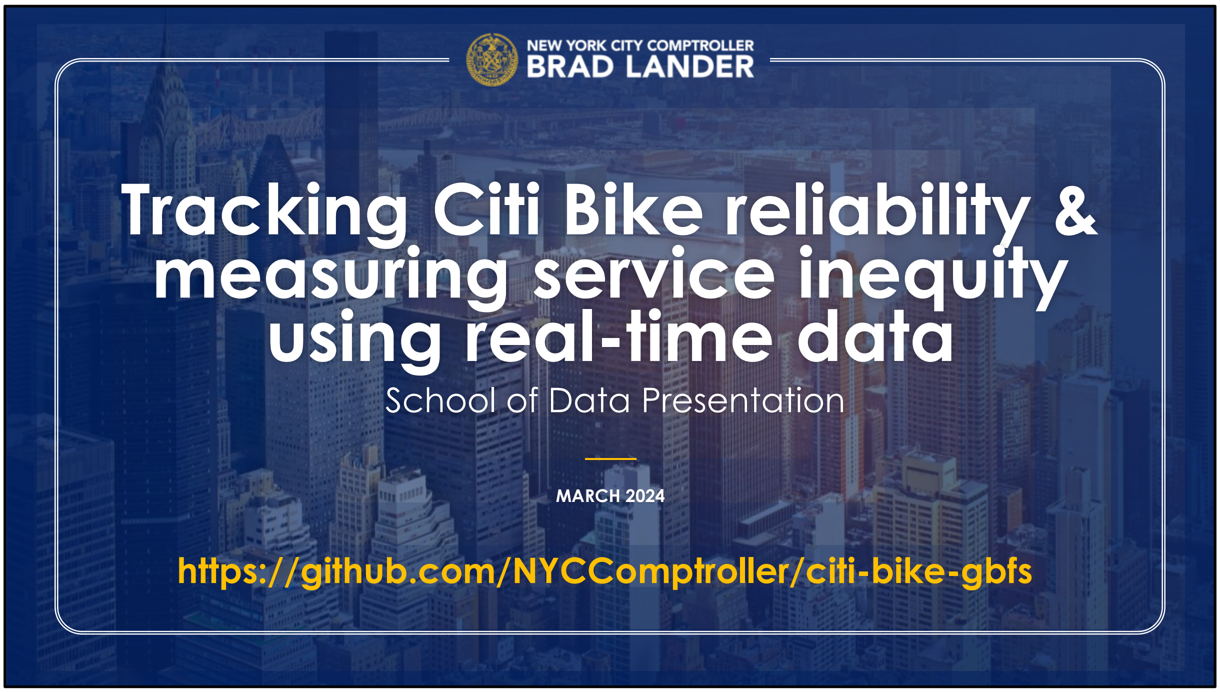


NEW YORK CITY COMPTROLLER
BRAD LANDER

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

School of Data Presentation

MARCH 2024

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

Agenda

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

Problem: unreliable Citi Bike

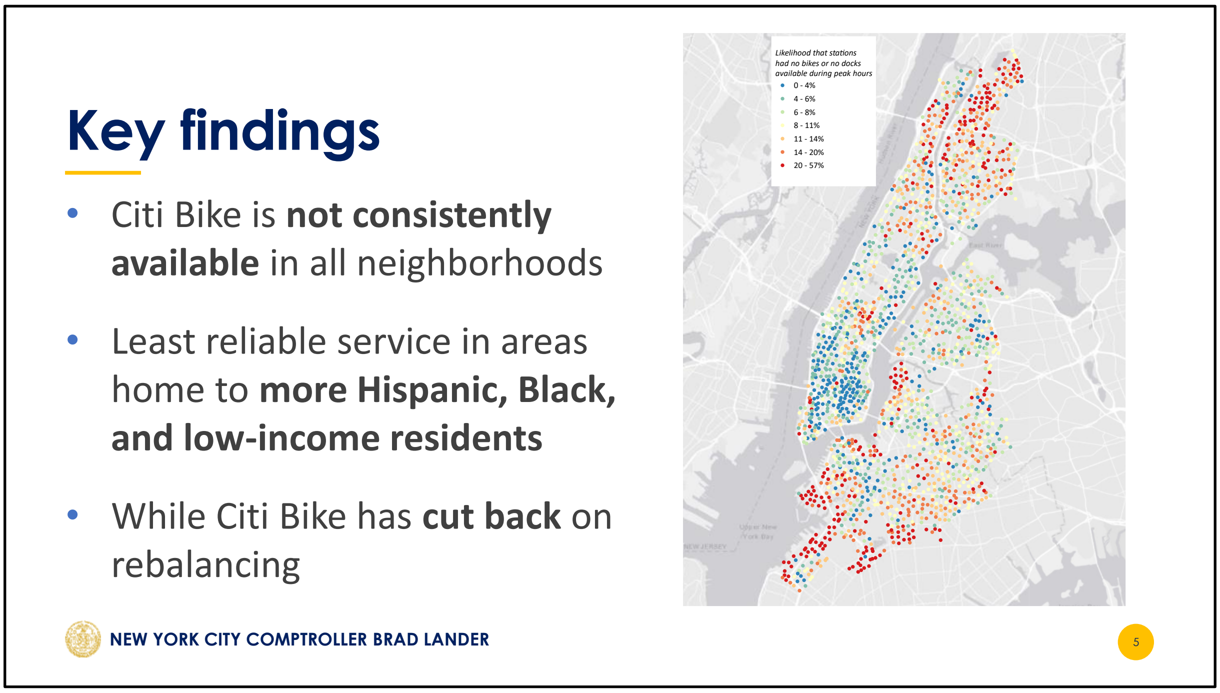
NEW YORK CITY COMPTROLLER BRAD LANDER

3

This project began when riders came to us with a problem: more often, they were finding stations empty, or full of broken bikes. We found a way to investigate Citi Bike’s reliability.

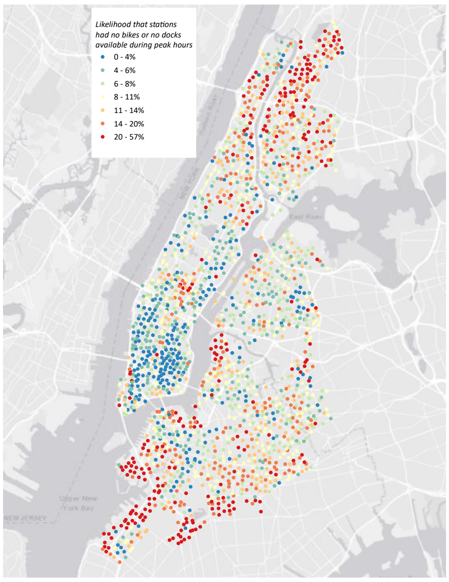
Key findings & policy recommendations

Photo: Elzbieta Sekowska / Shutterstock.com



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

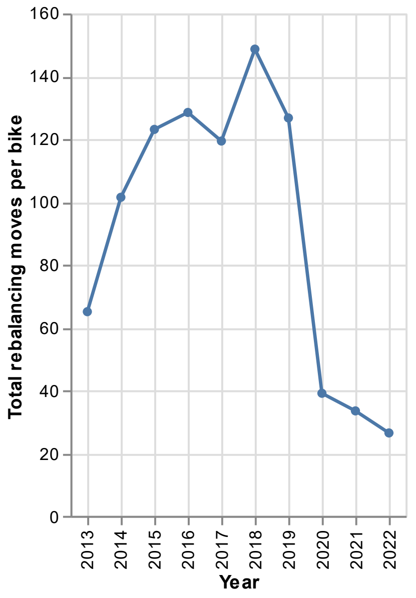


NEW YORK CITY COMPTROLLER BRAD LANDER

5

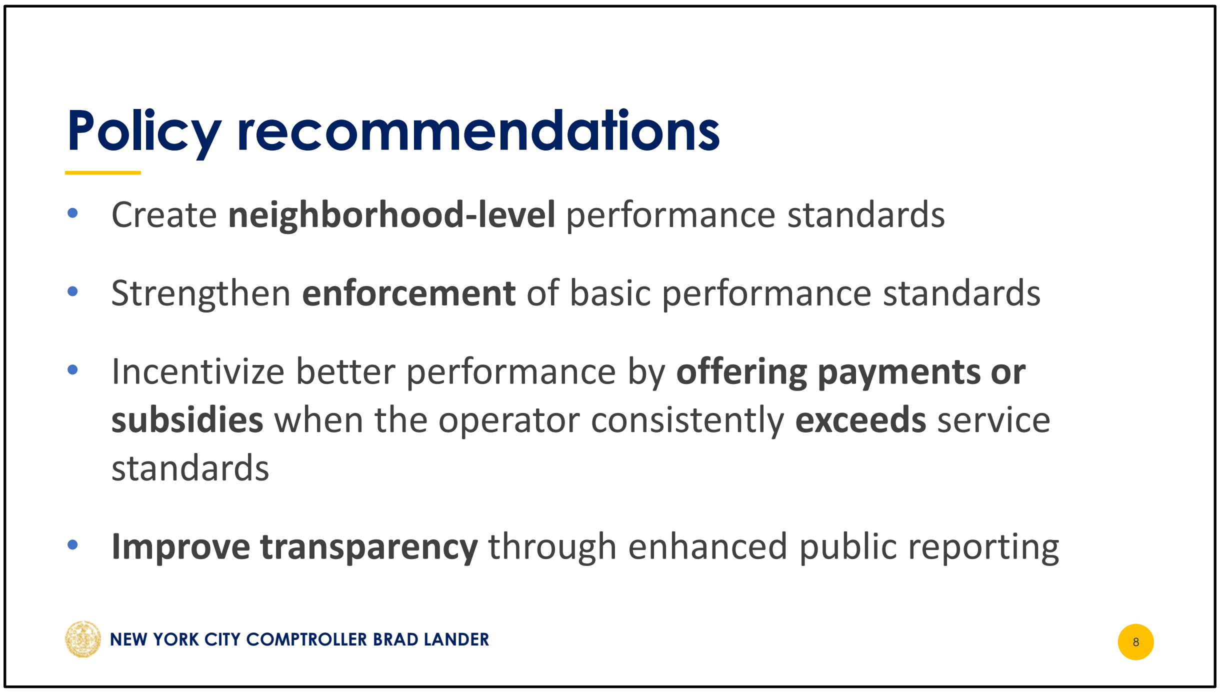
Riders in neighborhoods around the edges of the system in Brooklyn, upper Manhattan, and the Bronx are more likely to find stations unusable, because they have no bikes available or are completely full and can't accept returns. Map: likelihood that riders would find stations with no bikes or no docks, during peak morning and evening hours.

After Citi Bike cut rebalancing moves



NEW YORK CITY COMPTROLLER BRAD LANDER

Riders are seeing this unreliable service – bikes or docks not available often and for long periods – after Citi Bike has dramatically cut back on rebalancing moves. (rebalancing = manually restocking empty stations, moving bikes out of full stations, swapping broken bikes). This rebalancing would be one obvious way to ameliorate the problem.



