
Your Ideal New York City Bench

By: Cindy Setiadi, Tala Berro, Vidya Madhavan, & Paolo Rivas

Where should I sit?



Wi-Fi



Drinking
Fountain



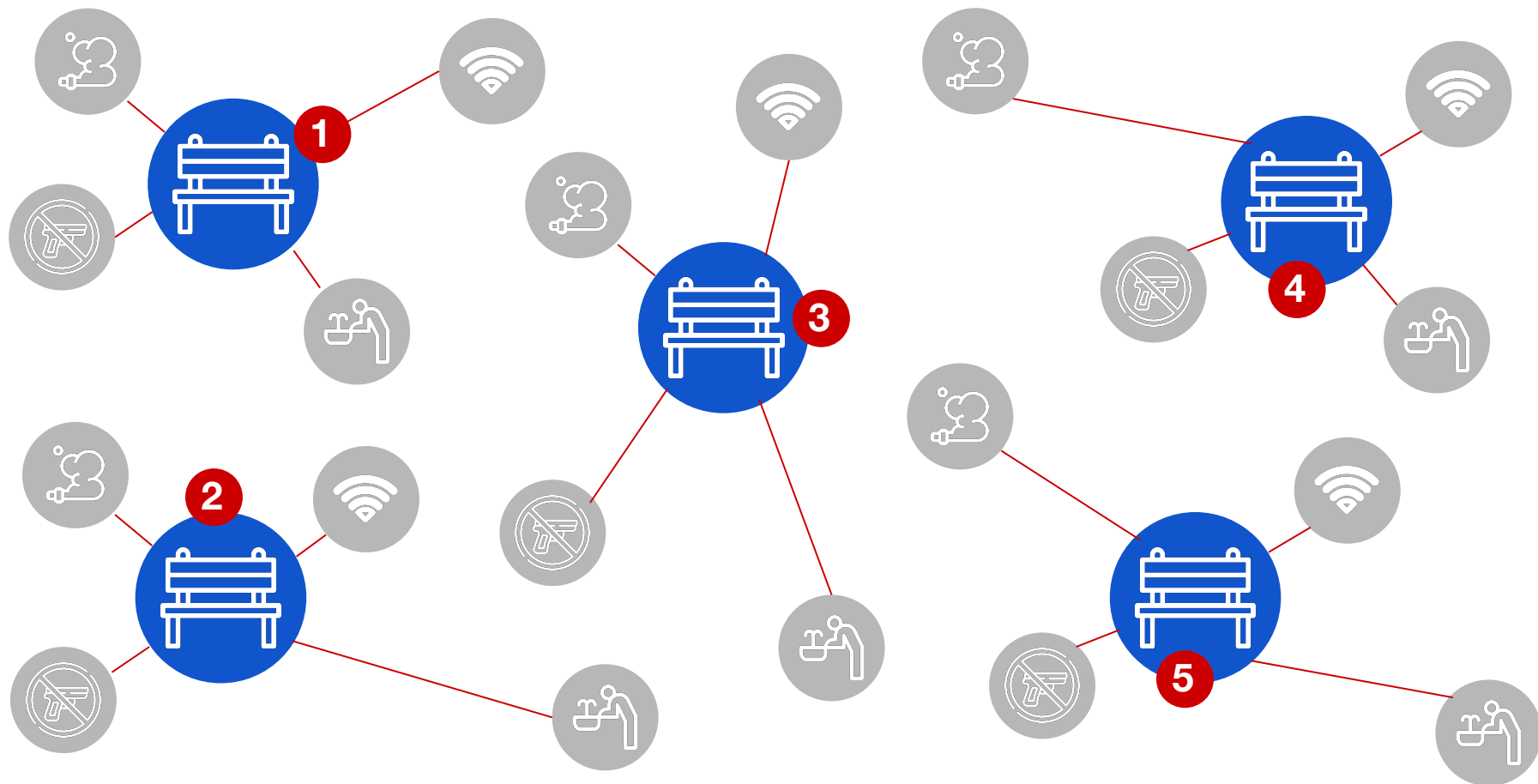
Air Quality



Crime History

Goal:

