Rotman

INTRO TO DATA VISUALIZATION

Part II Plotting Maps with GeoPandas & Matplotlib

September 6, 2024 Prepared by Jay Cao / TDMDAL

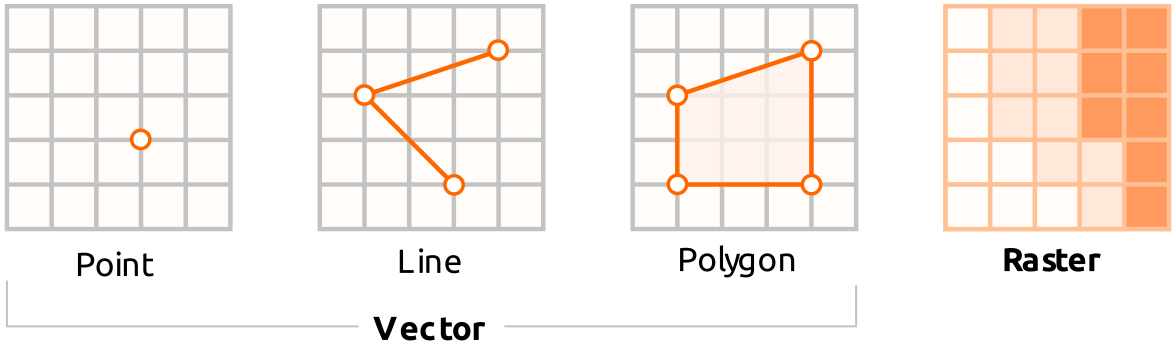
Website: https://tdmdal.github.io/mma-dv-2024/

Rotman School of Management
UNIVERSITY OF TORONTO

Spatial Data

- A spatial dataset is a combination of...
  - location data and spatial dimensions (the where)
  - attribute data (the what)

- Two most common forms of spatial data
  - Vector
  - Raster



Source/Ref: Introduction to (Q)GIS by Jeff Allen

Vector Data

- Use geographic coordinates, or a series of coordinates, to create points, lines, and polygons representing real-world features

- Two most common vector data format
  - [GeoJSON]
  - [Shapefile]
    - Shape (.shp) contains feature geometry
    - Shape index (.shx): facilitates fast search
    - Attribute (.dbf): contains attributes for each shape

An example of GeoJSON file.

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

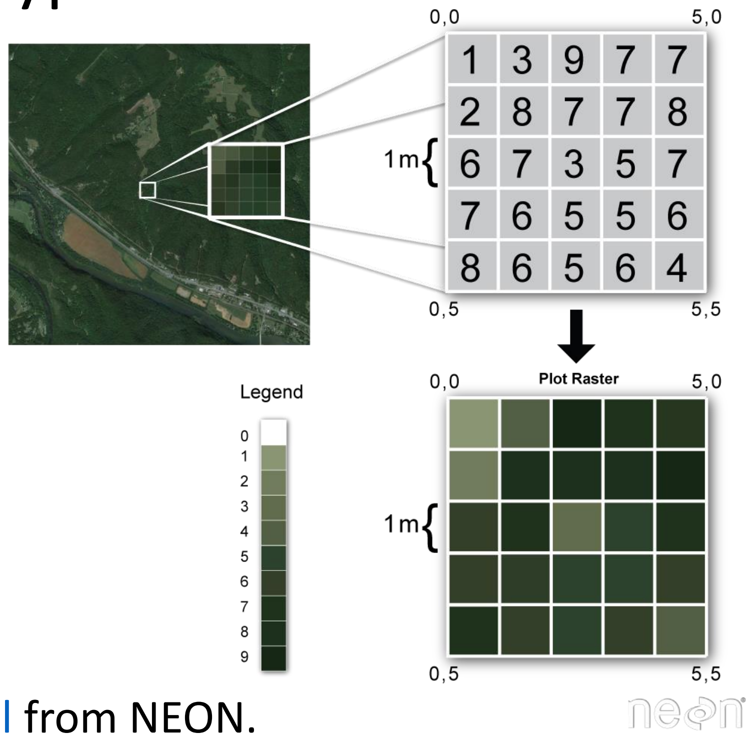
Source/Ref: Introduction to (Q)GIS by Jeff Allen

Raster Data

- Represents space as a continuous grid with equal cell sizes

- Each cell contains a value pertaining to the type of feature it represents
  - Quantitative value (e.g., elevation)
  - Categorical value (e.g., type of land use)

- Examples of raster data
  - digital elevation models (DEMs)
  - satellite imagery

