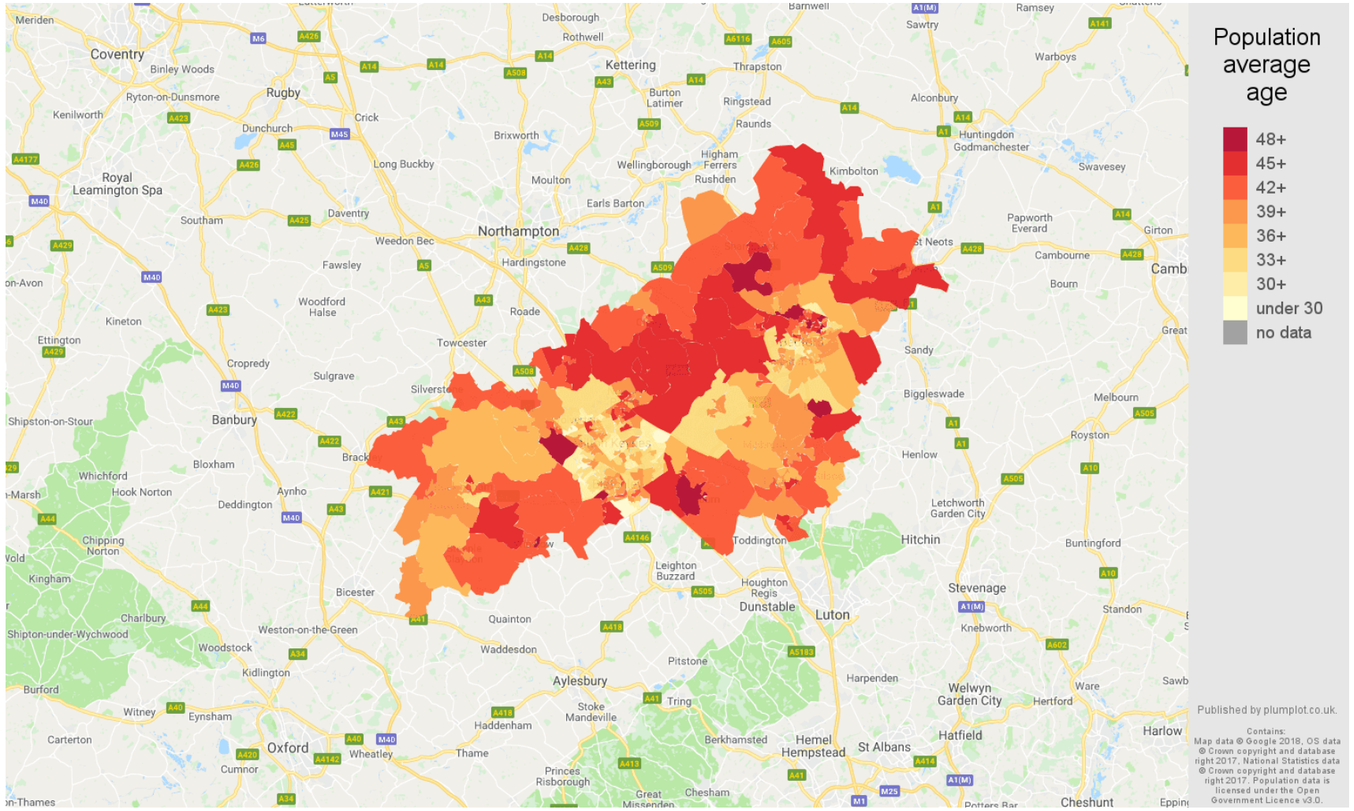