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

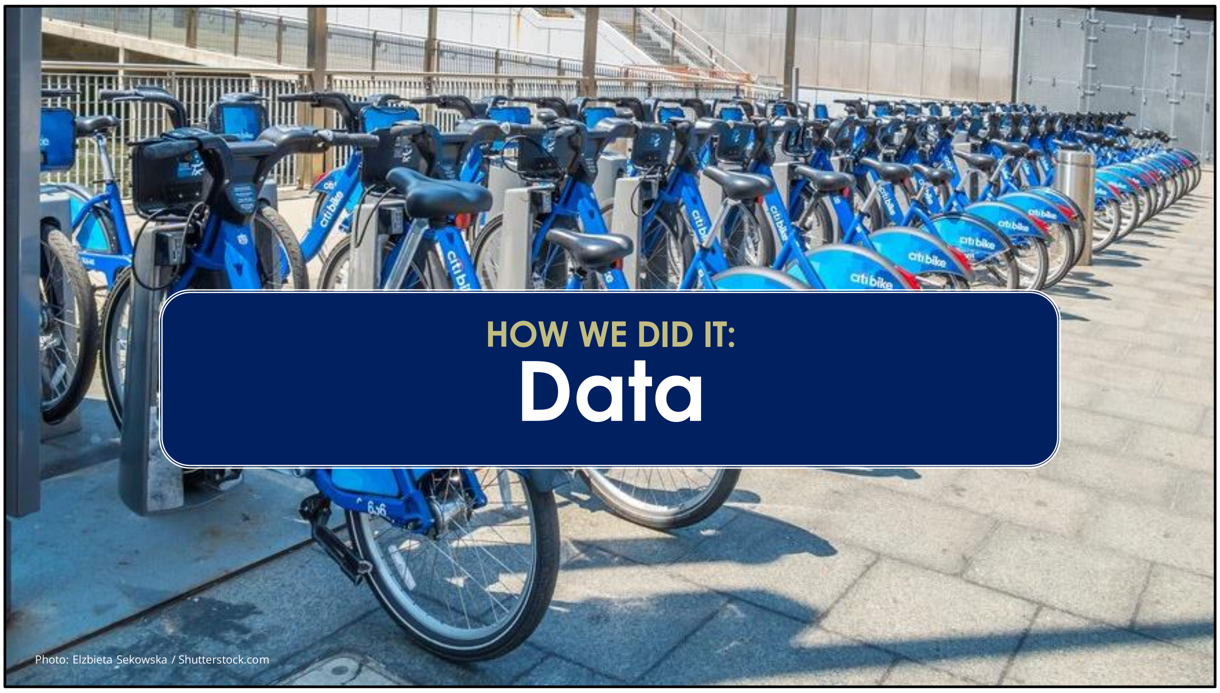
NEW YORK CITY COMPTROLLER BRAD LANDER

8

Citi Bike has become an essential transportation service and expanded its footprint: Ensuring Citi Bike's viability into the future requires optimizing the performance of the current system.
The City should overhaul the contract to set new standards for service.

1) We found big disparities in service between areas, but current service standards are only aggregated citywide. The contract should set local service standards (e.g. borough or community district level)
2) City should step up oversight and enforce service level agreements, including levying fines.
3) Fines are the only enforcement mechanism in the current contract, but this should be a partnership and sticks alone are probably not going to improve service the way we need. We should provide targeted incentives – carrots - to support better service
4) Citi Bike needs to be more accessible for lower-income riders. That means make the system reliable in neighborhoods home to more lower-income New Yorkers, and reduce the membership cost burden. Discounts are now available for NYCHA residents and SNAP recipients, but that misses some eligible low-income New Yorkers not enrolled in SNAP
5) It shouldn't take this much data wrangling to answer these basic questions about how the system is doing. Citi Bike should report more data and give the City detailed data to

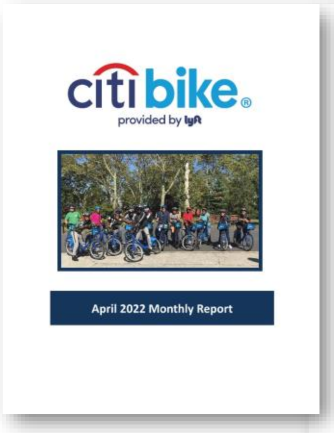
measure if it is meeting service level standards.

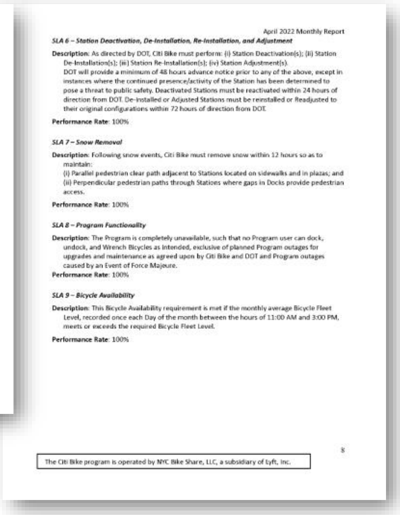

HOW WE DID IT:

Data

Photo: Elzbieta Sekowska / Shutterstock.com

Needed more granular data




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.

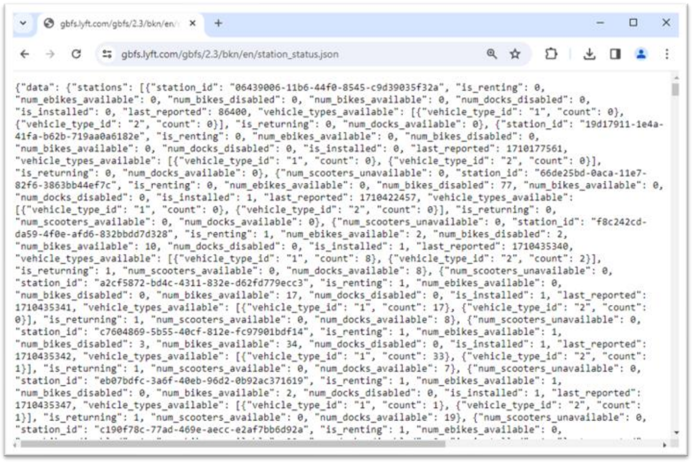
NEW YORK CITY COMPTROLLER BRAD LANDER

Monthly service reports provide only monthly overall average – not broken down by day, time of day, day of week, or location. And, are released with some lag (up to several months.

Real-time detail

Citi Bike GBFS

(General Bikeshare Feed Specification)


gbfs.lyft.com/gbfs/2.3/bkn/en/station_status.json

```
{"data": {"stations": [{"station_id": "06439006-11b6-44f0-8545-c9d39035f32a", "is_renting": 0,
"num_ebikes_available": 0, "num_bikes_disabled": 0, "num_bikes_available": 0, "num_docks_disabled": 0,
"is_installed": 0, "last_reported": 86400, "vehicle_types_available": [{"vehicle_type_id": "1", "count": 0},
{"vehicle_type_id": "2", "count": 0}], "is_returning": 0, "num_docks_available": 0}, {"station_id": "19d17911-1e4a-
41fa-b62b-719aa0a6182e", "is_renting": 0, "num_ebikes_available": 0, "num_bikes_disabled": 0,
"num_bikes_available": 0, "num_docks_disabled": 0, "is_installed": 0, "last_reported": 1710177561,
"vehicle_types_available": [{"vehicle_type_id": "1", "count": 0}, {"vehicle_type_id": "2", "count": 0}],
"is_returning": 0, "num_docks_available": 0}, {"num_scooters_unavailable": 0, "station_id": "66de25bd-0aca-11e7-
82f6-3863bb44ef7c", "is_renting": 0, "num_ebikes_available": 0, "num_bikes_disabled": 77, "num_bikes_available": 0,
"num_docks_disabled": 0, "is_installed": 1, "last_reported": 1710422457, "vehicle_types_available":
[{"vehicle_type_id": "1", "count": 0}, {"vehicle_type_id": "2", "count": 0}], "is_returning": 0,
"num_scooters_available": 0, "num_docks_available": 0}, {"num_scooters_unavailable": 0, "station_id": "f8c242cd-
da59-4f0e-afd6-832bbdd7d328", "is_renting": 1, "num_ebikes_available": 2, "num_bikes_disabled": 2,
"num_bikes_available": 10, "num_docks_disabled": 0, "is_installed": 1, "last_reported": 1710435340,
"vehicle_types_available": [{"vehicle_type_id": "1", "count": 8}, {"vehicle_type_id": "2", "count": 2}],
"is_returning": 1, "num_scooters_available": 0, "num_docks_available": 8}, {"num_scooters_unavailable": 0,
"station_id": "a2cf5872-bd4c-4311-832e-d62fd779ecc3", "is_renting": 1, "num_ebikes_available": 0,
"num_bikes_disabled": 0, "num_bikes_available": 17, "num_docks_disabled": 0, "is_installed": 1, "last_reported":
1710435341, "vehicle_types_available": [{"vehicle_type_id": "1", "count": 17}, {"vehicle_type_id": "2", "count":
0}], "is_returning": 1, "num_scooters_available": 0, "num_docks_available": 8}, {"num_scooters_unavailable": 0,
"station_id": "c7604869-5b55-40cf-812e-fc97901bdf14", "is_renting": 1, "num_ebikes_available": 1,
"num_bikes_disabled": 3, "num_bikes_available": 34, "num_docks_disabled": 0, "is_installed": 1, "last_reported":
1710435342, "vehicle_types_available": [{"vehicle_type_id": "1", "count": 33}, {"vehicle_type_id": "2", "count":
1}], "is_returning": 1, "num_scooters_available": 0, "num_docks_available": 7}, {"num_scooters_unavailable": 0,
"station_id": "eb07bdfc-3a6f-40eb-96d2-0b92ac371619", "is_renting": 1, "num_ebikes_available": 1,
"num_bikes_disabled": 0, "num_bikes_available": 2, "num_docks_disabled": 0, "is_installed": 1, "last_reported":
1710435347, "vehicle_types_available": [{"vehicle_type_id": "1", "count": 1}, {"vehicle_type_id": "2", "count":
1}], "is_returning": 1, "num_scooters_available": 0, "num_docks_available": 19}, {"num_scooters_unavailable": 0,
"station_id": "c190f78c-77ad-469e-aecc-e2af7bb6d92a", "is_renting": 1, "num_ebikes_available": 0,
```

NEW YORK CITY COMPTROLLER BRAD LANDER

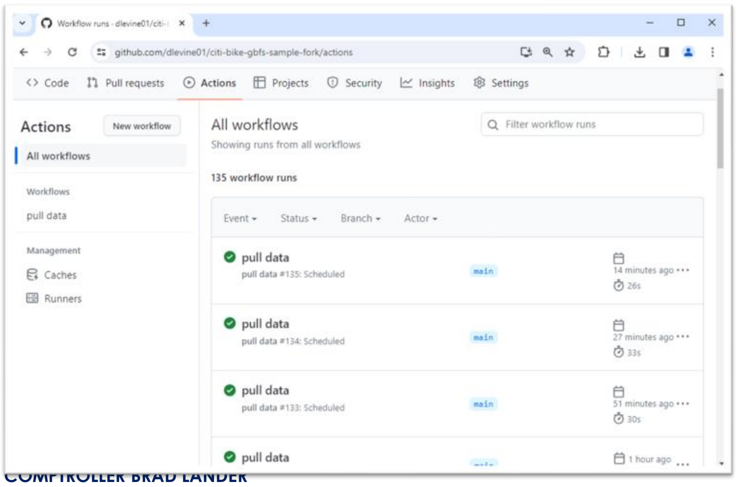
So we used Citi Bike's real-time GBFS (General Bikeshare Feed Specification) data: https://gbfs.citibikenyc.com/gbfs/2.3/gbfs.json

This feed powers navigation apps, offering real-time view of where bikes and docks are available. Updated every 5 seconds.

It looks like a bit of a mess, but is a structured json with a fixed schema, used by bike share systems around the world: https://gbfs.org/

The problem for our research was that this data is _only_ live, not historic.

