Part 3: Map APIs

Wenzheng Li

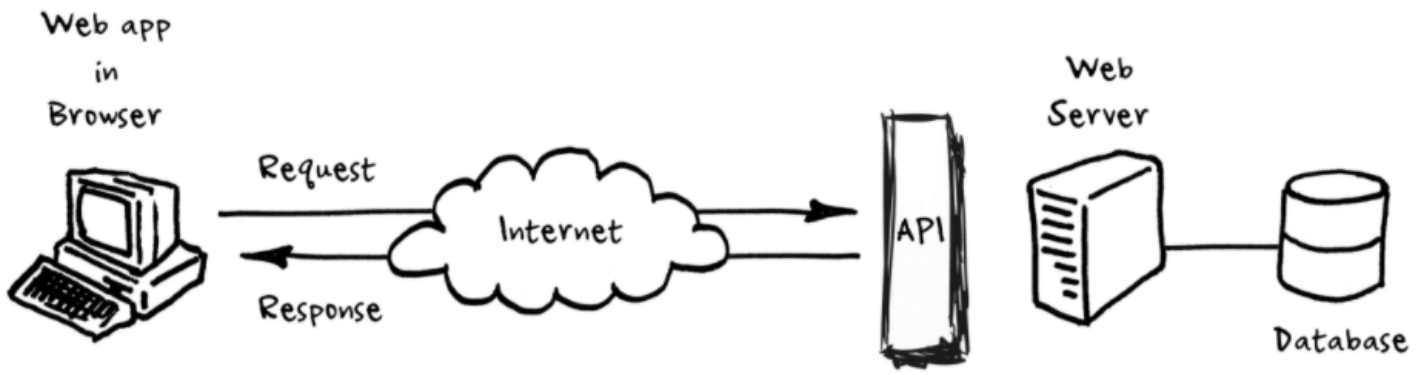
Schedule for today

- Part 1: The applications of Google Map APIs
  - the geocoding API and distance matrix API
- Part 2: Use OSMnx package to obtain OpenStreetMap dataset
- Part 3: Interactive Mapping

What is a web-based API (Application Programming Interface?

An API allows one program to talk to another program. Many websites or services provide an API so you can query for information or download datasets in an automated way.

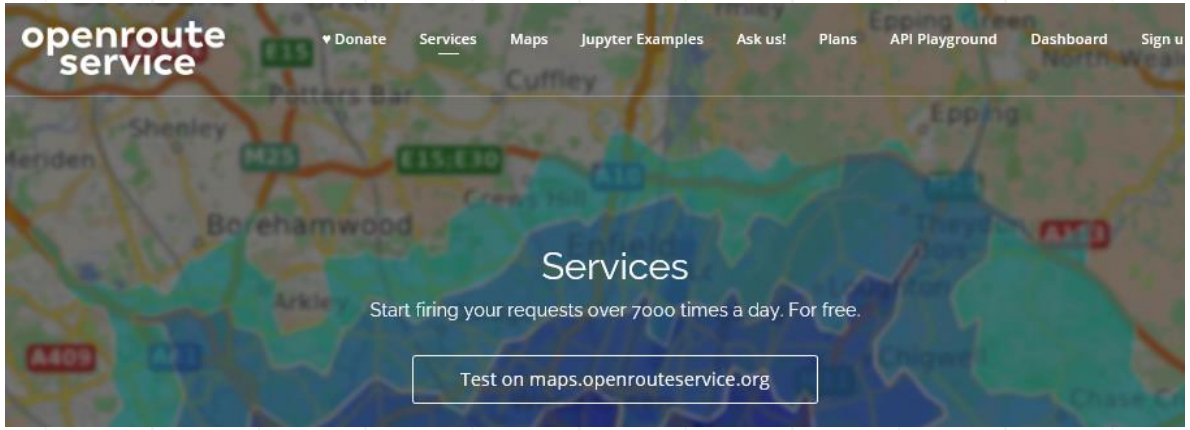
- ✓ We send an API a request detailing the information we want.
- ✓ API can return data in response to a request made by a client.



An API is not a database. It is an access point to an app that can access a database.