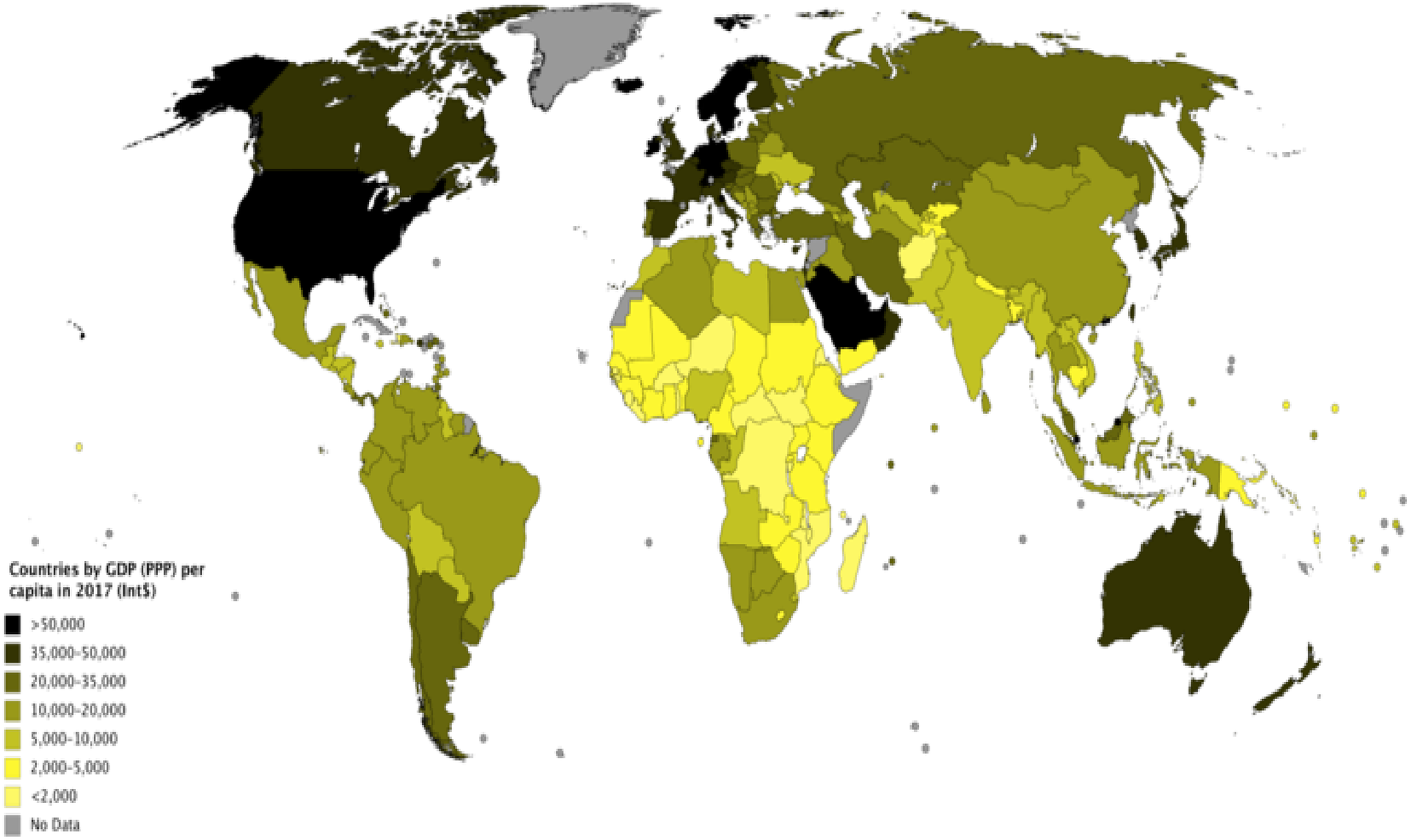
VISUALIZING GEOSPATIAL DATA IN PYTHON

