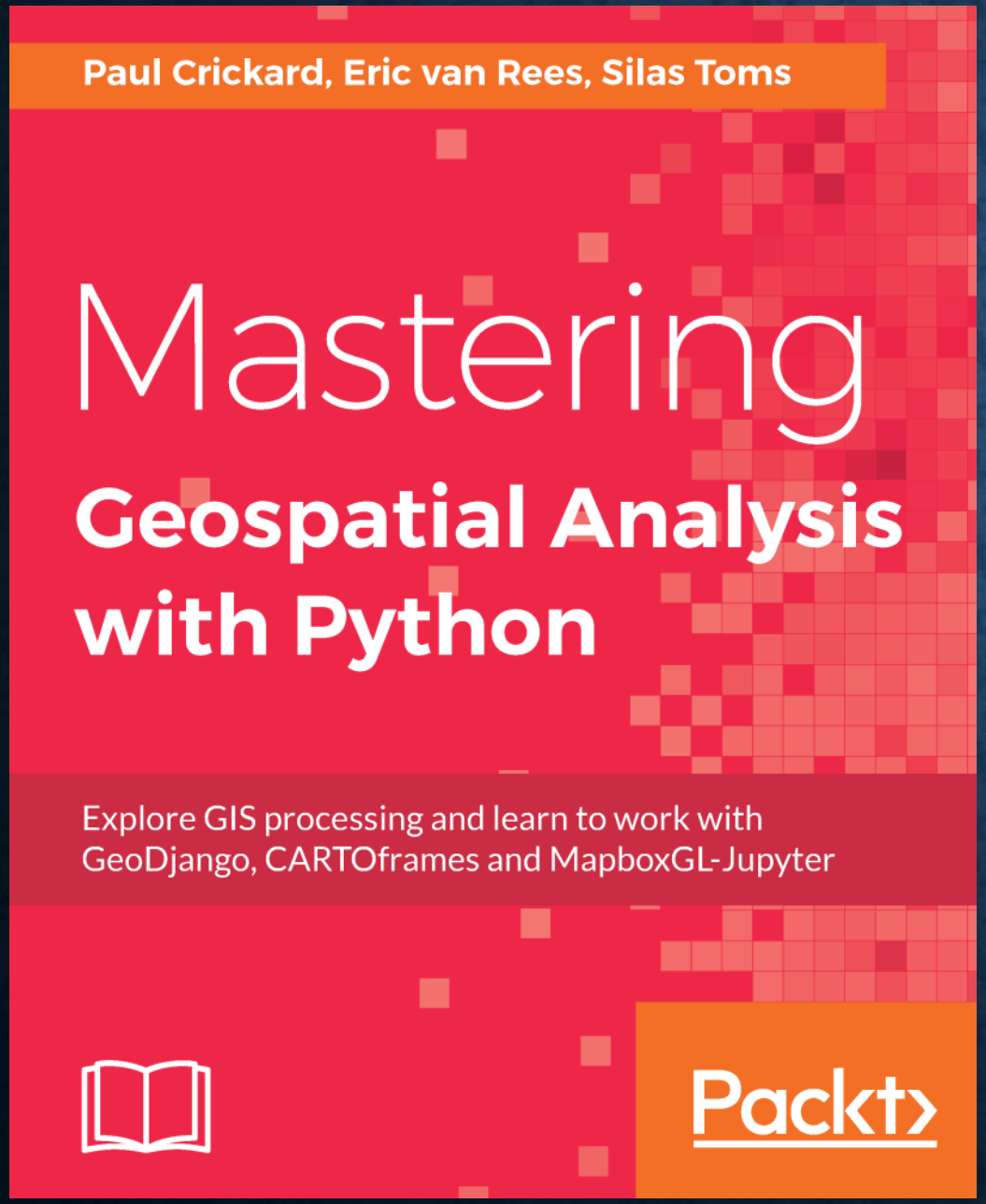
Jueves 20 de Septiembre

errant espacio cowork

OUT BARRIERS

ABOUT ME

- Born Dutch, studied Human Geography
- Self-taught Python developer
- Remote freelance technical writer
- Contact:
  www.ericvanrees.com
  eric@ericvanrees.com



- New book: Released in May 2018
- Paper + eBook
- Comes with examples code files
- 400 pages
- Learning by doing:
  - Software Tutorials
  - Best practices
  - Tips & Tricks

WHAT IS PYTHON?