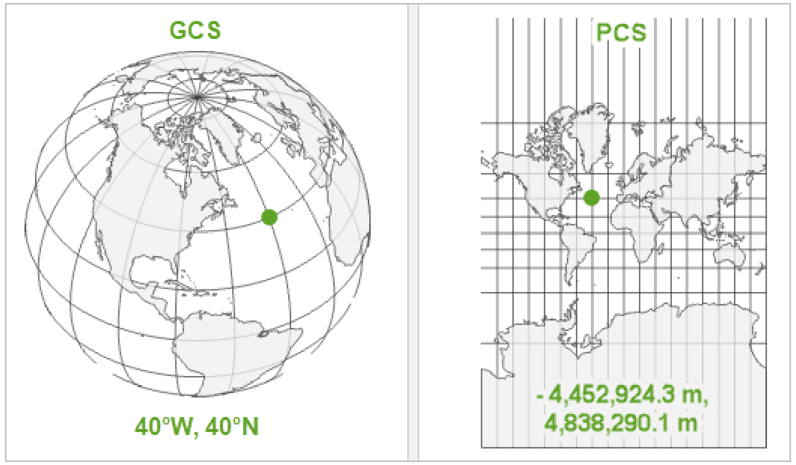


Source/Ref: 1) [Introduction to (Q)GIS](#) by [Jeff Allen](#); 2) [Raster 00 tutorial](#) from NEON.

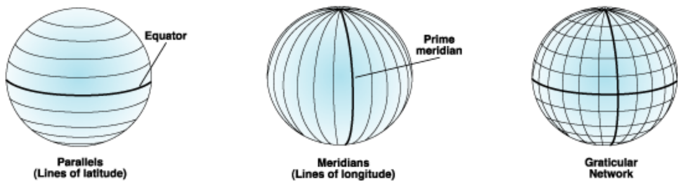
Coordinate Reference System (CRS)

- Any geo-spatial dataset comes with a CRS
- Without CRS, one cannot plot and process geospatial data correctly
- CRS is “a framework used to precisely measure locations on the surface of Earth as coordinates.” (wikipedia)
- CRS is a COMPLEX topic
    - Let’s attempt to understand some BASICS.

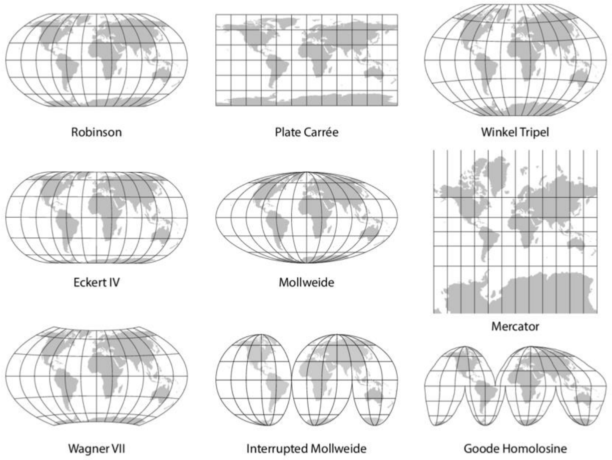
Two Main Types of CRS



- Geographic Coordinate Systems (GCS)
  - Round
  - Records locations in angular units (e.g., degrees)
    - Latitude: degrees north or south of equator
    - Longitude: degrees west or east of a prime meridian
  - Many different model the Earth surface -> Many CRS
    - e.g. World Geodetic System 1984 (WGS84)



- Projected Coordinate Systems (PCS)
  - Flat
  - Records locations in linear units (e.g., meters)
  - Many ways to project -> Many PCS
    - e.g., Universal Transverse Mercator (UTM)

