

Tracking Citi Bike reliability & measuring service inequity using real-time data

Dan Levine

Office of NYC Comptroller Brad Lander

<https://github.com/NYCCComptroller/citi-bike-gbfs>

nycsodata24.sched.com #nycSOdata #opendataweek





NEW YORK CITY COMPTROLLER
BRAD LANDER

Tracking Citi Bike reliability & measuring service inequity using real-time data

School of Data Presentation

MARCH 2024

<https://github.com/NYCCComptroller/citi-bike-gbfs>

Agenda

- Key findings and recommendations
- Data
- Analysis methods
- Hands-on how-to



Problem: unreliable Citi Bike



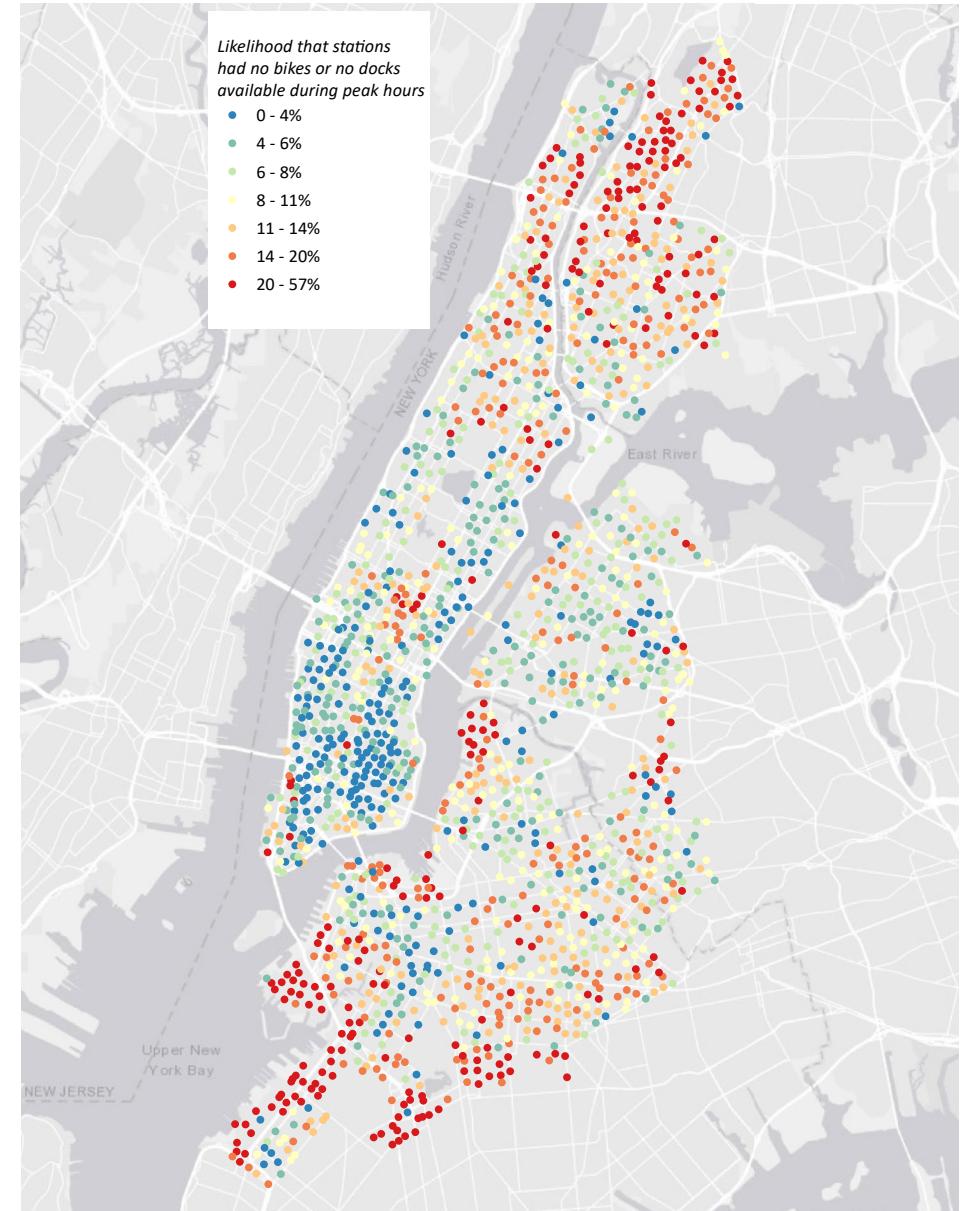
NEW YORK CITY COMPTROLLER BRAD LANDER

A photograph showing a large number of blue Citibikes parked in a long row at a city bike sharing station. The bikes are secured to grey metal posts. In the background, there's a modern building with glass windows and a set of stairs. The foreground shows the front wheel and handlebars of one of the bikes.

Key findings & policy recommendations

Key findings

- Citi Bike is **not consistently available** in all neighborhoods
- Least reliable service in areas home to **more Hispanic, Black, and low-income residents**
- While Citi Bike has **cut back** on rebalancing



Policy recommendations

- Create **neighborhood-level** performance standards
- Strengthen **enforcement** of basic performance standards
- Incentivize better performance by **offering payments or subsidies** when the operator consistently **exceeds** service standards
- **Improve transparency** through enhanced public reporting



A photograph showing a large number of blue Citibikes lined up at a docking station. The bikes are secured to grey metal posts. In the background, there's a modern building with a glass facade and some stairs. The foreground shows the front wheel and handlebars of one of the bikes.

HOW WE DID IT: Data

Needed more granular data



April 2022 Monthly Report

April 2022 Monthly Report

SLA 6 – Station Deactivation, De-Installation, Re-Installation, and Adjustment

Description: As directed by DOT, Citi Bike must perform: (i) Station Deactivation(s); (ii) Station De-Installation(s); (iii) Station Re-Installation(s); (iv) Station Adjustment(s). DOT will provide a minimum of 48 hours advance notice prior to any of the above, except in instances where the continued presence/activity of the Station has been determined to pose a threat to public safety. Deactivated Stations must be reactivated within 24 hours of direction from DOT. De-Installed or Adjusted Stations must be reinstalled or Readjusted to their original configurations within 72 hours of direction from DOT.

Performance Rate: 100%

SLA 7 – Snow Removal

Description: Following snow events, Citi Bike must remove snow within 12 hours so as to maintain:
(i) Parallel pedestrian clear path adjacent to Stations located on sidewalks and in plazas; and
(ii) Perpendicular pedestrian paths through Stations where gaps in Docks provide pedestrian access.

Performance Rate: 100%

SLA 8 – Program Functionality

Description: The Program is completely unavailable, such that no Program user can dock, undock, and Wrench Bicycles as intended, exclusive of planned Program outages for upgrades and maintenance as agreed upon by Citi Bike and DOT and Program outages caused by an Event of Force Majeure.

Performance Rate: 100%

SLA 9 – Bicycle Availability

Description: This Bicycle Availability requirement is met if the monthly average Bicycle Fleet Level, recorded once each Day of the month between the hours of 11:00 AM and 3:00 PM, meets or exceeds the required Bicycle Fleet Level.

Performance Rate: 100%

8

The Citi Bike program is operated by NYC Bike Share, LLC, a subsidiary of Lyft, Inc.