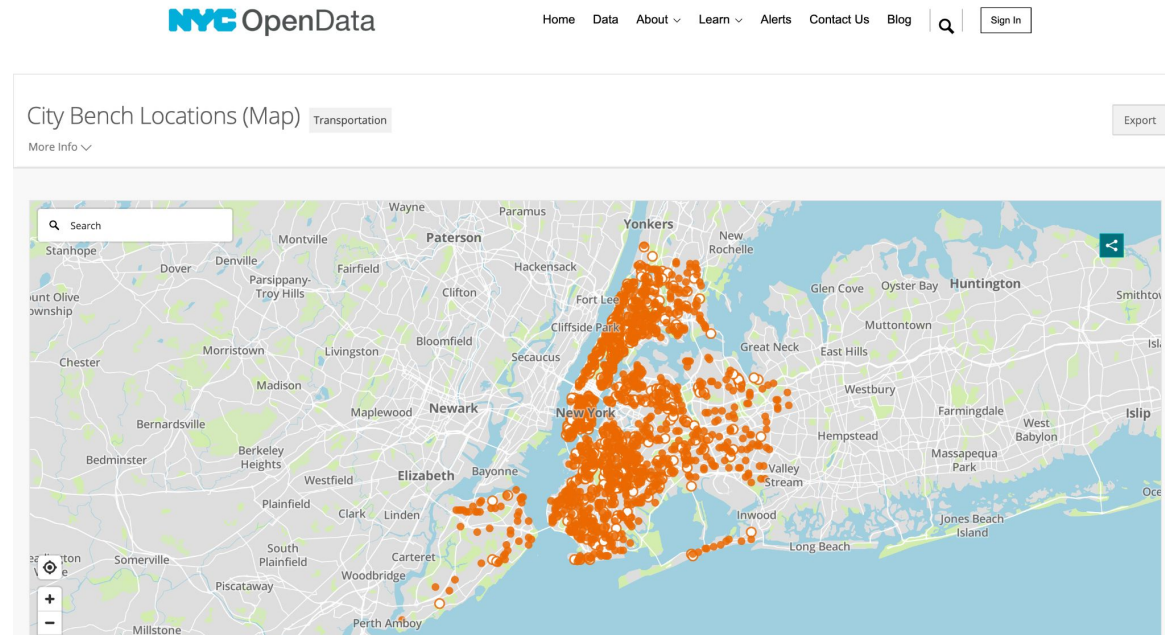
Use descriptive statistics and modeling to rank benches utilizing proximity algorithms.



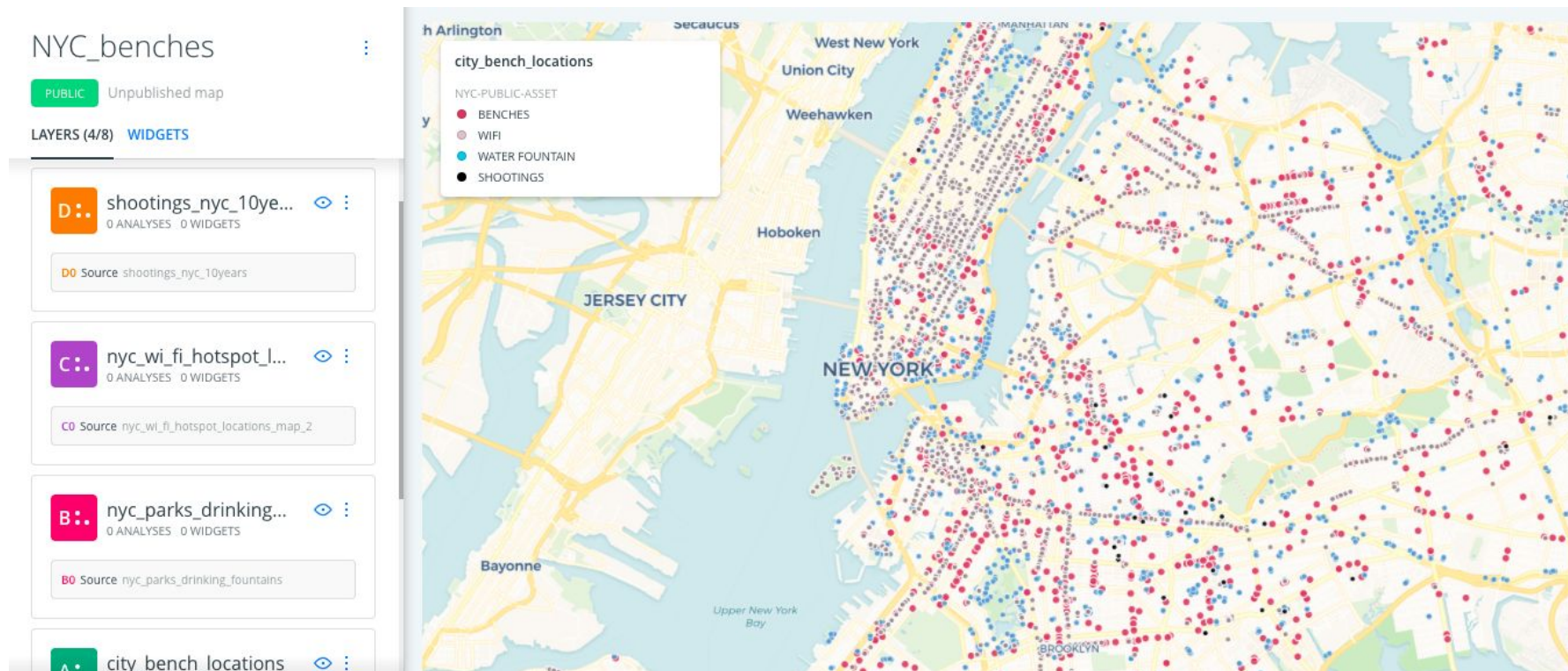
The source:

NYC Open Data


<https://data.cityofnewyork.us>



Collecting multiple layers of Data and Visualizing in a Map




Generate new data by:
1) calculating and creating clusters



☒ **Find Nearest**

Select points from second dataset nearest to the geometries in current layer.

