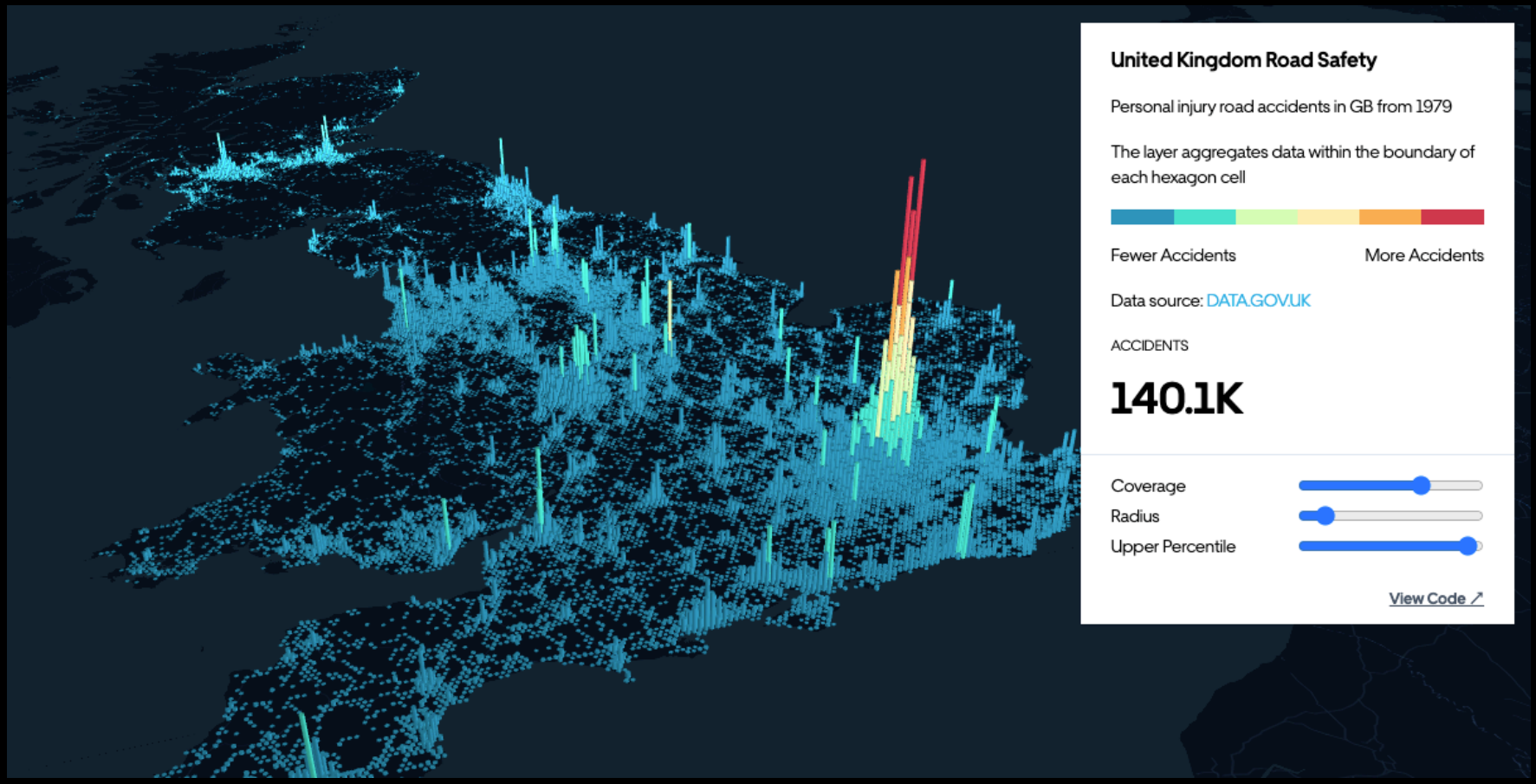
Part B:
Beyond Slippy Maps



1. Raster vs Vector

- Moving beyond the tiled square "slippy map" PNG images to pure vector data
  - It can still be delivered to the browser as squares of data
  - As it's vector, we can do a lot more with it
    - Change the style dynamically
    - Animate the data to show temporal change
    - Warp without pixellation
    - Dynamically extrude or otherwise show in 3D
    - Perform on-the-fly analysis, spatial calculations,
      - e.g. data summing across user-specified area.
- The browser has to do more work
  - but client computers are a lot more powerful now
  - browsers have been heavily optimized for large data processing.
- "Raster is faster but vector is corrector"

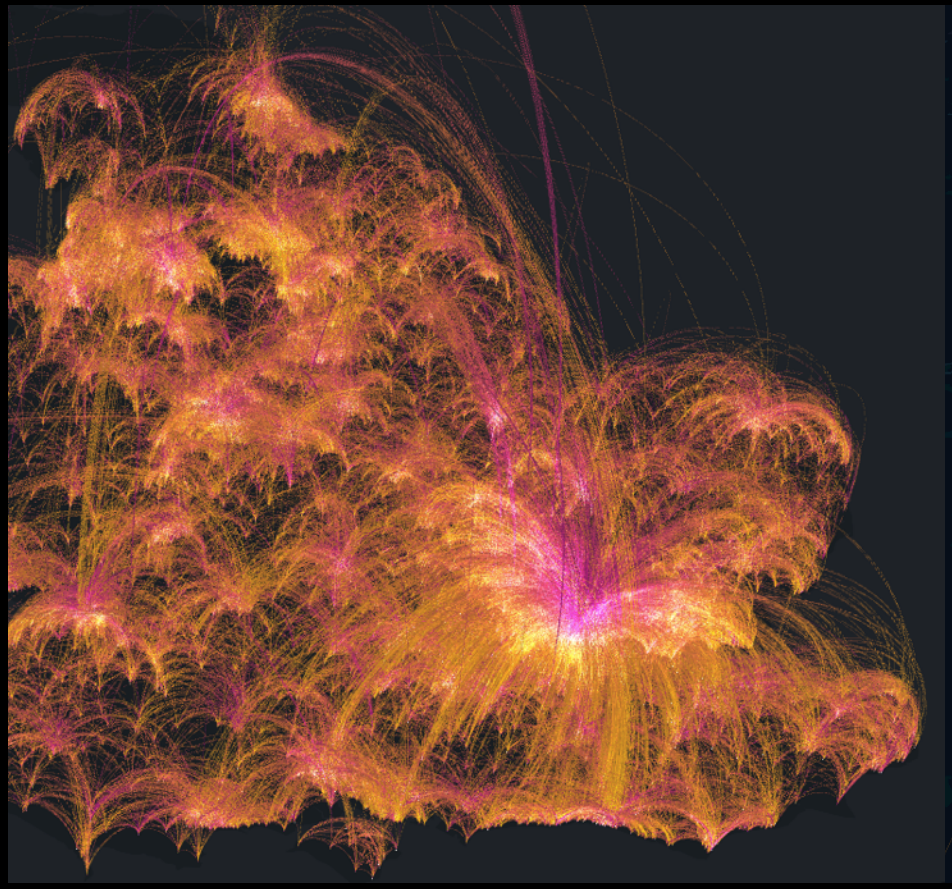
2. Deck.gl and Kepler.gl

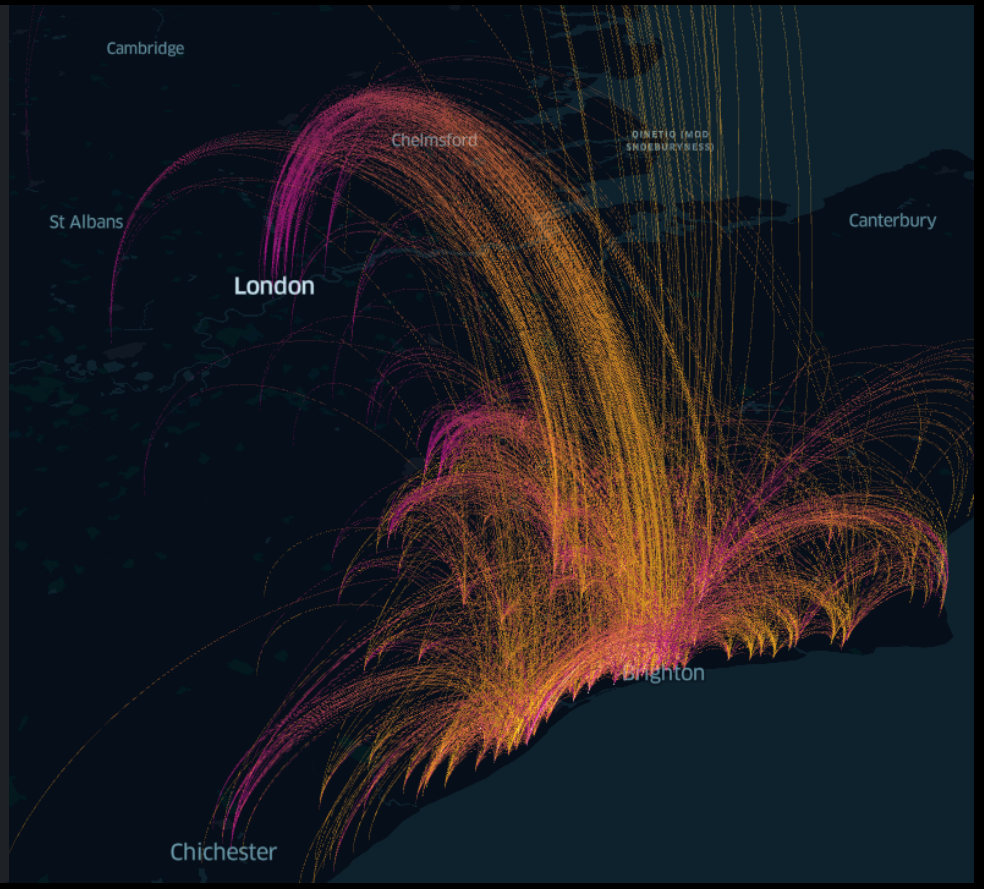
- Kepler.gl is Part of Uber Engineering's visualisation library
  - It is built on Deck.gl which is a high-performance WebGL application for visualizing large sets of geospatial data browser-side.
  - It provides a wrapper, heuristics and controls to quickly analyse basic geospatial datasets including O/D datasets (supply origin lat/lon, destination lat/lon and other attributes)
  - Open source + on GitHub: https://github.com/keplergl/kepler.gl
- It is client-side in your browser (although you can load data server-side from a URL if you prefer)

