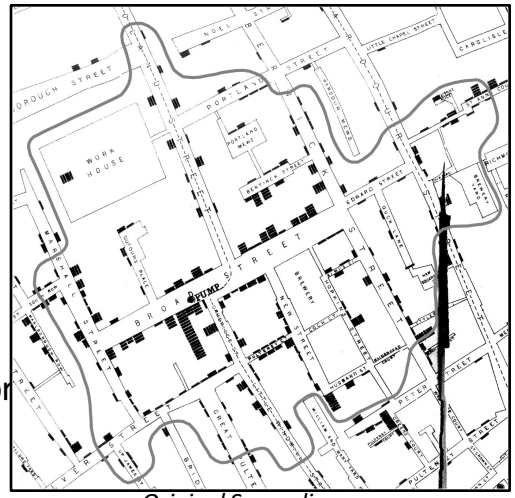
Geospatial Voronoi Analysis with Python

Dan Finkel PhD
Service Management Group (SMG)
Boston Python Meetup
December 20 2016

1854 Broad Street Cholera Outbreak

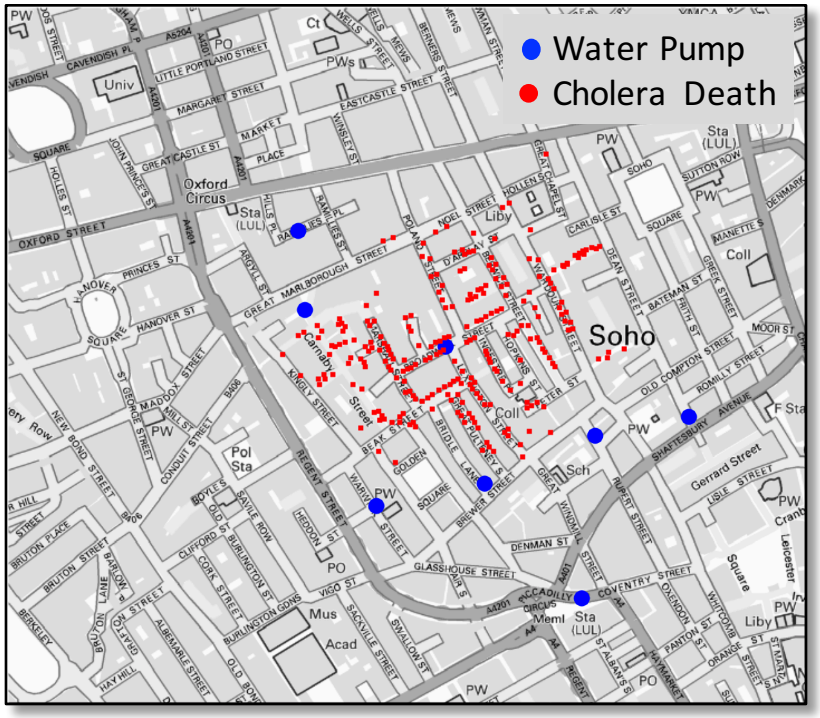

John Snow Memorial

- A major cholera outbreak killed over 600 London Soho residents and caused 75% of residents to flee the area
- A local physician, John Snow, connected the outbreak to filth polluting the water
  - Conclusions were based on graph and pattern analysis
  - Regarded as the founding event of epidemiology
- In the absence of hard biological evidence Snow relied on graph and pattern analysis
- Postscript
  - Snow's evidence caused Broad St well to be disabled
  - After the disease subsided Snow's theory was rejected and air pollution was blamed
  - "To accept Snow's theory of oral-foecal transmission of disease was too unpleasant for most of the public to contemplate"