Recording live data



Workflow runs · dlevine01/citi-

github.com/dlevine01/citi-bike-gbfs-sample-fork/actions

Code | Pull requests | Actions | Projects | Security | Insights | Settings

Actions — New workflow

All workflows

Workflows: pull data

Management: Caches, Runners

All workflows

Showing runs from all workflows

135 workflow runs

Event | Status | Branch | Actor

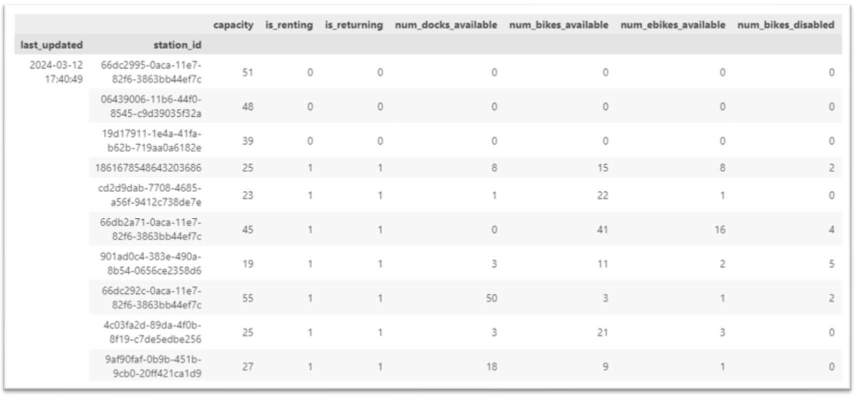
pull data pull data #135: Scheduled	main	14 minutes ago 26s
pull data pull data #134: Scheduled	main	27 minutes ago 33s
pull data pull data #133: Scheduled	main	51 minutes ago 30s
pull data		1 hour ago

So, we started recording the data feed.

Set up a GitHub Actions script that, every 15 minutes or so, would load the data and save it.

(You can do this too, by following the steps in the template repo)

Build detailed data



last_updated	station_id	capacity	is_renting	is_returning	num_docks_available	num_bikes_available	num_ebikes_available	num_bikes_disabled
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

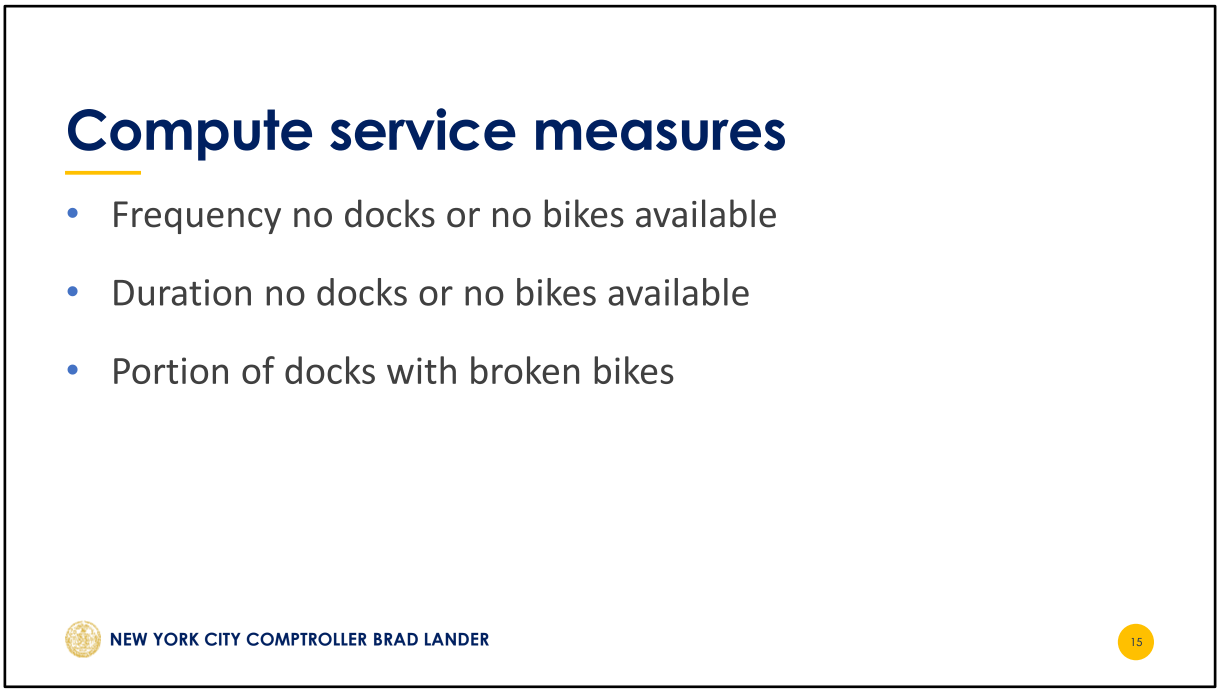
NEW YORK CITY COMPTROLLER BRAD LANDER

13

We started this investigation at the beginning of June, and after two months of peak summer ridership, thought we should have enough to look for patterns.
The next step was to join all those data samples together into a single dataset
(You can also repeat this with the code in the template repo)

HOW WE DID IT:
Analysis

Photo: Elzbieta Sekowska / Shutterstock.com



Compute service measures

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

NEW YORK CITY COMPTROLLER BRAD LANDER

15

Needed to convert the raw data into measures of the experience and reliability for riders.

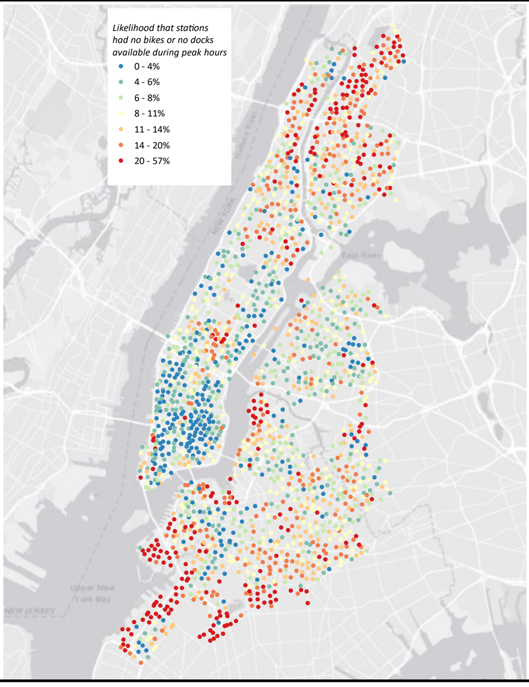
We explored many subsets of the data: e.g. where specifically bikes or docks were frequently unavailable in either morning or evening, peak or off-peak, weekday or weekend, etc.

We settled on three overall measures of station reliability:
- frequency that stations have no docks or no bikes (within morning (7:00-11:00 or evening (4:00-10:00 peak hours)
- median durations of outages when stations had no docks or no bikes (excepting overnight hours, midnight to 6:00AM)
- percent of docks that held a broken bike

These were measures we could cleanly draw from the data that we thought well-defined availability, or lack of availability, for riders

Explore the data:

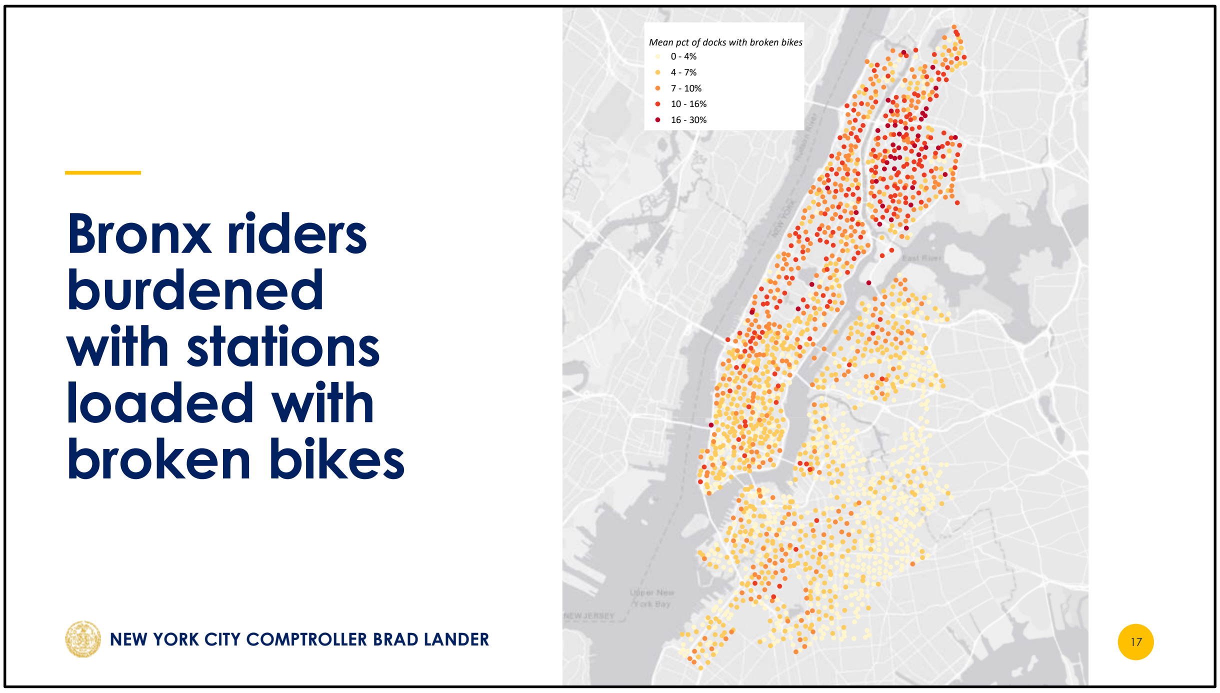
Unequal service between neighborhoods



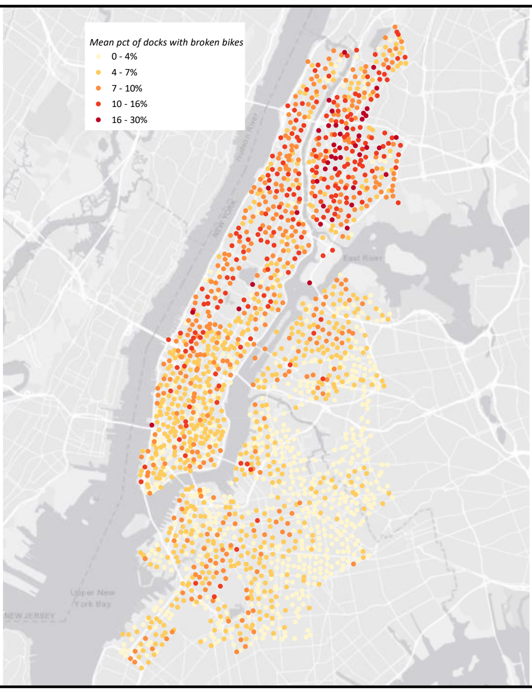
NEW YORK CITY COMPTROLLER BRAD LANDER

16

Seems to be a clear pattern, with more availability in core of Manhattan, less around edges of system.