What is a choropleth?

Mary van Valkenburg
Data Science Program Manager, Nashville Software School

Definition of a choropleth




Countries by GDP (PPP) per capita in 2017 (Int$)
>50,000
35,000-50,000
20,000-35,000
10,000-20,000
5,000-10,000
2,000-5,000
<2,000
No Data

Density

```
schools_in_districts.head(2)
```

```
district  geometry                    name                lat    lng
1         (POLYGON ((-86.77 36.38... Nashville Prep      36.16 -86.85
1         (POLYGON ((-86.77 36.38... Rocketship Prep     36.17 -86.79
```

Get counts

```
school_counts = schools_in_districts.groupby(['district']).size()
print(school_counts)
```

```
district
1    30
2    11
3    19
4    18
5    36
6    21
7    13
8    10
9    12
dtype: int64
```

Add counts

```
school_counts_df = school_counts.to_frame()
school_counts_df.reset_index(inplace=True)
school_counts_df.columns = ['district', 'school_count']
```

```
districts_with_counts = pd.merge(school_districts, school_counts_df,
                                 on = 'district')
districts_with_counts.head(2)
```

```
district geometry            school_count
1        (POLYGON ((-86.77 36.38...  30
3        (POLYGON ((-86.75 36.40...  19
```

Divide counts by area

```
districts_with_counts['area'] = districts_with_counts.geometry.area
```

```
districts_with_counts['school_density'] = districts_with_counts.apply(
    lambda row: row.school_count/row.area, axis = 1)

districts_with_counts.head(2)
```

```
district geometry            school_count   area     school_density
1        (POLYGON ((-86.77 36.38...    30     0.036641    818.745403
3        (POLYGON ((-86.75 36.40...    19     0.014205    1337.594495
```

