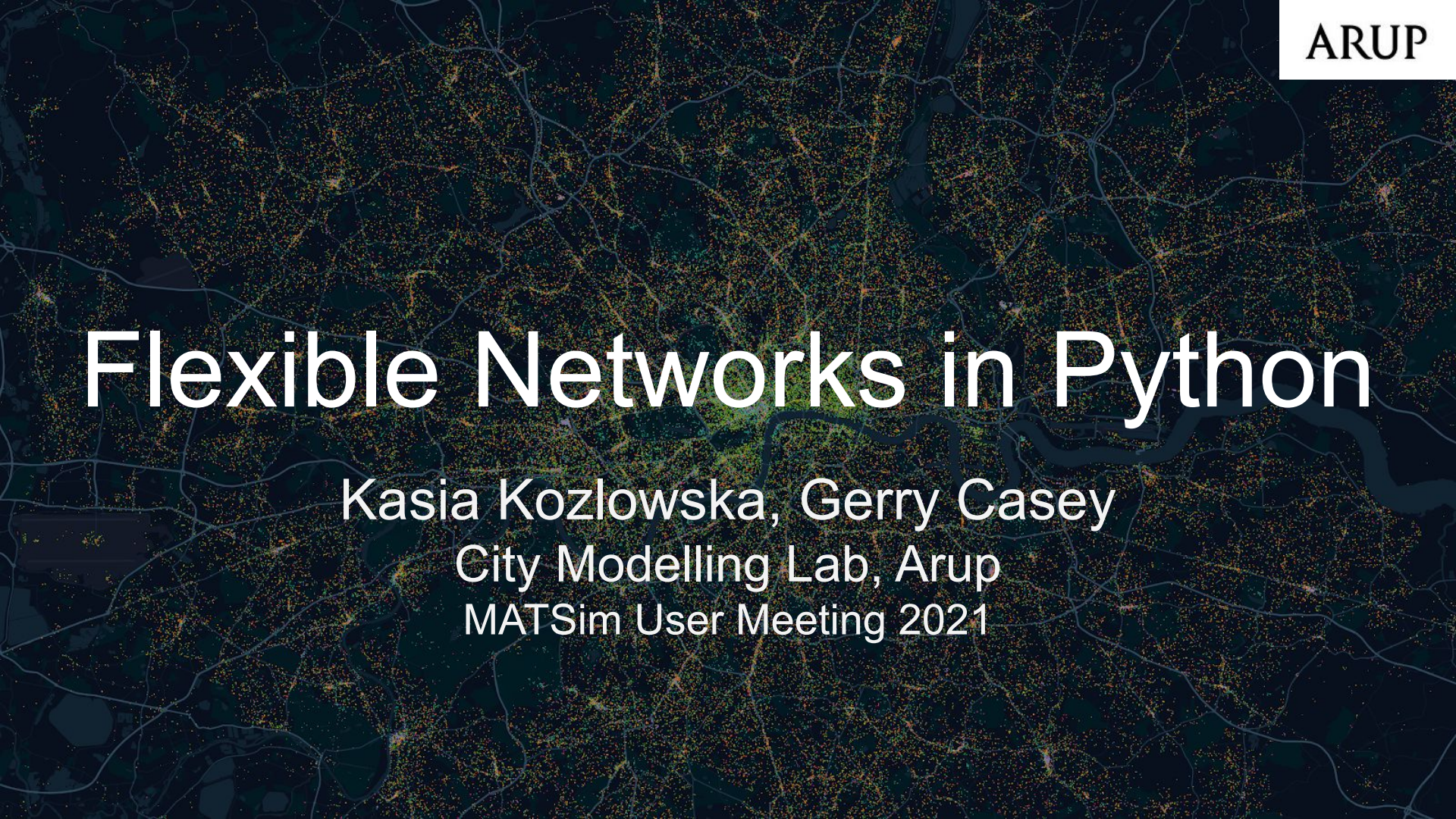


Flexible Networks in Python

Kasia Kozlowska, Gerry Casey
City Modelling Lab, Arup

MATSim User Meeting 2021

Hello from City Modelling Lab

Gerry Casey

Claire Fram

Fred Shone

Michael Fitzmaurice