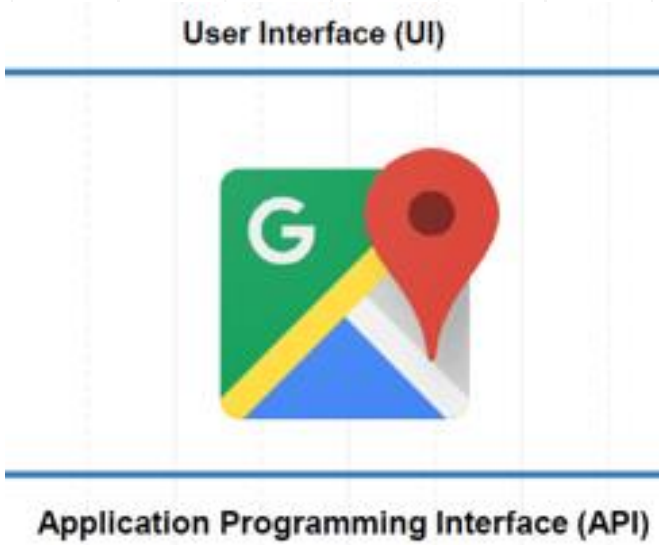
How API works. Source: Medium

Map APIs



Map APIs for spatial analysis:

- ✓ Getting travel navigation information (directions/distance/duration)
- ✓ Route optimization (shortest distance, least time spent)
- ✓ Geocoding (address to longitude and latitude)
- ✓ Downloading data (obtain spatial entities from OSM and Google MyPlace)
- ✓ Getting real-time weather data
- ✓....



```
// Compute the trend of nighttime lights from DMSP.

// Add a band containing image date as years since 1991.
function createTimeBand(img) {
  var year = ee.Date(img.get('system:time_start')).get('year').subtract(1991);
  return ee.Image(year).byte().addBands(img);
}

// Fit a linear trend to the nighttime lights collection.
var collection = ee.ImageCollection('NOAA/DMSP-OLS/NIGHTTIME_LIGHTS')
    .select('stable_lights')
    .map(createTimeBand);
var fit = collection.reduce(ee.Reducer.linearFit());
```

Commonly used Map APIs:

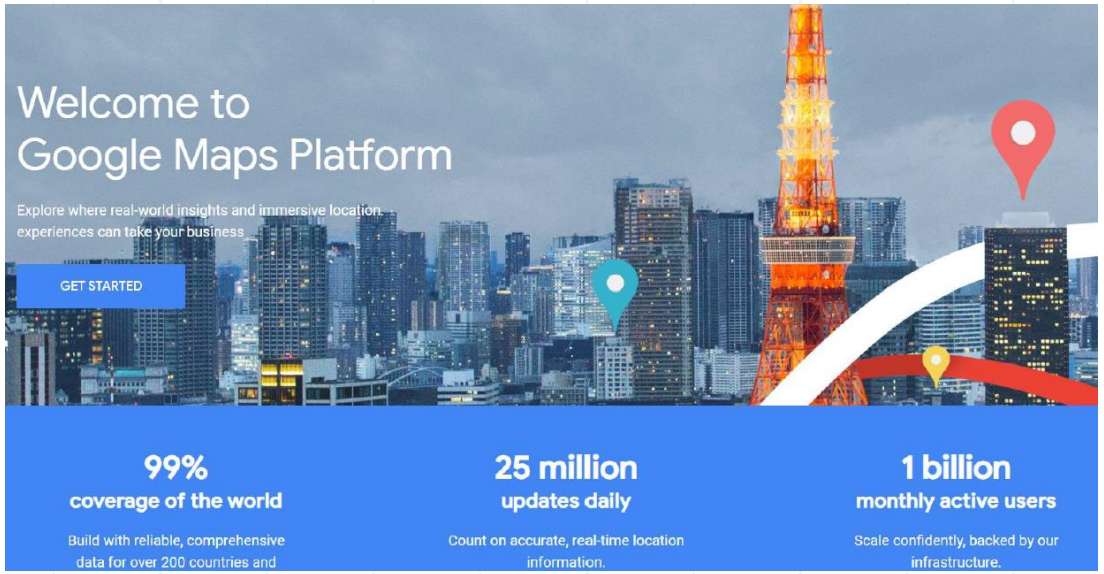
- ✓ OpenStreetMap Overpass APIs
- ✓ Google Maps APIs
- ✓ Google Earth Engine API
- ✓ Baidu APIs