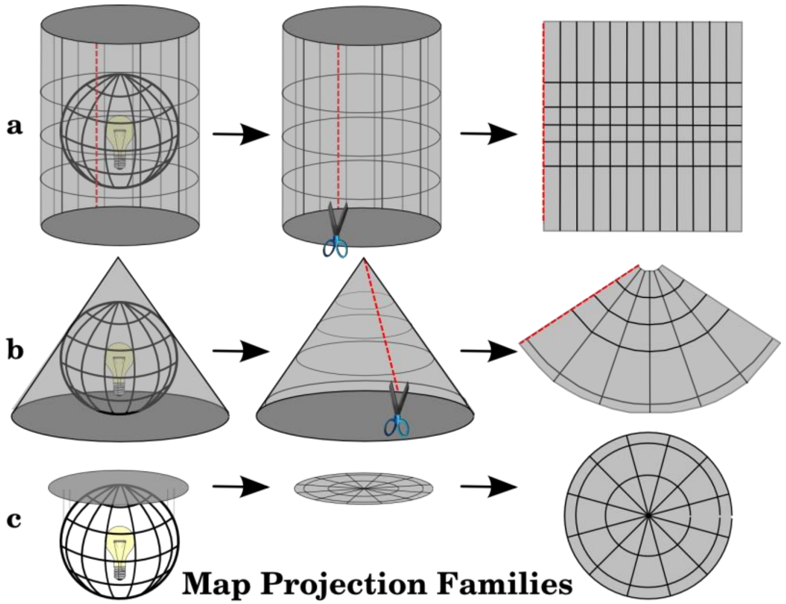


Source Here

Source/Ref: 1) CRS Wikipedia; 2) GCS vs PCS

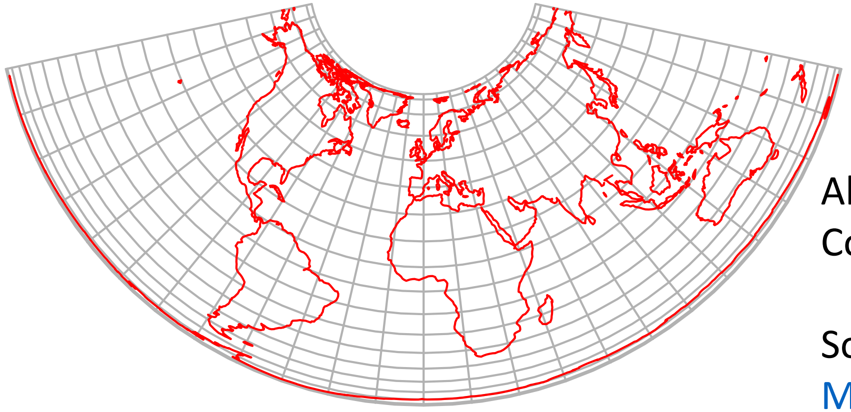
Projections

- MANY different projections
  - Three main families, but there are more
  - Within each family, there are many projections



Source: QGIS Doc

- A projection can preserve one or more properties below but never all
  - Direction
  - Distance
  - Area
  - Shape



Albers Equal-Area Conic Projection

Source: Wolfram MathWorld

Source/Ref: 1) List of map projections wiki; 2) Map projection wiki

Components of CRS

- Coordinate system
  - E.g., longitude & latitude, x & y, etc.
- Datum: binds abstract coordinate system to real space on the Earth
  - Datum usually consists of
    - an estimate of the shape of the Earth (usually an ellipsoid)
    - one or more anchor points for which the measurement is documented
  - Examples of dataum
    - Global datum: WGS84, North American Datum (NAD83)
    - Local datum: NAD27 (A local datum aligns its ellipsoid to closely fit the earth's surface in a particular area)
- Projection (if it is a PRS)

Source/Ref: CRS wiki

CRS is a “Stack” of Dependent Specifications

EPSG Code	Name	Ellipsoid	Horizontal Datum	CS Type	Projection	Origin	Axes	Unit of Measure
4326	GCS WGS 84	GRS 80	WGS 84	ellipsoidal (lat, lon)	N/A	equator/ prime meridian	equator, prime meridian	degree of arc
26717	UTM Zone 17N NAD 27	Clarke 1866	NAD 27	cartesian (x,y)	Transverse Mercator: central meridian 81°W, scaled 0.9996	500 km west of (81°W, 0°N)	equator, 81°W meridian	meter
6576	SPCS Tennessee Zone NAD 83 (2011) ftUS	GRS 80	NAD 83 (2011 epoch)	cartesian (x,y)	Lambert Conformal Conic: center 86°W, 34°20'N, standard parallels 35°15'N, 36°25'N	600 km grid west of center point	grid east at center point, 86°W meridian	US survey foot

Source/Ref: CRS wiki

GIS (Geographic Information Systems)

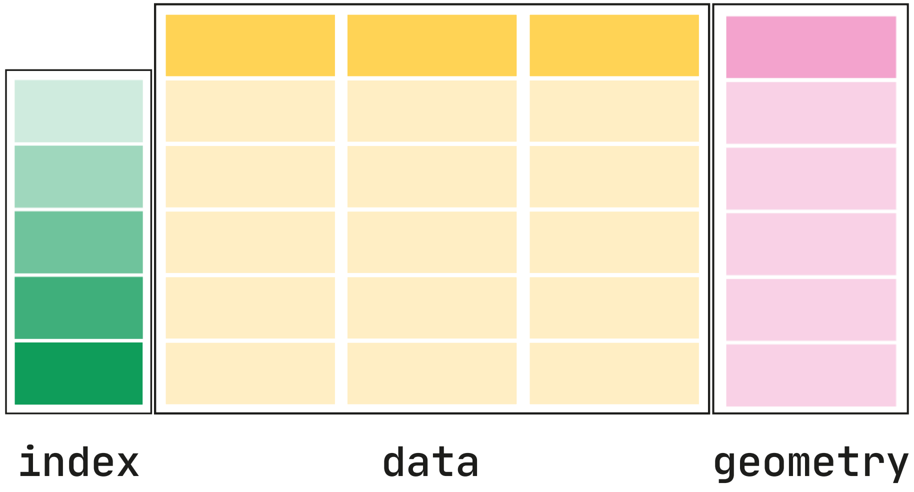
- GIS are tools to analyze, manipulate, and visualize spatial information on a computer
- Many GIS tools
  - QGIS, ArcGIS, MapBox, etc.
- We will only focus on visualization with GeoPandas's `plot()` function
  - GeoPandas's `plot()` is a method on `GeoSeries` or `GeoDataFrame`
  - GeoPandas's `plot()` builds on Matplotlib