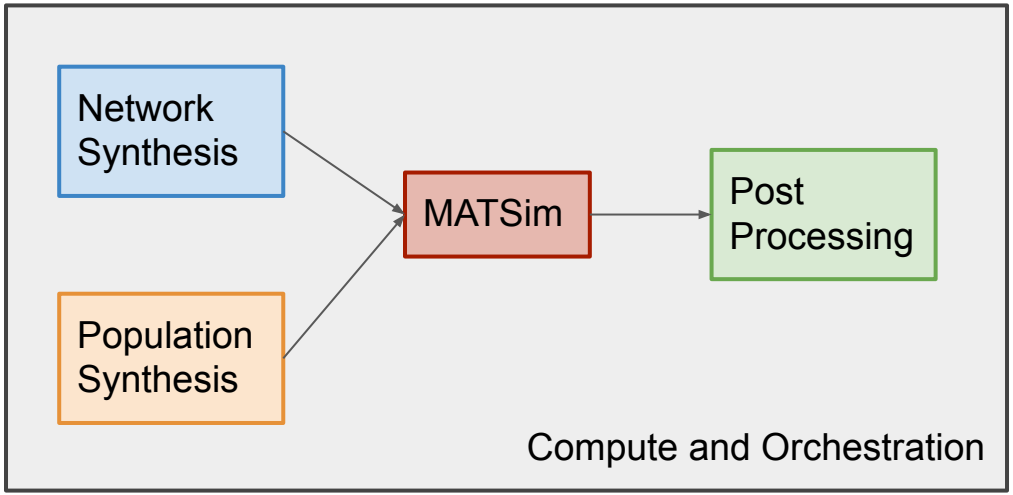
Rory Sedgwick

Theodore Chatziioannou

Kasia Kozlowska

medium.com/arupcitymodelling

Our framework



GeNet - Network Scenario Generator

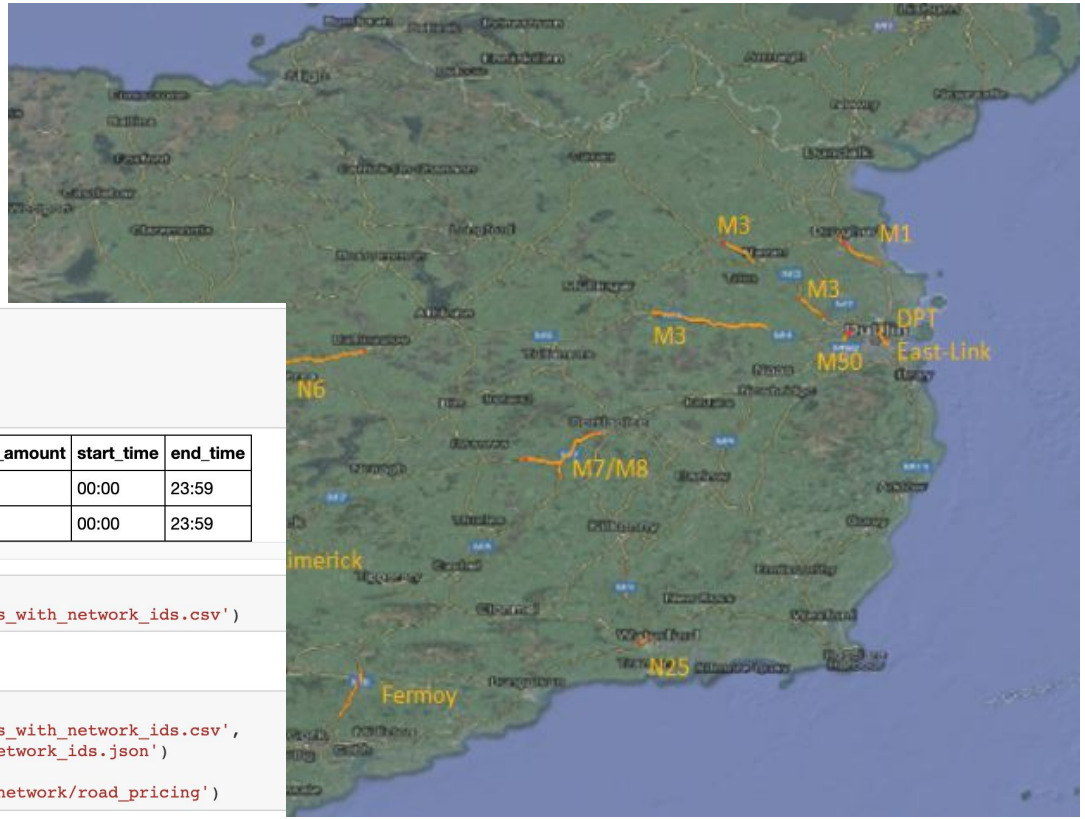
Open-source project: https://github.com/arup-group/genet