Kepler Examples

https://kepler.gl/demo/nyctrips

Kepler Examples





https://kepler.gl/demo/ukcommute

Data

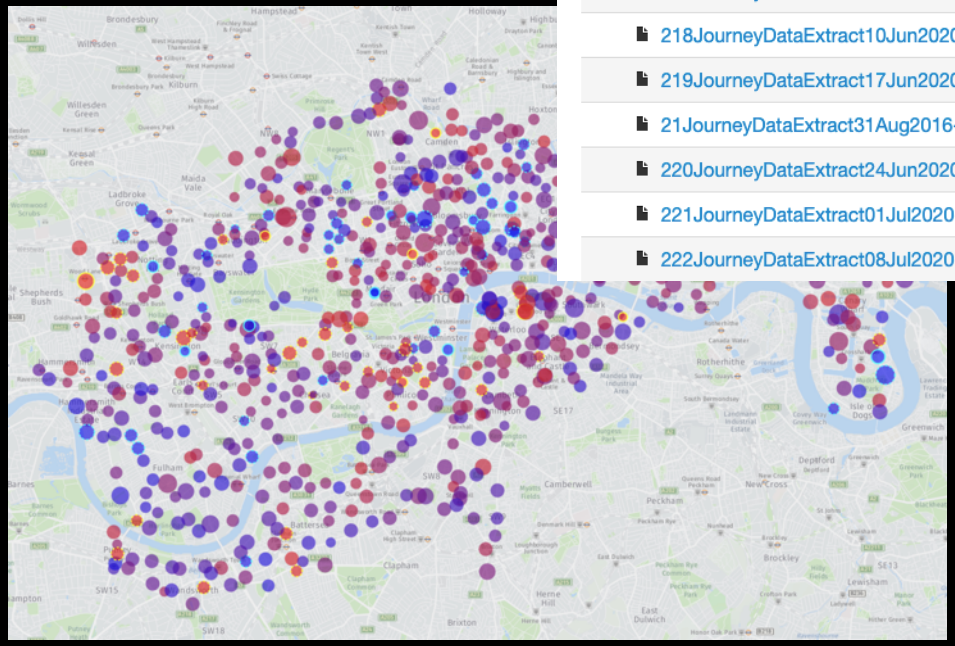
cycling.data.tfl.gov.uk

The public TfL data (or 'open data') released here is for open data users to use in their own software and services. We encourage software developers to use this data to present customer travel information in innovative ways - providing they adhere to the transport data terms and conditions https://tfl.gov.uk/corporate/terms-and-conditions/transport-data-service

Name

- CycleCounters/
  - CycleCounters/Blackfriars/
    - CycleCounters/Blackfriars/July/
      - Friday, Jul 13 2018.xls
      - Friday, Jul 20 2018.xls
      - Friday, Jul 27 2018.xls

211JourneyDataExtract22Apr2020-28Apr2020.csv	May 4th 2020, 01:20:54 pm	20.33 MB	CSV file
212JourneyDataExtract29Apr2020-05May2020.csv	May 27th 2020, 02:07:17 pm	16.63 MB	CSV file
213JourneyDataExtract06May2020-12May2020.csv	May 27th 2020, 02:07:32 pm	24.15 MB	CSV file
214JourneyDataExtract13May2020-19May2020.csv	May 27th 2020, 02:07:51 pm	27.70 MB	CSV file
215JourneyDataExtract20May2020-26May2020.csv	May 27th 2020, 02:08:12 pm	38.57 MB	CSV file
216JourneyDataExtract27May2020-02Jun2020.csv	Aug 5th 2020, 03:13:40 pm	41.49 MB	CSV file
217JourneyDataExtract03Jun2020-09Jun2020.csv	Aug 5th 2020, 03:13:59 pm	26.30 MB	CSV file
218JourneyDataExtract10Jun2020-16Jun2020.csv	Aug 5th 2020, 03:14:13 pm	32.29 MB	CSV file
219JourneyDataExtract17Jun2020-23Jun2020.csv	Aug 5th 2020, 03:14:27 pm	35.02 MB	CSV file
21JourneyDataExtract31Aug2016-06Sep2016.csv	Nov 30th 2016, 09:55:59 am	29.46 MB	CSV file
220JourneyDataExtract24Jun2020-30Jun2020.csv	Aug 5th 2020, 03:14:44 pm	36.30 MB	CSV file
221JourneyDataExtract01Jul2020-07Jul2020.csv	Aug 5th 2020, 03:14:59 pm	30.42 MB	CSV file
222JourneyDataExtract08Jul2020-14Jul2020.csv	Aug 5th 2020, 03:15:20 pm	31.16 MB	CSV file



We are using the ”out of the box” version of Kepler. Customising the code for cycling data in particular (e.g. the typical scales and data types) would enhance the app for our use case.

Data

Normally:

Edges

```
SELECT  journey_id,
        duration,
        bike_id,
        startdate,
        start_station_id,
        end_station_id
FROM    journeys;
```

Nodes

```
SELECT  station_id,
        lat,
        lon
FROM    dockingstations;
```

Data

- But, Kepler reqires the node and edge list joined together as a single file. So:

```
SELECT   rental_id,
         duration_s,
         bike_id,
         start_dt,
         a.lat as start_latitude,
         a.lon as start_longitude,
         b.lat as end_latitude,
         b.lon as end_longitude
FROM     journeys, dockingstations a, dockingstations b
WHERE    a.station_id = startstation_id
         and b.station_id = endstation_id;
```