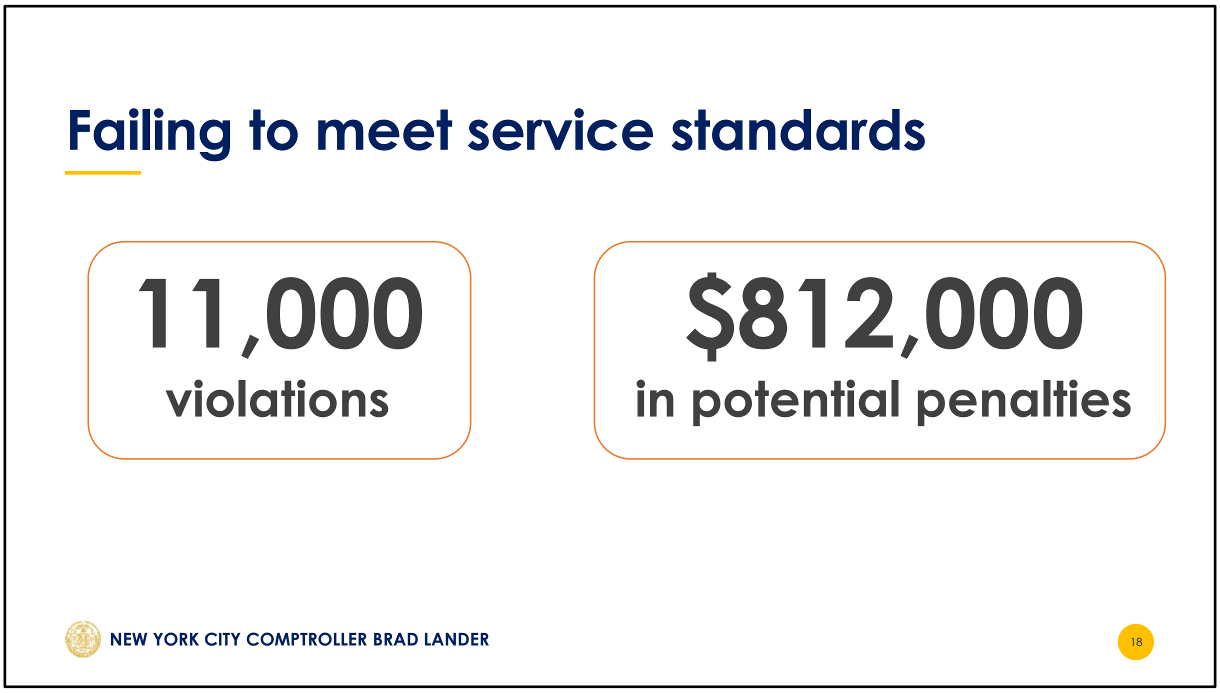
Bronx riders burdened with stations loaded with broken bikes



NEW YORK CITY COMPTROLLER BRAD LANDER

17

Also, we found shockingly high proportions of broken bikes: Across the Bronx, stations were routinely up to 1/3 full of out-of-service bikes – which means fewer bikes available for rent _and_ fewer docks



Failing to meet service standards

11,000 violations

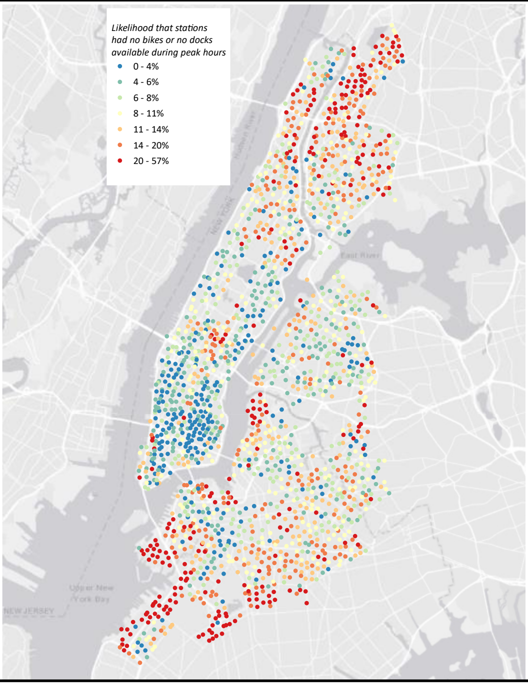
$812,000 in potential penalties

NEW YORK CITY COMPTROLLER BRAD LANDER

18

Citi Bike's operating agreement with the City includes service level agreements for bike and dock availability and a schedule of fines that can be levied against the system for outages or periods stations are not usable.
It does not appear the City has ever actually levied fines on Citi Bike for failing to meet these service levels.
We estimated just two service measures – the ones we could approximate from our data – peak times more than 1 hour with no bikes, or off-peak times more than 2 hours with no bikes – and estimate Citi Bike could be liable for over $800,000 in fines just for the months of June and July.

Unequal service between neighborhoods


Likelihood that stations had no bikes or no docks available during peak hours
● 0 - 4%
● 4 - 6%
● 6 - 8%
● 8 - 11%
● 11 - 14%
● 14 - 20%
● 20 - 57%

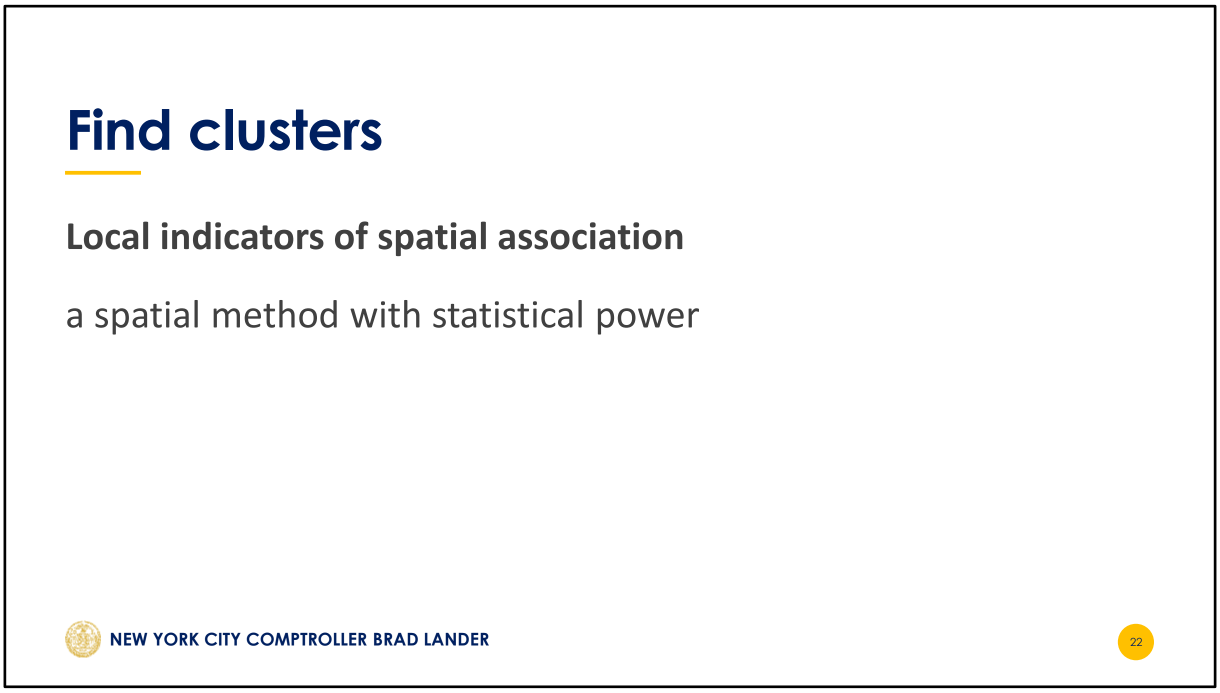
NEW YORK CITY COMPTROLLER BRAD LANDER

There certainly _appears_ to be a clear structure here, but we wanted to take the analysis one step further by computing clusters:

1) Because our human brains are good at seeing patterns in what could, in reality, be random chance
2) Because groups of stations with similar poor service are _even_worse_ for riders than a single unavailable station
3) In order to compress the variability into a binary, that is, highlight areas of _worst_ service

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

22

There are tons of ways to cluster data. We used a specifically _spatial_ method, i.e. one that accounts for the physical locations on stations (which many clustering methods don't do!) and one that gives a statistical representation of how unlikely the observed clusters are under random chance.:

Local Indicators of Spatial Association, specifically Local Moran's I.

Developed by Luc Anselin, a spatial econometrician and leader in this field.

Read the paper defining this method here:
https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1538-4632.1995.tb00338.x

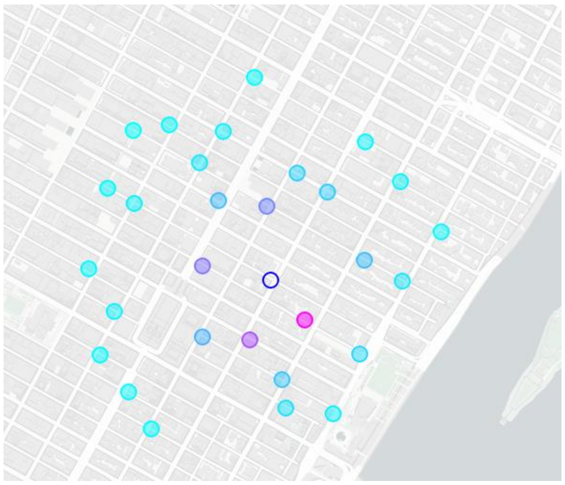
Or watch Anselin's lecture explaining this method here:
https://www.youtube.com/watch?v=zyt7djba4EM

Essentially, this method is:
- compare the measure of reliability at each location, with the average value of that measure at nearby locations.

- pick out the locations that are less reliable than average, and whose neighbors are also less reliable.
- filter those to locations that are most unlikely by random chance.

Neighbors

1. Compute weighted average of nearby stations

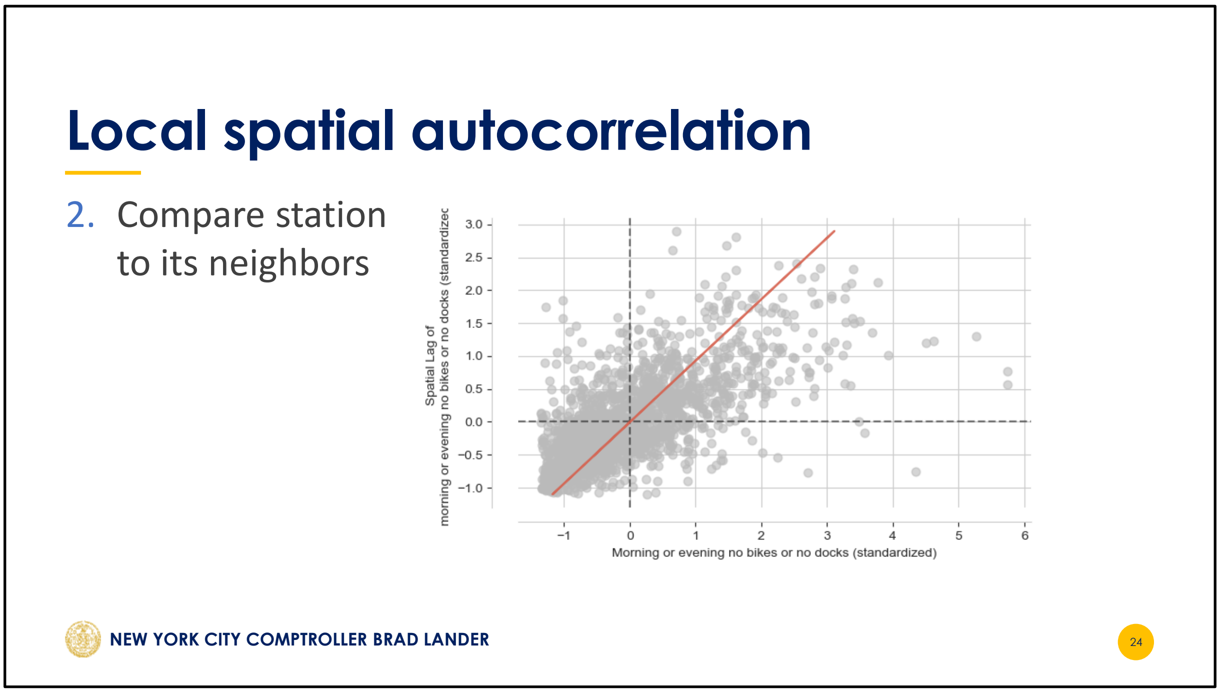


NEW YORK CITY COMPTROLLER BRAD LANDER

23

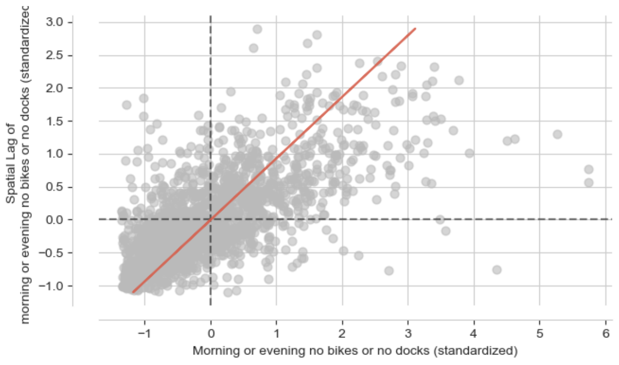
Need to choose how to define “neighboring” stations. (This is one of the most important considerations for this, and many other, spatial analysis methods)

We considered neighboring stations as any within a half-mile (straight-line) distance, and weighted by the inverse square of (straight-line) distance: meaning the nearest stations count for a lot more than the stations at the ½-mile boundary.