Goals:

- In-memory Python representation of a multimodal transport network, schedule and vehicles
  - Access via format-specific read & write methods
- Use to generate auxiliary files, e.g. for road pricing
- Modify network
  - Add/Remove/Modify data for graph links, nodes, transit stops, routes, services
  - Keep record of the changes
- Validate + Visualise

Using Network - Road Pricing

- We use OSM IDs stored on the network links to decide on the toll
- intermediate CSV format, for ease of use, which then gets written to xml



In [8]:
```
df_tolls['vehicle_type'] = 'type2'
df_tolls['toll_amount'] = '2.9'
df_tolls['start_time'] = '00:00'
df_tolls['end_time'] = '23:59'
df_tolls.head()
```

Out[8]:

	osm_ids	osm_refs	osm_names	network_id	vehicle_type	toll_amount	start_time	end_time
0	26997928	A400	Charing Cross Road	True	type2	2.9	00:00	23:59
1	546461337	A3211	Byward Street	True	type2	2.9	00:00	23:59

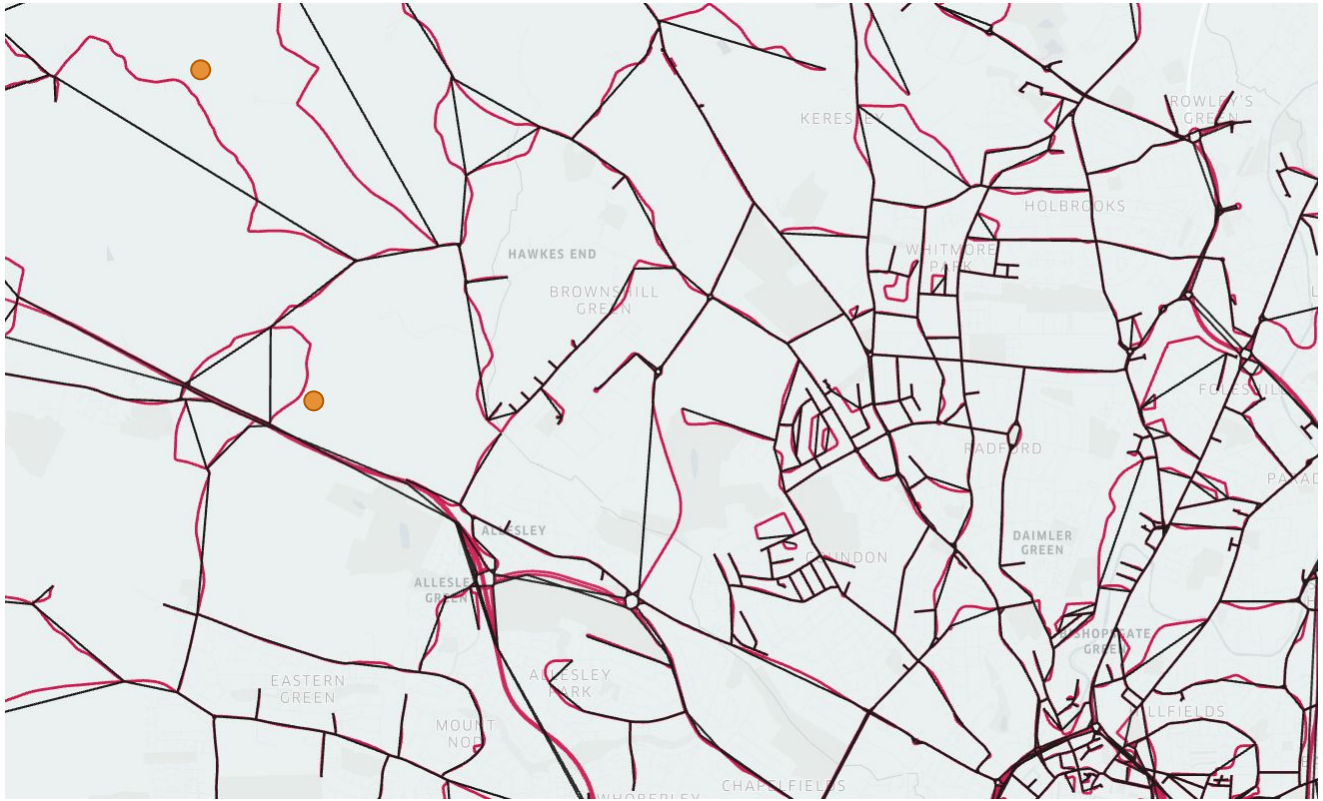
In [10]:
```
df_tolls.to_csv(
    '../example_data/pt2matsim_network/road_pricing/osm_tolls_with_network_ids.csv')
```