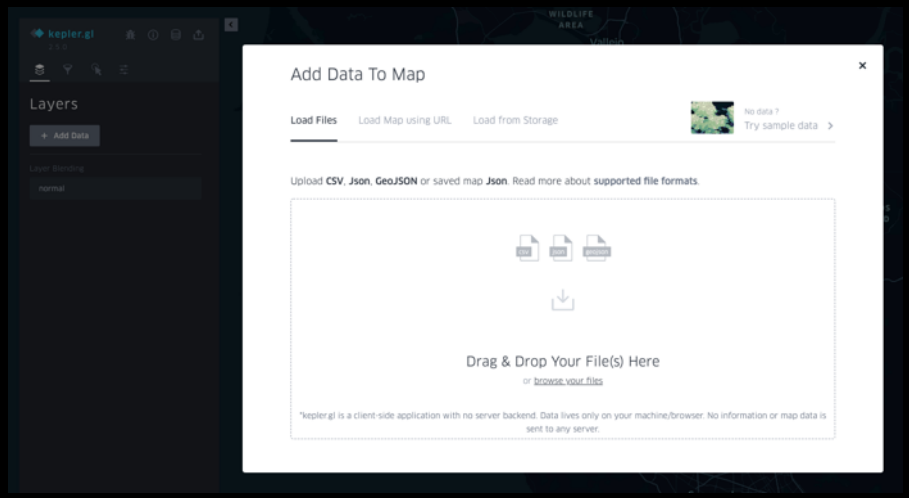
Data

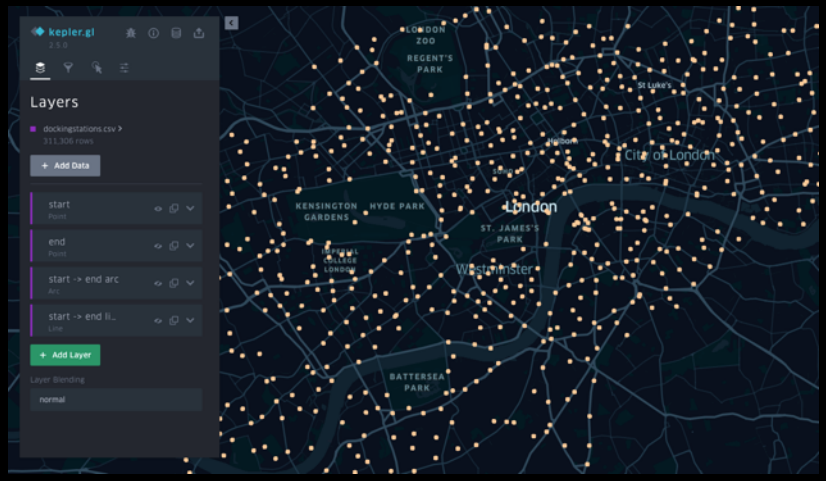
- Let's add some extra calculated columns, as Kepler doesn't (yet) have these built in – although may be straightforward to add in to the Kepler open source
- Also add a basic weighting column for getting simple flow aggregates

```
SELECT    rental_id,
          duration_s,
          bike_id,
          start_dt,
          weekday(start_dt) dayofweek,
          hour(start_dt) hourofday,
          day(start_dt) dayofmonth,
          CAST(degrees(atan2(cos(radians(a.lat))
                    *(b.lat-a.lat),b.lon-a.lon))
                    AS SIGNED) direction,
          a.lat as start_latitude,
          a.lon as start_longitude,
          b.lat as end_latitude,
          b.lon as end_longitude,
          1 as unity
FROM      journeys, dockingstations a,
          dockingstations b
WHERE     a.station_id = startstation_id
          and b.station_id = endstation_id;
```

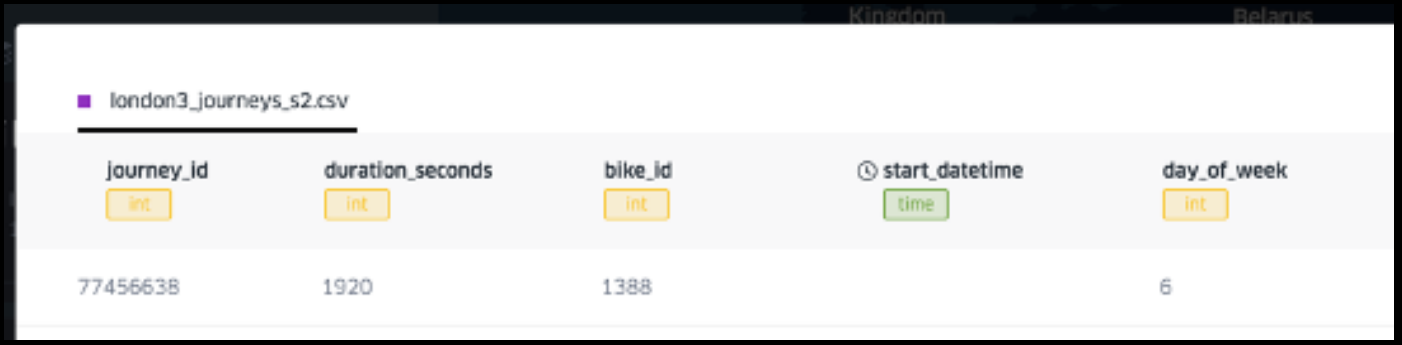
Kepler

- Just drag CSV of the data onto https://kepler.gl/#/demo or run a local copy with its built in webserver, via npm
- Example with 330k journeys (~35MB file) on London's Santander Cycles, from a busy week
  - Fast on this laptop (5 seconds to load/display)
  - Also tried 1 million rows (slow), 3 million rows (crashed)

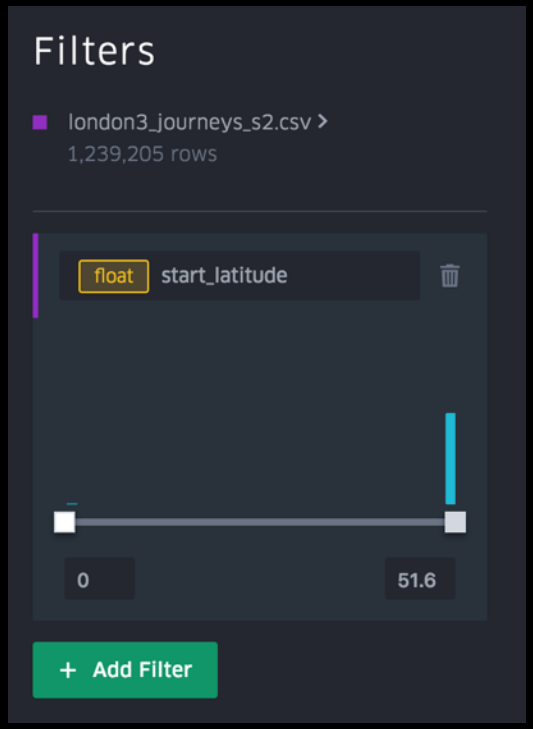




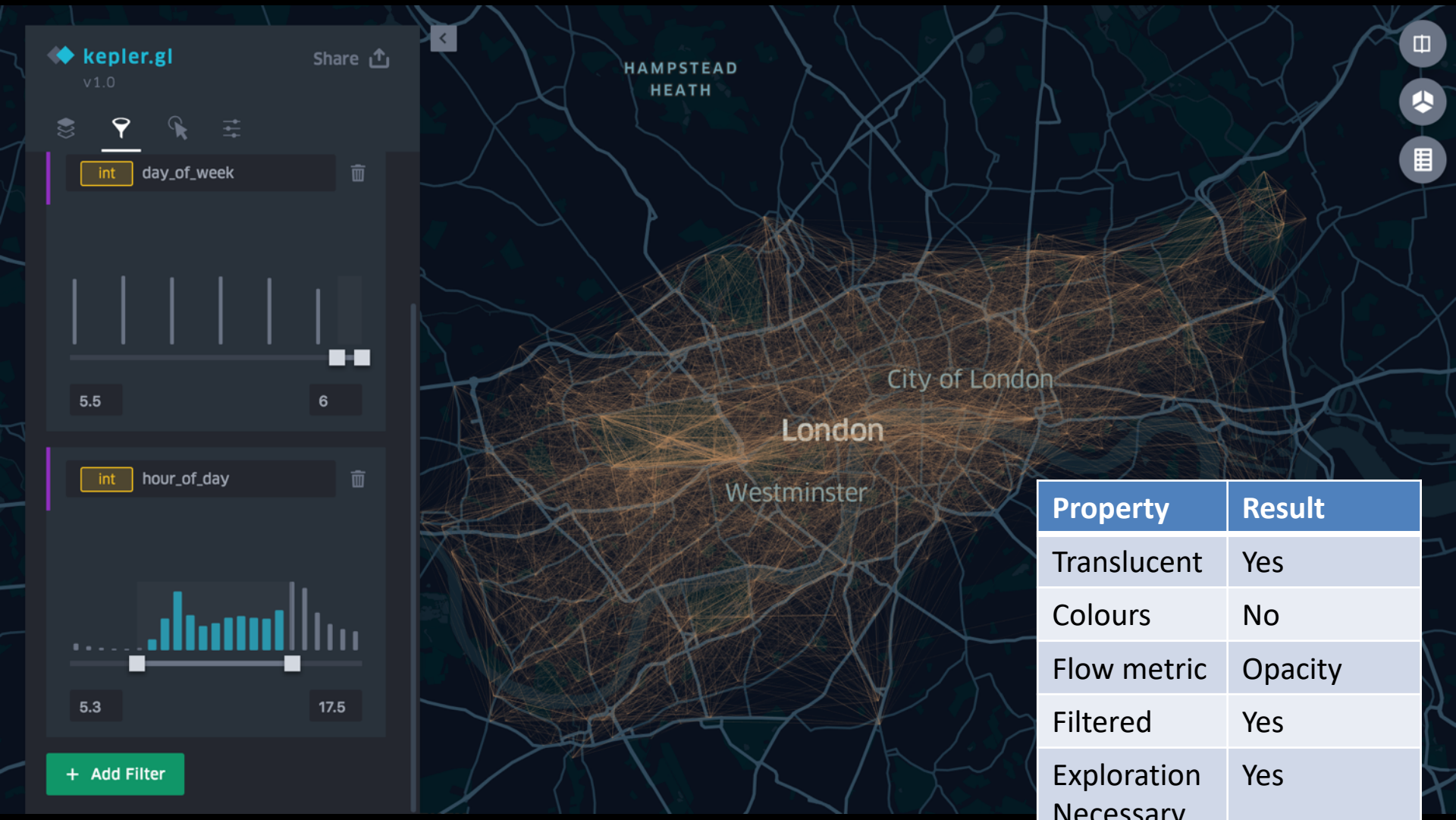
Field Types and Filtering



- Kepler attempts to autodetect field types (lat/lon pairs, datetimes), but add a header row to help it.
- Use filter to remove rows with missing location data (lat/lon = 0)
- Kepler doesn't aggregate flows (yet), so use translucency to simulate this (it can aggregate point data)