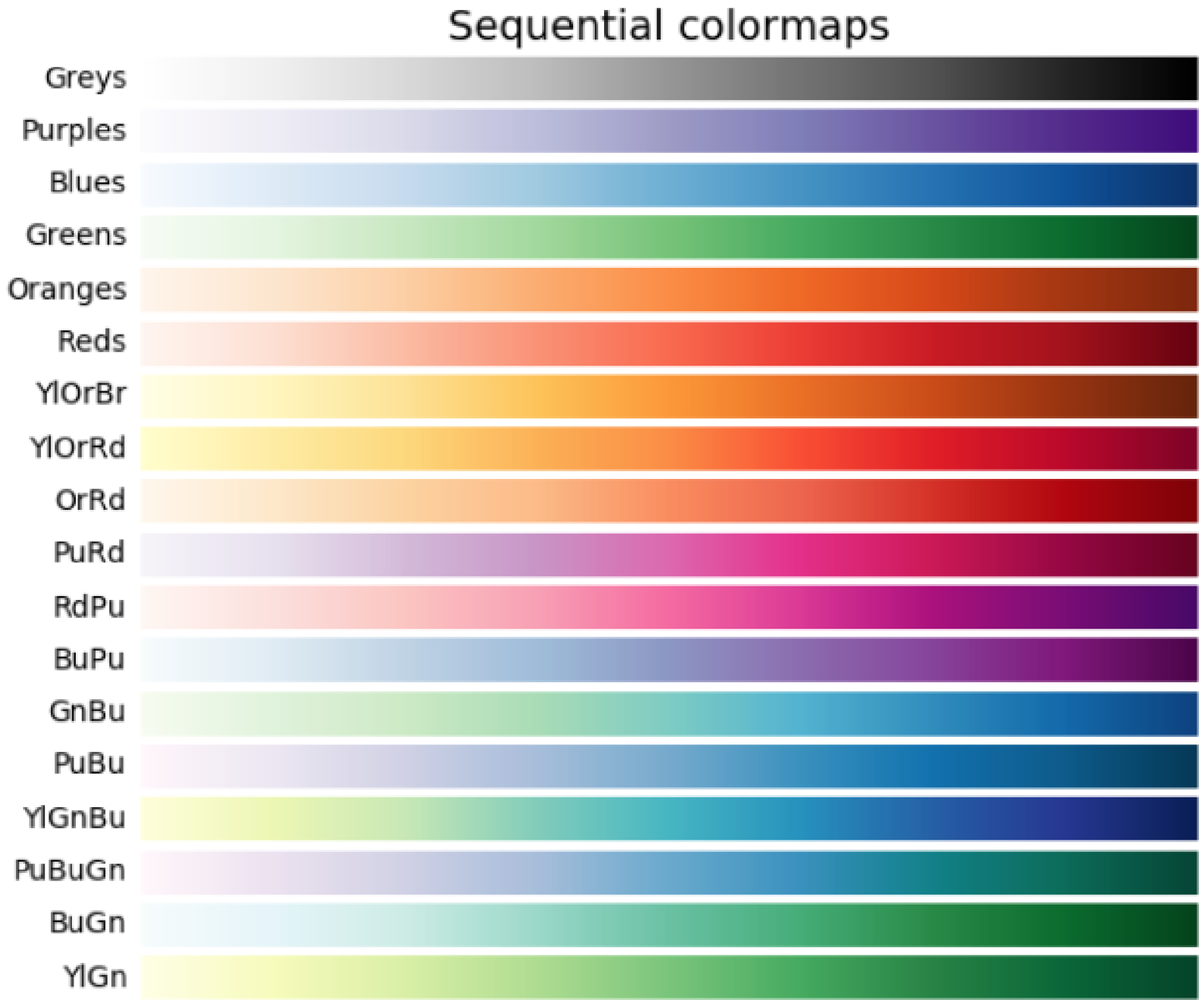
VISUALIZING GEOSPATIAL DATA IN PYTHON

Let's Practice!

VISUALIZING GEOSPATIAL DATA IN PYTHON

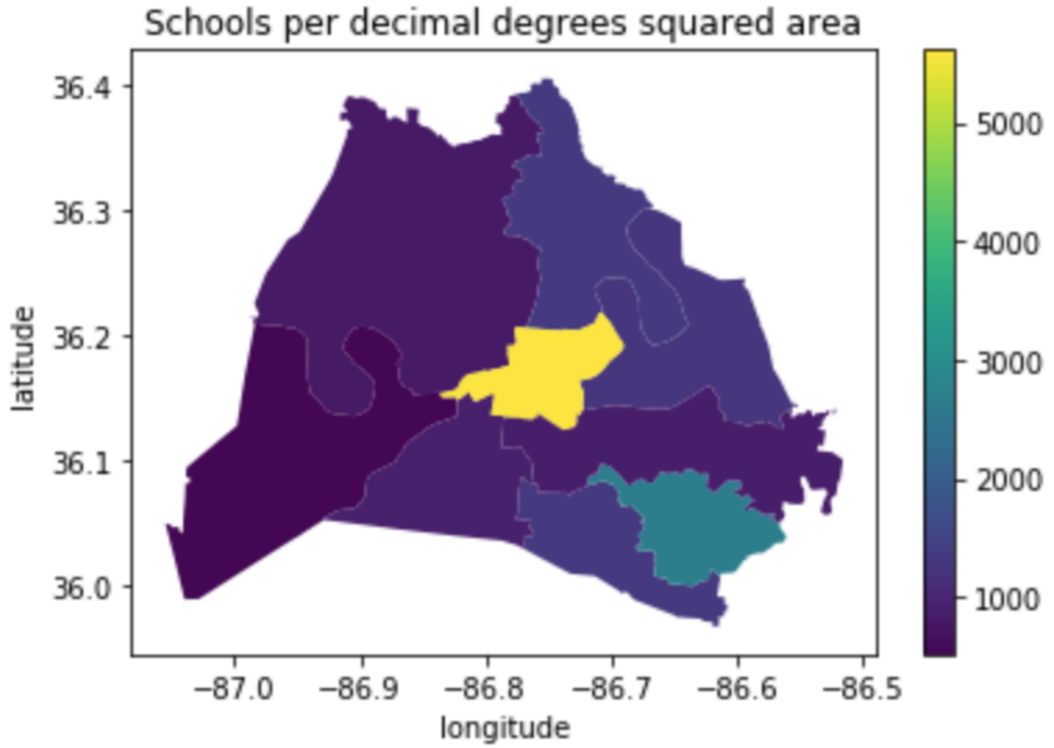
Choropleths with geopandas

Mary van Valkenburg
Data Science Program Manager, Nashville Software School


Sequential colormaps
Greys
Purples
Blues
Greens
Oranges
Reds
YlOrBr
YlOrRd
OrRd
PuRd
RdPu
BuPu
GnBu
PuBu
YlGnBu
PuBuGn
BuGn
YlGn