Next we can generate the road pricing file.

In [11]:
```
xml_tree = road_pricing.build_tree_from_csv_json(
    '../example_data/pt2matsim_network/road_pricing/osm_tolls_with_network_ids.csv',
    '../example_data/pt2matsim_network/road_pricing/osm_to_network_ids.json')

road_pricing.write_xml(xml_tree, '../example_data/pt2matsim_network/road_pricing')
```

Using Network - Benchmark snapping



GeNet network simplification retains original geometry, making spatial joins easier (e.g. snapping benchmark point counters)

Modifying Network - Road Scenarios

- Increasing capacity on roads of specific type
- Adding new roads (with a geometry, to match with the rest of the network)
- Remove roads (and then use methods to verify connectivity)

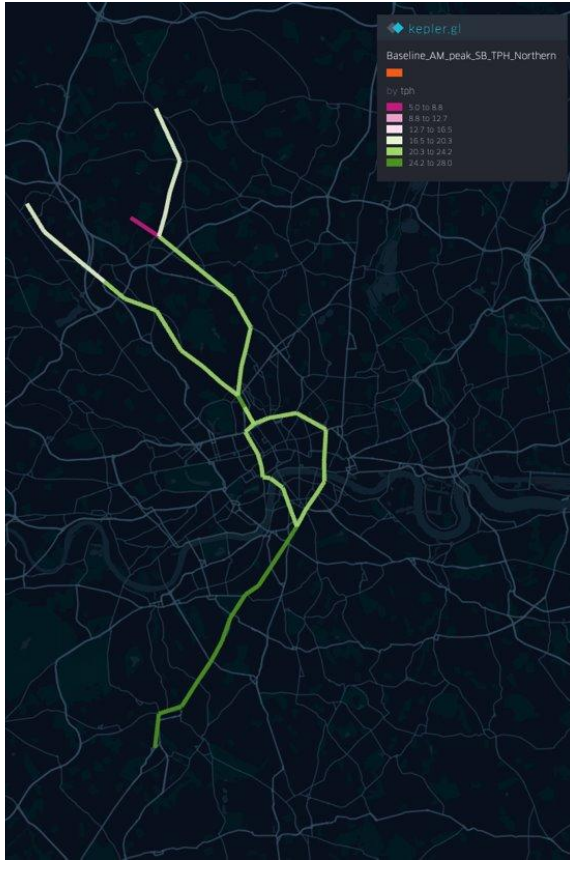
All changes (to network or schedule) are recorded in a changelog table