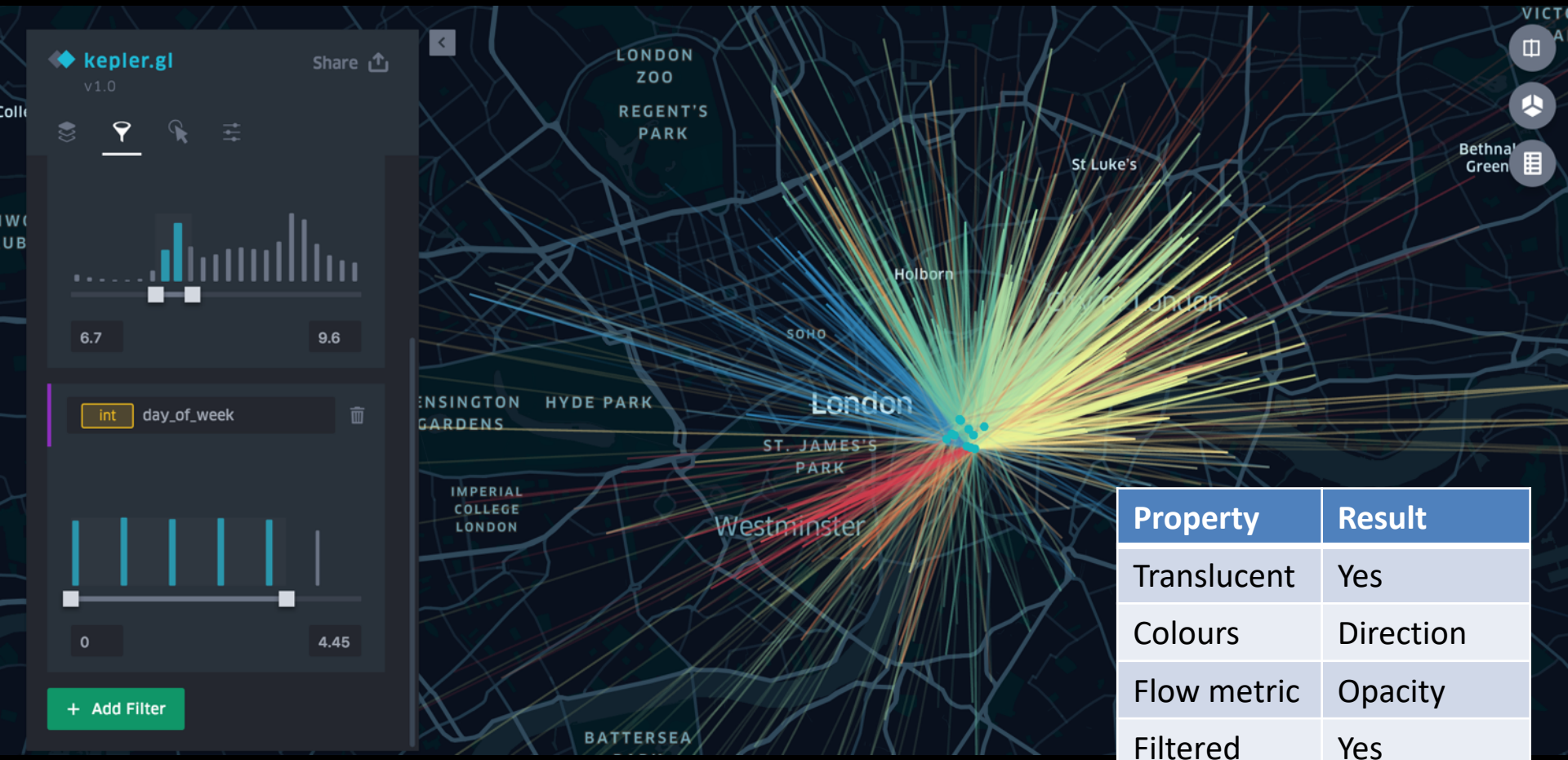


Lines



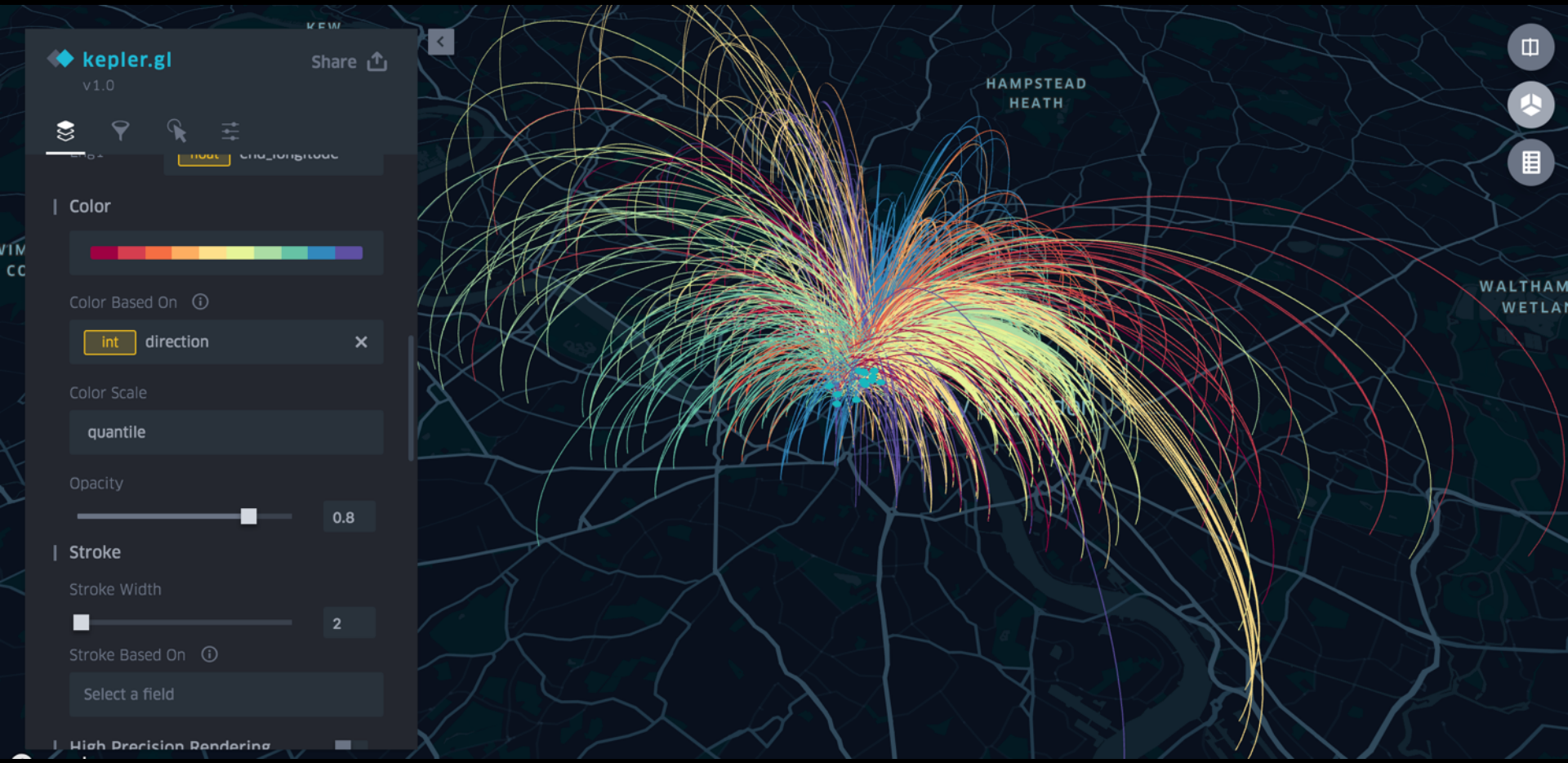
Property	Result
Translucent	Yes
Colours	No
Flow metric	Opacity
Filtered	Yes
Exploration Necessary	Yes