NEW YORK CITY COMPTROLLER BRAD LANDER

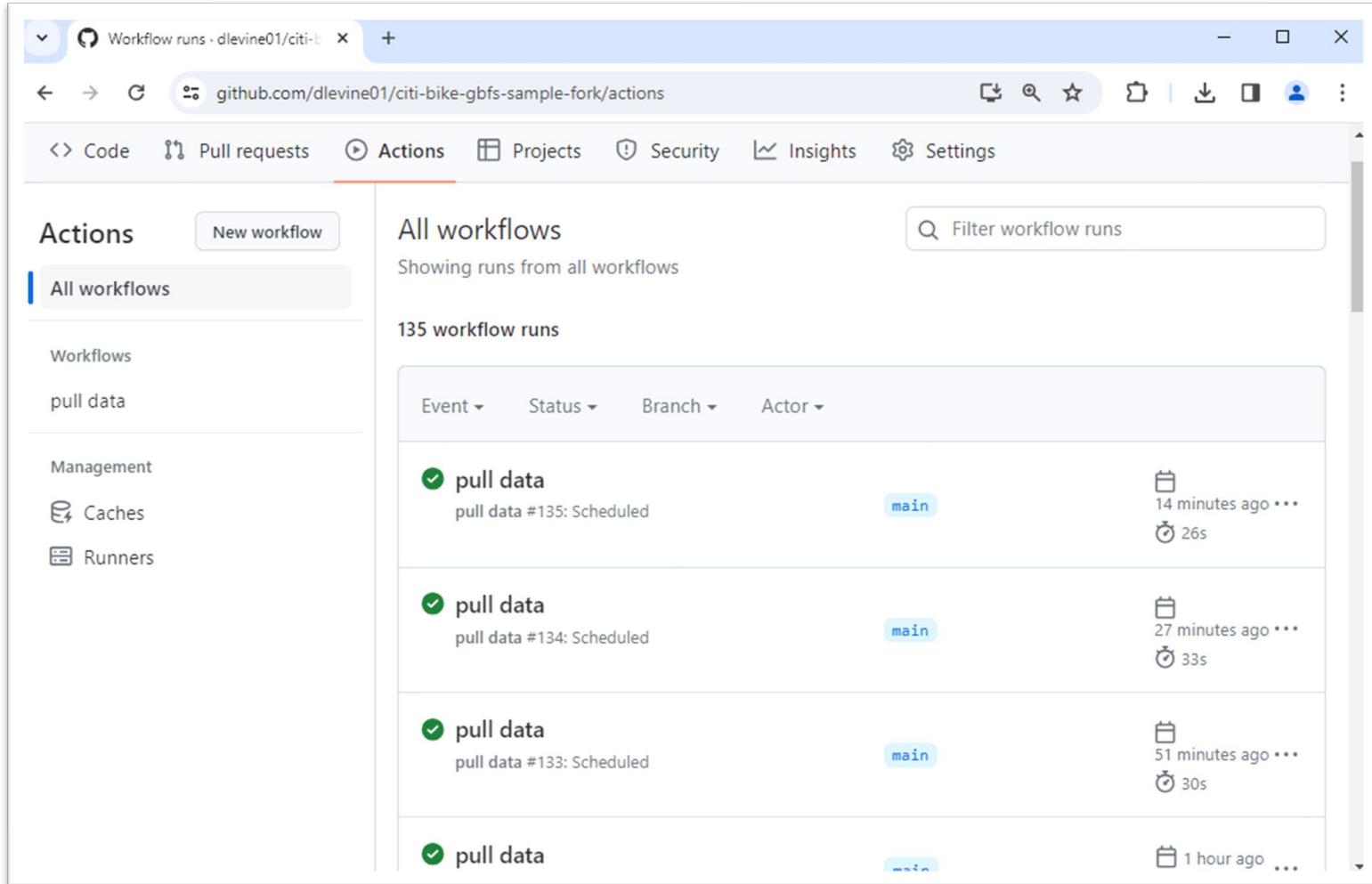
Real-time detail

Citi Bike GBFS

(General Bikeshare Feed Specification)



Recording live data



The screenshot shows a GitHub Actions interface for a repository named "citi-bike-gbfs-sample-fork". The "Actions" tab is selected in the navigation bar. On the left sidebar, under the "Actions" section, the "All workflows" tab is active. The main area displays a table of "135 workflow runs" for the "pull data" action. The table includes columns for Event, Status, Branch, and Actor. The first four rows of the table are as follows:

Event	Status	Branch	Actor	Timestamp	Duration
✓ pull data	Scheduled	main		14 minutes ago	... 26s
✓ pull data	Scheduled	main		27 minutes ago	... 33s
✓ pull data	Scheduled	main		51 minutes ago	... 30s
✓ pull data	In Progress	main		1 hour ago	...



Build detailed data

		capacity	is_renting	is_returning	num_docks_available	num_bikes_available	num_ebikes_available	num_bikes_disabled
last_updated	station_id							
2024-03-12 17:40:49	66dc2995-0aca-11e7-82f6-3863bb44ef7c	51	0	0	0	0	0	0
	06439006-11b6-44f0-8545-c9d39035f32a	48	0	0	0	0	0	0
	19d17911-1e4a-41fa-b62b-719aa0a6182e	39	0	0	0	0	0	0
	1861678548643203686	25	1	1	8	15	8	2
	cd2d9dab-7708-4685-a56f-9412c738de7e	23	1	1	1	22	1	0
	66db2a71-0aca-11e7-82f6-3863bb44ef7c	45	1	1	0	41	16	4
	901ad0c4-383e-490a-8b54-0656ce2358d6	19	1	1	3	11	2	5
	66dc292c-0aca-11e7-82f6-3863bb44ef7c	55	1	1	50	3	1	2
	4c03fa2d-89da-4f0b-8f19-c7de5edbe256	25	1	1	3	21	3	0
	9af90faf-0b9b-451b-9cb0-20ff421ca1d9	27	1	1	18	9	1	0



A photograph showing a large number of blue Citibikes lined up at a docking station. The bikes are secured to grey metal posts. The background shows a modern building with a glass and steel facade. A dark blue rectangular overlay contains the text.

HOW WE DID IT: Analysis

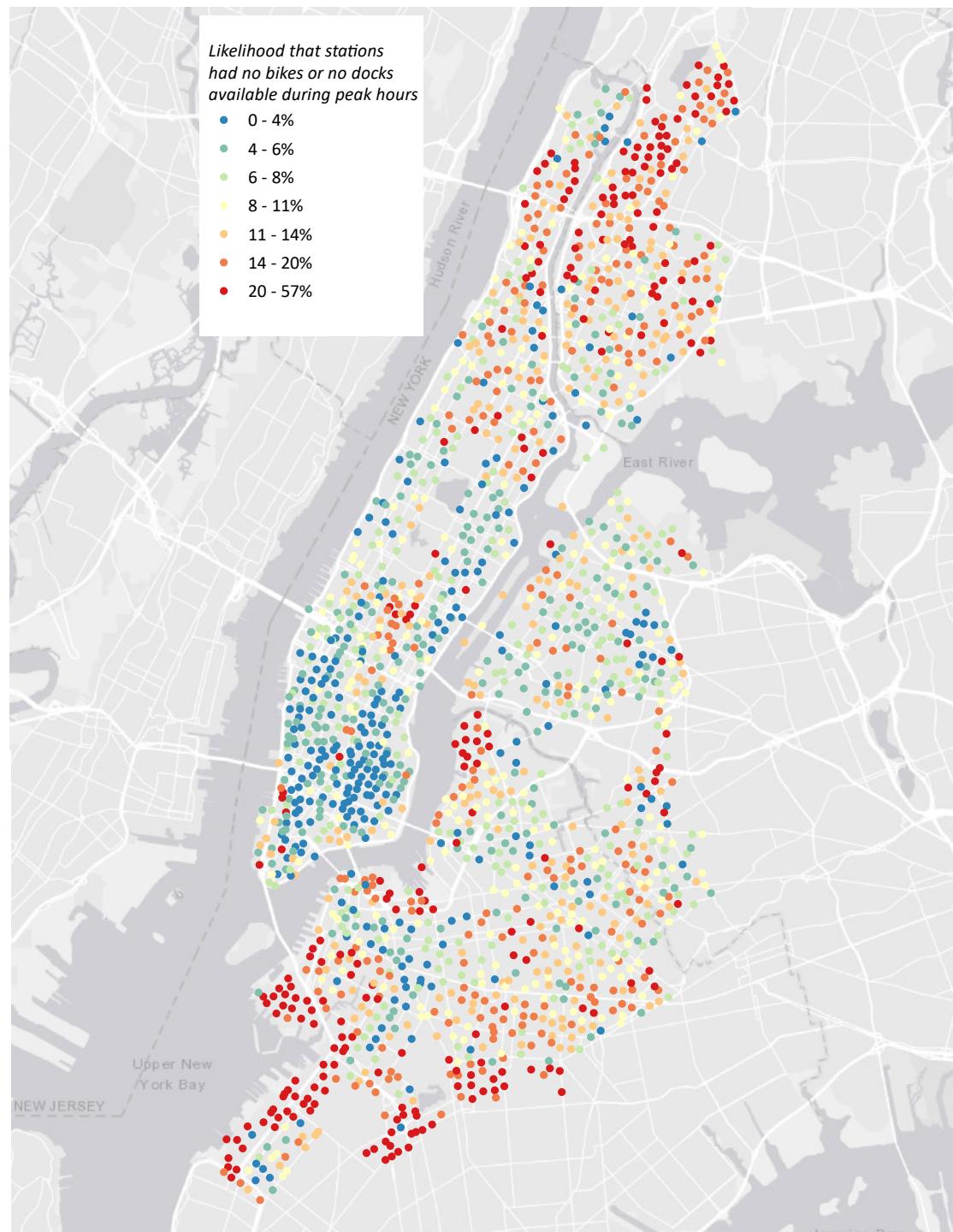
Compute service measures

- Frequency no docks or no bikes available
- Duration no docks or no bikes available
- Portion of docks with broken bikes