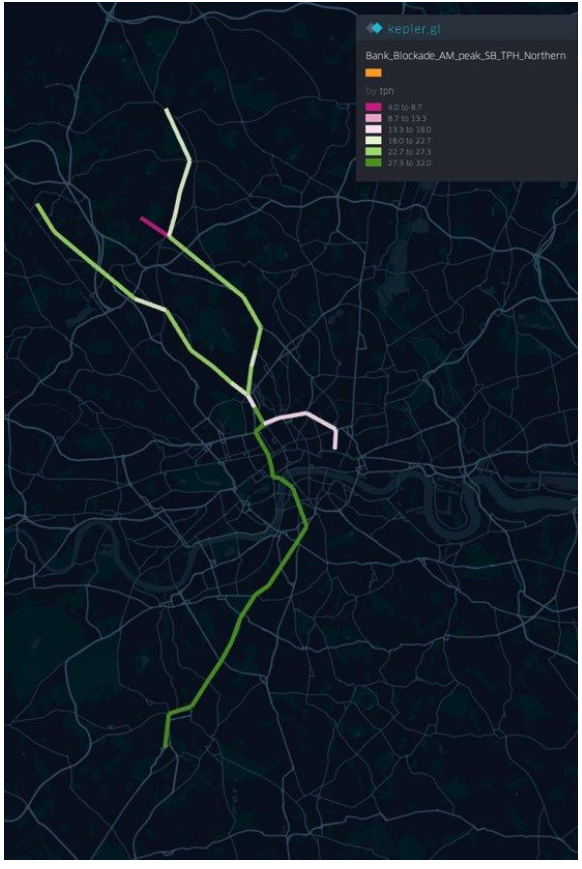
```
In [13]: n.change_log.head()
```

Out[13]:

	timestamp	change_event	object_type	old_id	new_id	old_attributes	new_attributes	diff
0	2021-02-23 20:44:24	add	link	None	proposed_index	None	{'from': '4356572310', 'to': '5811263955', 'id...	[(add, , [('from', '4356572310'), ('to', '5811...
1	2021-02-23 20:44:24	add	node	None	proposed_index	None	{'data': 'some_data'}	[(add, , [('data', 'some_data')]), (add, id, p...
2	2021-02-23 20:44:24	add	link	None	0	None	{'from': '4356572310', 'to': '5811263955', 'id...	[(add, , [('from', '4356572310'), ('to', '5811...
3	2021-02-23 20:44:24	modify	node	proposed_index	another_index	{'data': 'some_data'}	{'data': 'some_data', 'id': 'another_index'}	[(add, , [('id', 'another_index')]), (change, ...
4	2021-02-23 20:44:24	modify	node	proposed_index	proposed_index	{'data': 'some_data'}	{'data': 'some_data', 'id': 'another_index'}	[(add, , [('id', 'another_index')])]