Source/Ref: Introduction to (Q)GIS by Jeff Allen

Geopandas - 1

- Make working with geospatial data in Python easier

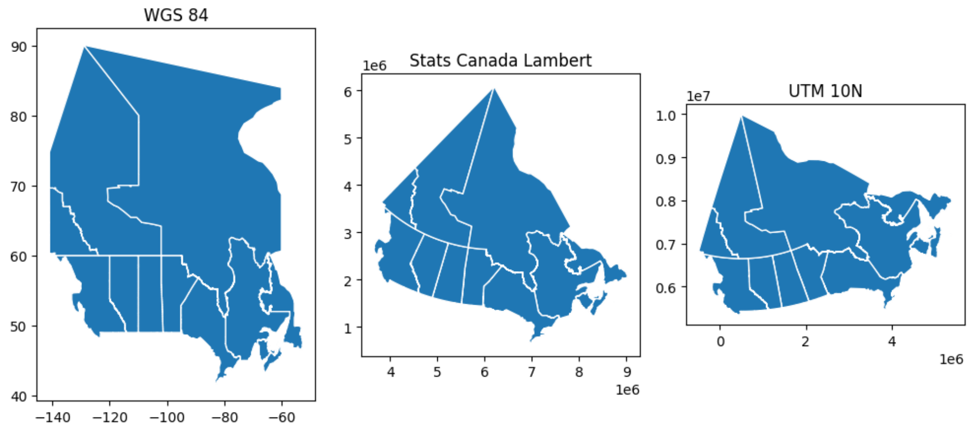
- Extends pandas to allow spatial operations on geometric types
  - Geometric operations built on shapely
  - File access built on fiona
  - Plotting built on matplotlib



Geopandas - 2

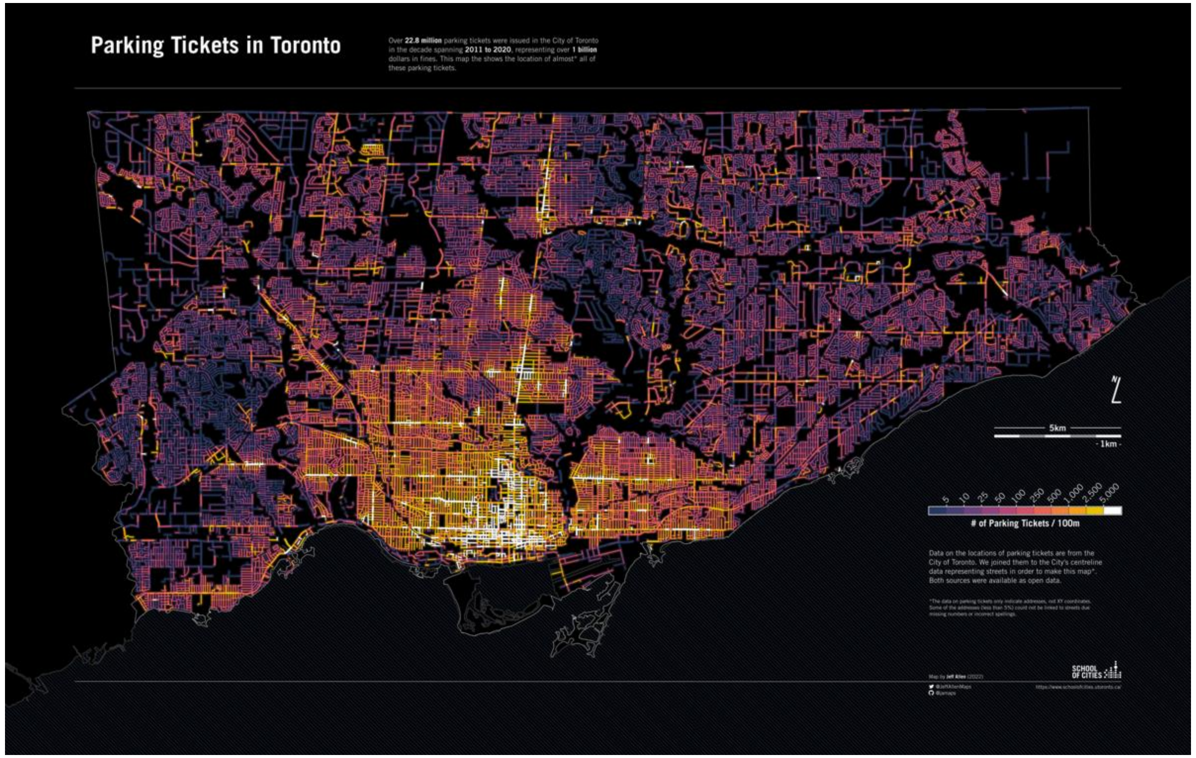
- Load dataset containing geometry info
- Manage Coordinate Reference System (CRS)
  - Store CRS info
  - translate between CRSs
- Geospatial operations
  - Calculate areas bounded by polygons
  - Spatial join
  - Spatial aggregation
  - Many more
- Plot