Lines

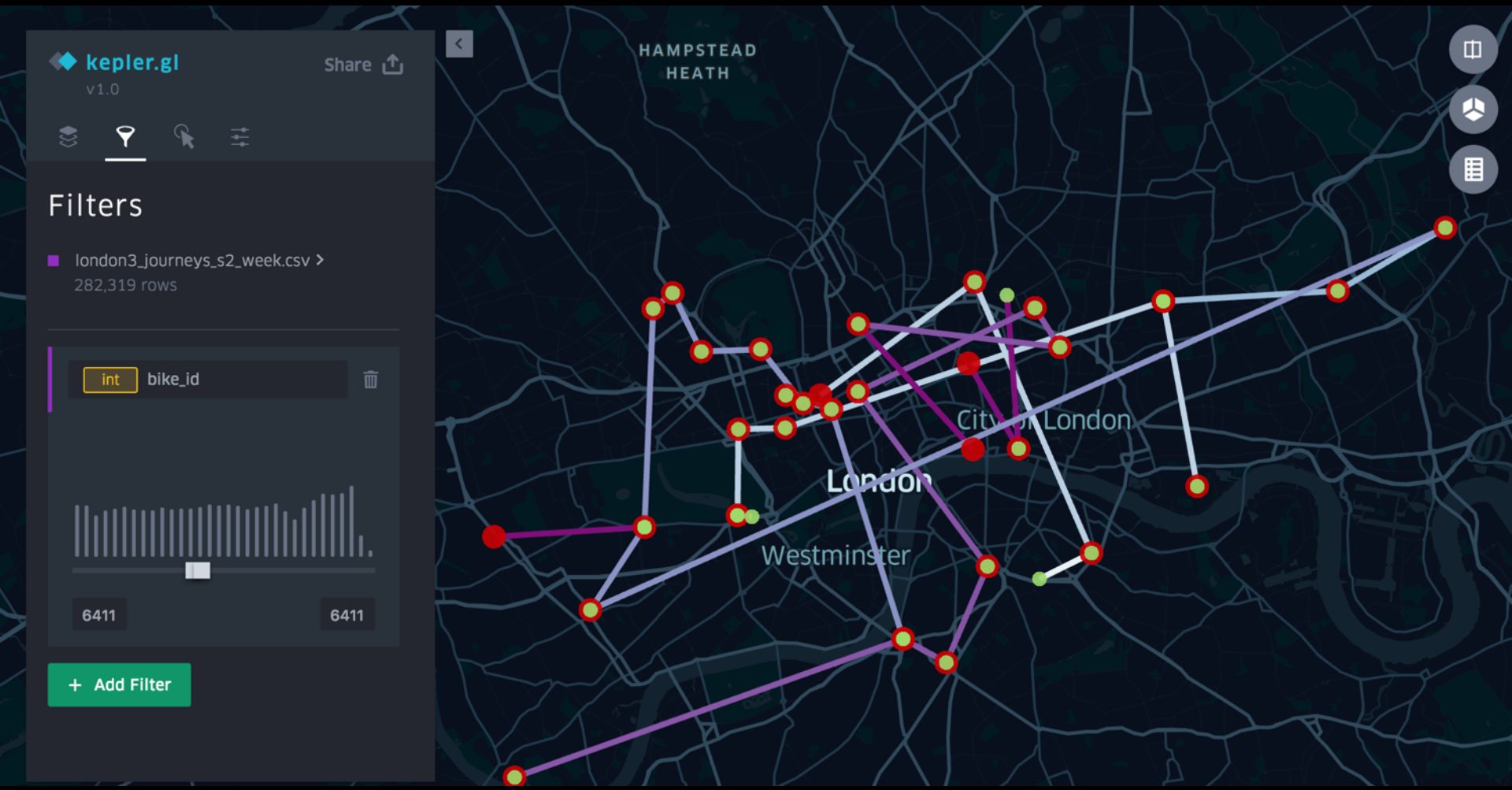


Property	Result
Translucent	Yes
Colours	Direction
Flow metric	Opacity
Filtered	Yes
Exploration Necessary	Yes

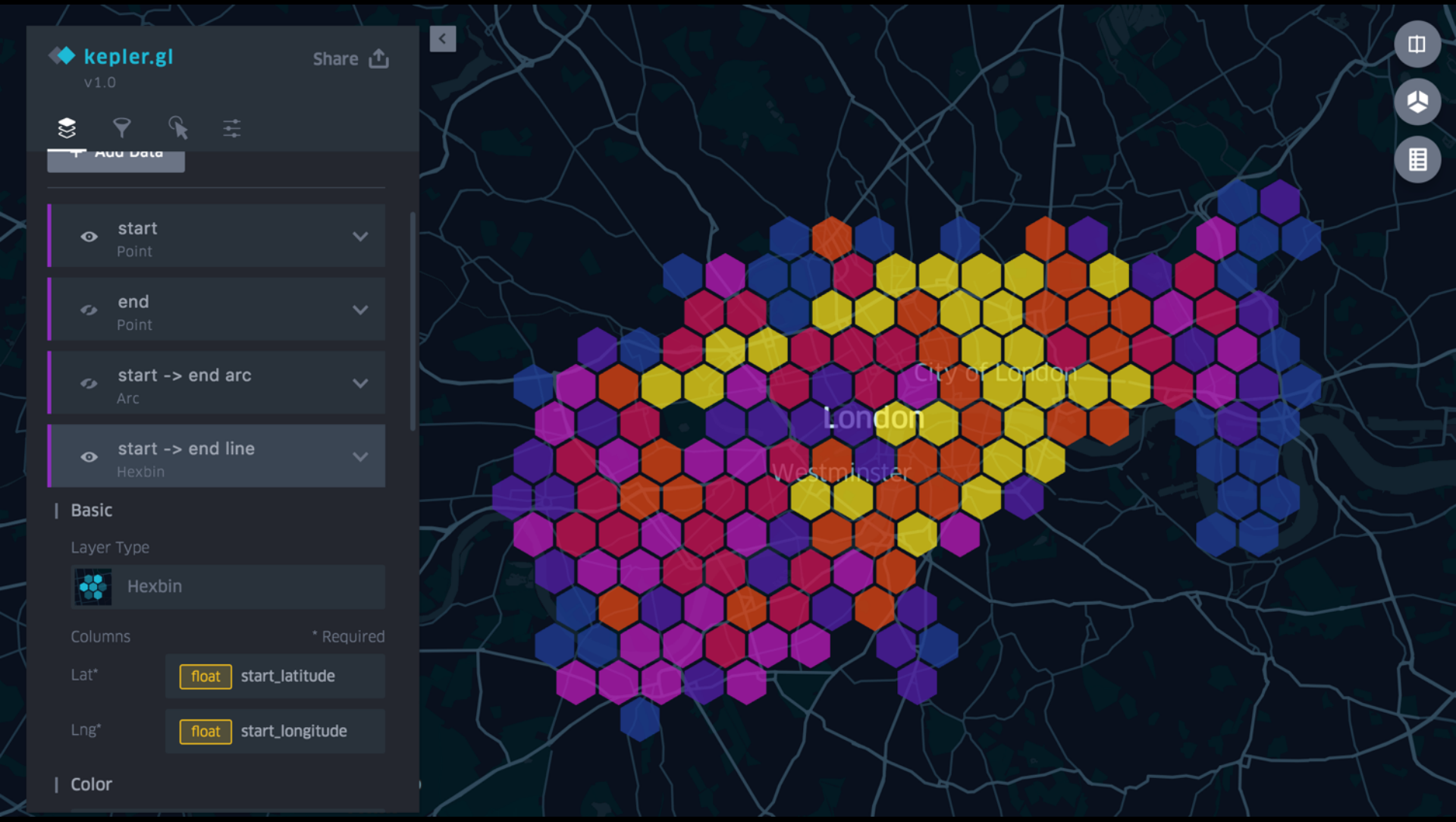
Arcs


kepler.gl
v1.0
Share
Color
Color Based On
int
direction
Color Scale
quantile
Opacity
0.8
Stroke
Stroke Width
2
Stroke Based On
Select a field
HAMPSTEAD
HEATH

Individual Bike Filter



Grids/Hexbins



Kepler and Bikeshare Flow Data

- We have identified ~30 cities around the world where there is journey data published as open data.
- We are looking to build a site to visualise this data – potentially by customizing elements of the Kepler/Deck.gl andor ROpenSci bikedata approaches to managing and displaying it.