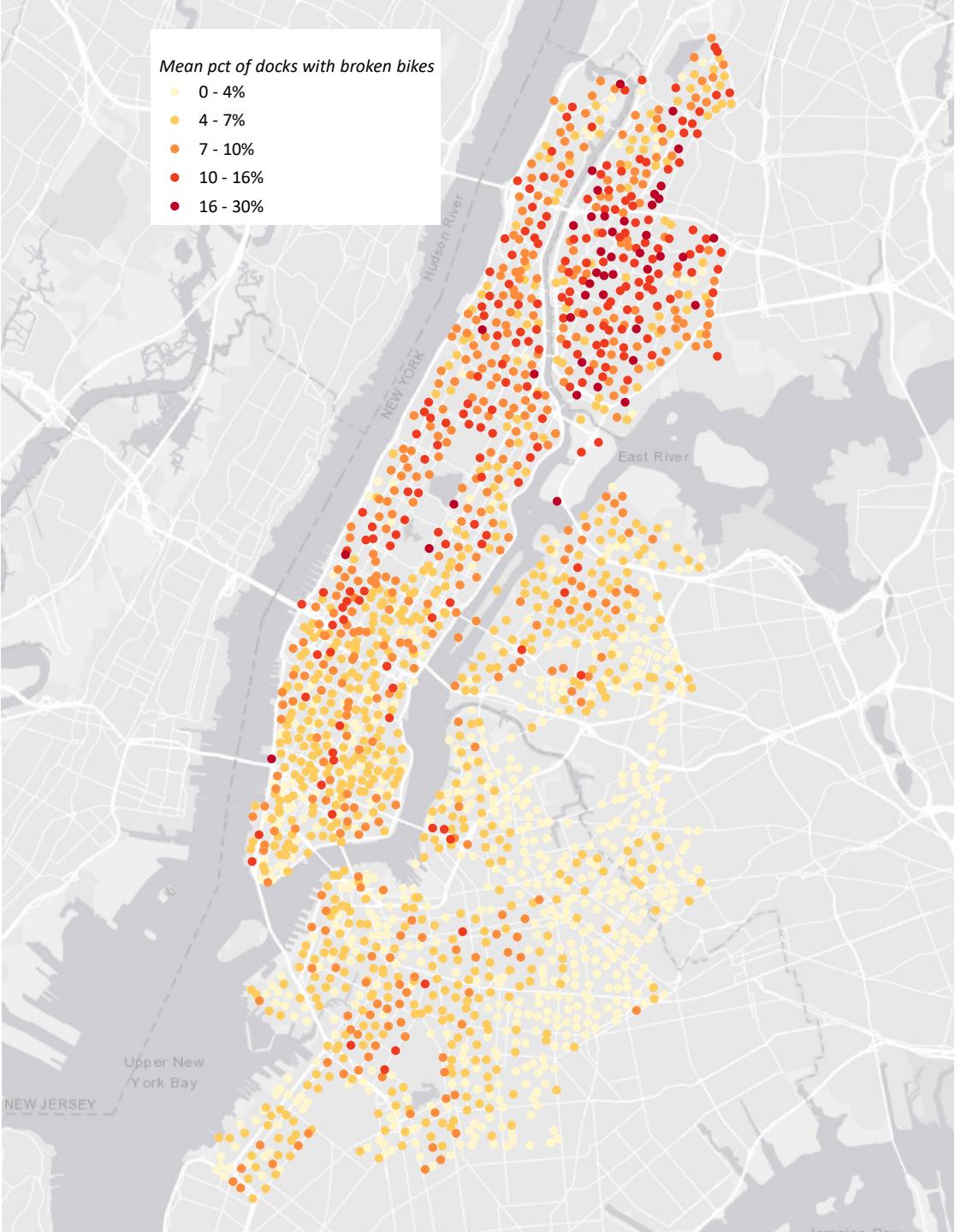
Explore the data:

Unequal service between neighborhoods



NEW YORK CITY COMPTROLLER BRAD LANDER

Bronx riders burdened with stations loaded with broken bikes



NEW YORK CITY COMPTROLLER BRAD LANDER

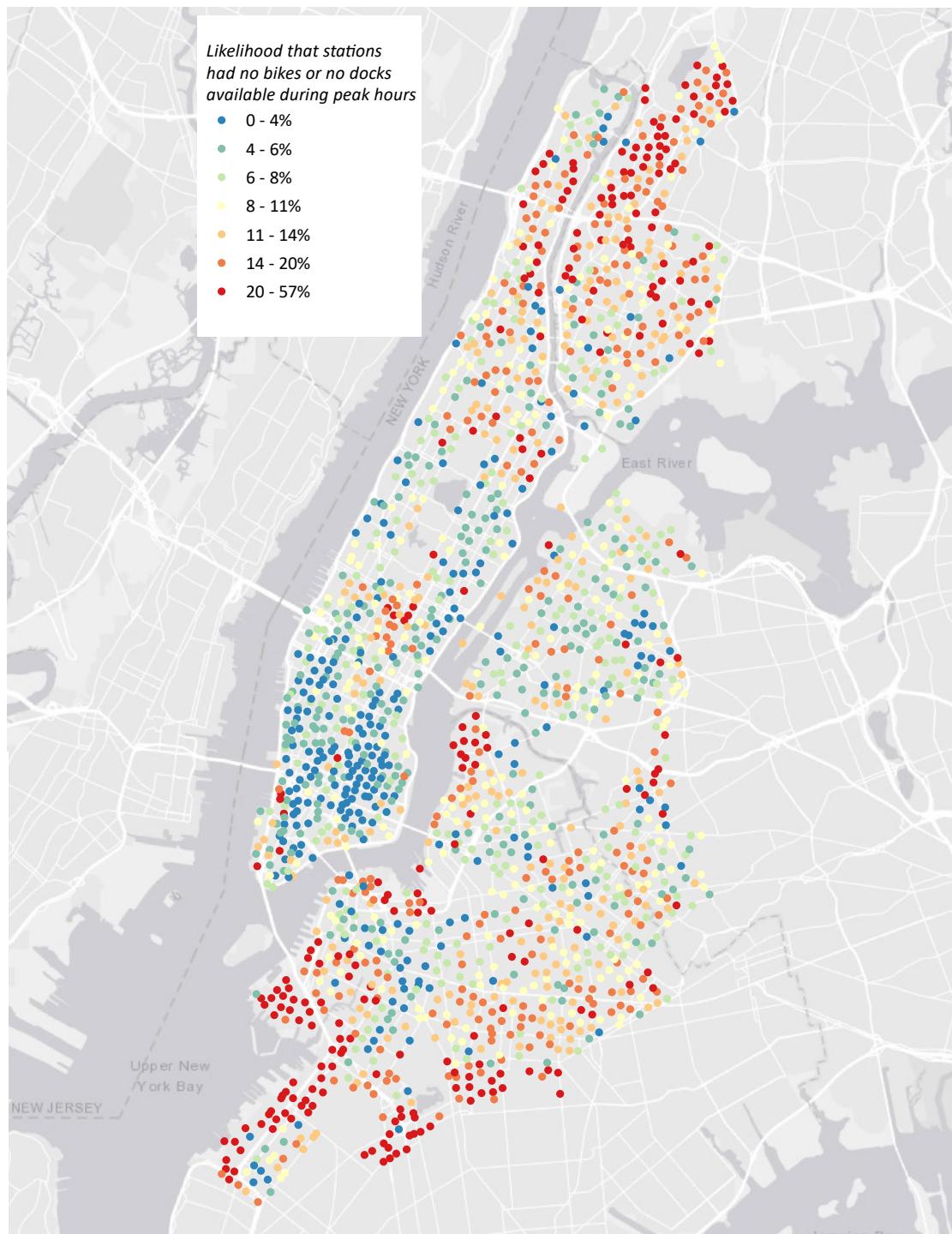
Failing to meet service standards

11,000
violations

\$812,000
in potential penalties



Unequal service between neighborhoods



NEW YORK CITY COMPTROLLER BRAD LANDER



HOW WE DID IT:
**Compute clusters of
poor service**

Find clusters

Local indicators of spatial association