Google Map API (googlemaps library)

The Google Maps library provides us a list of applications using Google Map APIs:

- Directions API
- Distance Matrix AI
- Elevation API
- Geocoding API
- Geolocation API
- Time Zone API
- Roads API
- Places API
- Maps Static API

for more: https://github.com/googlemaps/google-maps-services-python



Installation of Google Map API:

create a Google API account with an associated API Key. Please get your own API keys following this instruction: https://developers.google.com/maps/gmp-get-started#api-key.

the Python library for Google API is called *googlemaps*. To install the library, we type !*pip install googlemaps*

Once you have created a billing account and project you are eligible for the Google Cloud Platform $300 free trial and Google Maps Platform recurring $200 monthly credit. https://cloud.google.com/maps-platform/pricing/sheet/. Geocoding API: 0.005 USD per call

Basic Applications

In this section, we will introduce two basic applications

- the **Geocoding API** which returns the corresponding longitude and latitude in terms of the address

Geocoding

'1225 Gerard Avenue Bronx, New York 10452' → WGS84: (-73.92 40.83621)

	address
0	292 EAST THIRD STREET
1	269 EAST 7TH STREET
2	234 EAST 2ND STREET
3	215 EAST 3RD STREET
4	209 EAST 4 STREET
5	168 EAST 7TH STREET
6	182 AVENUE A
7	506 EAST 13 STREET
8	543 EAST 12TH STREET
9	EAST 9TH STREET

User Interface (UI)

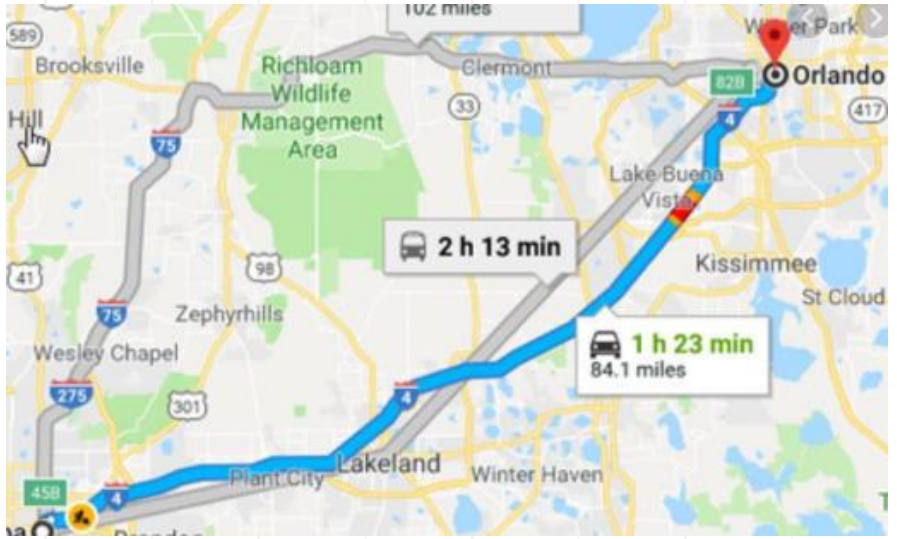
Application Programming Interface (API)

longitude	latitude
-73.979781	40.721129
-73.977156	40.723515
-73.982108	40.721635
-73.982280	40.722538
-73.983623	40.723936
-73.981924	40.725162
-73.981228	40.728710
-73.980370	40.729315
-73.979306	40.728478
-73.985810	40.728578

Basic Applications

In this section, we will introduce two basic applications

- the **Distance Matrix API** which calculates the travel distance/time from one location to another in terms of travel modes.