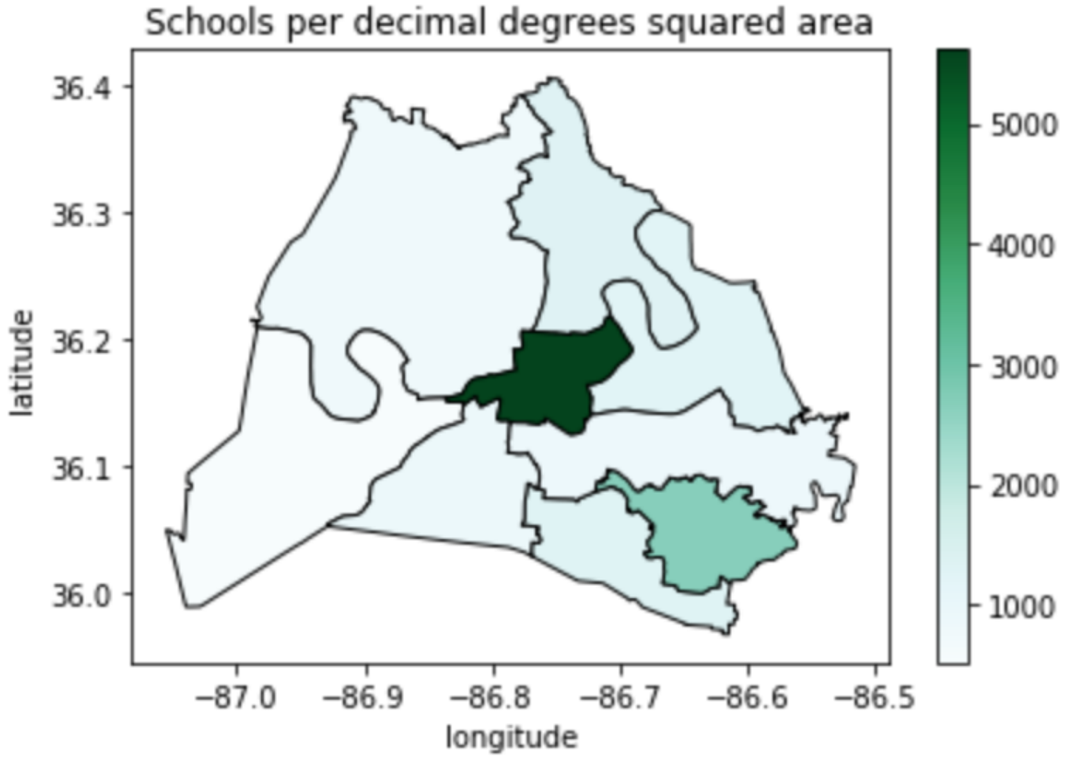
Choropleth with GeoDataFrame.plot()

```
districts_with_counts.plot(column = 'school_density', legend = True)
plt.title('Schools per decimal degrees squared area')
plt.xlabel('longitude')
plt.ylabel('latitude');
```



Choropleth with GeoDataFrame.plot()

```
districts_with_counts.plot(column = 'school_density', cmap = 'BuGn',
                           edgecolor = 'black', legend = True)
plt.title('Schools per decimal degrees squared area')
plt.xlabel('longitude')
plt.ylabel('latitude');
```



Area in Kilometers Squared

```
# starting CRS
print(school_districts.crs)
```

```
{'init': 'epsg:4326'}
```

```
# convert to EPSG 3857
school_districts = school_districts.to_crs(epsg = 3857)
print(school_districts.crs)
```

```
{'init': 'epsg:3857', 'no_defs': True}
```

Area in Kilometers Squared

```
# define a variable for m^2 to km^2
sqm_to_sqkm = 10**6

school_districts['area'] = school_districts.geometry.area / sqm_to_sqkm
school_districts.head(2)
```

```
district    geometry                                area
1          (POLYGON ((-965.055 4353528.766...    563.134380
3          (POLYGON ((-965.823 4356392.677...    218.369949
```