- Python is a high-level programming language
- Has human-readable syntax
- Is very popular among beginning programmers
- Has a huge ecosystem of freely available code libraries
- Has an active and helpful community (newsgroups, local /national/int. events)
- Can be used for scripting purposes, as well as building applications
- Many free, open source spatial Python libraries available for automating and executing spatial analysis (raster + vector)
- These integrate very well with other Python libraries:
- … for example Machine Learning, Data Science…

WHY DID WE WRITE THIS BOOK?

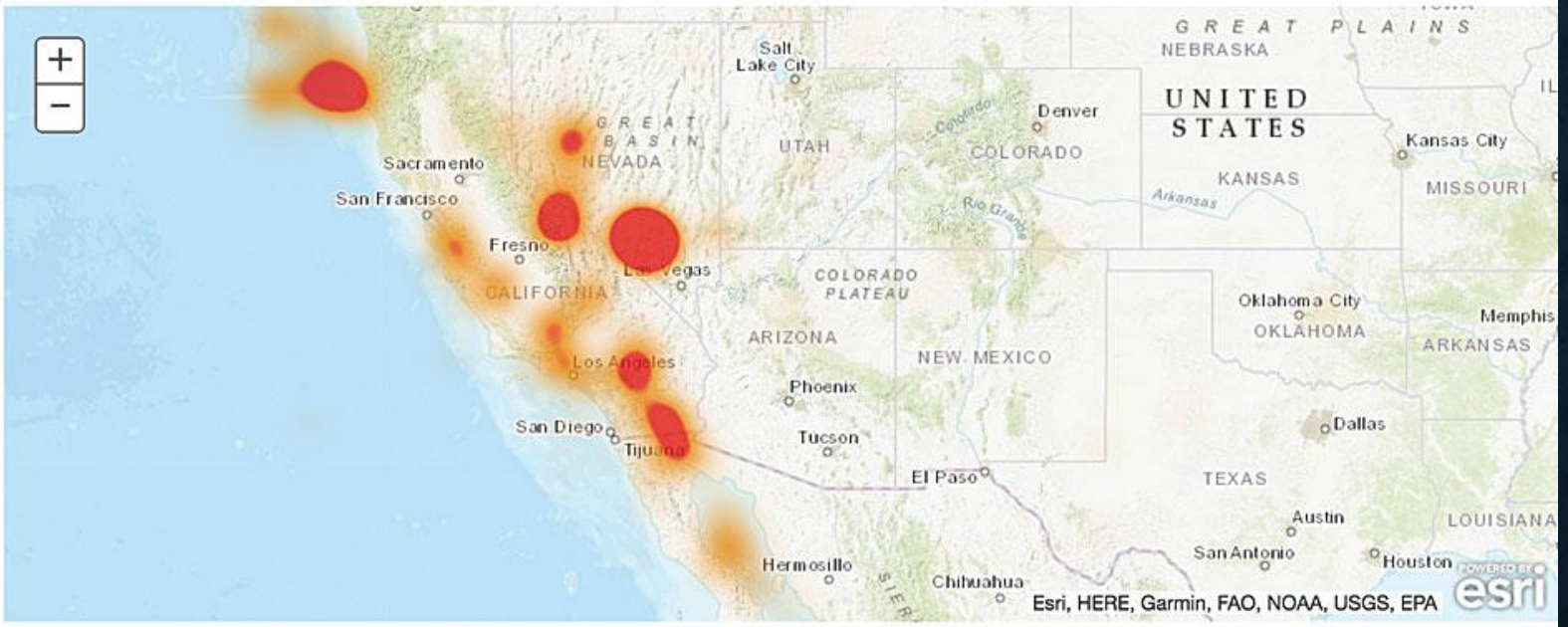
- No book available yet that covers Python 3 spatial libraries, only Python 2
- Nothing available on data visualization with Jupyter Notebooks (or JupyterLab)
- Nothing available on new cool Python libraries from Esri, Carto and Mapbox
- No high-level overview available for spatial libraries

Out[12]:

earthquakes_2
Major earthquake information

Feature Layer Collection by arcgis_python
Last Modified: July 03, 2017
0 comments, 2 views

In [10]:
```
world_eq_map = gis.map('California, USA')
world_eq_map.add_layer(eq_item, {'renderer':'HeatmapRenderer', 'opacity':0.75})
world_eq_map
```