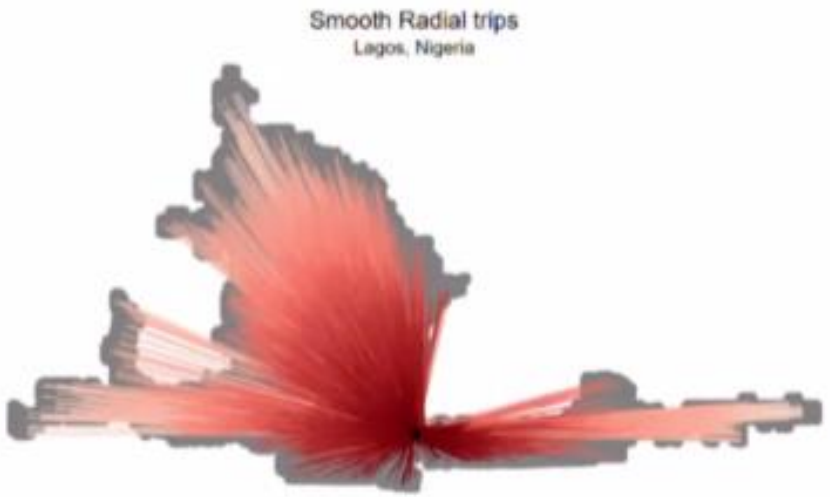
	origin	destination	driving distance	driving duration	bicycling distance	bicycling duration
0	New York, NY, USA	Los Angeles, CA, USA	4,494 km	1 day 17 hours	4,720 km	10 days 5 hours
1	New York, NY, USA	Chicago, IL, USA	1,272 km	12 hours 11 mins	1,471 km	3 days 9 hours
2	New York, NY, USA	Houston, TX, USA	2,618 km	1 day 0 hours	2,910 km	6 days 11 hours
3	New York, NY, USA	Phoenix, AZ, USA	3,873 km	1 day 12 hours	4,111 km	8 days 22 hours
4	Los Angeles, CA, USA	Chicago, IL, USA	3,243 km	1 day 5 hours	3,449 km	7 days 5 hours
5	Los Angeles, CA, USA	Houston, TX, USA	2,490 km	22 hours 16 mins	2,582 km	5 days 8 hours
6	Los Angeles, CA, USA	Phoenix, AZ, USA	600 km	5 hours 38 mins	661 km	1 day 10 hours
7	Chicago, IL, USA	Houston, TX, USA	1,744 km	16 hours 13 mins	1,823 km	3 days 20 hours
8	Chicago, IL, USA	Phoenix, AZ, USA	2,822 km	1 day 2 hours	2,838 km	5 days 23 hours
9	Houston, TX, USA	Phoenix, AZ, USA	1,889 km	16 hours 48 mins	1,957 km	4 days 1 hour

More Advanced Applications:

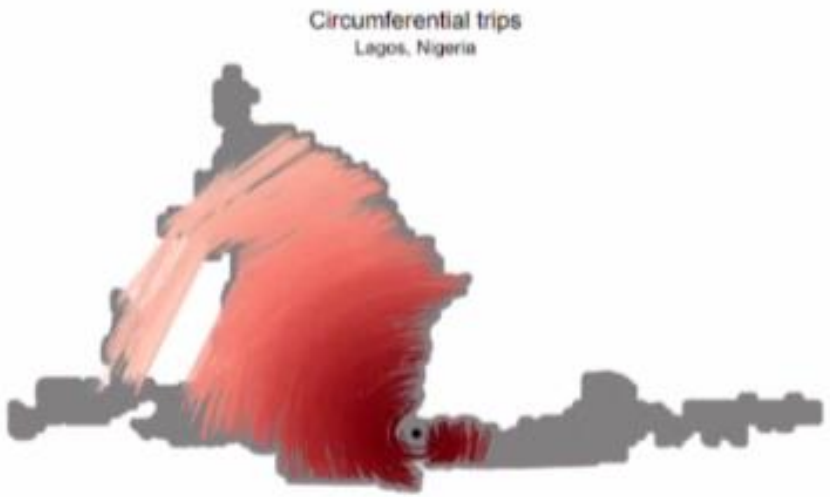
Akbar et al, 2018:

- Obtain POIs from Google MyPlace
- 900M simulated “real time traffic” trips from Google Map
- Compare the real-traffic time/distance with the hypothetic state of no traffic
- Calculate traffic congestion for global cities