Latitude and longitude in decimal degrees

```
# change crs back to 4326
school_districts = school_districts.to_crs(epsg = 4326)
print(school_districts.crs)
```

```
{'init': 'epsg:4326', 'no_defs': True}
```

```
print(school_districts.head(2))
```

```
district      geometry                          area
1             (POLYGON ((-86.771 36.383...   563.134380
3             (POLYGON ((-86.753 36.404...   218.369949
```

```
# spatial join to get districts that contain schools
schools_in_districts = gpd.sjoin(school_districts,
                                 schools_geo, op = 'contains')
```

Counting schools in each district

```
# aggregate to get counts
school_counts = schools_in_districts.groupby(['district']).size()
```

```
# convert school_counts to a df
school_counts_df = school_counts.to_frame()
school_counts_df.reset_index(level=0, inplace=True)
school_counts_df.columns = ['district', 'school_count']
```

```
# merge
districts_with_counts = pd.merge(school_districts,
                                 school_counts_df, on = 'district')
```

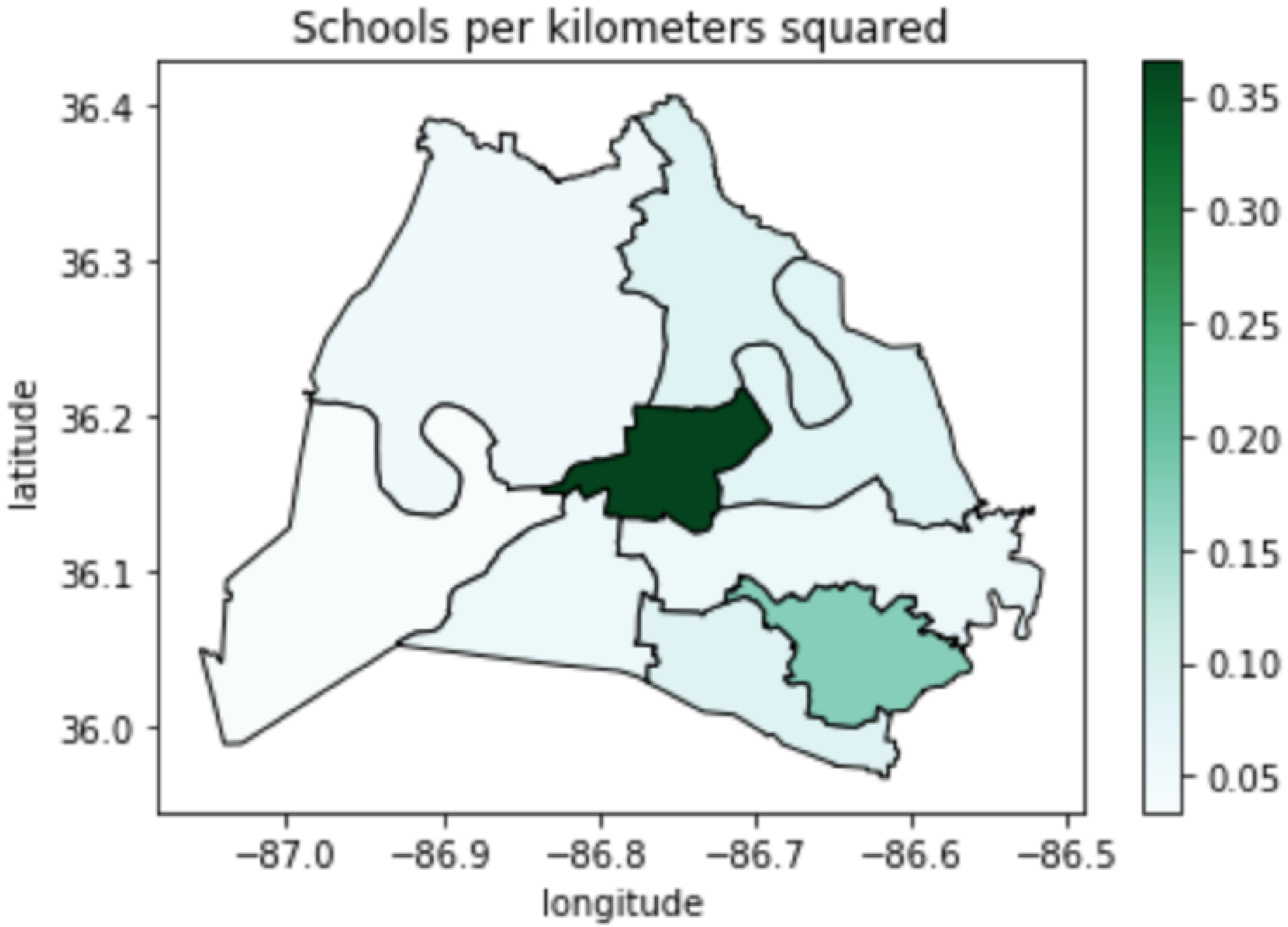
```
districts_with_counts.head(2)
```

```
district  geometry                        area        school_count
1         (POLYGON ((-86.771 36.383..     563.134380    30
3         (POLYGON ((-86.753 36.404...    218.369949    19
```