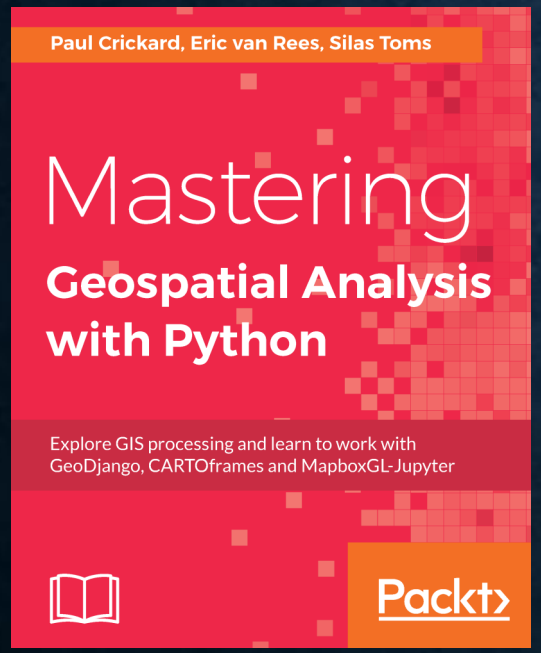


In [15]:
```
eq_flayer = eq_item.layers[0]
eq_df = eq_flayer.query().df
eq_df.head(5)
```

Out[15]:

	datetime	depth	distance	eventid	gap	latitude	longitude	magnitude	magtype	nbstations	objectid	occurrence_time	rms	source	SHAP
0	1898/06/29 18:36:00.00	0.0	None	None	None	52.00	172.0	7.6	ML	0.0	1	-2256528240000	None	AK	{'x': 172, 'y': 5
1	1898/10/11 16:37:32.70	0.0	None	None	None	50.71	-179.5	6.9	ML	0.0	2	-2247549747000	None	AK	{'x': -179.5, 'y 50.70999999999
2	1899/07/14 13:32:00.00	0.0	None	None	None	60.00	-150.0	7.2	ML	0.0	3	-2223714480000	None	AK	{'x': -150, 'y': 6
3	1899/09/04 00:22:00.00	25.0	None	None	None	60.00	-142.0	8.3	ML	0.0	4	-2219269080000	None	AK	{'x': -142, 'y': 6
4	1899/09/04	0.0	None	None	None	60.00	-142.0	6.9	ML	0.0	5	-2219253600000	None	AK	{'x': -142, 'y': 6

LOOKING FOR A PLACE TO START?



Learn Essential Spatial Python libraries:

- **GDAL/OGR/Geoss/Shapely: read/write/data conversion/geographical operations/reprojections**
- **More "pythonic libraries" that do the same thing: Fiona, Shapely, Rasterio**

PYTHON IS THE "GLUE" THAT TIES EVERYTHING TOGETHER:

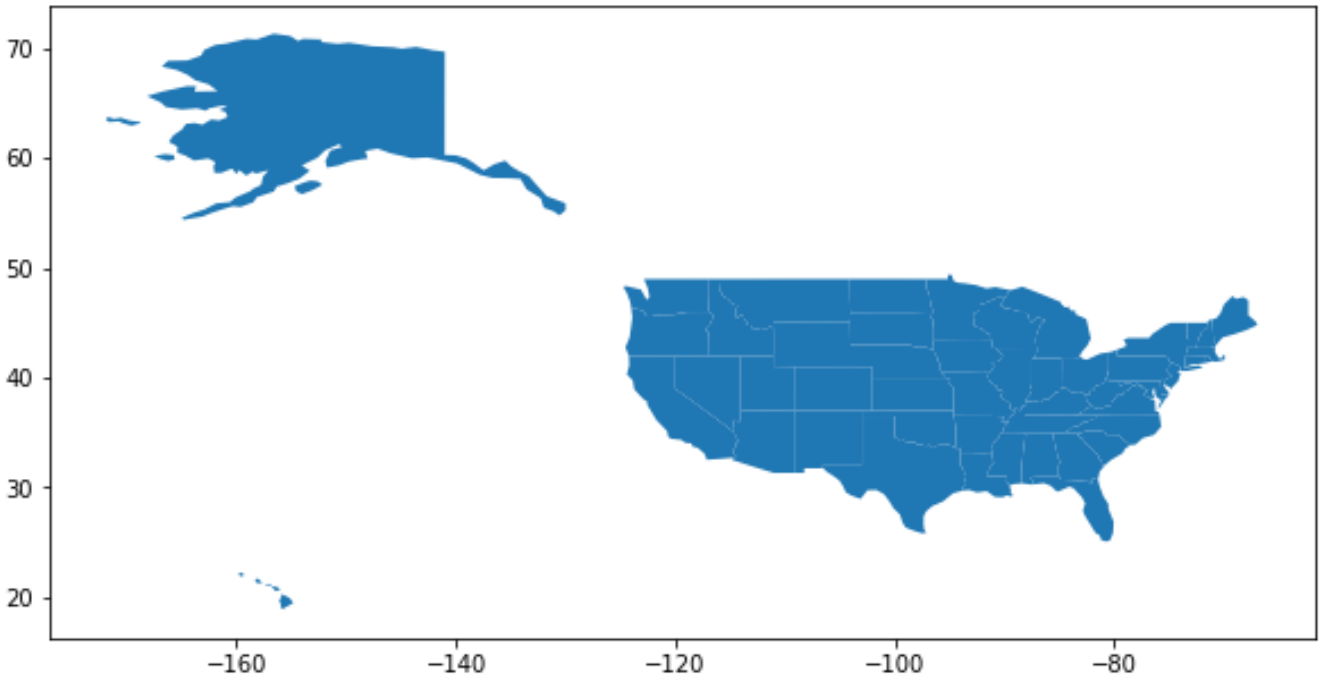
Desktop:	**GIS + Python + Spatial Database (PostGIS) OR: Python only (w/ optional spatial database)**
Web:	**Server-based GIS, Python (Flask, GeoDjango) + Local Database/Files**
Cloud:	**Cloud-based GIS Platform, w/Python and Desktop Client / OR: cloud big data processing**