Illustration: Lagos in Nigeria



Radial trips

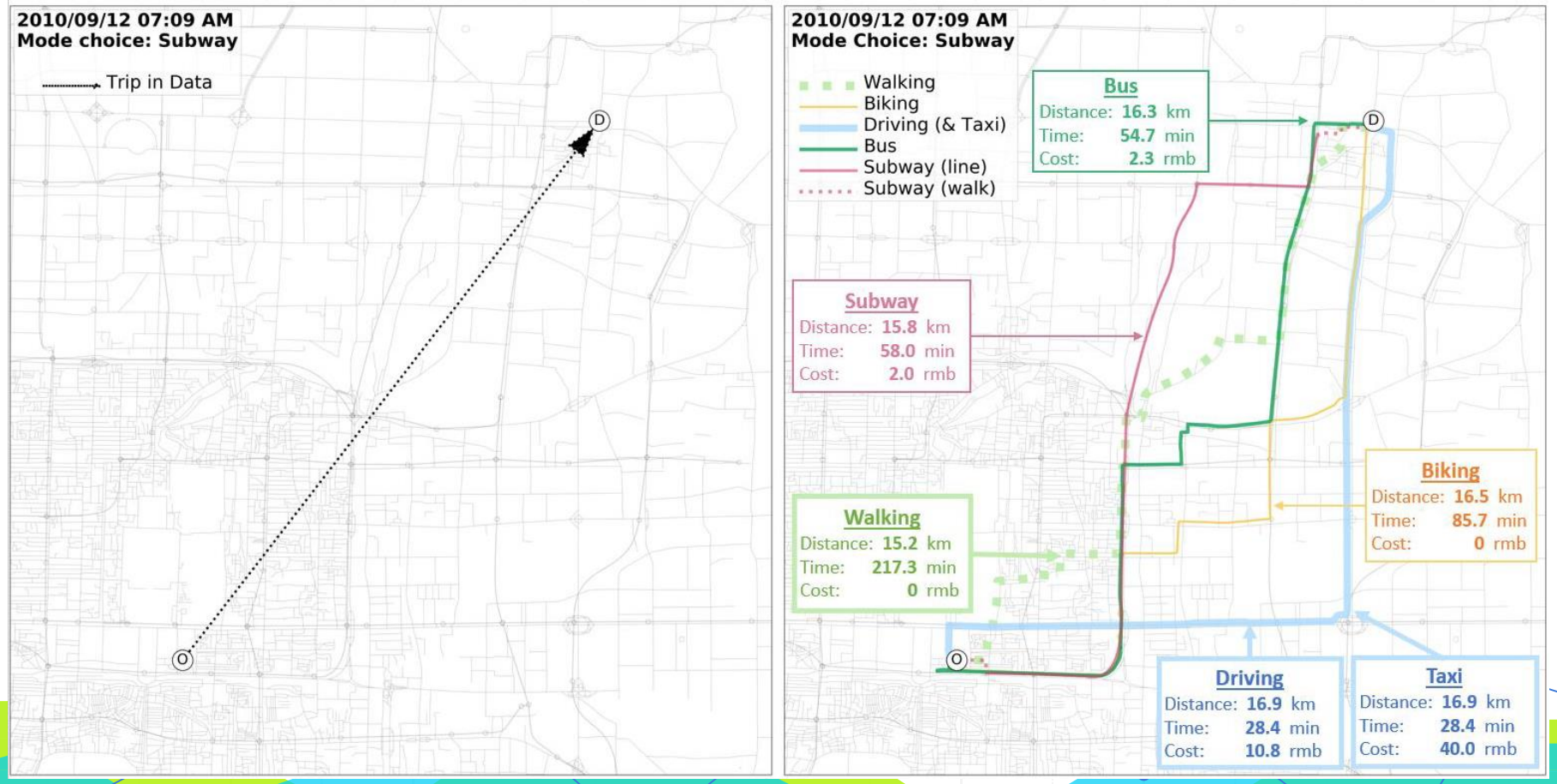


Circumferential trips

More Advanced Applications:

Li et al, 2020:

- obtain the travel time, distance and costs based on different travel modes (driving, walking, bus, biking, and walking) for each OD pair
- Value travel of time in Beijing



In today's class:

1. Google Map API Geocoding application:
   go to the Lab folder and load the COVID-19 NYC Vaccination sites dataset (COVID19_Vaccination.xlsx).

```
# import the googlemaps library
import googlemaps

# create a Client object
gmaps = googlemaps.Client(key='AIzaSyDfN_8wTkwEC2ZHFzKDJIpCTbKG-rtYgks')
```

```
# Geocoding the first address
geocode_result = gmaps.geocode(lst_address[0])
geocode_result
```

```
print(geocode_result[0]['formatted_address'],
      geocode_result[0]['geometry']['location']['lat'],
      geocode_result[0]['geometry']['location']['lng'])
```