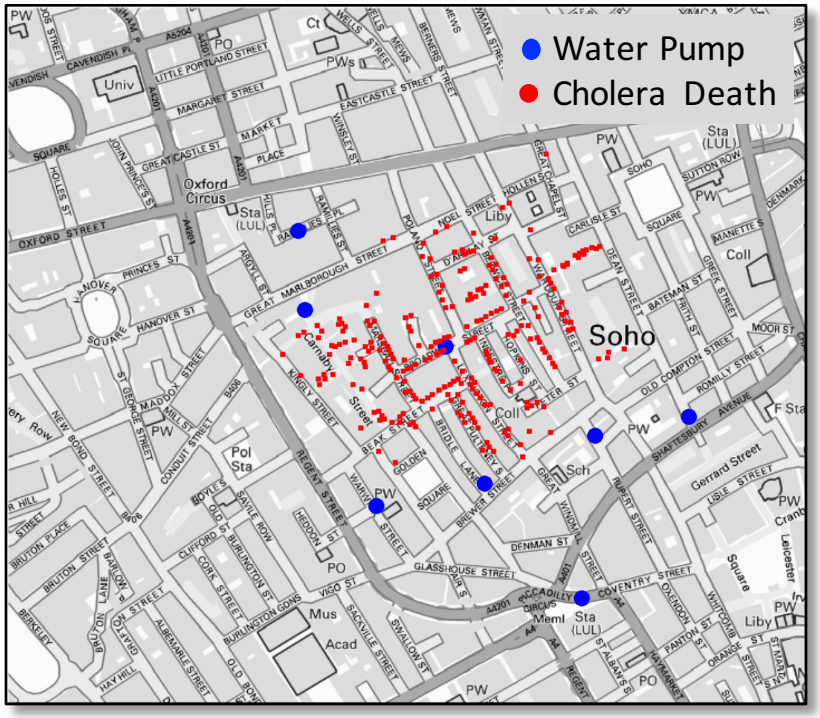
Original Snow diagrams

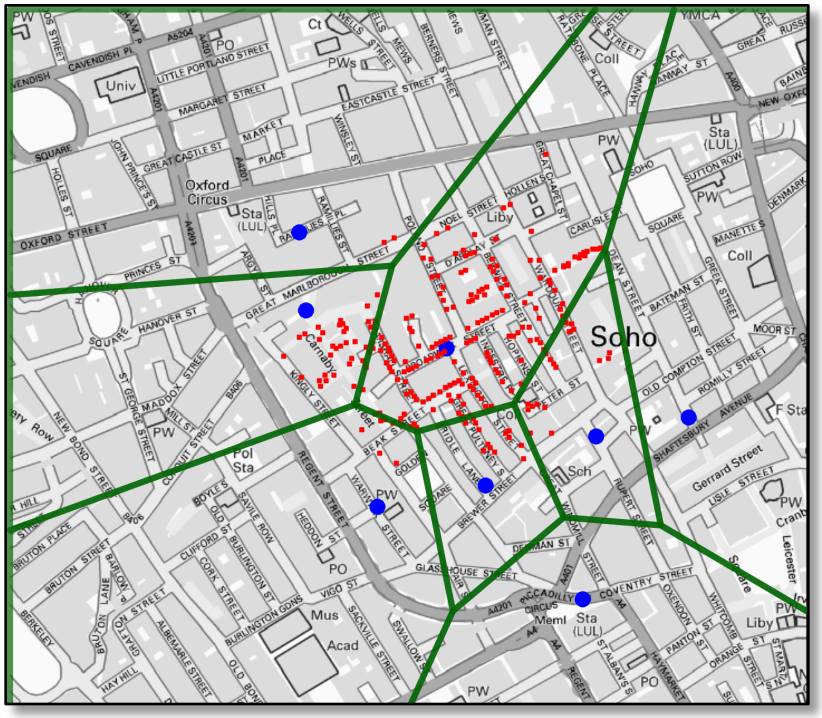
Visualizing the Cholera Outbreak by Water Pump Region



Data obtained from http://blog.rtwilson.com/john-snows-famous-cholera-analysis-data-in-modern-gis-formats/

Visualizing the Cholera Outbreak by Water Pump Region





Planar partitions by pump proximity helped Snow argue outbreak was due to polluted water

Data obtained from http://blog.rtwilson.com/john-snows-famous-cholera-analysis-data-in-modern-gis-formats/