Modifying Network - Change existing service

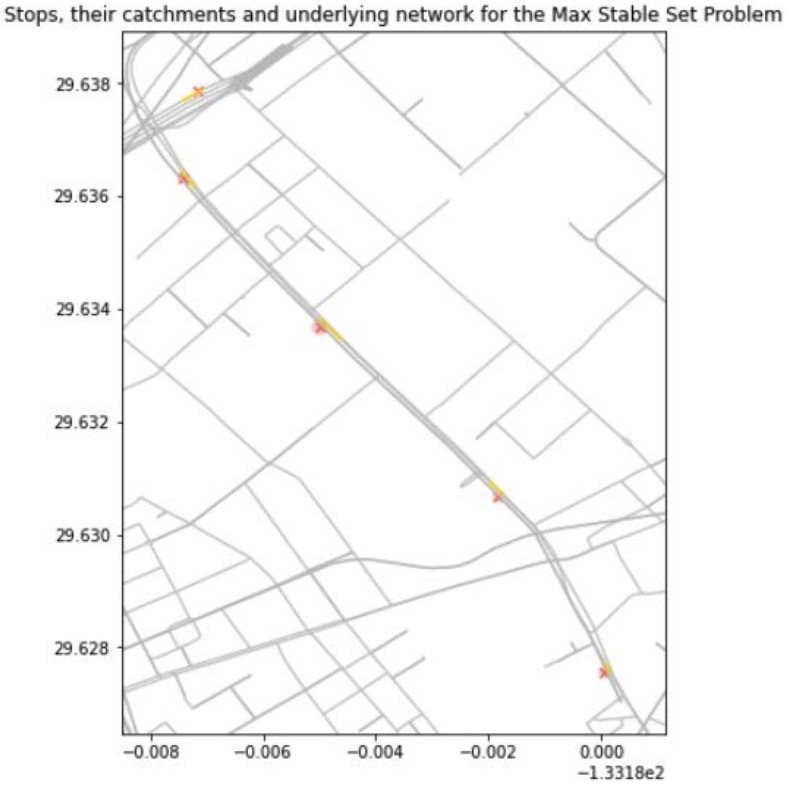


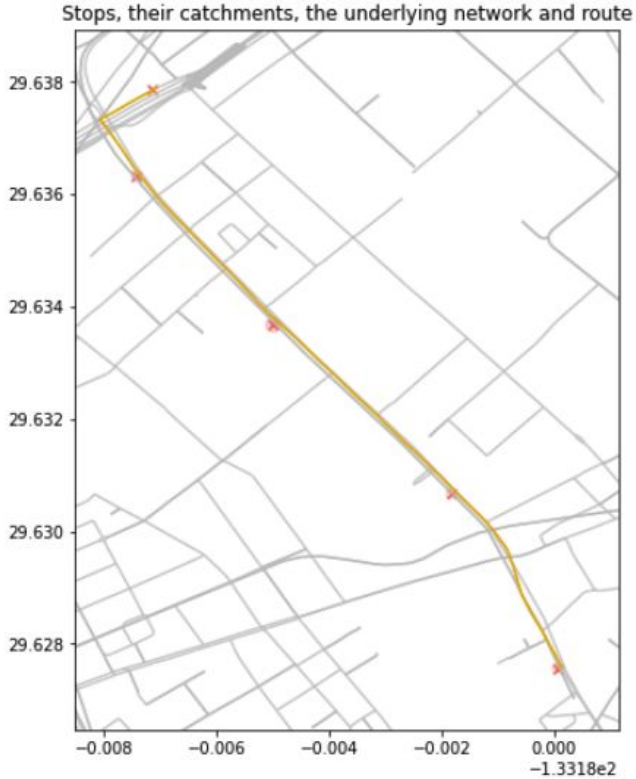


We closed a section of the Jubilee Line Service to verify model behaviour.

Modifying Network - Add new service

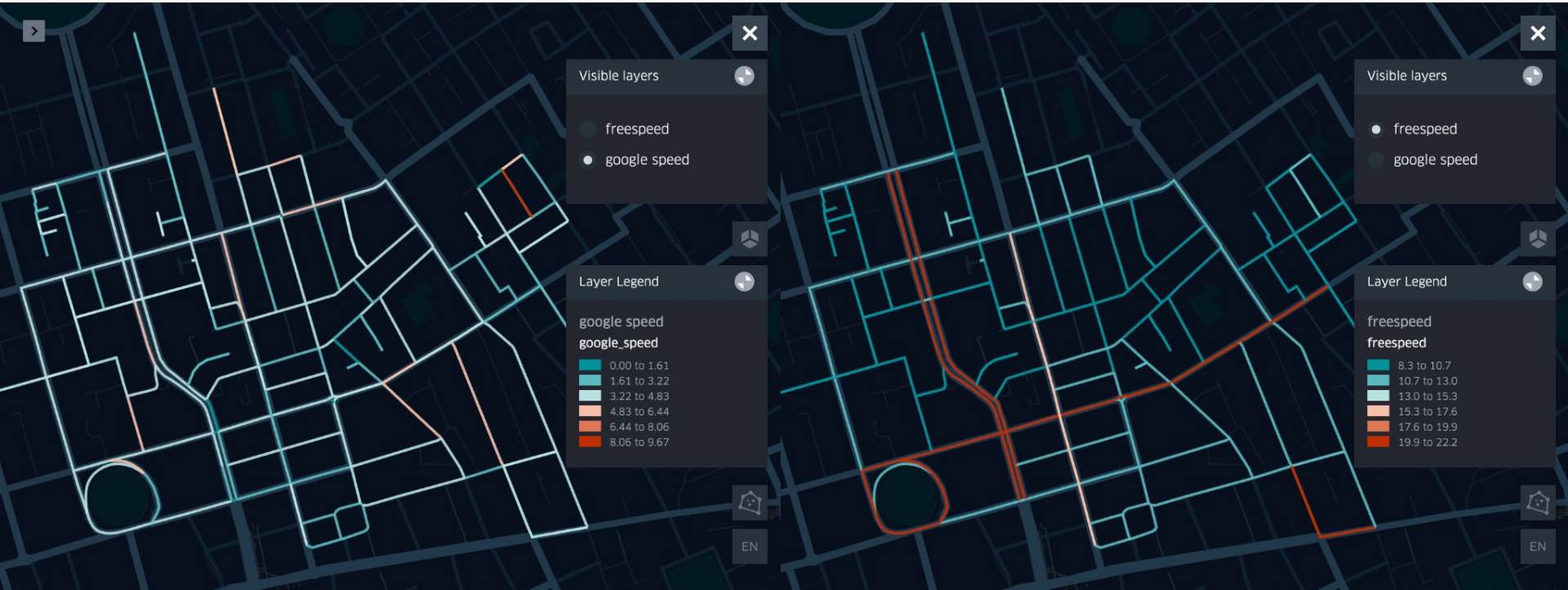
1. Read GTFS
2. Add relevant Service(s)
3. Snap to existing network