2. Distance Matrix API
Create city pairs: OD matrix of top five US cities

```
from itertools import combinations
lst_city = ["New York City", "Los Angeles", "Chicago", "Houston", "Phoenix"]
city_pair = list(combinations(lst_city, 2)) # Nonrepeating city pairs
```

```
# importing googlemaps module
import googlemaps

# Requires API key
gmaps = googlemaps.Client(key='AIzaSyDfN_8wTkwEC2ZHFzKDJIpCTbKG-rtYgks')

# Requires cities name
my_dist = gmaps.distance_matrix(city_pair[0][0],city_pair[0][1], mode="driving")

# Printing the result
my_dist
```

```
# obtain the destination
print(my_dist["destination_addresses"][0])

# obtain the driving distance (in km)
print(my_dist["rows"][0]['elements'][0]['distance']['text'])

# obtain the driving duration (in days and hours)
print(my_dist["rows"][0]['elements'][0]['duration']['text'])
```

Part 2: Collecting OpenStreetMap Dataset using OSMnx

Wenzheng Li, Yujin Lee

OSMnx?

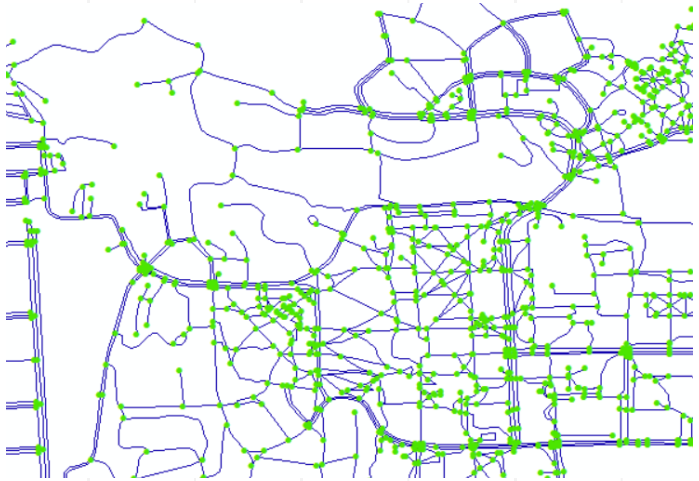
Author: **Geoff Boeing** Urban planning and spatial analysis professor at USC

OSMnx features:

✓ retrieve, analyze, and visualize street networks and other spatial data from OpenStreetMap.

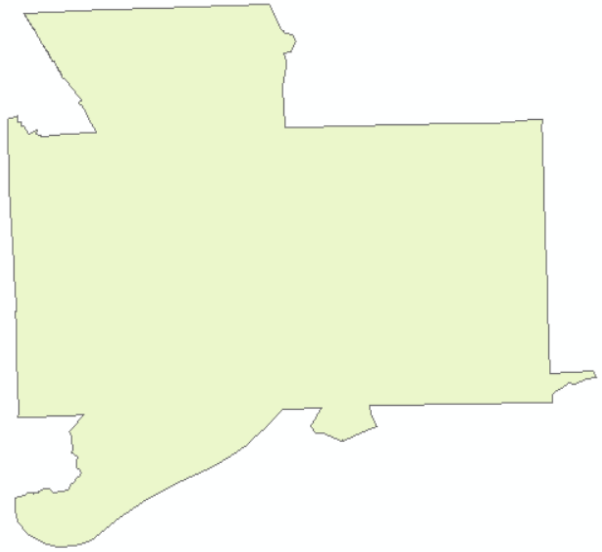
OSMnx is on GitHub and you can install it with conda.