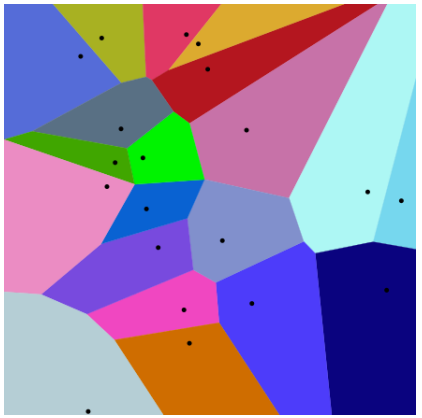
Voronoi Diagram Overview

Georgy Voronoy
1868 - 1908

- A Voronoi diagram is a planar partition based on distance to a set of points ("seeds") within the plane
- Each seed defines a partition as those points in the plane closer to that seed than any other

$$R_k = \{x \in X \mid d(x, P_k) \leq d(x, P_j) \text{ for all } j \neq k\}$$

- Algorithms / implementations
  - Typically O(n log n)
  - Can scale to multiple dimensions
  - Python implementation in scipy
    - Utilizes Qhull library
    - Computes via it's dual (Delaunay triangulation)



Voronoi Diagram Example (Wikipedia)

Constructing Voronoi Polygons in Python

Simple Functionality

1. Import the library

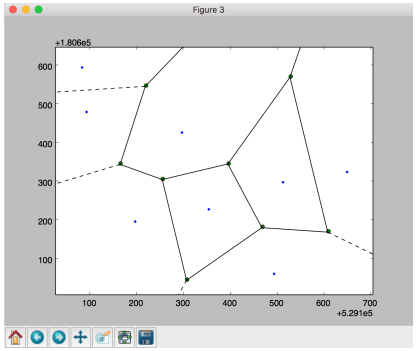
```
> from scipy.spatial import Voronoi, voronoi_plot_2d
```

2. Instantiate

```
> vor = Voronoi(pumps_xy)
```

3. Plot / Visualize

```
> voronoi_plot_2d(v)
```



Constructing Voronoi Polygons in Python

Simple Functionality

1. Import the library

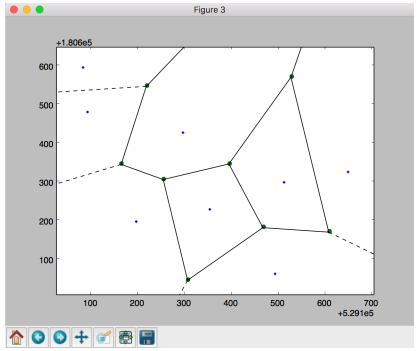
```
> from scipy.spatial import Voronoi, voronoi_plot_2d
```

2. Instantiate

```
> vor = Voronoi(pumps_xy)
```

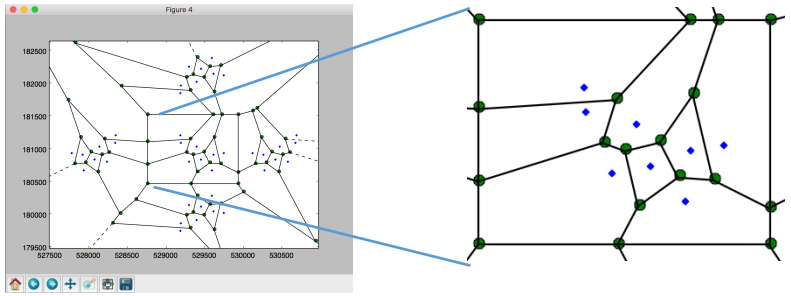
3. Plot / Visualize

```
> voronoi_plot_2d(v)
```



Additional Modifications

Mirror pumps and build Voronoi off expanded set (to create hard boundaries)



Convert Voronoi regions into polygons

```
def build_vor_polys(vor, soho_poly, pumps_xy):
    polys = []

    # Build all lines in the voronoi region set
    lines = [shapely.geometry.LineString(vor.vertices[line])
             for line in vor.ridge_vertices
             if -1 not in line]

    # polygonize the lines and check if
    # they are original or copies
    for poly in shapely.ops.polygonize(lines):
        if is_in(poly, pumps_xy):
            poly = poly.intersection(soho_poly)
            x, y = poly.exterior.xy
            polys.append(poly)
    return polys
```

Shapely Geospatial functions

- Compute area
- Intersect, overlap with other polygons
- Join / union with other objects