Local spatial autocorrelation

2. Compare station to its neighbors



NEW YORK CITY COMPTROLLER BRAD LANDER

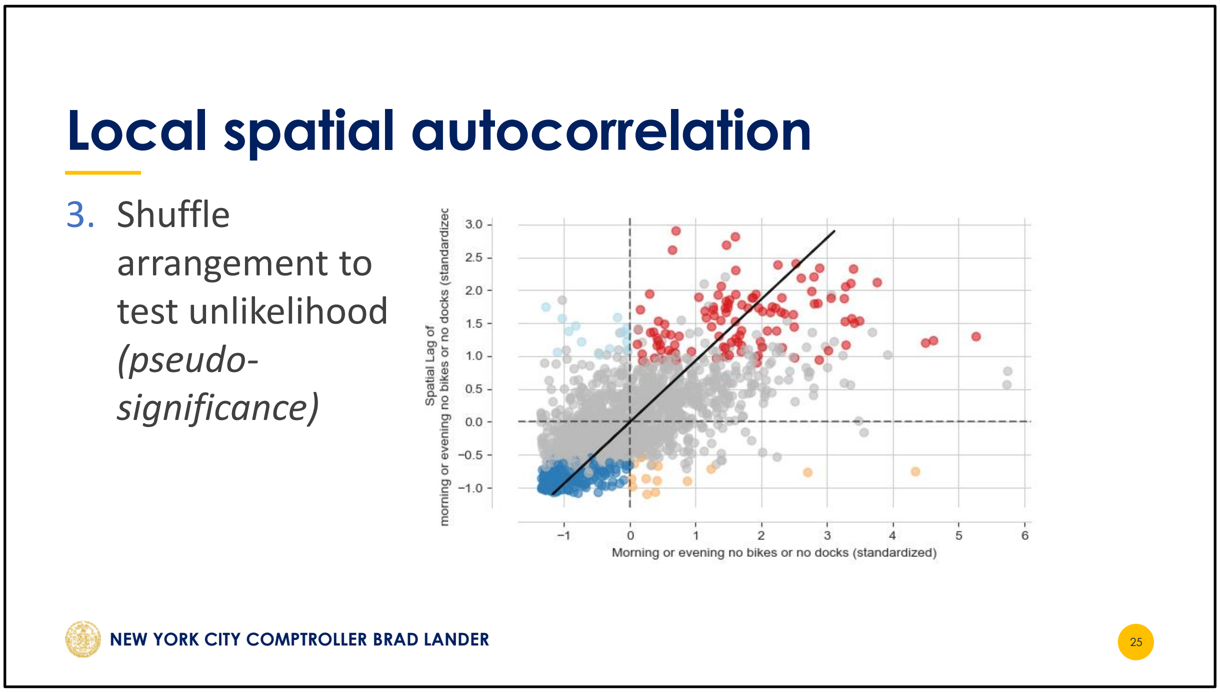
24

For each station:
- We have the measure of reliability at each station: that is plotted along the x axis.
- We compute the weighted average reliability of the nearby stations: that is plotted on the y axis

So this is _auto_ correlation: correlation against the same variable at nearby locations
(if you are familiar with lags in time series analysis this can also be thought of as a _spatial_ lag)
Note: values are standardized, so these are better or worse reliability compared to the mean (not to any absolute threshold)

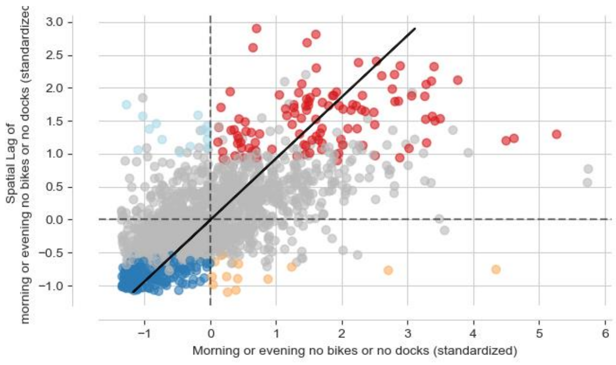
We already see something interesting: the positive trend line (the red fit line) means that overall, systemwide, stations with good availability are mostly surrounded by other stations with good availability; whereas stations with poor availability are mostly surrounded by other stations with poor availability.

Those points in the upper right quadrant all represent stations that are unreliable and are surrounded by unreliable stations. But which are the worst?
We'll take a statistical approach to worst: which are the ones most unlikely to be observed?



Local spatial autocorrelation

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



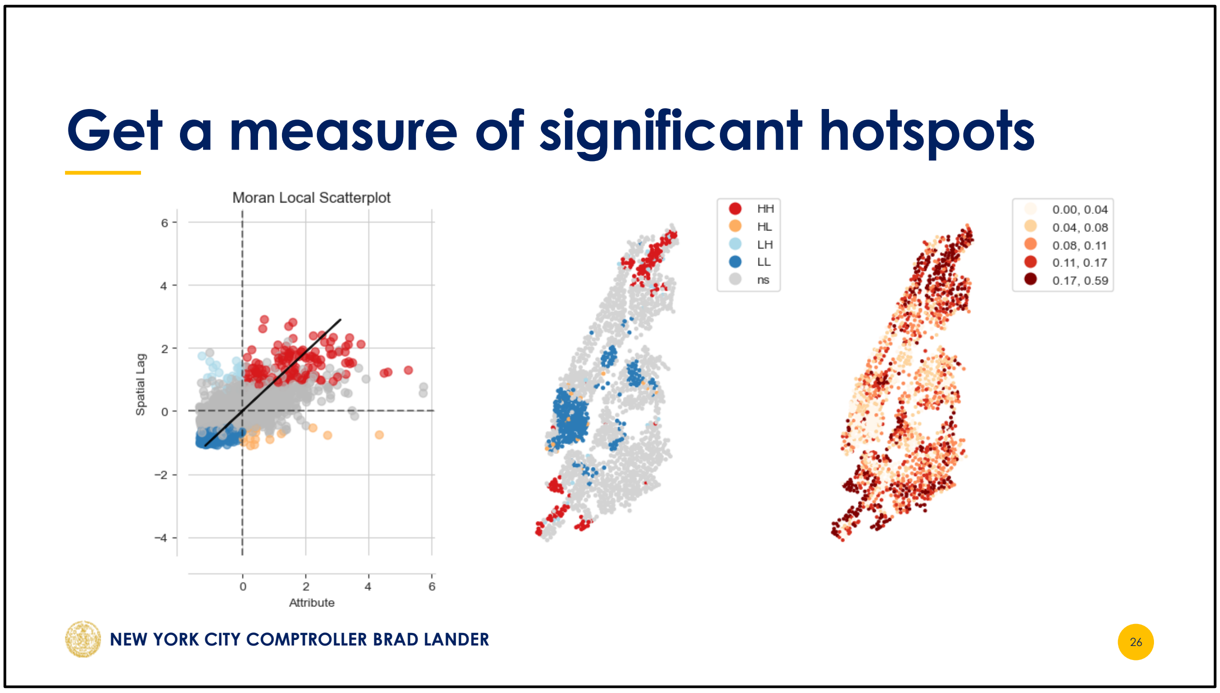
NEW YORK CITY COMPTROLLER BRAD LANDER

25

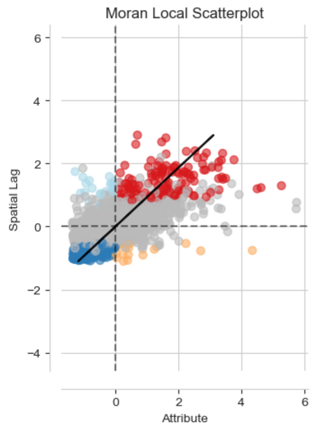
We estimate the statistics computationally: that is, for each point, we shuffle the map 1000 times, moving actual data values to randomly selected actual other station locations. This gives us a reference distribution for relationships of unreliable stations near other unreliable stations.

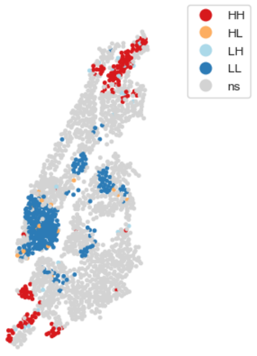
We need to choose a threshold for how unlikely we want to consider: we used 1%. The shaded pointes then represent stations whose structure of unreliable service, surrounded by other stations with unreliable service, is less than 1% likely under spatial randomness.