Calculating school density

```
# create school_density
districts_with_counts['school_density'] = districts_with_counts.apply(
    lambda row: row.school_count/row.area, axis = 1)
```

```
# plot it
districts_with_counts.plot(column = 'school_density', cmap = 'BuGn',
                           edgecolor = 'black', legend = True)
plt.title('Schools per kilometers squared')
plt.xlabel('longitude')
plt.ylabel('latitude');
```


Schools per kilometers squared
36.4
36.3
36.2
36.1
36.0
latitude
−87.0
−86.9
−86.8
−86.7
−86.6
−86.5
longitude
0.35
0.30
0.25
0.20
0.15
0.10
0.05

VISUALIZING GEOSPATIAL DATA IN PYTHON

Let's practice!

VISUALIZING GEOSPATIAL DATA IN PYTHON

Choropleths with folium

Mary van Valkenburg
Data Science Program Manager, Nashville Software School

folium.Map choropleth

```
# Construct a map object for Nashville
nashville = [36.1636,-86.7823]
m = folium.Map(location=nashville, zoom_start=10)

# Create a choropleth
m.choropleth(...)
```

Arguments of the folium choropleth

- `geo_data` - the source data for the polygons (geojson file or a GeoDataFrame)
- `name` - the name of the geometry column (or geojson property) for the polygons
- `data`- the source DataFrame or Series for the normalized data
- `columns`- a list of columns: one that corresponds to the polygons and one that has the value to plot