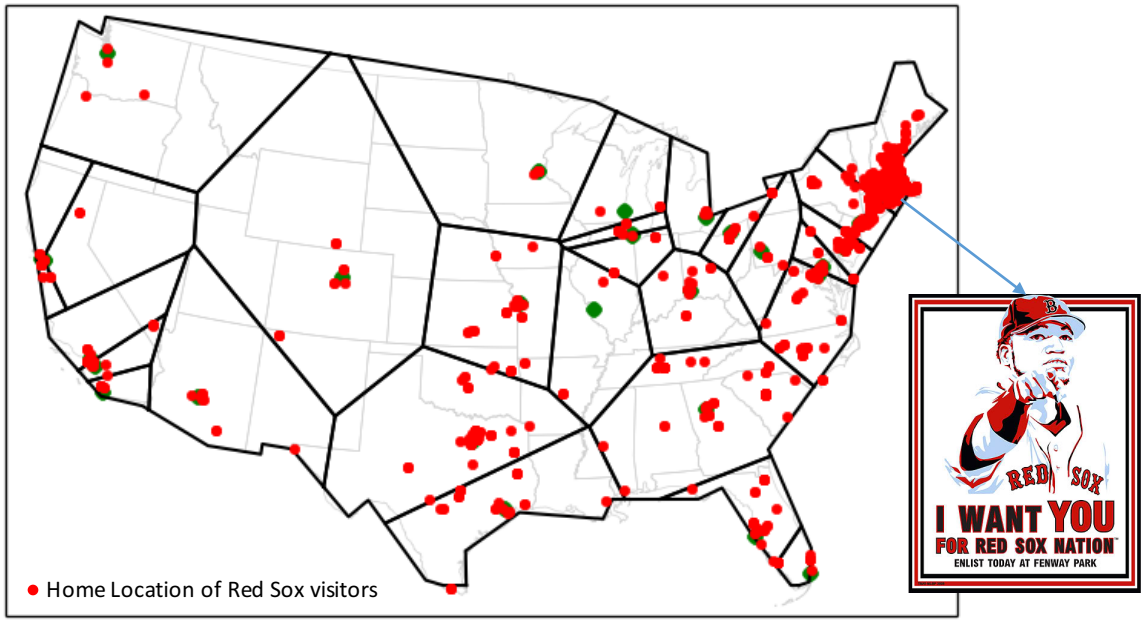
All code available on http://github.com/danfinkel/python

MLB Stadium Market Research Example

- SMG maintains a large panel of individuals who report behavior and customer experiences
- During 2016 MLB season we monitored stadium visits to identify geospatial patterns and preferences

smg service management group®

Behavioral Analysis
Red Sox Nation



Boston Red Sox Stadium Visitors Live in Many Different Parts of the US

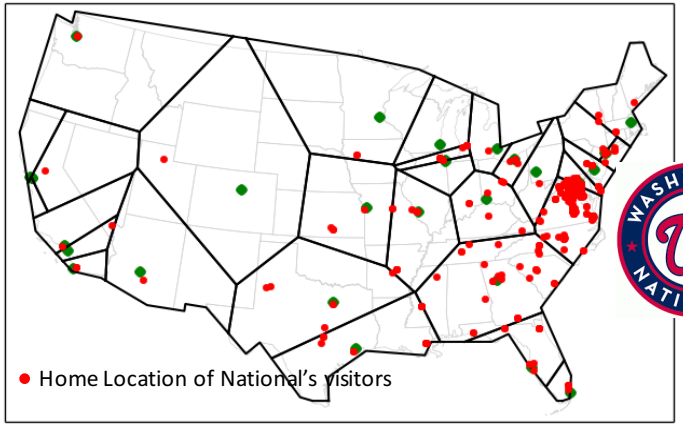
SMG Panelists Visit to Voronoi "Home" Team (2016)