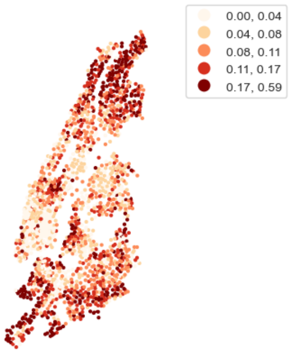
(This is functionally equivalent to a p-value and an alpha threshold, but true statisticians will chide that it is not a true p-value because it is estimated computationally instead of derived analytically)



Get a measure of significant hotspots



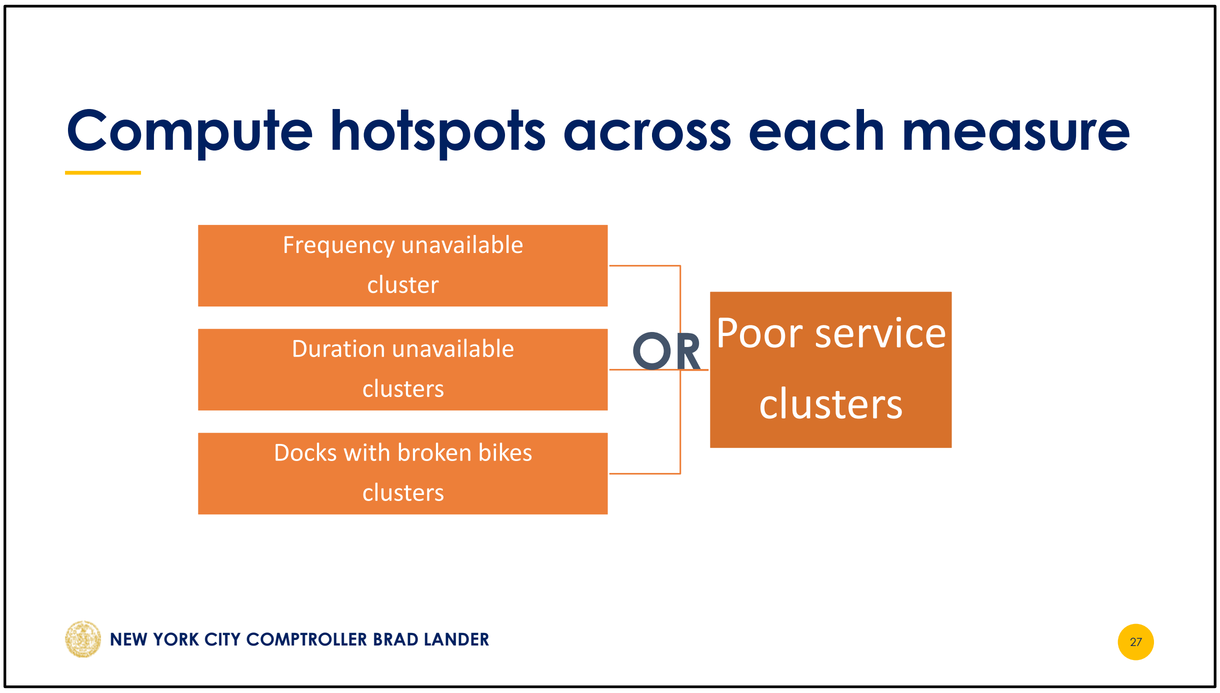




NEW YORK CITY COMPTROLLER BRAD LANDER

26

So, on the right is the continuous choropleth we stated with and in the center are the new hotspots that we just computed.



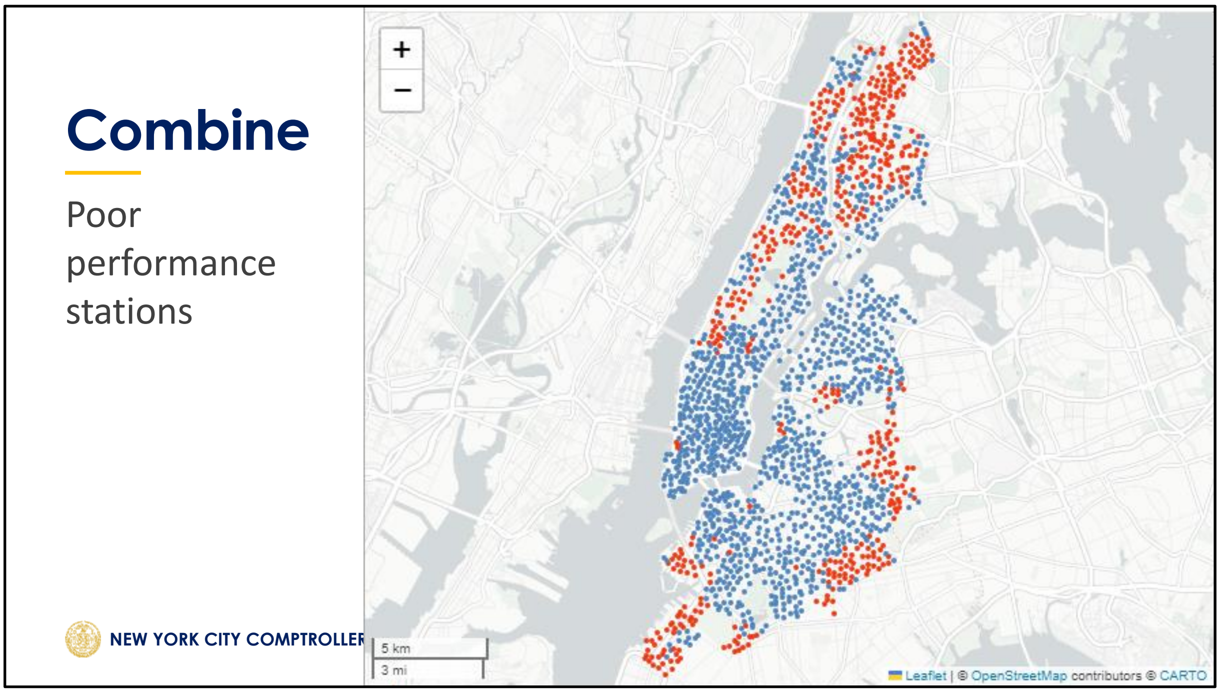
Compute hotspots across each measure

Frequency unavailable cluster
Duration unavailable clusters
Docks with broken bikes clusters
OR
Poor service clusters

NEW YORK CITY COMPTROLLER BRAD LANDER

27

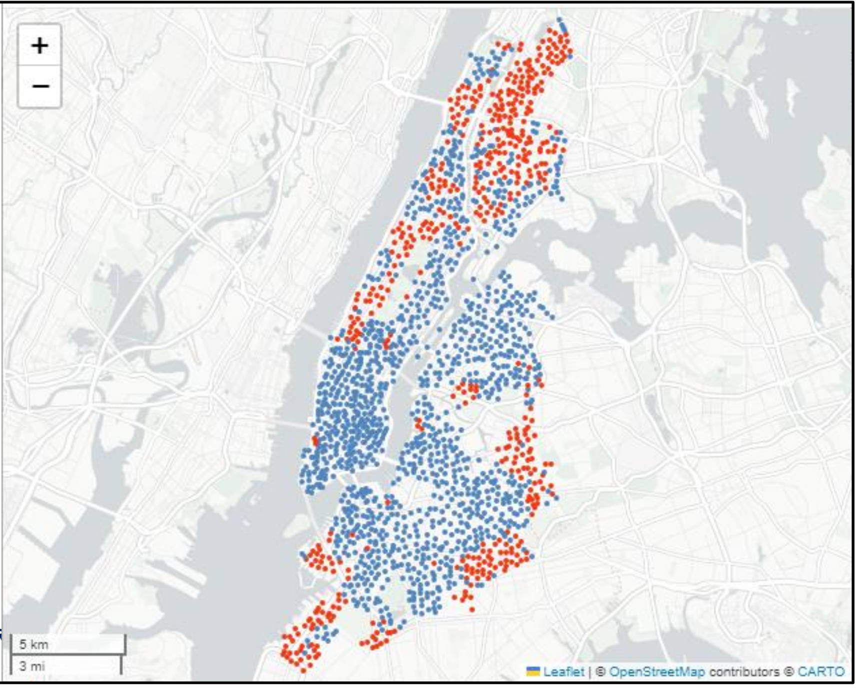
We run that same process across each of our three performance measures and count a station as one with poor service if it is a significant hotspot for any of the three measures.



Combine

Poor performance stations

NEW YORK CITY COMPTROLLER



Here is our result.

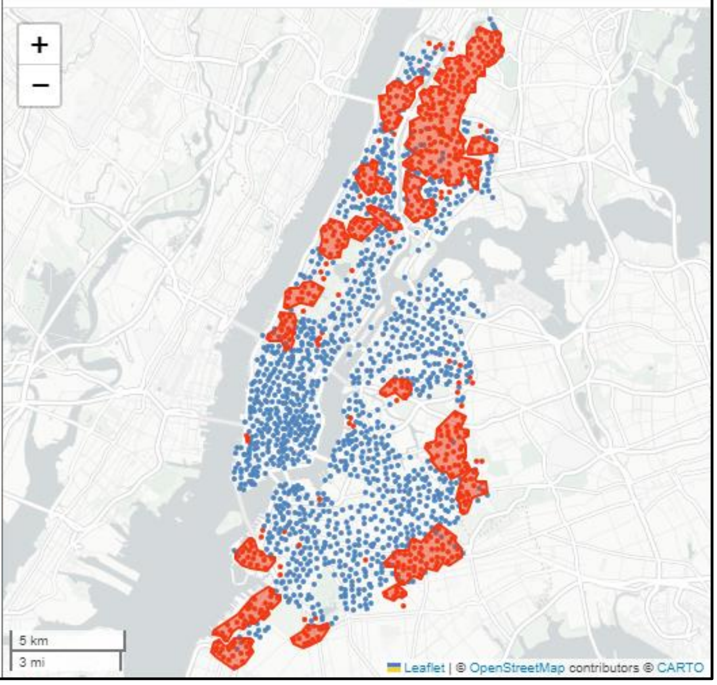
That concluded the spatial statistics, but we still need to do some map work. These are _points_ representing stations that are poor service hotspots. But we want _areas_ of poor service.

Find clusters

Keep groups with at least 5 stations within ¼ mile

Build concave hulls as boundaries

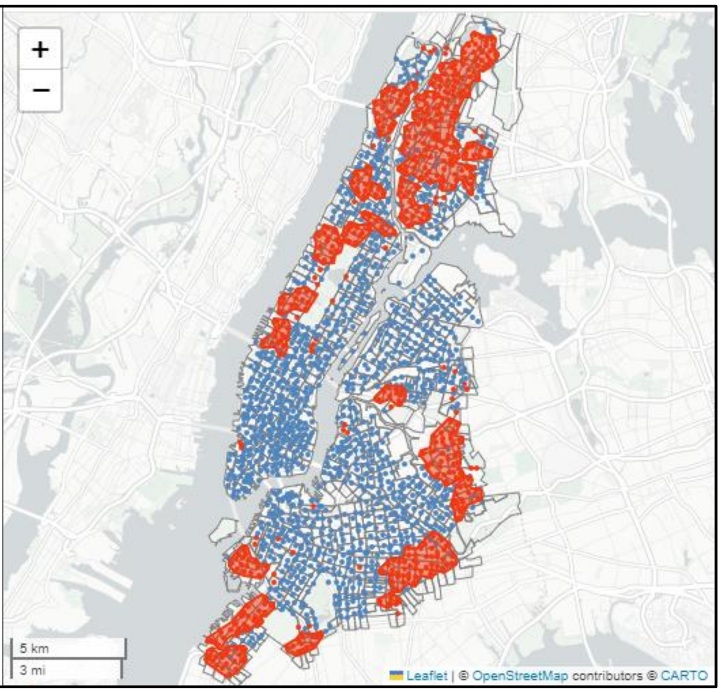
NEW YORK CITY COMPTROLLER BRAD LANDER



Build a network graph of all poor service stations, networked to all other poor service stations within ¼ mile
(creates many, disconnected networks)
Drop networks that have fewer than 5 stations

Create a concave hull linking the stations that are the ~outside border of each grouping
Buffer these borders 500 ft to create a poor service area