Additional arguments of the folium choropleth

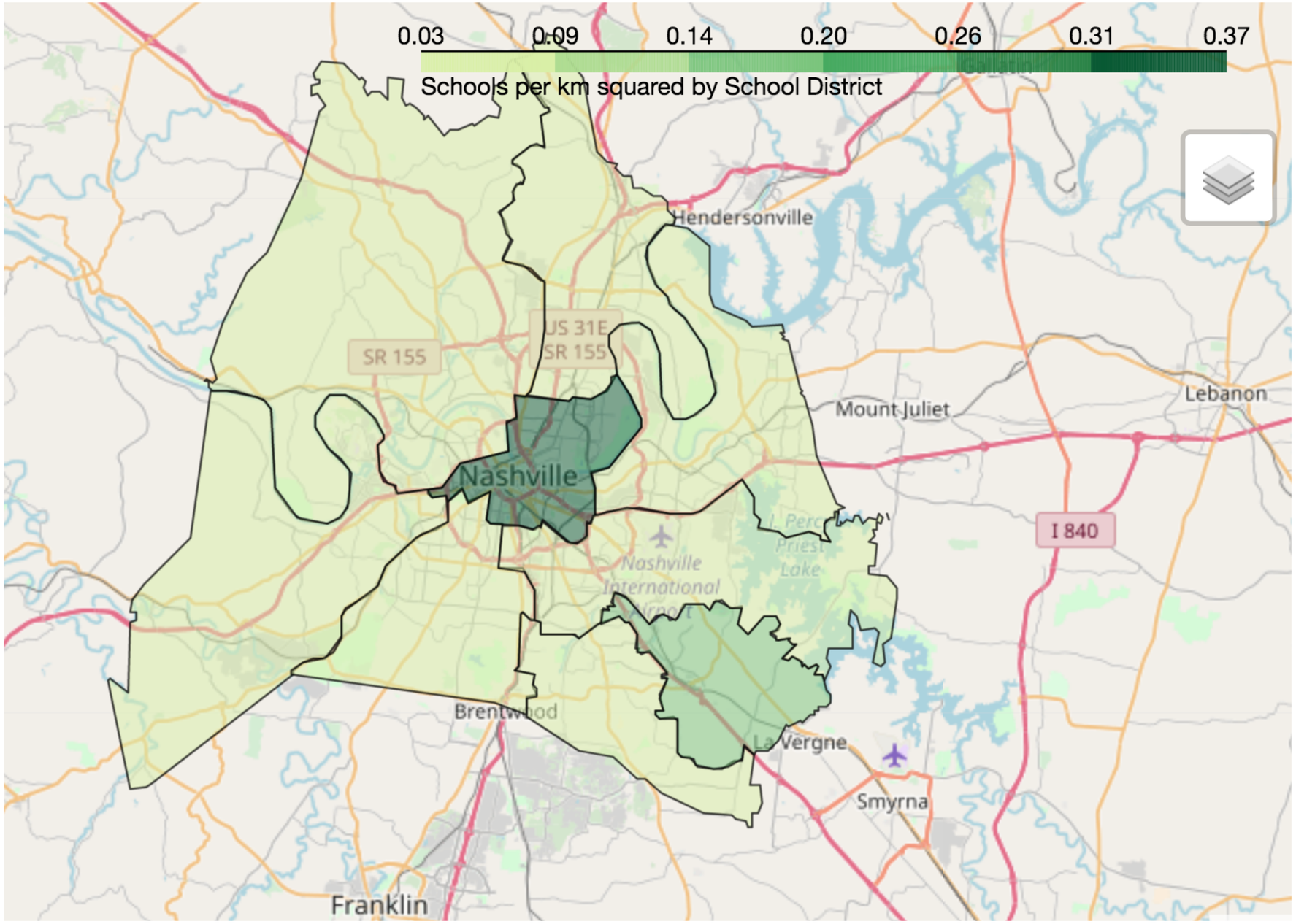
- `key_on` - a GeoJSON variable to bind the data to (always starts with `feature`)
- `fill_color` - polygon fill color (defaults to blue)
- `fill_opacity` - range between 0 (transparent) and 1 (completely opaque)
- `line_color` - color of polygon border lines (defaults to black)
- `line_opacity` - range between 0 (transparent) and 1 (completely opaque)
- `legend_name` - creates a title for the legend

Folium choropleth of school density

```
# Center point and map for Nashville
nashville = [36.1636,-86.7823]
m = folium.Map(location=nashville, zoom_start=10)
```

```
# Define a choropleth layer for the map
m.choropleth(
    geo_data=districts_with_counts,
    name='geometry',
    data=districts_with_counts,
    columns=['district', 'school_density'],
    key_on='feature.properties.district',
    fill_color='YlGn',
    fill_opacity=0.75,
    line_opacity=0.5,
    legend_name='Schools per km squared by School District'
   )
```

```
# Add layer control and display
folium.LayerControl().add_to(m)
display(m)
```


0.03
0.09
0.14
0.20
0.26
0.31
0.37
Schools per km squared by School District
Hendersonville
SR 155
US 31E
SR 155
Mount Juliet
Lebanon
Nashville
I 840
Nashville International
Priest Lake
Brentwood
La Vergne
Smyrna
Franklin

VISUALIZING GEOSPATIAL DATA IN PYTHON

Let's Practice!

VISUALIZING GEOSPATIAL DATA IN PYTHON

Congratulations!

Mary van Valkenburg
Data Science Program Manager, Nashville Software School

Skills list

- how to work with shapefiles and GeoJSON
- how to work with geometries
- how to use geopandas, shapely, and folium to extract meaning from geospatial data
- how to create beautiful and informative geospatial visualizations


WARNING!
NASHVILLE
#PeaceLoveGoodDeeds
PeaceLoveGoodDeeds.com
CASH FOR CLUES TO CRIME
EASTSIDEMURALS

VISUALIZING GEOSPATIAL DATA IN PYTHON

Goodbye