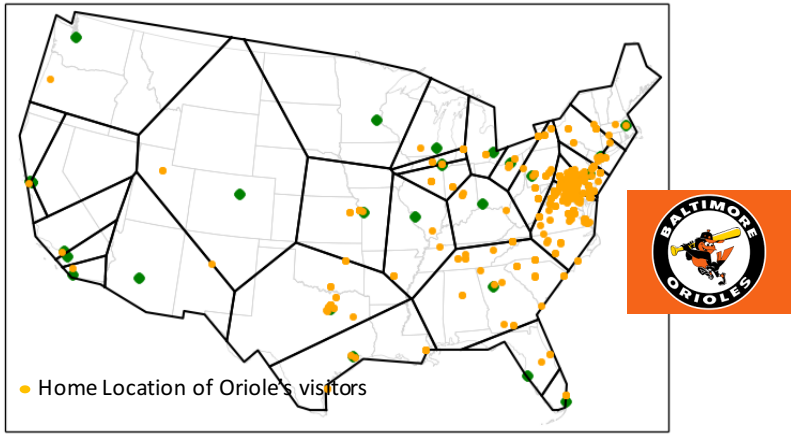
Stadium	Home Ratio	Away Ratio	Rival Ratio
Nationals Park	80.56%	15.10%	4.34%
Citizens Bank Park	83.74%	16.26%	0.00%
Citi Field	63.02%	18.63%	18.35%
Safeco Field	79.41%	20.59%	0.00%
AT&T Park	20.98%	33.26%	45.76%
Tropicana Field	82.35%	17.65%	0.00%
PETCO Park	62.79%	37.21%	0.00%
Great American Ball Park	88.94%	11.06%	0.00%
U.S. Cellular Field	59.51%	13.59%	26.90%
Comerica Park	84.26%	15.74%	0.00%
Chase Field	85.46%	14.54%	0.00%
Fenway Park	62.82%	37.18%	0.00%
Progressive Field	82.36%	17.64%	0.00%
Busch Stadium	79.67%	20.33%	0.00%
Minute Maid Park	88.64%	11.36%	0.00%
Globe Life Park in Arlington	94.28%	5.72%	0.00%
Camden Yards	56.89%	20.38%	22.72%
Angel Stadium of Anaheim	69.47%	15.09%	15.44%
PNC Park	83.82%	16.18%	0.00%
Target Field	90.75%	9.25%	0.00%
Turner Field	90.70%	9.30%	0.00%
Wrigley Field	35.34%	29.33%	35.34%
Kauffman Stadium	93.11%	6.89%	0.00%
Yankee Stadium	64.24%	16.19%	19.57%
O.co Coliseum	60.10%	30.39%	9.51%
Miller Park	81.01%	18.99%	0.00%
Coors Field	82.99%	17.01%	0.00%
Marlins Park	0.97%	99.03%	0.0
Dodger Stadium	66.47%	12.60%	20.93%

Mapping Error

Behavioral Analysis

Rivals Compete for Guests





Stadium	Home Ratio	Away Ratio	Rival Ratio
Nationals Park	80.56%	15.10%	4.34%
Citizens Bank Park	83.74%	16.26%	0.00%
Citi Field	63.02%	18.63%	18.35%
Safeco Field	79.41%	20.59%	0.00%
AT&T Park	20.98%	33.26%	45.76%
Tropicana Field	82.35%	17.65%	0.00%
PETCO Park	62.79%	37.21%	0.00%
Great American Ball Park	88.94%	11.06%	0.00%
U.S. Cellular Field	59.51%	13.59%	26.90%
Comerica Park	84.26%	15.74%	0.00%
Chase Field	85.46%	14.54%	0.00%
Fenway Park	62.82%	37.18%	0.00%
Progressive Field	82.36%	17.64%	0.00%
Busch Stadium	79.67%	20.33%	0.00%
Minute Maid Park	88.64%	11.36%	0.00%
Globe Life Park in Arlington	94.28%	5.72%	0.00%
Camden Yards	56.89%	20.38%	22.72%
Angel Stadium of Anaheim	69.47%	15.09%	15.44%
PNC Park	83.82%	16.18%	0.00%
Target Field	90.75%	9.25%	0.00%
Turner Field	90.70%	9.30%	0.00%
Wrigley Field	35.34%	29.33%	35.34%
Kauffman Stadium	93.11%	6.89%	0.00%
Yankee Stadium	64.24%	16.19%	19.57%
O.co Coliseum	60.10%	30.39%	9.51%
Miller Park	81.01%	18.99%	0.00%
Coors Field	82.99%	17.01%	0.00%
Marlins Park	~~0.97%~~	~~99.03%~~	~~0.00%~~
Dodger Stadium	66.47%	12.60%	20.93%

Asymmetry an opportunity for Nationals?

Mapping Error

Thanks!

Dan Finkel
Dan.finkel@gmail.com
http://github.com/danfinkel/python