a spatial method with statistical power



NEW YORK CITY COMPTROLLER BRAD LANDER

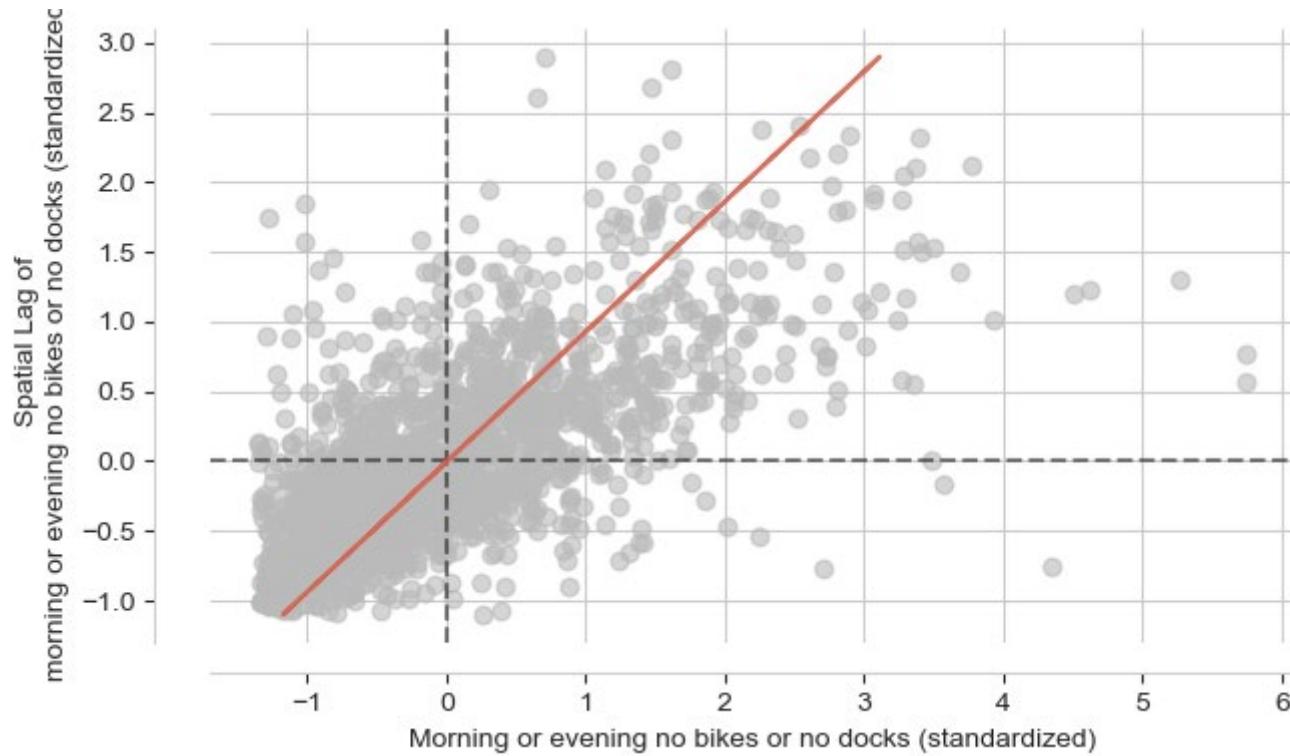
Neighbors

1. Compute weighted average of nearby stations



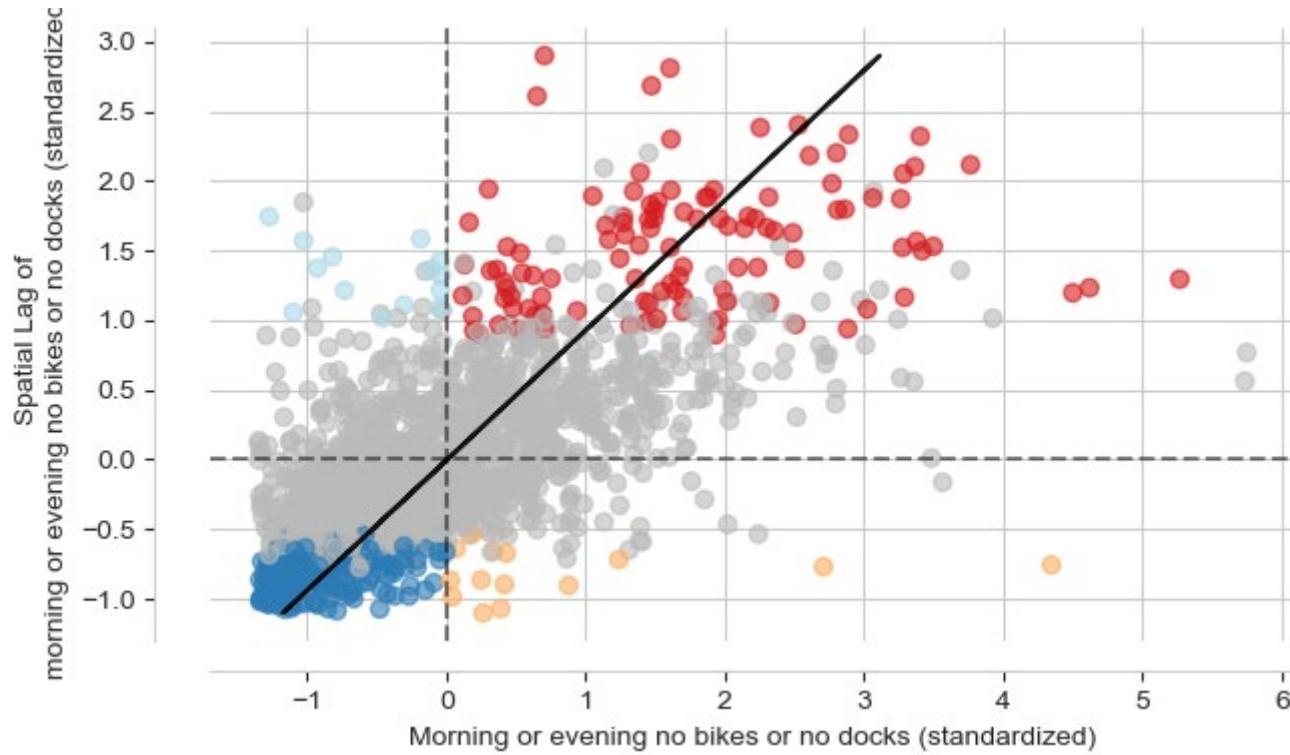
Local spatial autocorrelation

2. Compare station to its neighbors

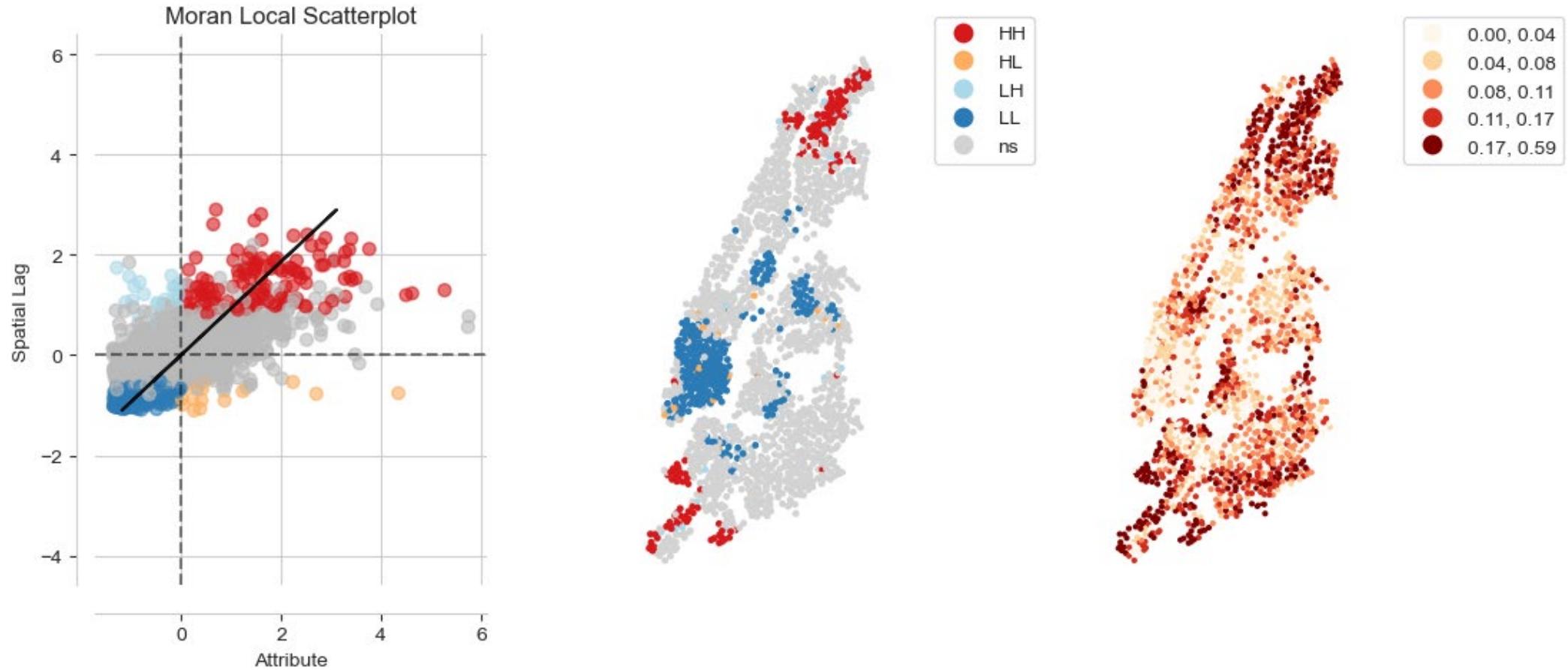


Local spatial autocorrelation

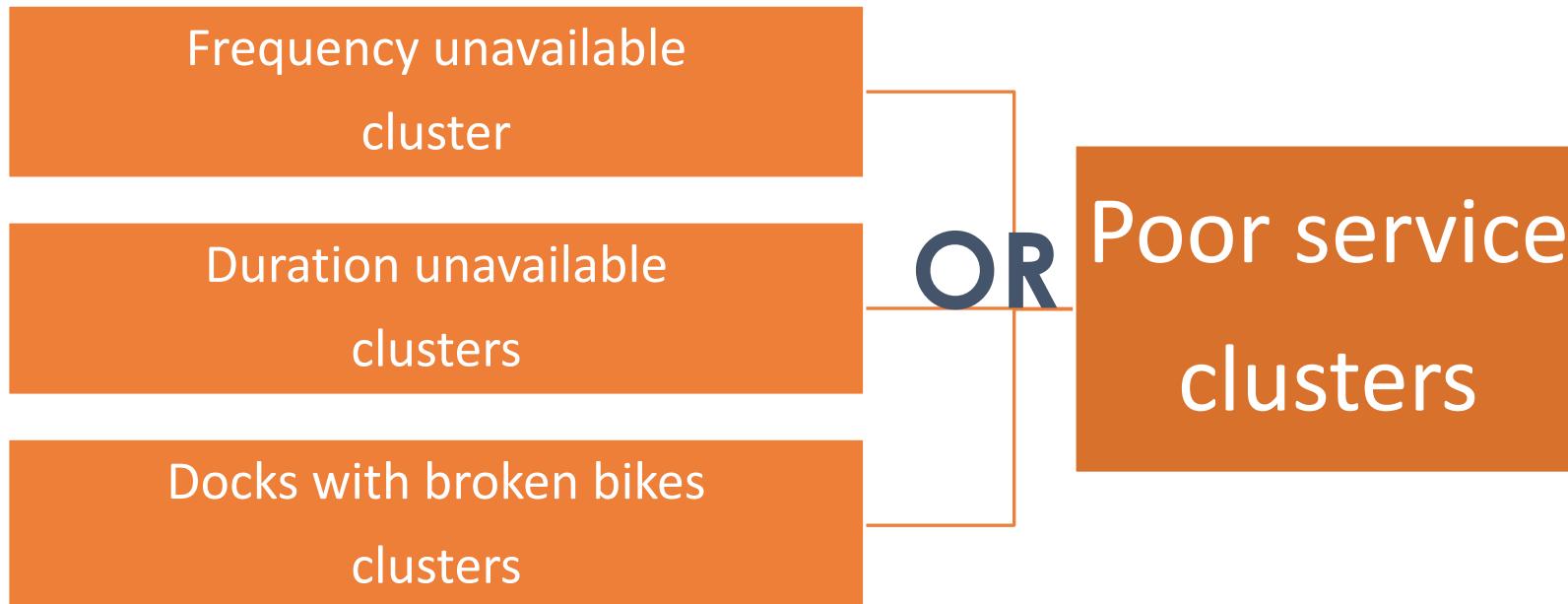
3. Shuffle
arrangement to
test unlikelihood
(*pseudo-*
significance)