- Download street networks anywhere in the world with a single line of code
- Download other infrastructure types, place boundaries, building footprints, and points of interest
- Download by city name, polygon, bounding box, or point/address + network distance
- Download drivable, walkable, bikeable, or all street networks
- Download node elevations and calculate edge grades (inclines)
- Impute missing speeds and calculate graph edge travel times
- Simplify and correct the network's topology to clean-up nodes and consolidate intersections
- Fast map-matching of points, routes, or trajectories to nearest graph edges or nodes
- Save networks to disk as shapefiles, GeoPackages, and GraphML
- Save/load street network to/from a local .osm xml file
- Conduct topological and spatial analyses to automatically calculate dozens of indicators
- Calculate and visualize street bearings and orientations
- Calculate and visualize shortest-path routes that minimize distance, travel time, elevation, etc
- Visualize street network as a static map or interactive leaflet web map
- Visualize travel distance and travel time with isoline and isochrone maps
- Plot figure-ground diagrams of street networks and/or building footprints

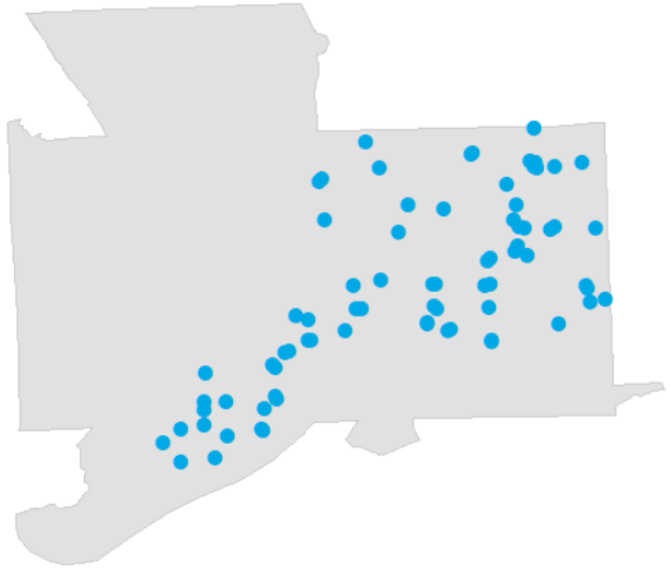


Street Network (edges and nodes)