Validating Network - Google Directions API Speeds

GeNet generates Google Directions API requests based on a network and computes speeds based on the responses.



Validating Network and Schedule

```
# read sample network
from genet import Network
import os

n = Network('epsg:27700')
path_to_matsim_network = '../tests/test_data/matsim'
n.read_matsim_network(os.path.join(path_to_matsim_network, 'network.xml'))
n.read_matsim_schedule(os.path.join(path_to_matsim_network, 'schedule.xml'))
```

```
report = n.generate_validation_report()
```

```
2020-12-17 12:17:23,744 - Checking validity of the Network
2020-12-17 12:17:23,748 - Checking validity of the Network graph
2020-12-17 12:17:23,749 - Checking network connectivity for mode: car
2020-12-17 12:17:23,753 - Checking network connectivity for mode: walk
2020-12-17 12:17:23,757 - Checking network connectivity for mode: bike
2020-12-17 12:17:23,760 - Checking validity of the Schedule
2020-12-17 12:17:23,762 - Not all stops reference network link ids.
2020-12-17 12:17:23,765 - Not all stops reference network link ids.
2020-12-17 12:17:23,768 - Not all stops reference network link ids.
2020-12-17 12:17:23,775 - This schedule is not valid
2020-12-17 12:17:23,777 - Not all stops reference network link ids.
2020-12-17 12:17:23,785 - Not all stops reference network link ids.
2020-12-17 12:17:23,788 - Service id=10314 is not valid
2020-12-17 12:17:23,790 - Not all stops reference network link ids.
2020-12-17 12:17:23,792 - Not all stops reference network link ids.
2020-12-17 12:17:23,795 - Route id=VJbd8660f05fe6f744e58a66ae12bd66acbca88b98 under Service id=10314 is not valid
2020-12-17 12:17:23,798 - Some link ids in Route: VJbd8660f05fe6f744e58a66ae12bd66acbca88b98 don't accept the route's mode: bus
2020-12-17 12:17:23,806 - Some link ids in Route: VJbd8660f05fe6f744e58a66ae12bd66acbca88b98 don't accept the route's mode: bus
```

Validating Network and Schedule

Summary of Test Coverage

Test	Purpose	Source	Warning or Error
Travel time warning	Checks if the scheduled time is possible on the mapped route	PT2MatSim	Warning
Artifical Link	Checks if the link was made by puma and not provided by the road network	PT2MatSim	Warning
Loop warning	Checks if a route revisits the same node more than once	PT2MatSim	Warning
Direction Change Warning	Checks for abrupt changes in direction	PT2MatSim	Warning
Disconnected Routes Warning	Checks if the route contains broken components	GeNet	Warning
Invalid Routes Warning	Checks if the route has more than 1 stop, the stops are correctly ordered, the route has valid arrival and departure offsets and that it does not contain self loops	GeNet	Warning
Invalid Schedule Warning	Checks if all the routes are valid and that all the routes are uniquely indexed	GeNet	Warning
Invalid Routing Warning	Checks if the route objects contain routes, if the link ids exist in the graph, if the link ids form a connected chain and whether the modes of each link id is the same as the route's mode	GeNet	Warning

Thanks

https://github.com/arup-group/genet - take a look at the wiki

https://medium.com/arupcitymodelling - blog