Get a measure of significant hotspots

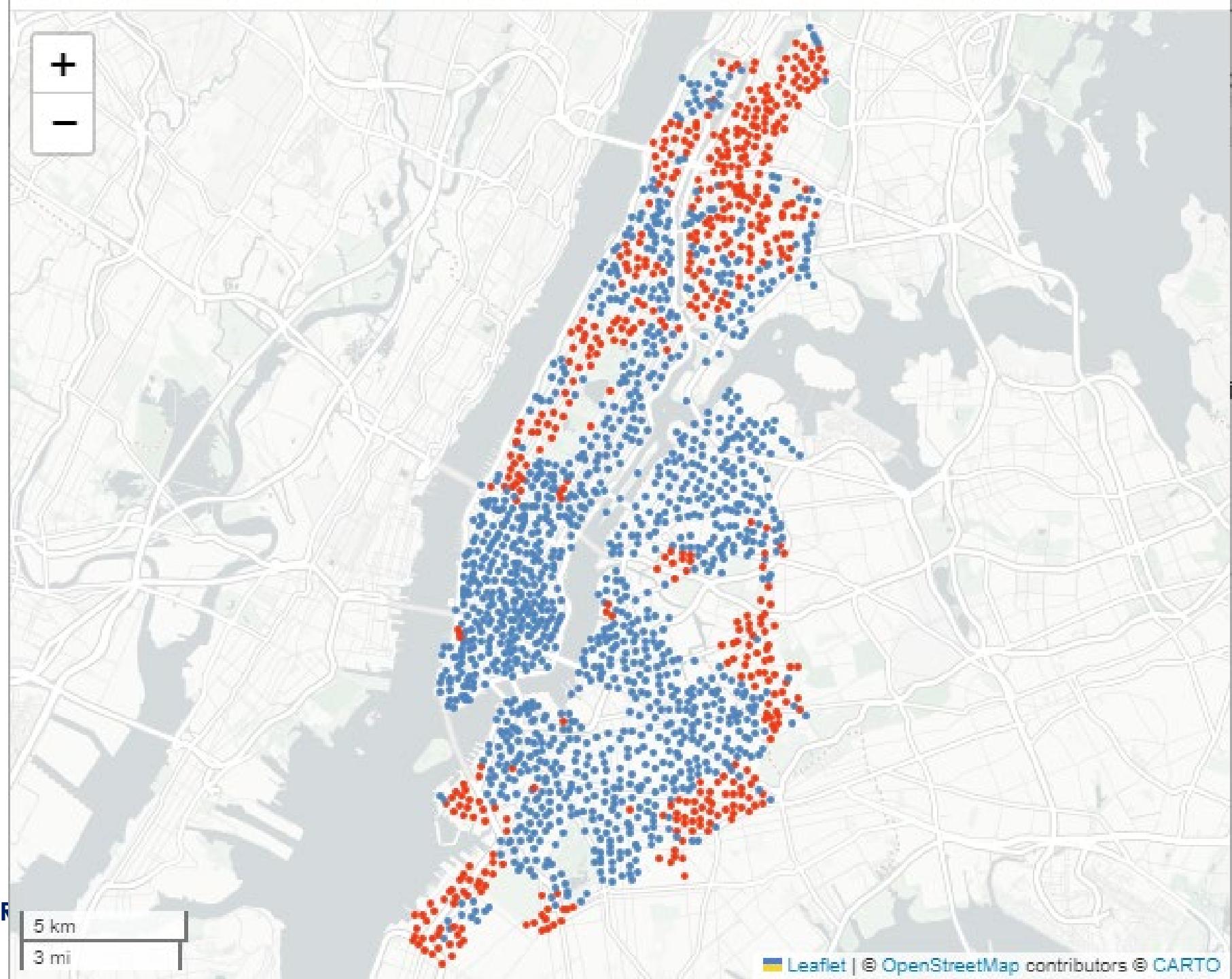


Compute hotspots across each measure



Combine

Poor
performance
stations



NEW YORK CITY COMPTROLLER

5 km

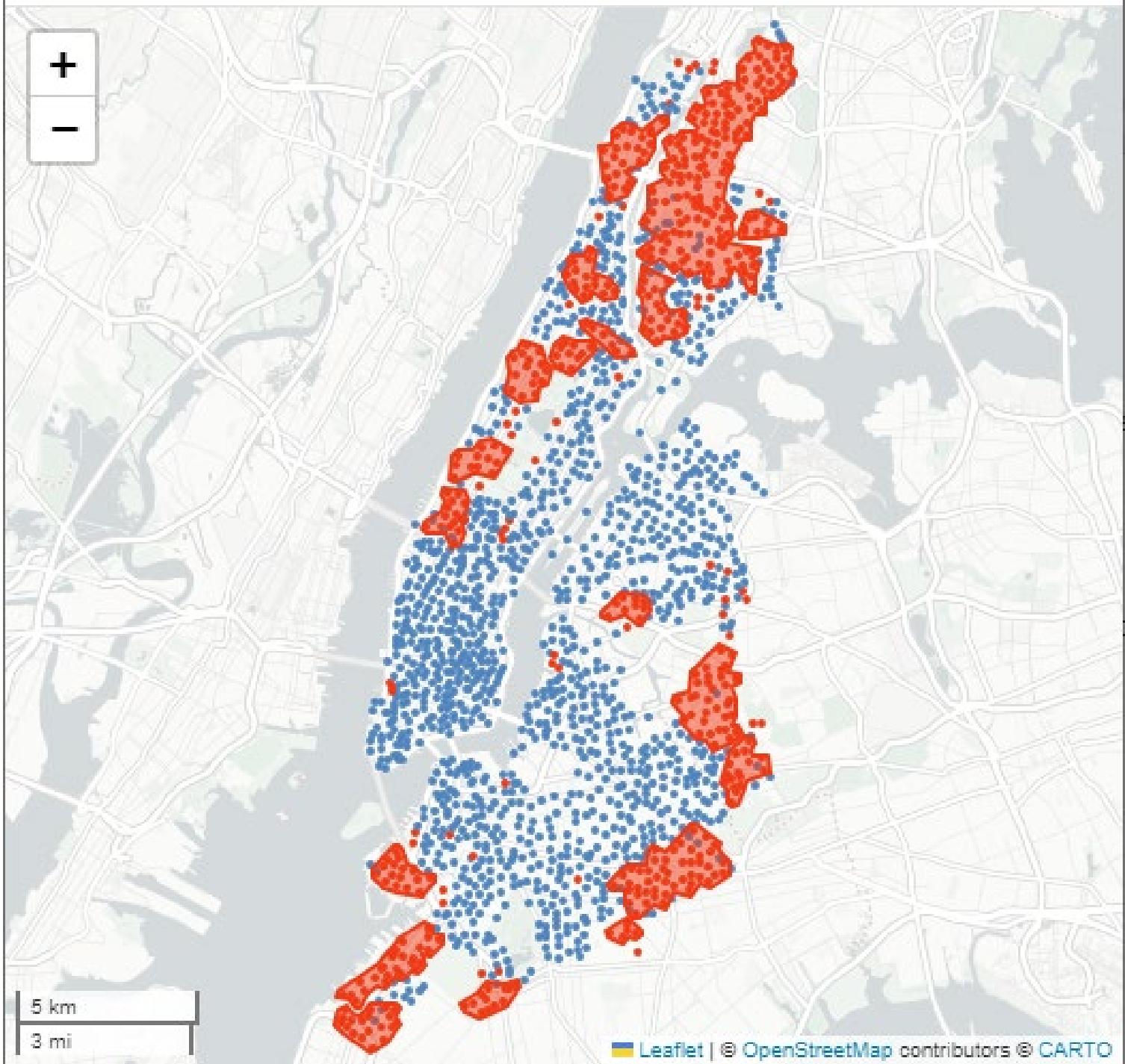
3 mi

Leaflet | © OpenStreetMap contributors © CARTO

Find clusters

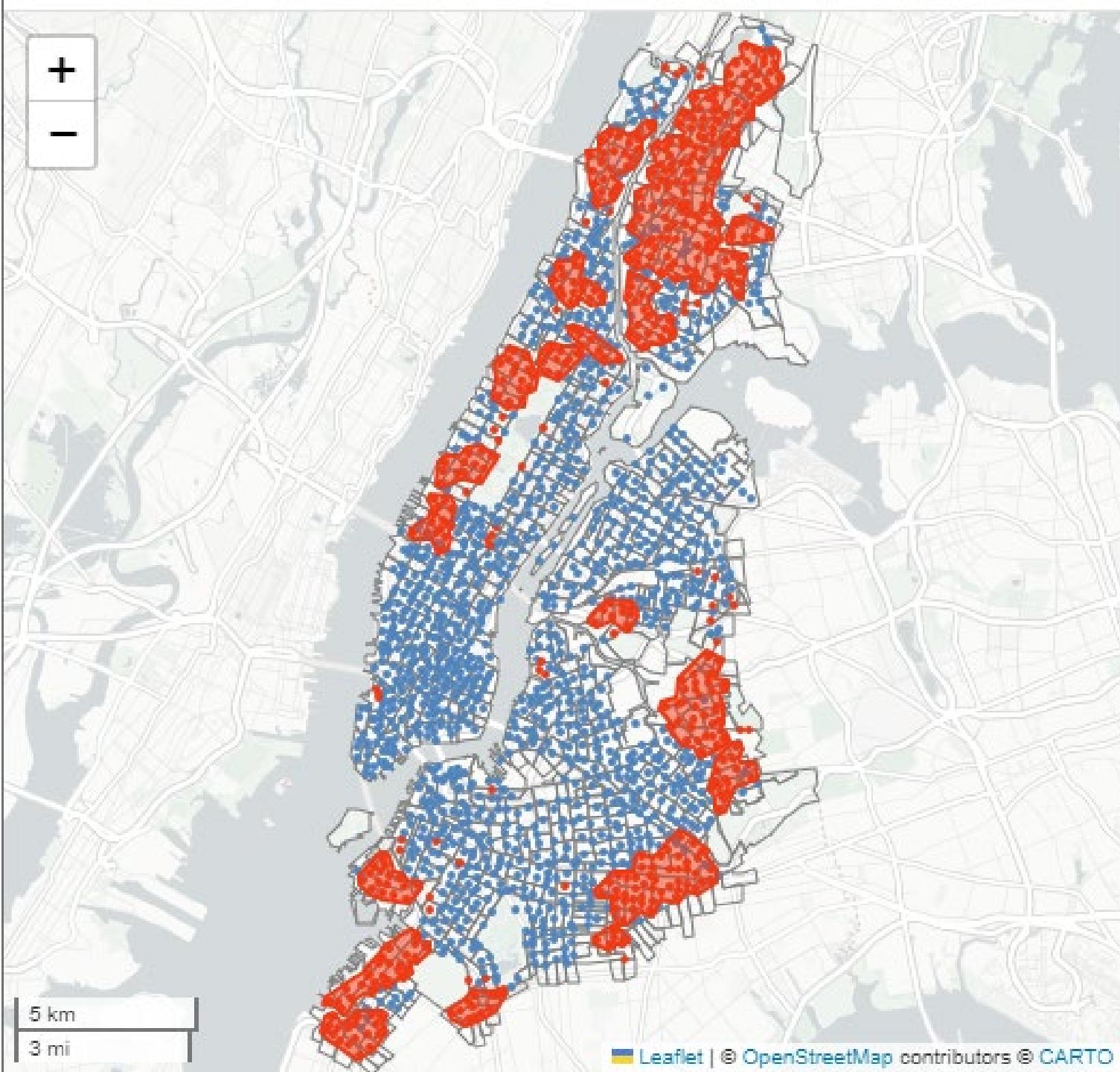
Keep groups with at least 5 stations within $\frac{1}{4}$ mile