Join Tracts

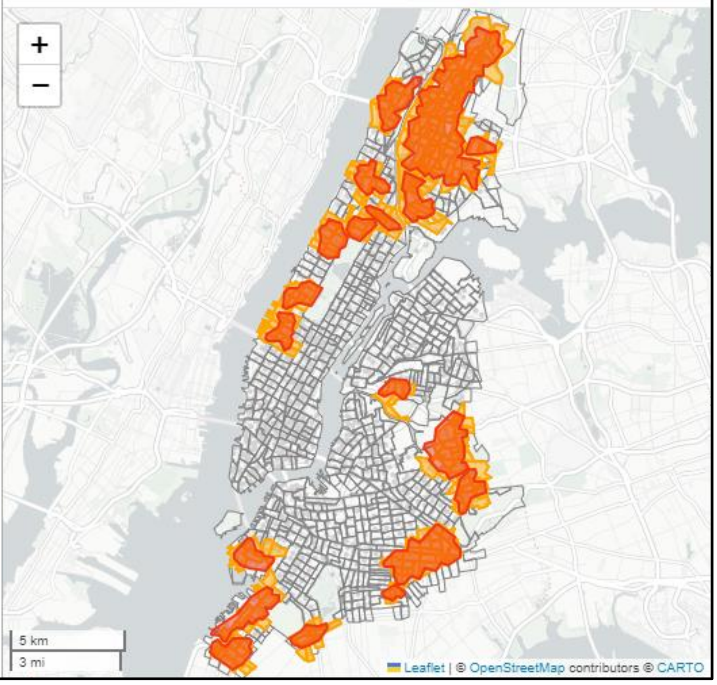


NEW YORK CITY COMPTROLLER BRAD LANDER

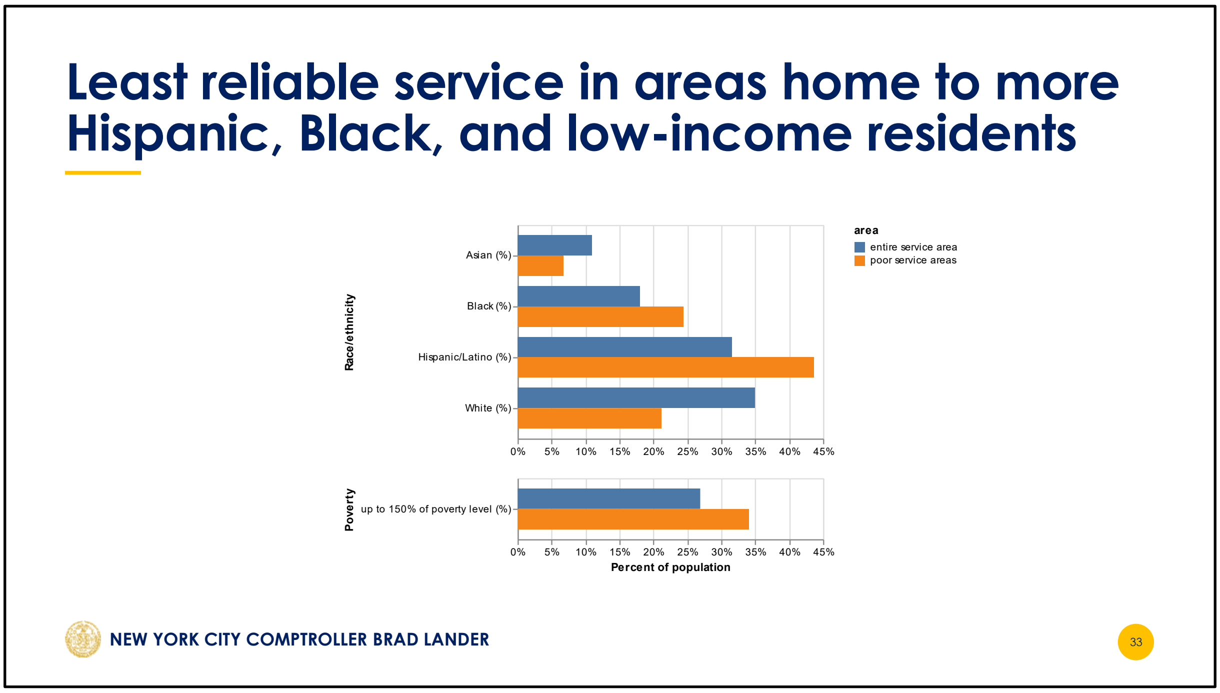
Then spatially join Census Tracts

Join Tracts

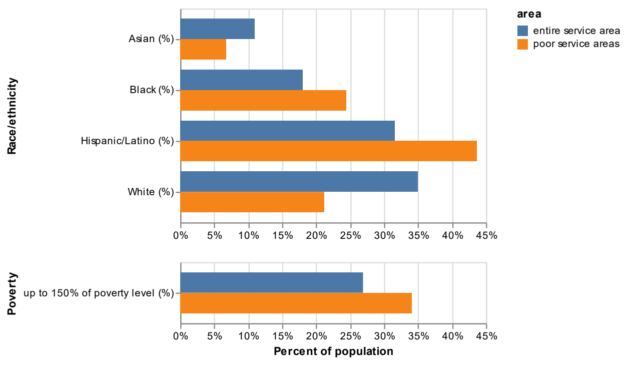
Tracts intersecting poor service clusters and entire service area



Find Census Tracts intersecting the entire service area and the poor service areas.



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



NEW YORK CITY COMPTROLLER BRAD LANDER

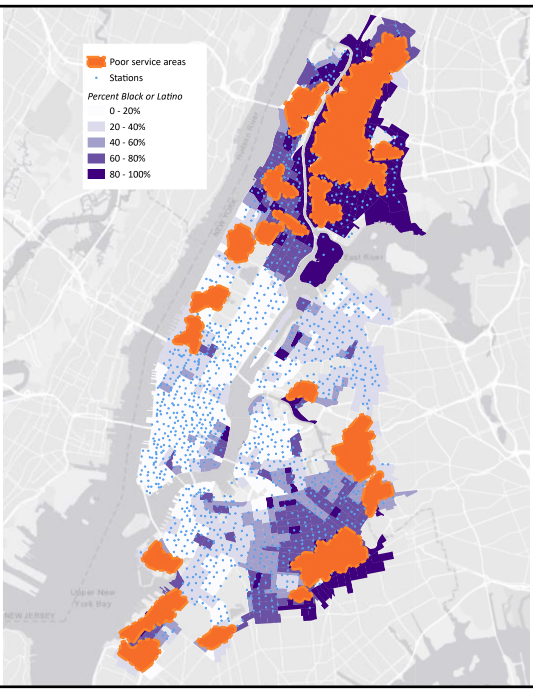
33

Aggregate demographics within the entire service area and the poor service areas and compare.

Clusters of least-reliable stations were in places home to more Hispanic and/or Latino, Black, and lower-income residents than the entire Citi Bike service area.

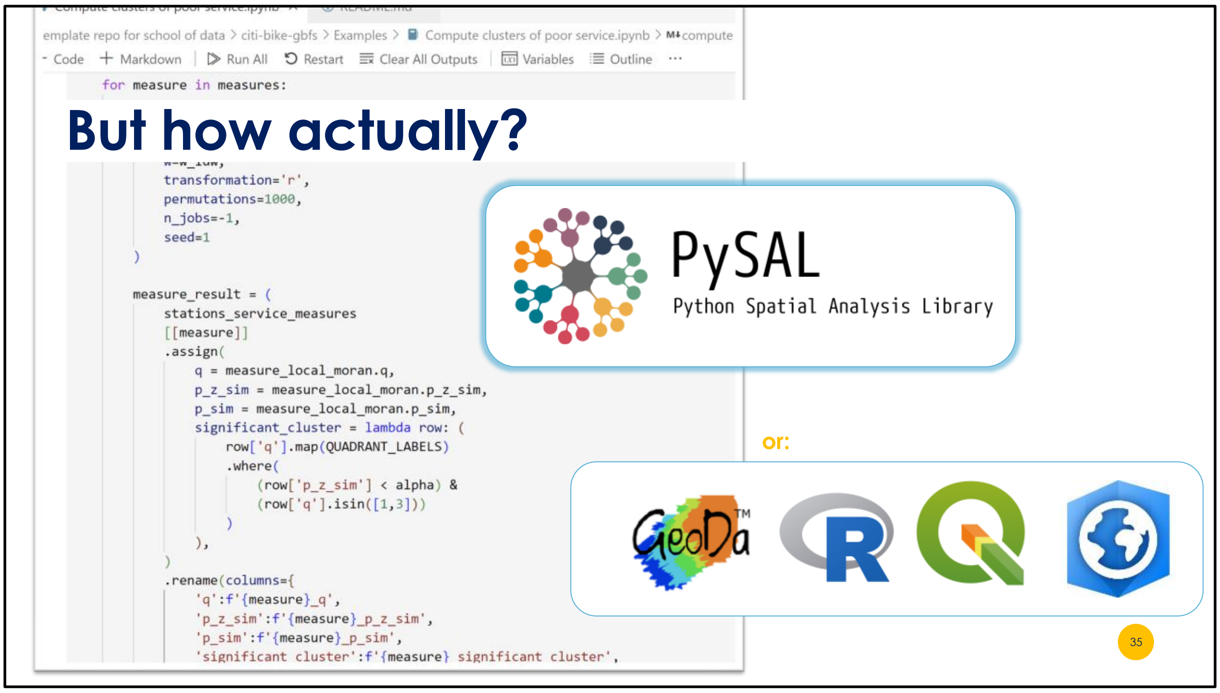
We _could_ extend this further with spatially-adjusted regression, but for our purposes, this was a clear and succinct demonstration of the inequity.

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



NEW YORK CITY COMPTROLLER BRAD LANDER

Clusters of least-reliable stations were in places home to more Hispanic. Latino, and/or Black residents than the entire Citi Bike service area.



But how actually?

emplate repo for school of data > citi-bike-gbfs > Examples > Compute clusters of poor service.ipynb > M↓compute

```
for measure in measures:
            w=w_idw,
            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',
```

or:

35

I showed maps, but the analysis was all done with code. Using Python, including common libraries pandas and geopandas, and pysal for spatial analysis

pysal is a great set of libraries: they run very fast and use numpy arrays and/or geopandas GeoDataFrames so fit well into other workflows

See my sample jupyter notebooks in the repo `Examples` folder to see how it was implemented.

But also, this is just math and has been implemented in R, QGIS, and ArcGIS also.

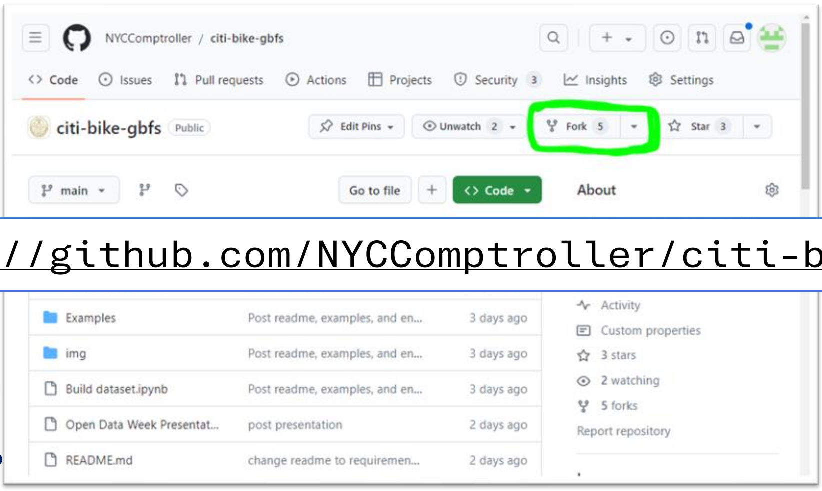
And also check out GeoDa, which is a standalone free desktop software specifically for spatial statistical analysis, including local indicators of spatial association/local Moran’s.

Q&A

36

You can too!