USA:
Atlanta
Austin
Boise
Boston
Boulder
Chattanooga
Chicago
Columbus
Denver
Hoboken
Jersey City
Los Angeles
Minneapolis
New York
Philadelphia
Pittsburgh
Portland
San Francisco
San Jose
Tampa

Canada:
Montreal

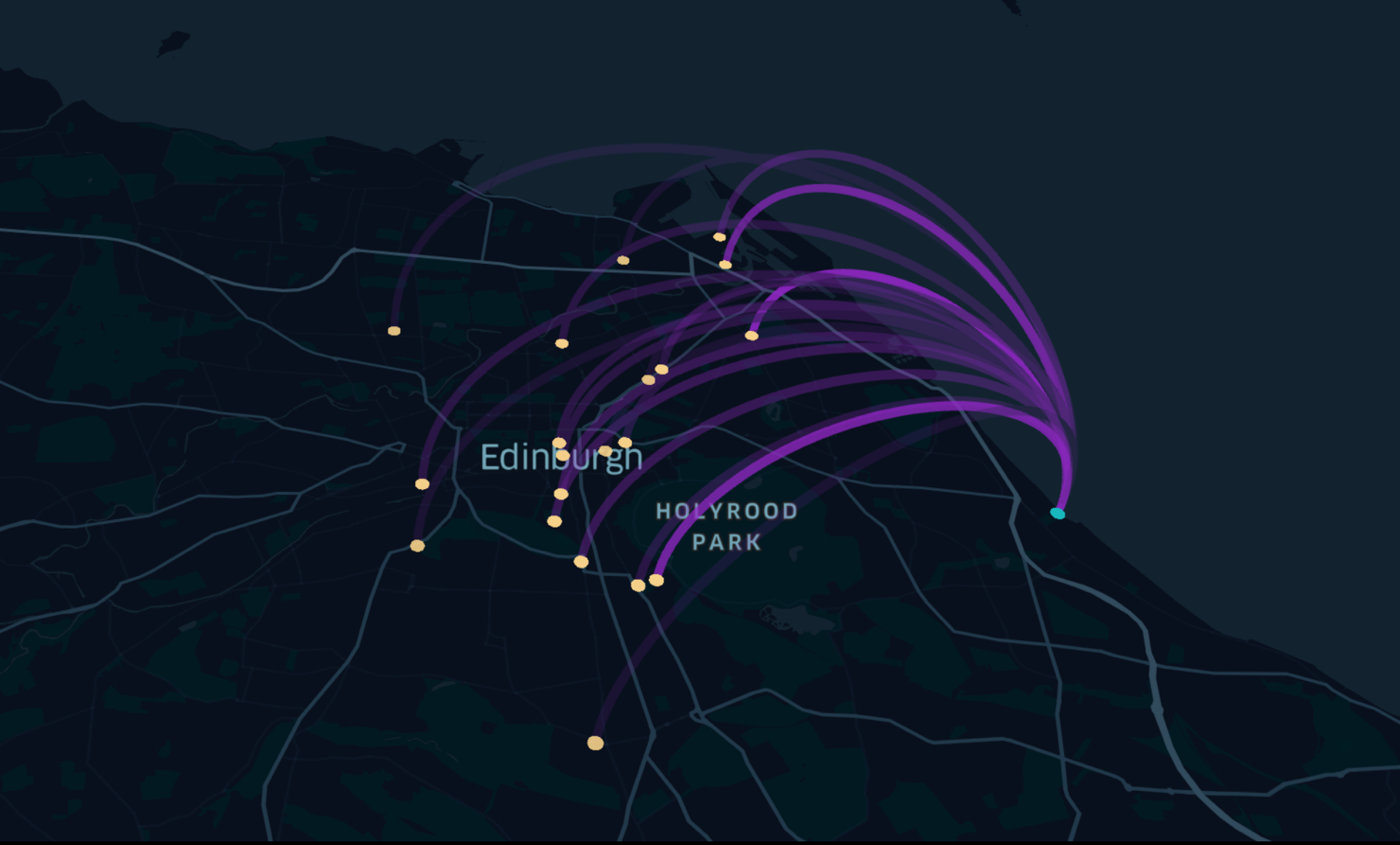
Mexico:
Guadalajara
Mexico City

South Korea:
Daejeon

Norway:
Bergen
Oslo
Trondheim

UK:
London
Edinburgh

Spain:
Madrid



- https://edinburghcyclehire.com/open-data/historical

3. D3

- "D3.js is a JavaScript library for manipulating documents based on data."
- D3 does do maps too
  - But you may find that OpenLayers/Leaflet etc cover your mapping needs and D3 will best visualise your data in other ways
  - I've used D3 for maps in the forthcoming "Worldnames 2" project due to its excellent support for obscure coordinate reference systems – I didn't want to use "Spherical Mercator" or slippy maps due to "first world" area distortion issues and unnecessary complexity.
- Relatively steep learning curve but immensely powerful. It is "lower level" than the mapping APIs I have mentioned.

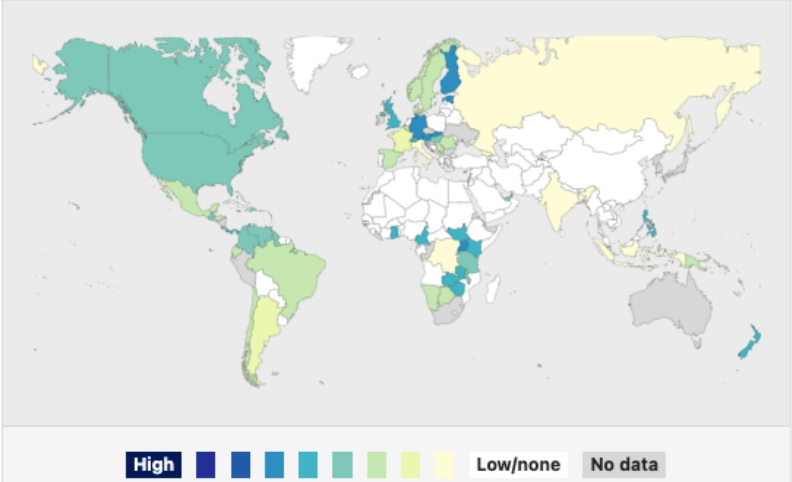
D3

OLIVER

Forename

Map

Countries | Regions



High — Low/none — No data

Top Countries

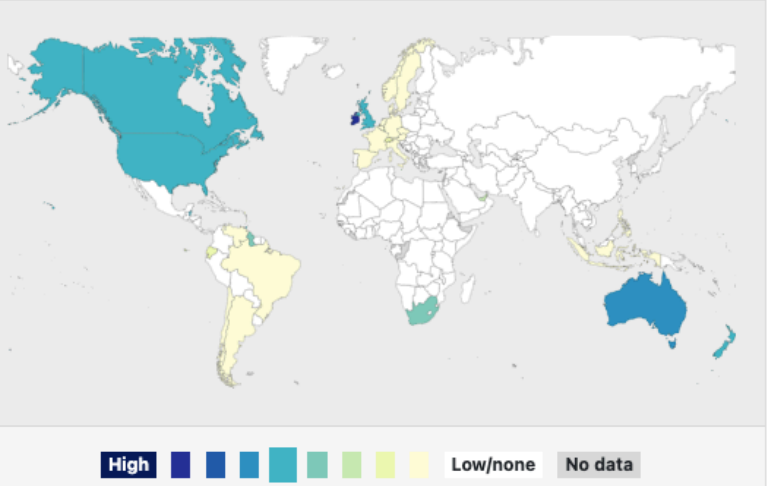
1. Estonia — 2,788
2. Uganda Map — 2,263