	PRENAME	PREABBR	LANDAREA	geometry
0	Newfoundland and Labrador	N.L.	3.581704e+05	POLYGON ((7644464.869 2980078.217, 7648864.423...
1	Prince Edward Island	P.E.I.	5.681179e+03	POLYGON ((8427184.671 1638777.314, 8427169.677...
2	Nova Scotia	N.S.	5.282471e+04	MULTIPOLYGON (((8693947.526 1494906.237, 86935...
3	New Brunswick	N.B.	7.124850e+04	POLYGON ((8188457.820 1707919.583, 8188444.194...
4	Quebec	Que.	1.298600e+06	MULTIPOLYGON (((8396476.571 1754341.151, 83956...
5	Ontario	Ont.	8.924118e+05	POLYGON ((6378273.940 2296884.400, 6378455.637...
6	Manitoba	Man.	5.403102e+05	POLYGON ((6039718.643 2636909.880, 6039717.720...
7	Saskatchewan	Sask.	5.770604e+05	POLYGON ((5248633.914 2767057.263, 5249285.640...
8	Alberta	Alta.	6.346583e+05	POLYGON ((5228304.177 2767597.891, 5228098.463...
9	British Columbia	B.C.	9.206866e+05	POLYGON ((4018904.414 3410247.271, 4019429.869...
10	Yukon	Y.T.	4.723454e+05	POLYGON ((4561932.471 4312865.174, 4564007.580...
11	Northwest Territories	N.W.T.	1.127712e+06	POLYGON ((5689672.257 4324508.314, 5685498.029...
12	Nunavut	Nvt.	1.836994e+06	POLYGON ((7297737.369 3983558.454, 7316653.440...



Walk-through – Toronto Parking Ticket

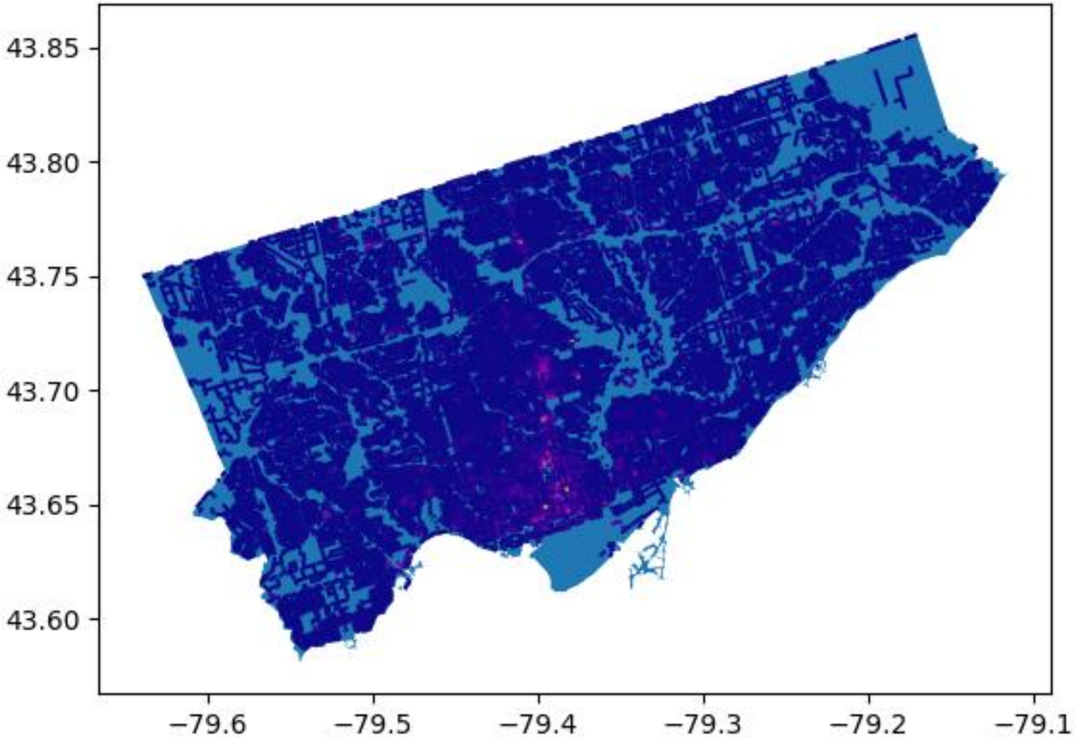
- Reproduce (almost) Jeff Alan’s Toronto Parking Tickets Visualization
- I believe Jeff’s plot is done using QGIS and Inkscape
- We will use GeoPandas with Matplotlib