Fork the repo



Examples	Post readme, examples, and en...	3 days ago
img	Post readme, examples, and en...	3 days ago
Build dataset.ipynb	Post readme, examples, and en...	3 days ago
Open Data Week Presentat...	post presentation	2 days ago
README.md	change readme to requiremen...	2 days ago

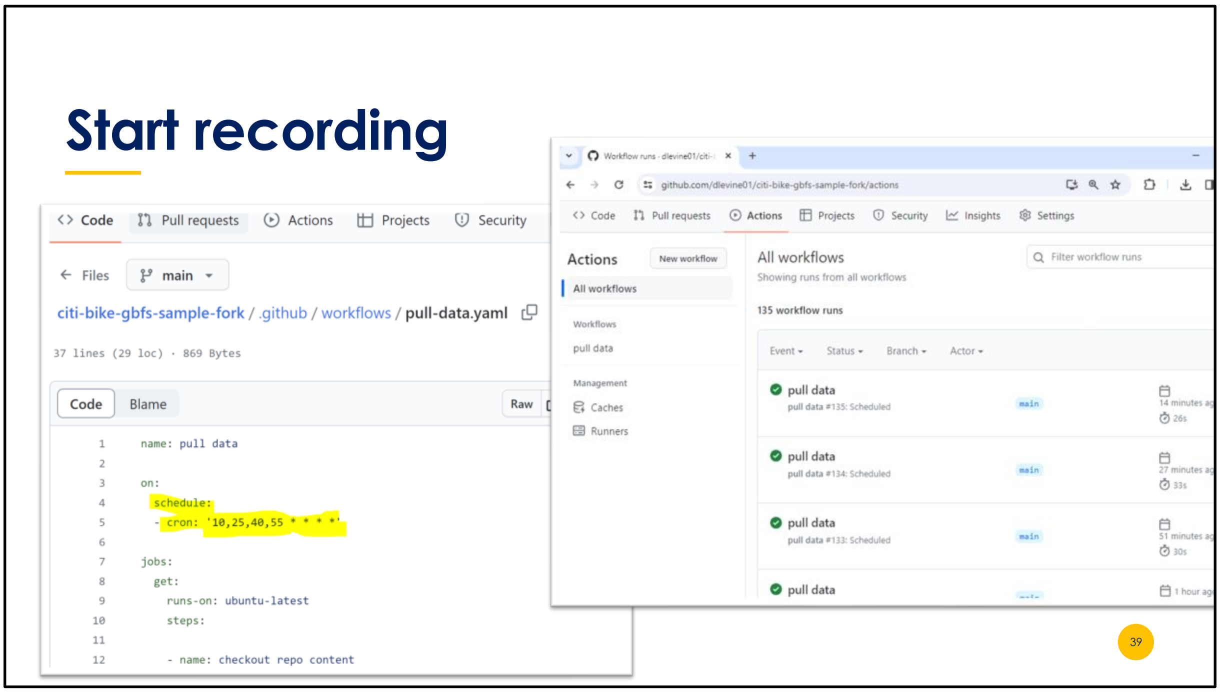
https://github.com/NYCComptroller/citi-bike-gbfs

NEW YO

38

These tools are here for you!
A template repo let's you start recording the data feed to GitHub and then build a dataset.

GitHub 101: go to the repo, click the 'Fork' button to make your own copy.



Start recording

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

37 lines (29 loc) · 869 Bytes

```yaml
name: pull data

on:
  schedule:
    - cron: '10,25,40,55 * * * *'

jobs:
  get:
    runs-on: ubuntu-latest
    steps:

      - name: checkout repo content
```

github.com/dlevine01/citi-bike-gbfs-sample-fork/actions

All workflows

Showing runs from all workflows

135 workflow runs

pull data pull data #135: Scheduled	main	14 minutes ag 26s
pull data pull data #134: Scheduled	main	27 minutes ag 33s
pull data pull data #133: Scheduled	main	51 minutes ag 30s
pull data		1 hour ag

39

All you need to do is set the interval for fetching data and turn on the action. Follow the detailed instruction in the README

(
FYI, the full action is defined by this yaml file. It defines how GitHub Actions will:
- set up a cloud compute instance
- checks out the repo
- set up python and install dependencies
- run shell commands, here just running a python .py script, which in turn gets the data, finds the last-updated timestamp, and saves new files to store the snapshot
- commits and pushed the new files back to the repo on GutHub
)

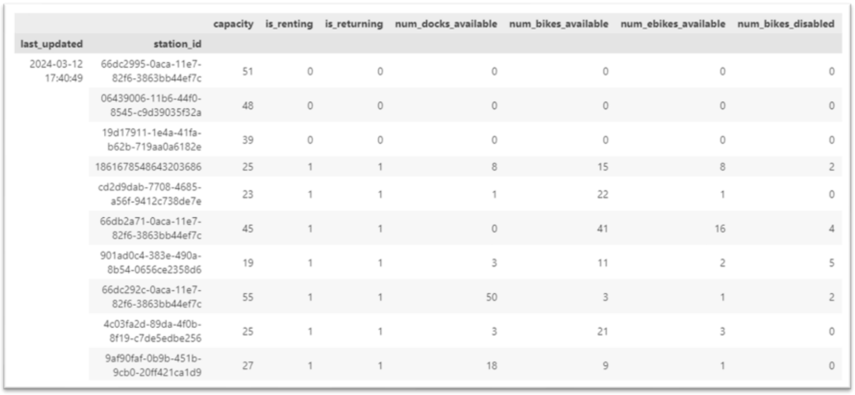
GitHub Actions is great because it is a completely free cloud compute instance you can set up to get and store data on a schedule.

But note a limitation: your code has to wait in line, it won't necessarily run at exactly the scheduled time. That's a bit of a problem for this analysis, when we would prefer to have all the data from even sample times and if we miss a data snapshot, it's gone forever.

(knowing this, we built in so checks to throw out durations that go over 40 minutes without new data)

If you want to get exact timing, you can run this on a dedicated cloud instance or use crontab to schedule the action on a Mac OS/unix/linux machine.

Build dataset

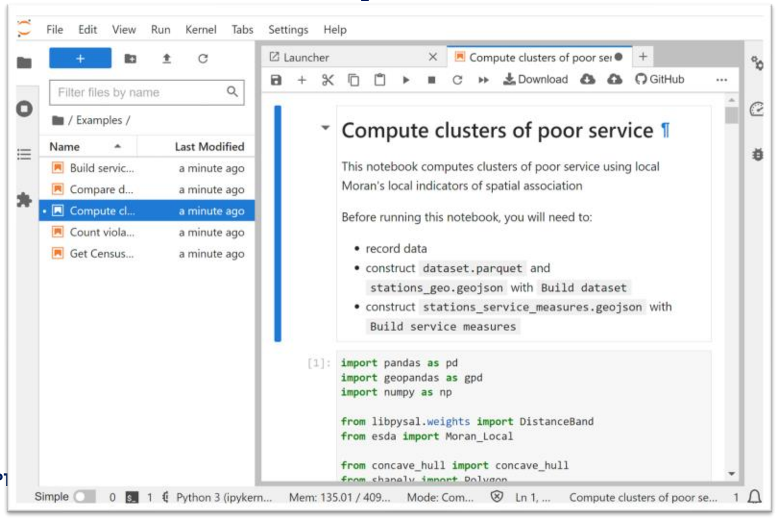


		capacity	is_renting	is_returning	num_docks_available	num_bikes_available	num_ebikes_available	num_bikes_disabled
last_updated	station_id							
2024-03-12 17:40:49	66dc2995-0aca-11e7-82f6-3863bb44ef7c	51	0	0	0	0	0	0
	06439006-11b6-44f0-8545-c9d39035f32a	48	0	0	0	0	0	0
	19d17911-1e4a-41fa-b62b-719aa0a6182e	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

NEW YORK CITY COMPTROLLER BRAD LANDER

Once you have recorded some data, clone the repo (with the data now stored on GitHub Actions), then run the `Build dataset` notebook to stick the data snapshot files together into a single dataset.

Review or extend analysis code


File Edit View Run Kernel Tabs Settings Help
Filter files by name
/ Examples /
Launcher
Compute clusters of poor ser
Download
GitHub
Simple 0 1 Python 3 (ipykern... Mem: 135.01 / 409... Mode: Com... Ln 1, ... Compute clusters of poor se... 1

Name	Last Modified
Build servic...	a minute ago
Compare d...	a minute ago
Compute cl...	a minute ago
Count viola...	a minute ago
Get Census...	a minute ago

Compute clusters of poor service ¶

This notebook computes clusters of poor service using local Moran's local indicators of spatial association

Before running this notebook, you will need to:

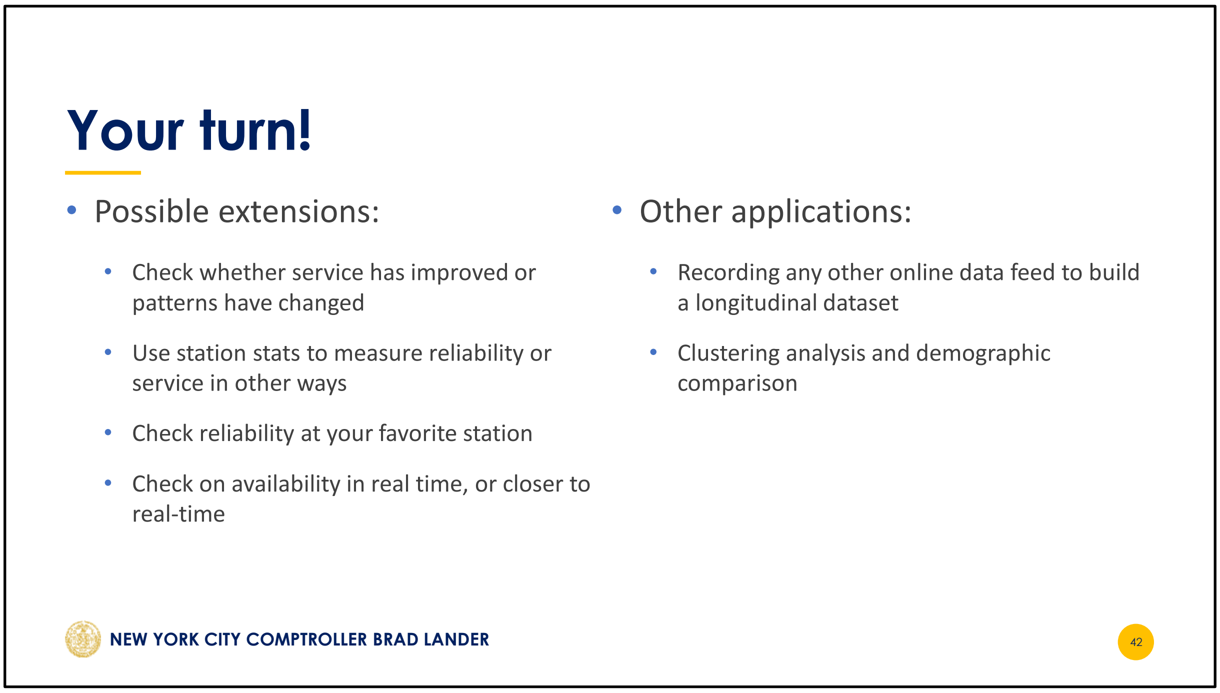
- record data
- construct `dataset.parquet` and `stations_geo.geojson` with `Build dataset`
- construct `stations_service_measures.geojson` with `Build service measures`

```
[1]: import pandas as pd
     import geopandas as gpd
     import numpy as np

     from libpysal.weights import DistanceBand
     from esda import Moran_Local

     from concave_hull import concave_hull
     from shapely import Polygon
```

NEW YORK CITY COMPT

41

The code we used for this analysis is in Jupyter notebooks in the Examples folder of the repo. Clone the repo to explore locally (with data you record). Or switch to the ‘sample-data’ branch of the repo with includes some data recoded in Mar 2024.



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

NEW YORK CITY COMPTROLLER BRAD LANDER

42

Thank you!

Stay in touch
at beta.nyc/links

βetaNYC

NYC SCHOOL OF DATA

nycsodata24.sched.com #nycSOdata #opendataweek

NYC SCHOOL OF DATA
OPEN DATA WEEK
Powered by NYC OpenData