CODE EXAMPLE: GEOPANDAS

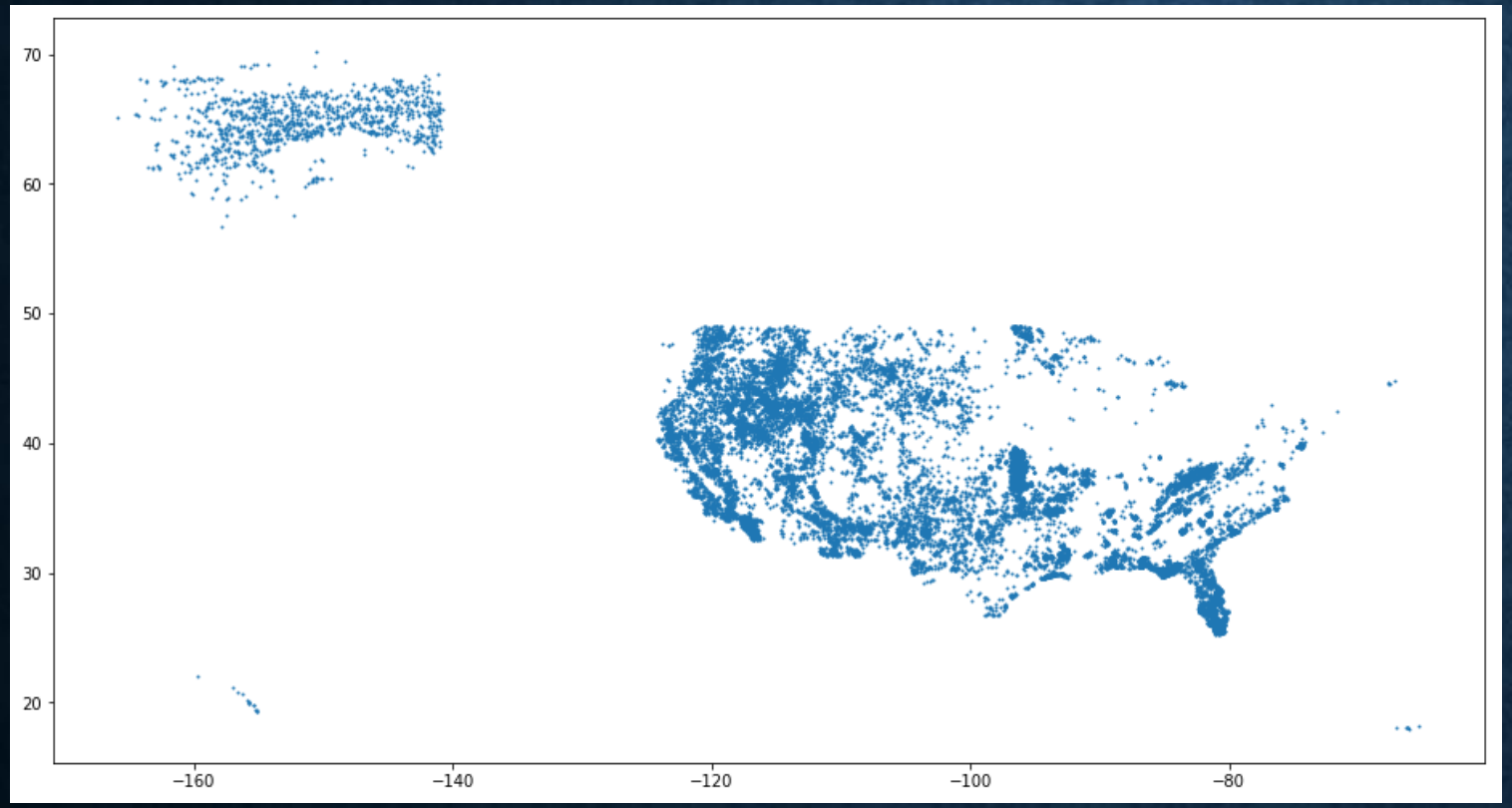
- Pandas cannot read spatial data, but GeoPandas does
- Adds 2 subclasses to pandas data objects: GeoSeries and GeoDataframes
- Read and visualize shapefile attribute tables easily – no more for loops
- Perform spatial analysis and plot the results on an map