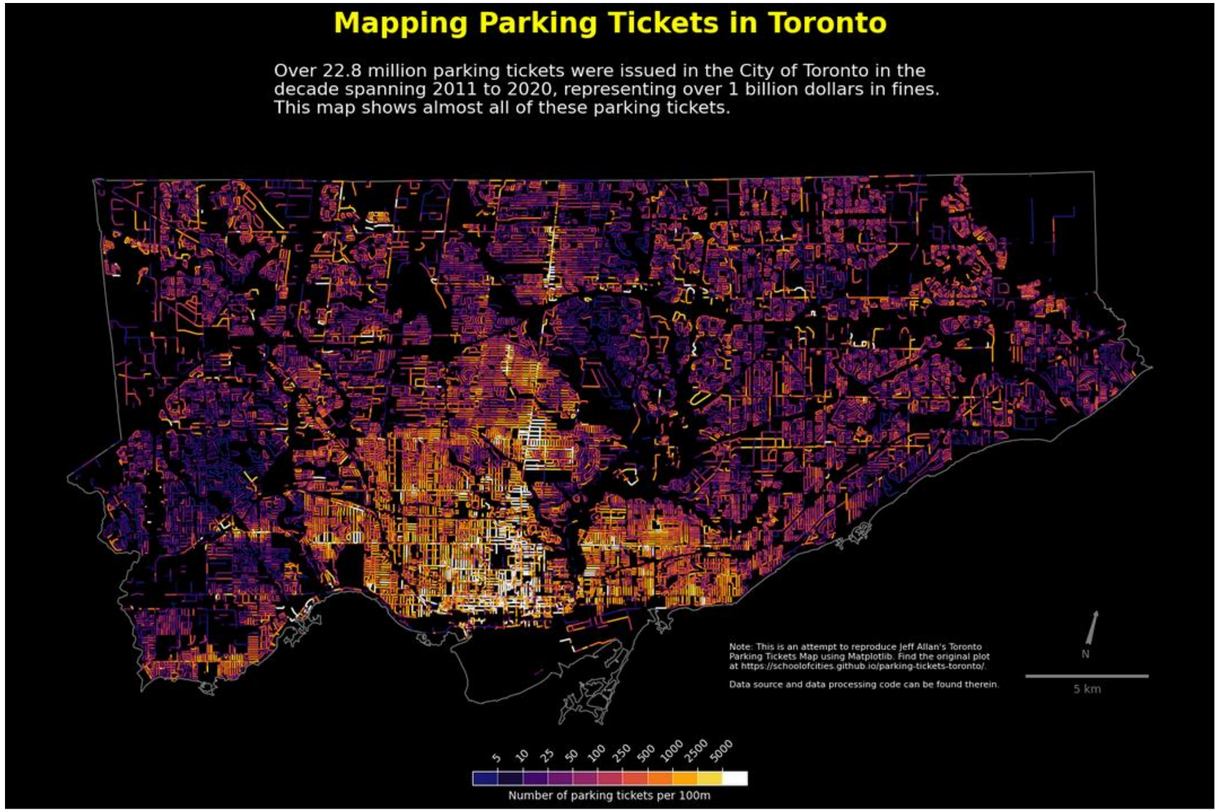
Parking Tickets in Toronto by Jeff Alan

Ref: 1) https://schoolofcities.github.io/parking-tickets-toronto/
2) https://github.com/schoolofcities/parking-tickets-toronto/tree/main

Walk-through – Our Implementation



the default plot from Geopandas/Matplotlib



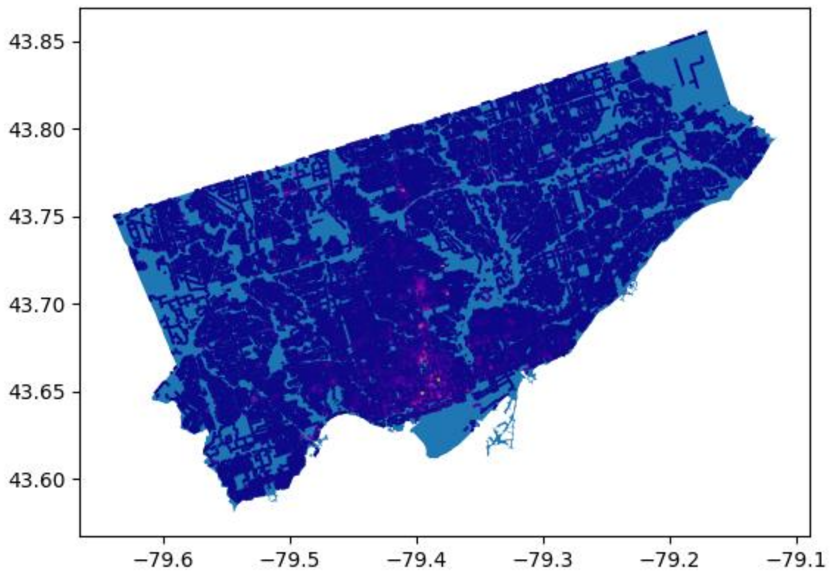
the refined plot

Default Plot

```
…

# draft a heatmap plot
fig, ax = plt.subplots()

boundary.plot(ax=ax)
centreline.plot(ax=ax, column='count_all', cmap='plasma')

…
```