Build concave hulls as boundaries



NEW YORK CITY COMPTROLLER BRAD LANDER

Join Tracts



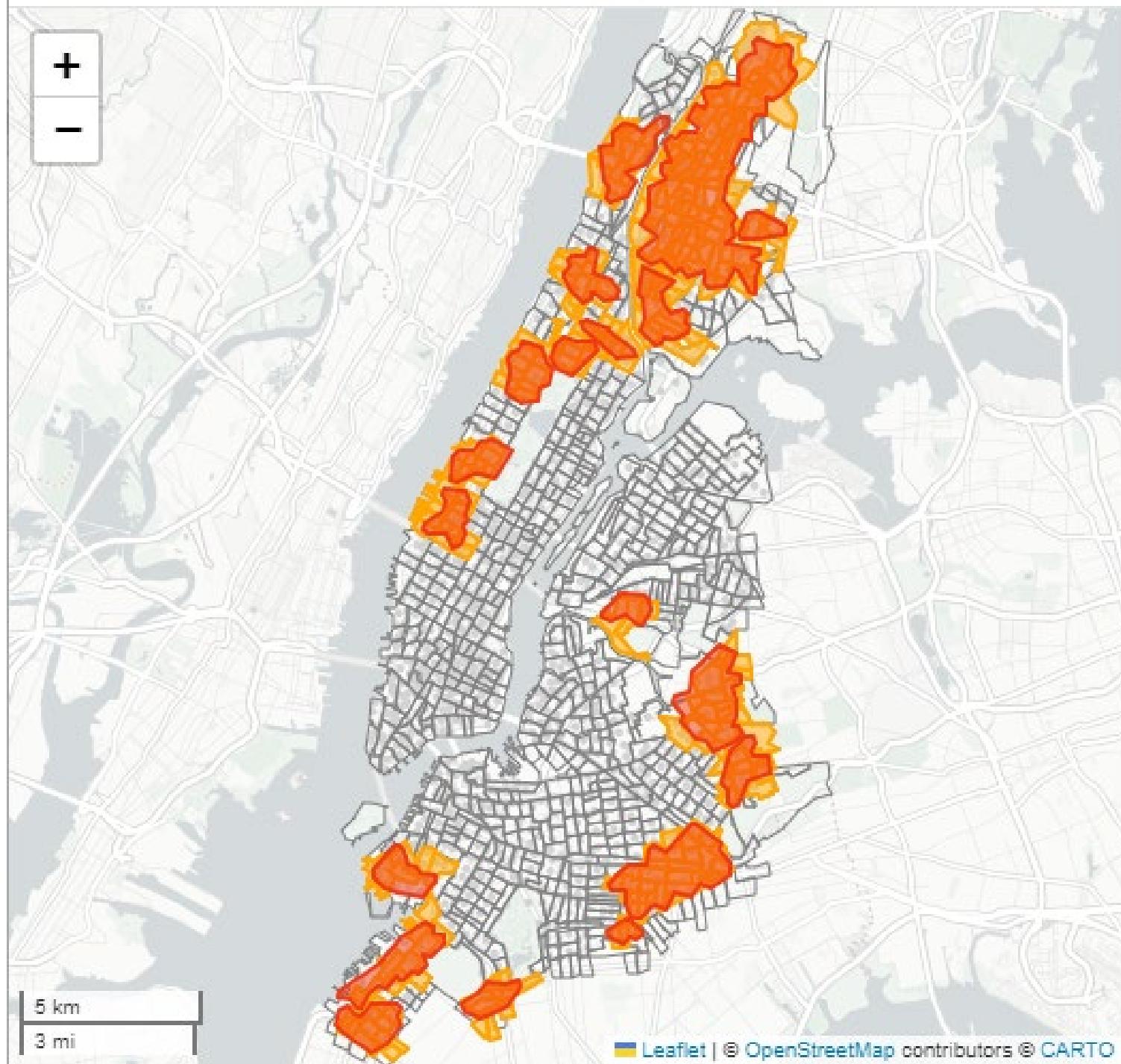
NEW YORK CITY COMPTROLLER BRAD LANDER

5 km

3 mi

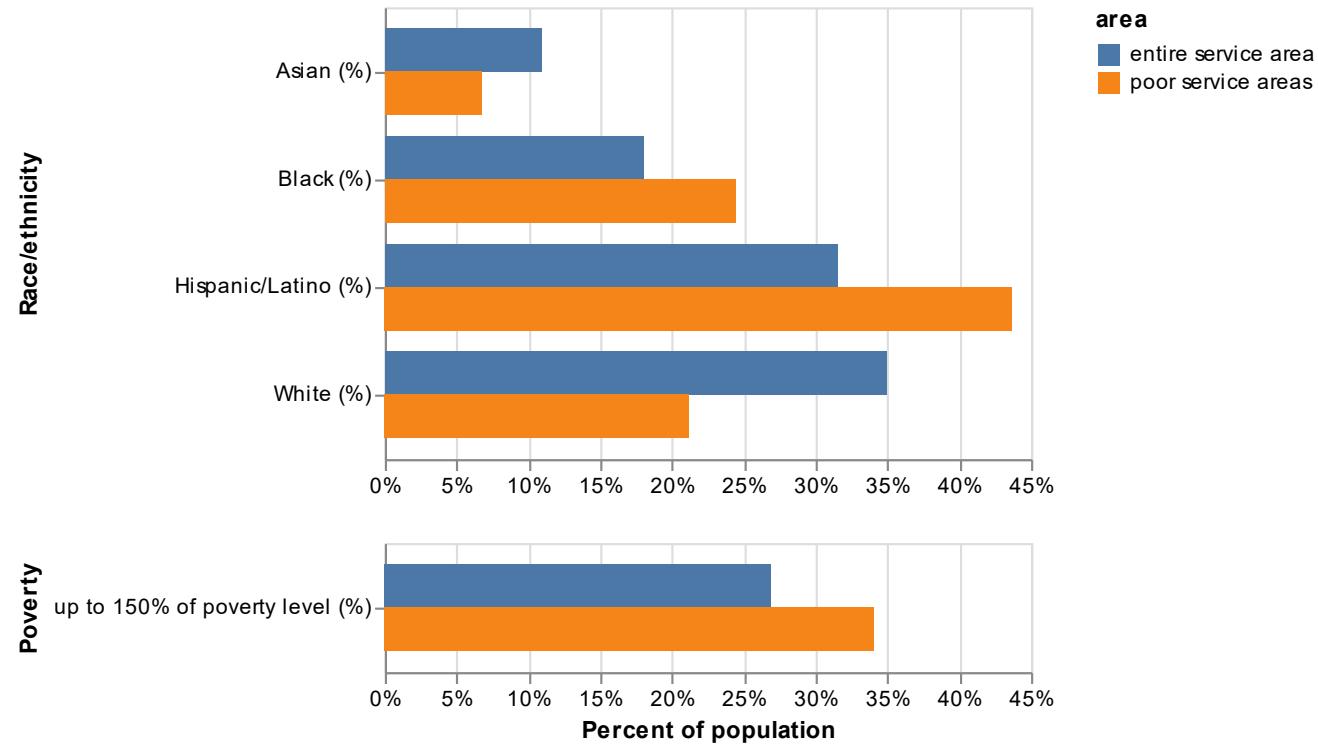
Join Tracts

Tracts intersecting poor service clusters and entire service area

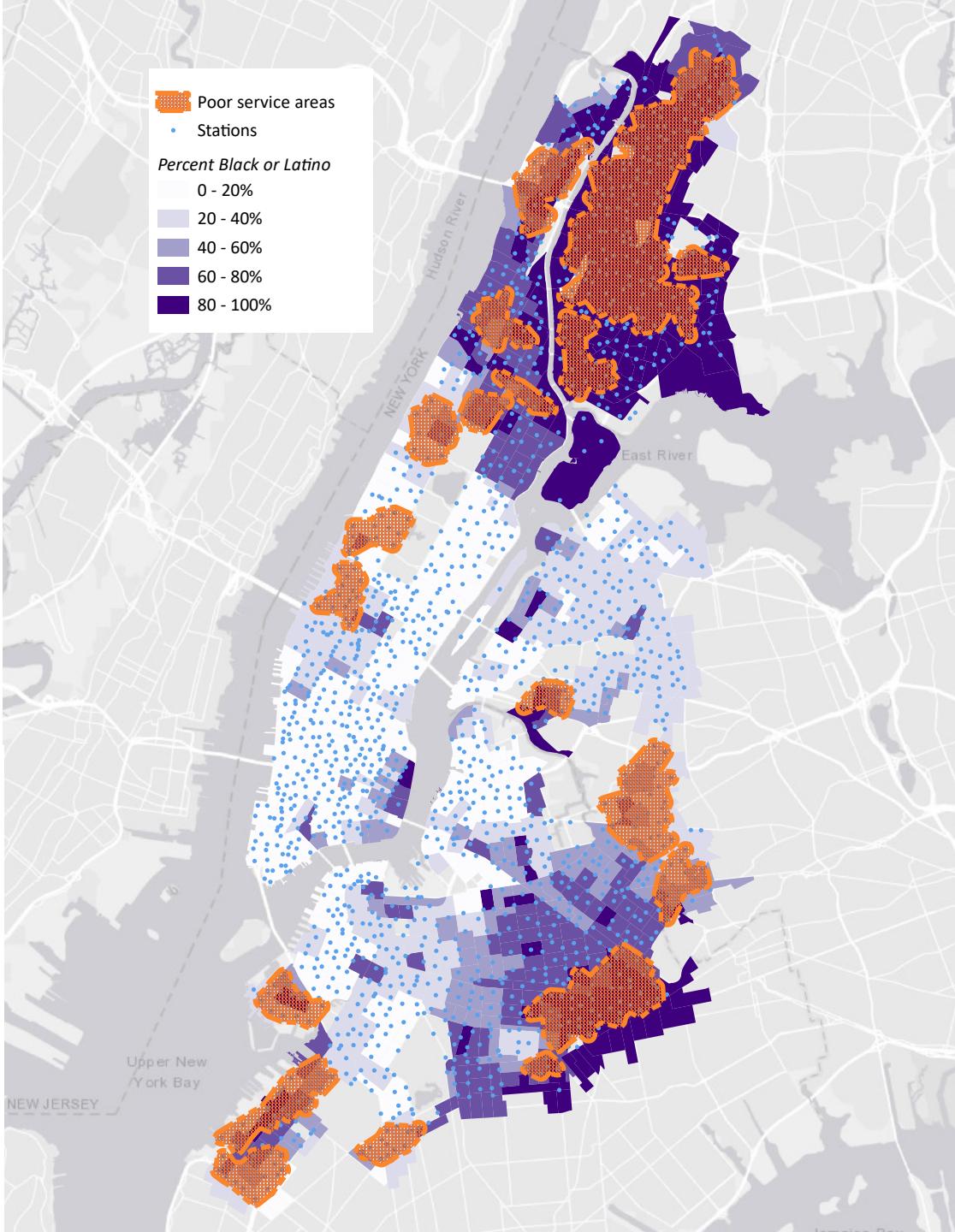


NEW YORK CITY COMPTROLLER BRAD LANDER

Least reliable service in areas home to more Hispanic, Black, and low-income residents



Least reliable service in areas home to more Hispanic & Black residents



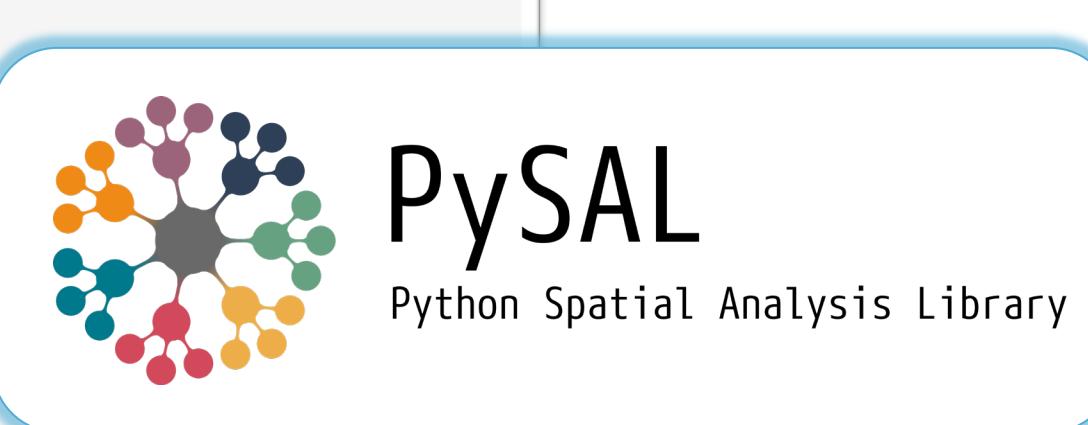
NEW YORK CITY COMPTROLLER BRAD LANDER

```
for measure in measures:
```

But how actually?

```
    w_w_tow,
    transformation='r',
    permutations=1000,
    n_jobs=-1,
    seed=1
)

measure_result = (
    stations_service_measures
    [[measure]]
    .assign(
        q = measure_local_moran.q,
        p_z_sim = measure_local_moran.p_z_sim,
        p_sim = measure_local_moran.p_sim,
        significant_cluster = lambda row: (
            row['q'].map(QUADRANT_LABELS)
            .where(
                (row['p_z_sim'] < alpha) &
                (row['q'].isin([1,3]))
            )
        ),
    )
    .rename(columns={
        'q':f'{measure}_q',
        'p_z_sim':f'{measure}_p_z_sim',
        'p_sim':f'{measure}_p_sim',
        'significant cluster':f'{measure} significant cluster',
    })
)
```



PySAL
Python Spatial Analysis Library

or:



Q&A

A photograph showing a large number of blue Citibikes lined up at a docking station. The bikes are secured to grey metal posts. The background shows a modern building with a glass and steel facade. A prominent blue rectangular overlay with a white border is centered over the middle of the bike row. Inside the overlay, the text "You can too!" is written in a large, white, sans-serif font.

You can too!

Fork the repo

NYCComptroller / citi-bike-gbfs

<https://github.com/NYCComptroller/citi-bike-gbfs>

citi-bike-gbfs Public Edit Pins Unwatch 2

main 1 Branch 0 Tags Go to file t + Code

File	Description	Time
.github/workflows	rename action	last week
Examples	Post readme, examples, and environment	11 minutes ago
img	Post readme, examples, and environment	11 minutes ago

Start recording

Code Pull requests Actions Projects Security

Files main

citi-bike-gbfs-sample-fork / .github / workflows / pull-data.yaml

37 lines (29 loc) · 869 Bytes

Code Blame Raw

```
1 name: pull data
2
3 on:
4   schedule:
5     - cron: '10,25,40,55 * * * *'
6
7 jobs:
8   get:
9     runs-on: ubuntu-latest
10    steps:
11      - name: checkout repo content
```

Workflow runs · dlevine01/citi-bike-gbfs-sample-fork/actions

Actions Projects Security Insights Settings