Building Footprints



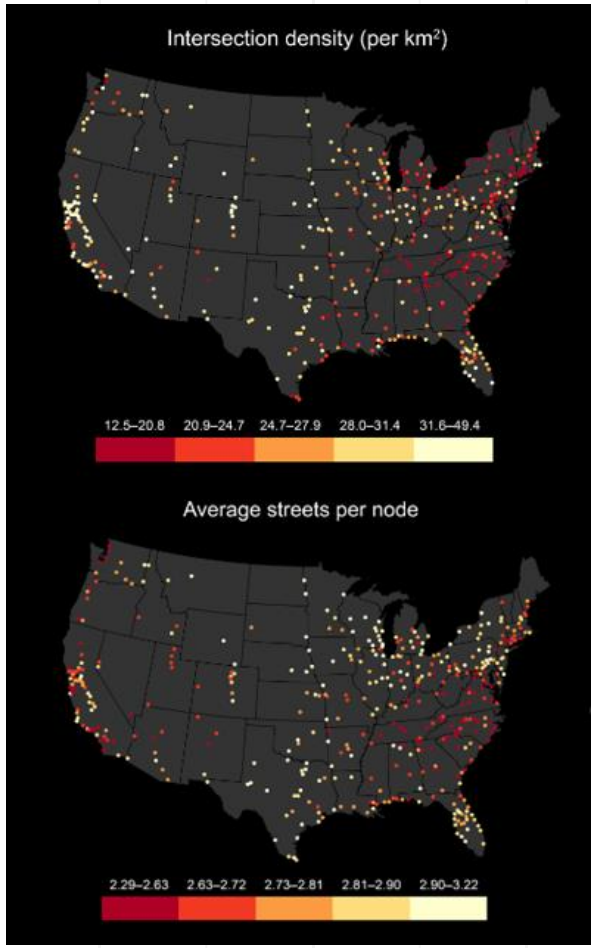
City Boundary



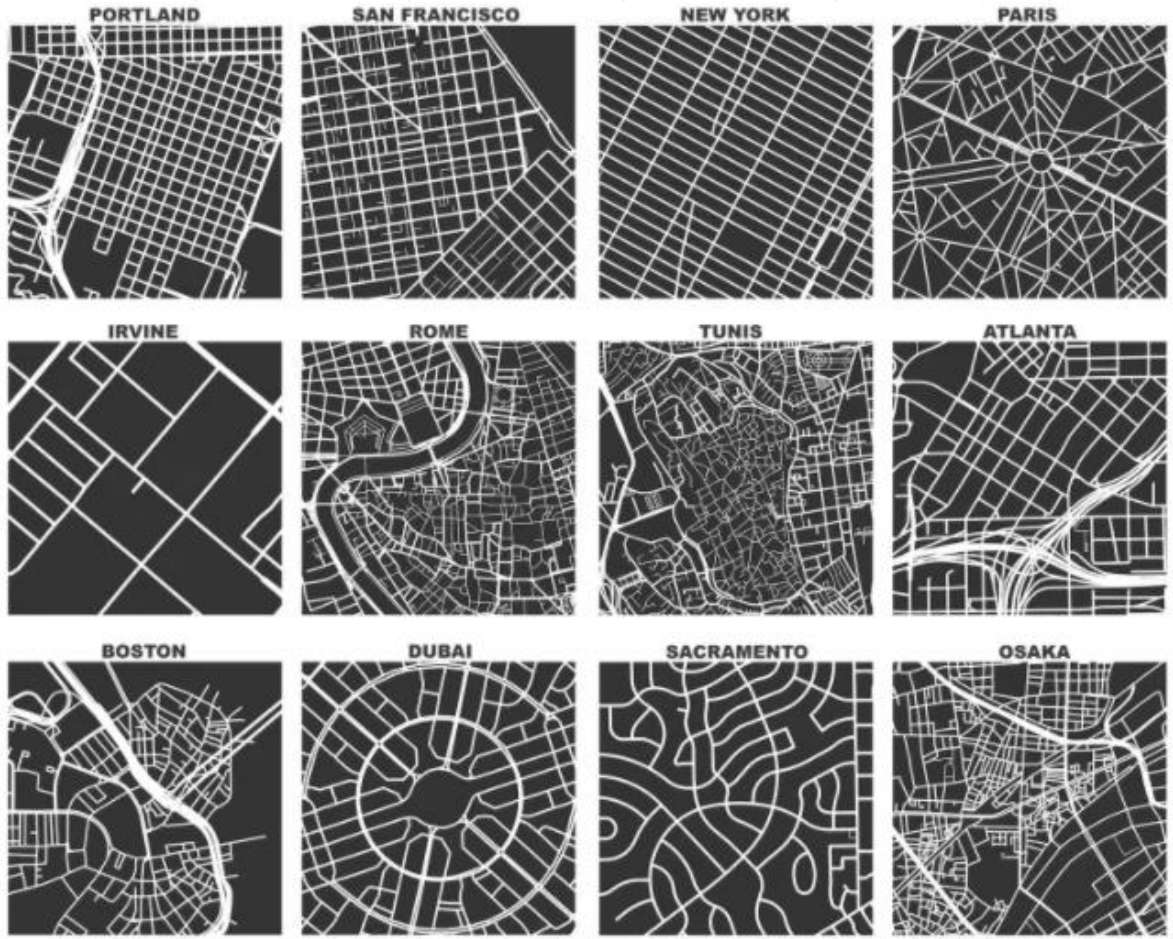
Bus stops

Interactive maps using folium

Advanced Applications:



Intersection density and average streets per node per urbanized area in the contiguous US



Cities' urban forms (Allan Jacobs, Geoff Boeing reproduce these figures using OSMnx)

Boeing, G. 2020. "A Multi-Scale Analysis of 27,000 Urban Street Networks: Every US City, Town, Urbanized Area, and Zillow Neighborhood." *Environment and Planning B: Urban Analytics and City Science*, 47 (4)

What we will do using OSMnx today?

(1) download street network dataset (both edges and nodes) of city of Ithaca

```
ox.graph_from_place(“Ithaca, New York, USA”)
nodes, edges = ox.graph_to_gdfs(graph) // convert to gdf
```

(2) download the boundary of City of Ithaca

```
ox.geocode_to_gdf (“Ithaca, New York, USA”)
```

(3) download building footprints

```
ox.geometries_from_place(query = “Ithaca, New York, USA”, tags = {'building': True})
```

(4) download POIs: bus stops

```
ox.geometries_from_place(query = “Ithaca, New York, USA”, tags = {'highway':'bus_stop’})
```

(6) visualize the OSMnx dataset

Questions?

More Readings: https://geoffboeing.com/