OBRIEN

Surname

Map

Countries | Regions

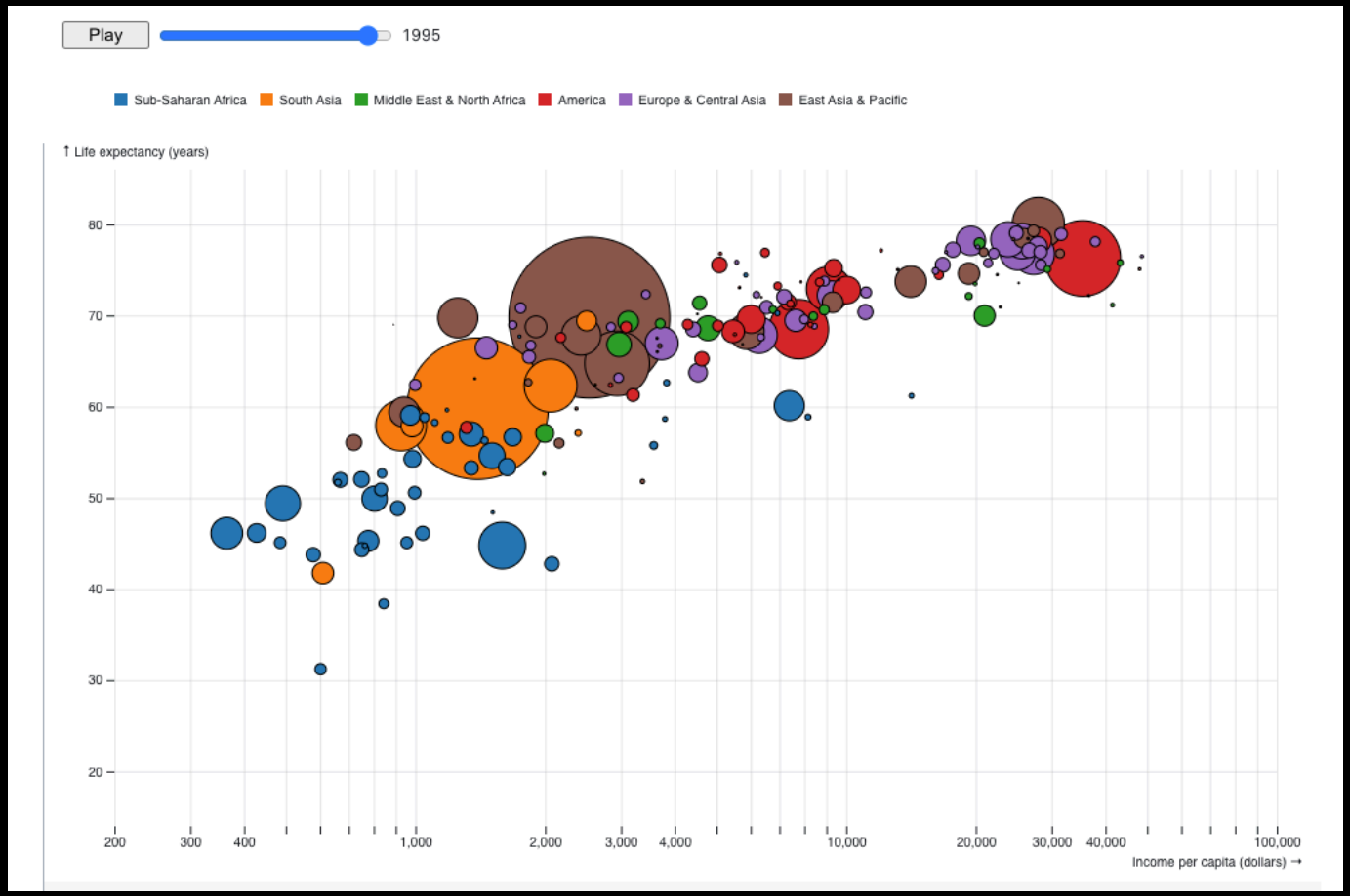


High — Low/none — No data

Top Countries

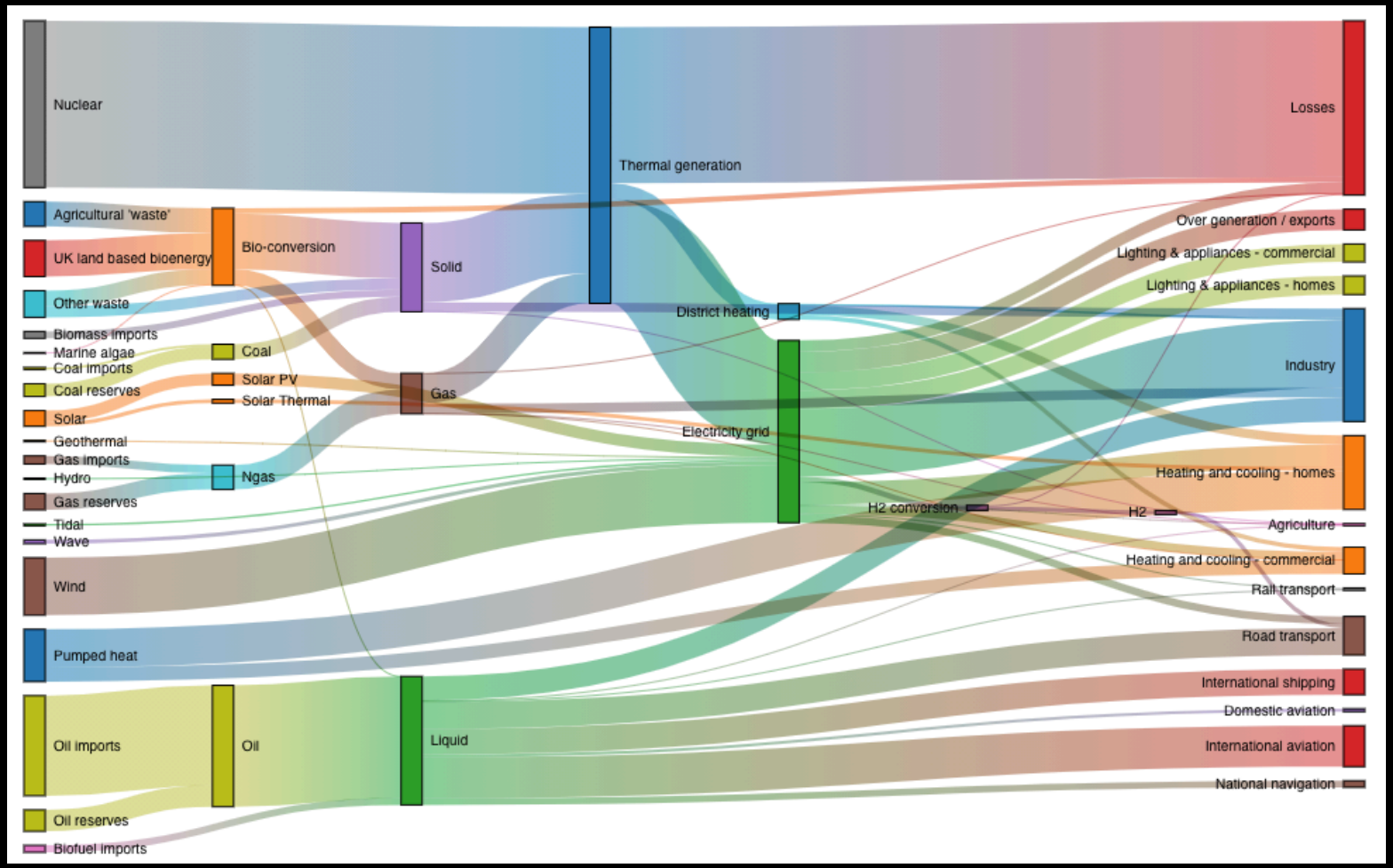
1. Ireland Map — 10,640
2. Australia Map — 1,956