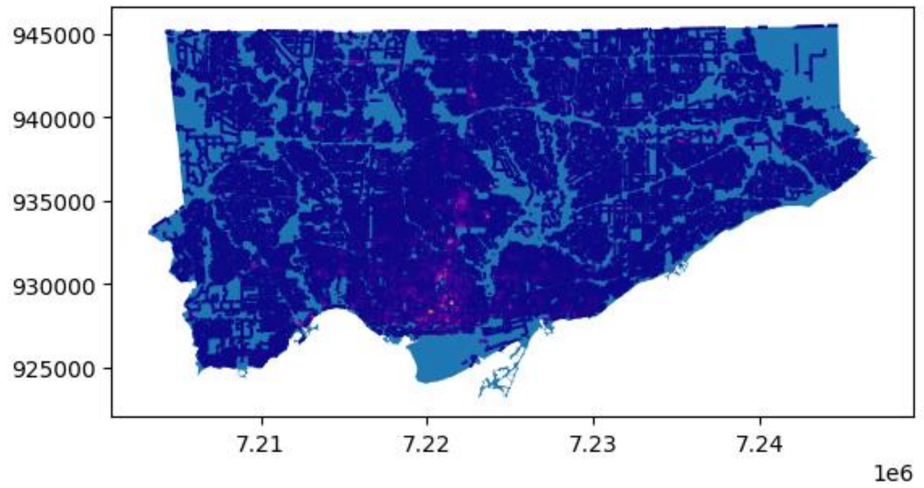

```
centreline.head()
```

	CENTRELINE_ID	FEATURE_CODE_DESC	geometry	count_all
0	914600	Local	MULTILINESTRING ((-79.50875 43.59744, -79.5098...	8.0
1	914601	Local	MULTILINESTRING ((-79.50987 43.59720, -79.5103...	355.0
2	7862398	Local	MULTILINESTRING ((-79.51087 43.59697, -79.5113...	708.0
3	914587	Major Arterial	MULTILINESTRING ((-79.51805 43.59795, -79.5191...	205.0
4	6735911	Major Arterial	MULTILINESTRING ((-79.51914 43.59770, -79.5202...	325.0



Ref: 1) https://geopandas.org/en/stable/docs/reference/api/geopandas.GeoDataFrame.plot.html
2) https://matplotlib.org/stable/users/explain/colors/colormaps.html