All workflows Showing runs from all workflows

135 workflow runs

Event	Status	Branch	Actor
pull data	Scheduled	main	14 minutes ago 26s
pull data	Scheduled	main	27 minutes ago 33s
pull data	Scheduled	main	51 minutes ago 30s
pull data	In Progress	main	1 hour ago

Build dataset

		capacity	is_renting	is_returning	num_docks_available	num_bikes_available	num_ebikes_available	num_bikes_disabled
last_updated	station_id							
2024-03-12 17:40:49	66dc2995-0aca-11e7-82f6-3863bb44ef7c	51	0	0	0	0	0	0
	06439006-11b6-44f0-8545-c9d39035f32a	48	0	0	0	0	0	0
	19d17911-1e4a-41fa-b62b-719aa0a6182e	39	0	0	0	0	0	0
	1861678548643203686	25	1	1	8	15	8	2
	cd2d9dab-7708-4685-a56f-9412c738de7e	23	1	1	1	22	1	0
	66db2a71-0aca-11e7-82f6-3863bb44ef7c	45	1	1	0	41	16	4
	901ad0c4-383e-490a-8b54-0656ce2358d6	19	1	1	3	11	2	5
	66dc292c-0aca-11e7-82f6-3863bb44ef7c	55	1	1	50	3	1	2
	4c03fa2d-89da-4f0b-8f19-c7de5edbe256	25	1	1	3	21	3	0
	9af90faf-0b9b-451b-9cb0-20ff421ca1d9	27	1	1	18	9	1	0



View on Binder

The screenshot shows a Jupyter Notebook interface with the following details:

- File Bar:** File, Edit, View, Run, Kernel, Tabs, Settings, Help.
- Launcher:** Shows a list of recent notebooks:
 - Build service...
 - Compare d...
 - Compute cl... (highlighted)
 - Count viola...
 - Get Census...Each item is timestamped "a minute ago".
- Search Bar:** Filter files by name.
- Code Cell:** [1]:

```
import pandas as pd
import geopandas as gpd
import numpy as np

from libpsyal.weights import DistanceBand
fromesda import Moran_Local

from concave_hull import concave_hull
from shapely import Polygon
```
- Output Cell:** Compute clusters of poor service
- Help:** This notebook computes clusters of poor service using local Moran's local indicators of spatial association.
- Instructions:** Before running this notebook, you will need to:
 - record data
 - construct `dataset.parquet` and `stations_geo.geojson` with `Build dataset`
 - construct `stations_service_measures.geojson` with `Build service measures`
- Bottom Status Bar:** Simple 0 \$ 1 Python 3 (ipykern... Mem: 135.01 / 409... Mode: Com... Ln 1, ... Compute clusters of poor se... 1
- Bottom Right Corner:** A yellow circle containing the number 42.



Your turn!

- Possible extensions:
 - Check whether service has improved or patterns have changed
 - Use station stats to measure reliability or service in other ways
 - Check reliability at your favorite station
 - Check on availability in real time, or closer to real-time
- Other applications:
 - Recording any other online data feed to build a longitudinal dataset
 - Clustering analysis and demographic comparison



Thank you!

Stay in touch
at beta.nyc/links

BetaNYC



nycsodata24.sched.com

#nycSodata #opendataweek