GOAL: COUNT WILDFIRES PER STATE

Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x234eb127e48>



Map #1:US state polygons

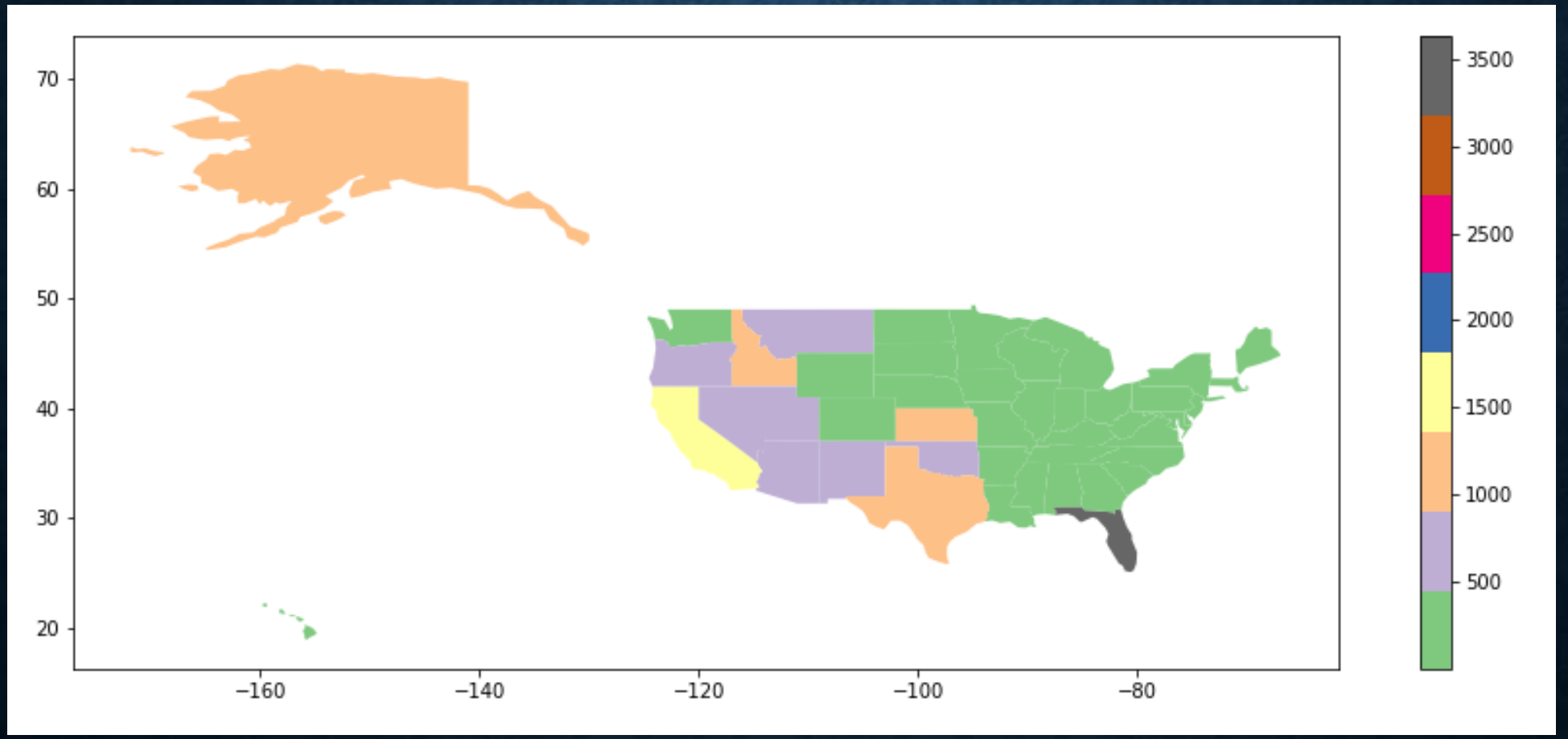


Map #2, point data of all wildfires 1984-2015

AS OUR GEODATAFRAME IS A PANDAS OBJECT, WE CAN COUNT AND RANK THE COUNTS:

```
Out[54]: name
         Florida       3635
         California    1577
         Idaho         1278
         Kansas        1124
         Alaska        1062
         Texas         1011
         Arizona        836
```

FINALLY, THIS MAP RANKS ALL STATE BASED THE TOTAL COUNT PER STATE FIELD:



SPATIAL JOIN OF POINTS AND POLYGONS

Fire_Type	geometry	index_right	name
WF	POINT (-141.851 65.29600000000001)	50	Alaska
WF	POINT (-162.314 67.75700000000001)	50	Alaska
WF	POINT (-141.217 65.05)	50	Alaska
WF	POINT (-146.817 62.698)	50	Alaska
WF	POINT (-156.362 64.077)	50	Alaska
WF	POINT (-143.9 64.137)	50	Alaska
WF	POINT (-144.441 64.545)	50	Alaska

```
# import module
import geopandas
# import the shapefile with all of the state boundaries
states = geopandas.read_file(r"C:\data\gdal\NE\110m_cultural\ne_110m_admin_1_states_provinces.shp")
# import the shapefile with wildfire data
fires = geopandas.read_file(r"C:\data\mtbs_fod_pts_data\mtbs_fod_pts_201705
01.shp")
# plot the data
fires.plot(markersize=1, figsize=(17,17))
# compare spatial reference:
fires.crs
# Out: {'init': 'epsg:4269'}
states.crs
# Out: {'init': 'epsg:4326'}
# reproject fire data so that it lines up correctly with states:
fires = fires.to_crs({'init': 'epsg:4326'})
```

```
# perform spatial join:
state_fires = geopandas.sjoin(fires,states[['name','geometry']].copy(),op='within')
# count and sort wildfires per state:
counts_per_state = state_fires.groupby('name').size()
counts_per_state.sort_values(axis=0, ascending=False)
# add these values to the original shapefile as a new field
states = states.merge(counts_per_state.reset_index(name='number_of_fires'))
states.head()
# create and plot a choropleth map for wildfire count per state:
ax = states.plot(column='number_of_fires', figsize=(15, 6), cmap='OrRd', legend=True)
# fine-tune the map a little further, by adding a title and dropping the x-axis and y-axis:
import matplotlib.pyplot as plt
f, ax = plt.subplots(1, figsize=(18,6))
ax = states.plot(column='number_of_fires', cmap='Accent',
legend=True, ax=ax)
lims = plt.axis('equal')
f.suptitle('US Wildfire count per state in 1984-2015')
ax.set_axis_off()
plt.show()
```

Thank You!

https://www.packtpub.com/application-development/mastering-geospatial-analysis-python

Any questions?

ericvanrees@gmail.com
eric@ericvanrees.com
Twitter: @geospatialprof