[MORE INFO](#)



☐ **Calculate Clusters of Points**

Augment with cluster_no column to spatially separate points into a specified number of groups.

[MORE INFO](#)

2) Finding the **nearest data point** and **estimating the distance** between our target and associated variables.

Project Flow

1. Exporting Dataset based on attributes of interest

Bench ID	Water-Fountain	Wifi	Crime Scene	Borough AQI (0 - 500)
1	0.2	0.1	0.7	424
2	0.6	0.4	0.8	320
3	0.4	0.3	0.4	190
4	0.8	0.8	0.4	15
5	0.9	0.4	0.5	4
6	0.2	0.1	0.8	100

Example dataset, values*

2. Minimization/Maximization of attributes

We minimise the attributes wifi and water-fountain (distance) as it is preferable for them to be closer to the bench.

We maximise the attribute crime scene (distance) as we want the bench to be further from scenes of crime.

We minimise the attribute AQI (0-500) as we prefer the bench to be in a location with less air pollution.

3. Normalization of attributes within the same range

We normalize each attribute between the same range (0 -10) to overcome differences in scale and units. We also want the direction of goodness to be similar (irrespective of the attribute). Hence after normalization, values near maximum of range (say, 10) should mean that bench is good in that attribute and lower values (say, near 0) means they are bad. We do this with the following formulae:

Maximization

Let A be an attribute which we want to maximize, and its elements are: $[a_1 \ a_2 \ \dots \ a_n]$; $1 \leq i \leq n$

Then, $maximize(a_i) = maximizeFunc(a_i, A)$

$$\text{where, } maximizeFunc(a_i, A) = \begin{cases} \frac{a_i}{sum(A)}, & \text{sum normalization} \\ \frac{a_i}{max(A)}, & \text{max normalization} \\ \frac{a_i - min(A)}{max(A) - min(A)}, & \text{max-min scaling} \\ \dots & \end{cases}$$

Minimization

Let A be an attribute which we want to minimize, and its elements are: $[a_1 \ a_2 \ \dots \ a_n]$; $1 \leq i \leq n$

Then, $minimize(a_i) = minimizeFunc(a_i, A)$

$$\text{where, } minimizeFunc(a_i, A) = \begin{cases} \frac{1}{maximizeFunc(a_i, A)}, & \text{inverse} \\ 1 - maximizeFunc(a_i, A), & \text{subtract} \\ \dots & \end{cases}$$

normalization logic for maximizing and minimizing an attribute values

4. Implementation: Packages and algorithms

We load the dataset onto the **sci-kit criteria** package in python, which provides many algorithms for multi-criteria decision-making problems. We will choose the **WeightedSum algorithm** where the individual scores are combining by summing up.

This algorithm takes in 2 parameters:

mnorm - The value maximization logic (minimization is the inverse of the same)

wnorm - The weight normalization logic

```
criteria_data = Data(  
    nycparkbench_data.iloc[:, 1:],          # the pandas dataframe  
    [MAX, MAX, MIN, MAX, MIN],             # direction of goodness for each column  
    anames = nycparkbench_data['Bench ID'], # each entity's name, here Bench ID  
    cnames = nycparkbench_data.columns[1:], # attribute/column name  
    # weights=[1,1,1,1,1]                 # weights for each attribute (optional)  
)  
criteria_data
```

```
from skcriteria.madm import simple  
# weighted sum  
dm = simple.WeightedSum(mnorm="sum")  
dec = dm.decide(criteria_data)  
dec
```

5. Result and further application

The dataset generated gives us the ranking table of benches in NYC.

Weighted Sum (mnorm = sum, wnorm = sum)

Bench ID	Water-Fountain (min)	Wifi (min)	Crime Scene (max)	Borough AQI (0 - 500) (min)	Rank
1	0.2	0.1	0.7	424	2
2	0.6	0.4	0.8	320	6
3	0.4	0.3	0.4	190	4
4	0.8	0.8	0.4	15	3
5	0.9	0.4	0.5	4	1
6	0.2	0.1	0.8	100	5

Example dataset, values*

We can then extract the bench location and ranking as a separate dataset, to build clusters of benches by micro-locations. An input tool will allow users to choose a micro-location, and provide the output of **top 15 ranked** benches within that micro-location.

References:

Source data and data extraction

- [NYC open data](#)
- [Carto - Spatial Analysis Tool](#)

Datasets used

- [City Bench Locations Dataset](#)
- [Wifi Hotspot Dataset](#)
- [Wifi Hotspot Public Dataset](#)
- [Drinking Fountains Dataset](#)
- [Shootings Dataset NYC](#)

Implementation and Analysis Documentation

- [Resource on multi-criteria decision making using Python](#)
- [Github Repo and codebook for methods of multi-criteria decision making using the scikit criteria package](#)