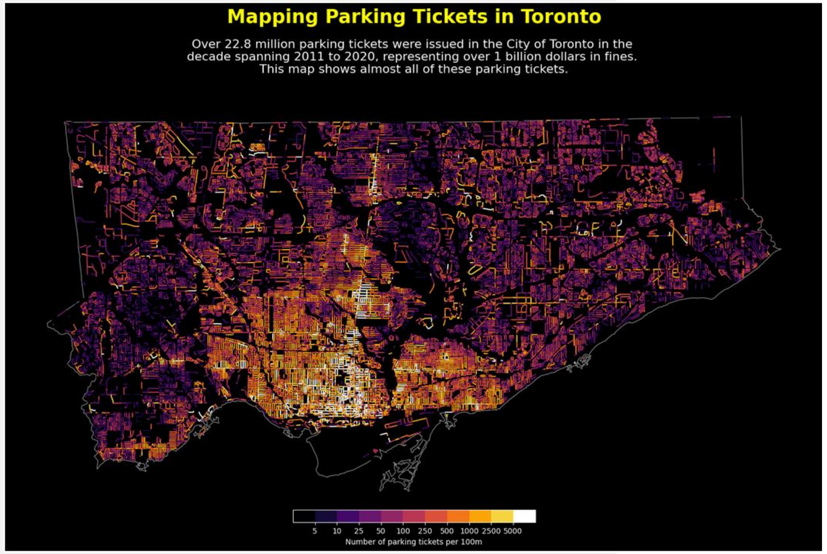
Refine 1. Better Orientation

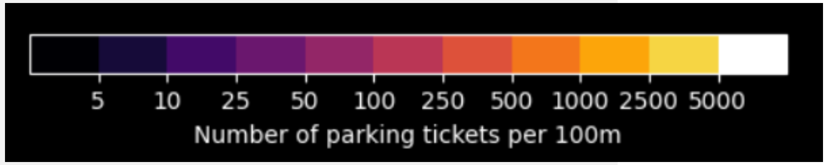



```
# rotate centreline
# turn EPSG:4326 to EPSG:3347 first to avoid shape
# distortion after rotation
cline_3347 = centreline.to_crs(epsg=3347)

# rotate with respect to the centroid of all centrelines
cline_3347_rotated = cline_3347.rotate(-28, origin=cline_3347.unary_union.centroid)
    .rename("geometry_3347_rotate")

# combine the original centreline GeoDataFrame with the rotated GeoSeries
centreline_rotated = centreline.join(cline_3347_rotated)

…

# rotate boundary
# turn EPSG:4326 to EPSG:3347 first to avoid shape distortion after rotation
# rotate with respect to the centroid of all centrelines to match centreline rotation centroid
boundary_rotated = boundary.to_crs(epsg=3347).rotate(-28, origin=cline_3347.unary_union.centroid)

…
```

Refine 2. Colormap on Discrete Intervals

```python
…

# Generate a colormap index based on discrete intervals
# https://matplotlib.org/stable/api/_as_gen/matplotlib.colors.Colormap.html
# https://matplotlib.org/stable/api/_as_gen/matplotlib.colors.BoundaryNorm.html
cmap = plt.colormaps['inferno'].with_extremes(over="white")
bounds = [5, 10, 25, 50, 100, 250, 500, 1000, 2500, 5000]
norm = BoundaryNorm(bounds, cmap.N, extend='both')

# plot centreline heatmap
centreline_rotated.plot(ax=ax, column='count_all',
                        cmap=cmap,
                        norm=norm,
                        markersize=0.5,
                        legend=True,
                        legend_kwds={
                            'shrink': 0.3,
                            'orientation': 'horizontal',
                            'pad': 0,
                            'anchor': (0.5, 1),
                            'extendfrac': 'auto',
                            'extendrect': True,
                            'label': 'Number of parking tickets per 100m'})
```



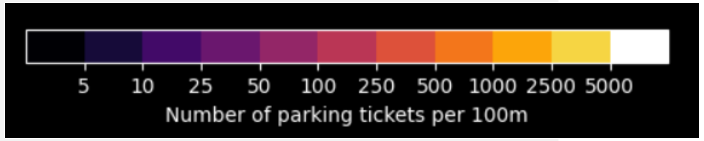


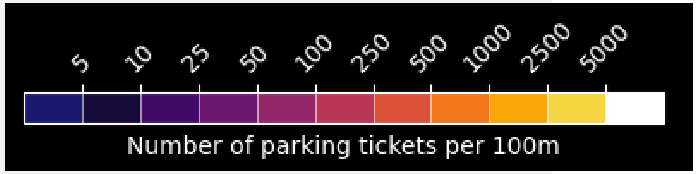
Ref: 1) Discrete and extended colorbar with continuous colorscale

2) In general, you can use mapclassify to auto-generate classification schemes for choropleth maps.

Refine 3. Better Colorbar

```
…

# plot coloarbar legend separately so as to customize its look
cbar = fig.colorbar(ScalarMappable(norm=norm, cmap=cmap),
                    ax=ax,
                    orientation='horizontal',
                    shrink=0.3,
                    pad=-0.02,
                    anchor=(0.5, 1),
                    extendfrac='auto',
                    extendrect=True,
                    drawedges=True,
                    label='Number of parking tickets per 100m')

cbar.ax.tick_params('x',
                    bottom=False, labelbottom=False,
                    top=True, labeltop=True,
                    labelrotation=45)

…
```





Refine 4. Map Scale, North Arrow, & Notes

```
# add scalebar
# https://geopandas.org/en/stable/gallery/matplotlib_scalebar.html
scale = ScaleBar(dx=1,
                 location='lower right',
                 color='grey',
                 box_alpha=0,
                 width_fraction=0.005,
                 border_pad=5)

_ = ax.add_artist(scale)

# add north arrow
# https://matplotlib.org/stable/users/explain/text/annotations.html
_ = ax.annotate("N",
                xy=(0.91, 0.25), xycoords='figure fraction',
                xytext=(0.9, 0.19), textcoords='figure fraction',
                ha='center',
                color='gray',
                arrowprops=dict(arrowstyle="fancy", color="gray"))

# add notes
_ = ax.text(0.6, 0.11,
            ("Note: This is an attempt to reproduce Jeff Allan's Toronto \nParking Tickets Map using Matplotlib. "
             "Find the original plot \nat https://schoolofcities.github.io/parking-tickets-toronto/.\n\n"
             "Data source and data processing code can be found therein."),
            transform=ax.transAxes,
            wrap=True,
            fontsize=8,
            horizontalalignment='left',
            bbox=dict(boxstyle='square', pad=1, facecolor='black', edgecolor='black'))

…
```