D3



https://observablehq.com/@mbostock/the-wealth-health-of-nations

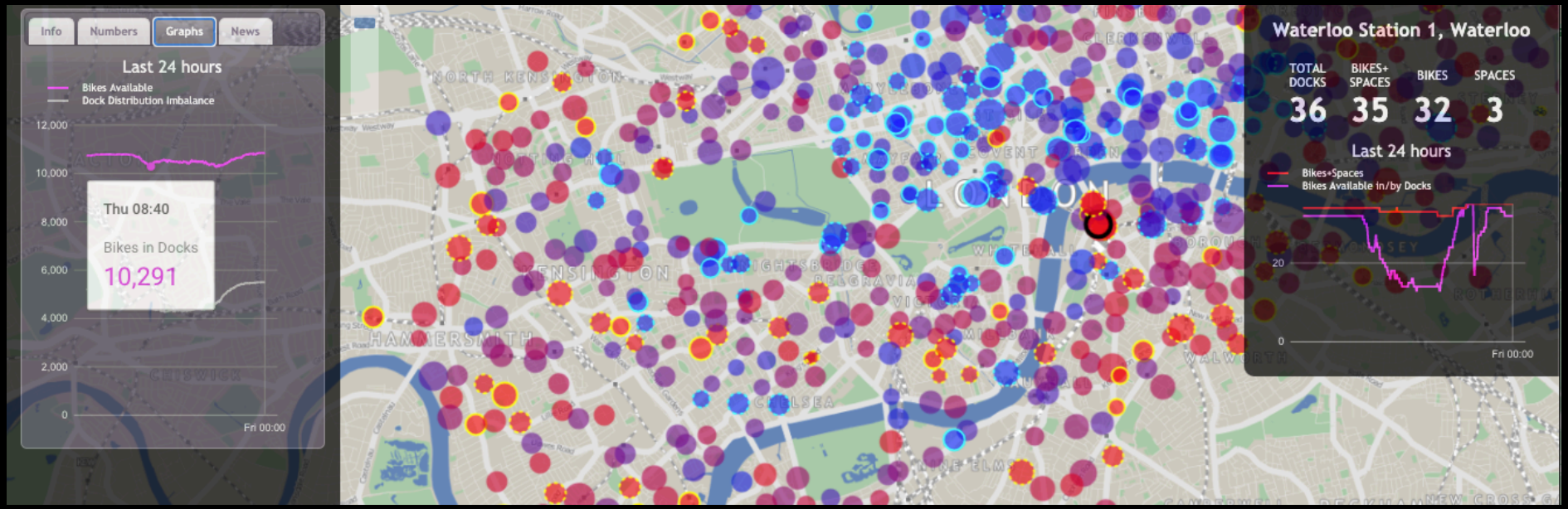
D3



https://observablehq.com/@d3/sankey-diagram

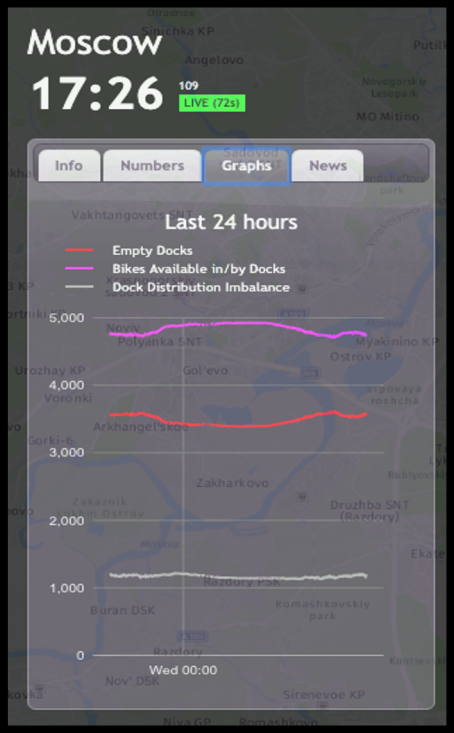
4. Google Charts + OpenLayers

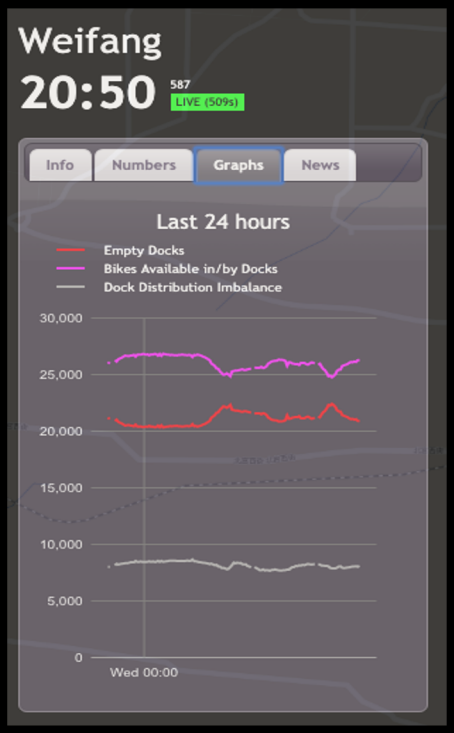
- I have mainly focused on maps.
- Chart/graph based visualization is simpler and there are many more toolkits available for it. I've chosen the one I use

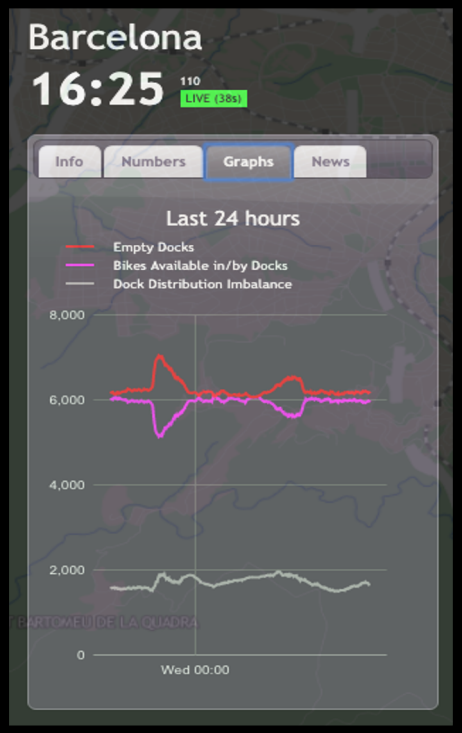


https://bikesharemap.com/london/

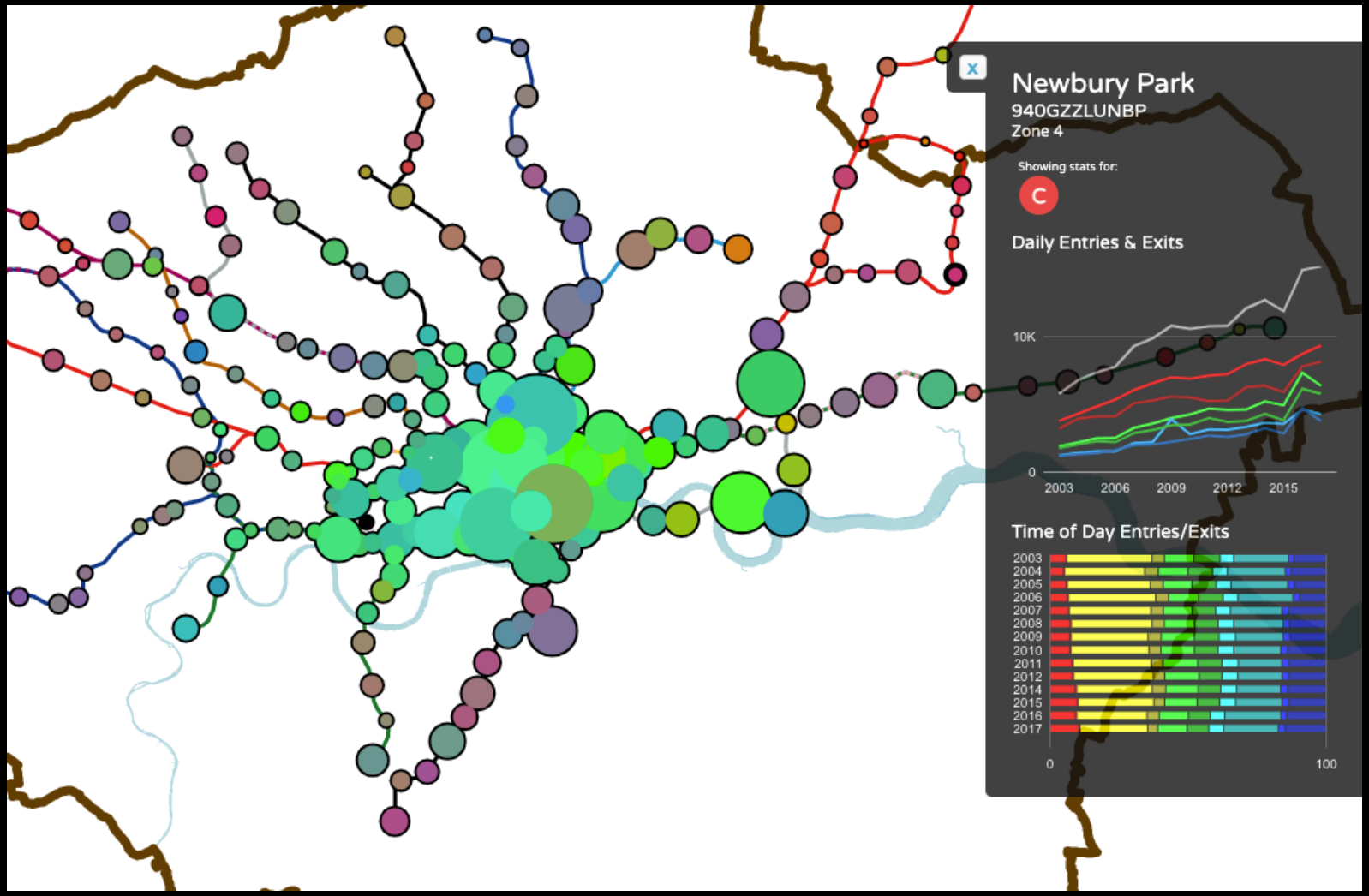
Google Charts





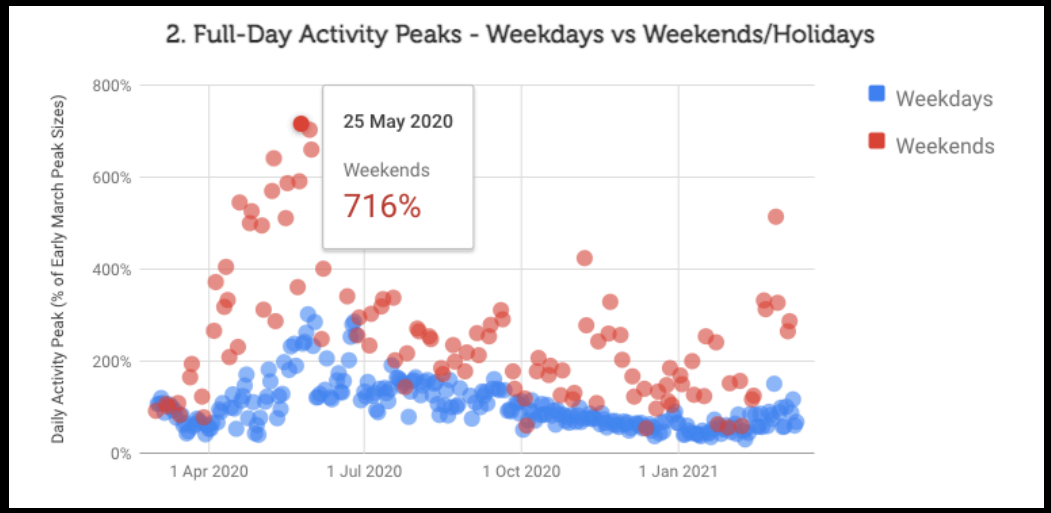


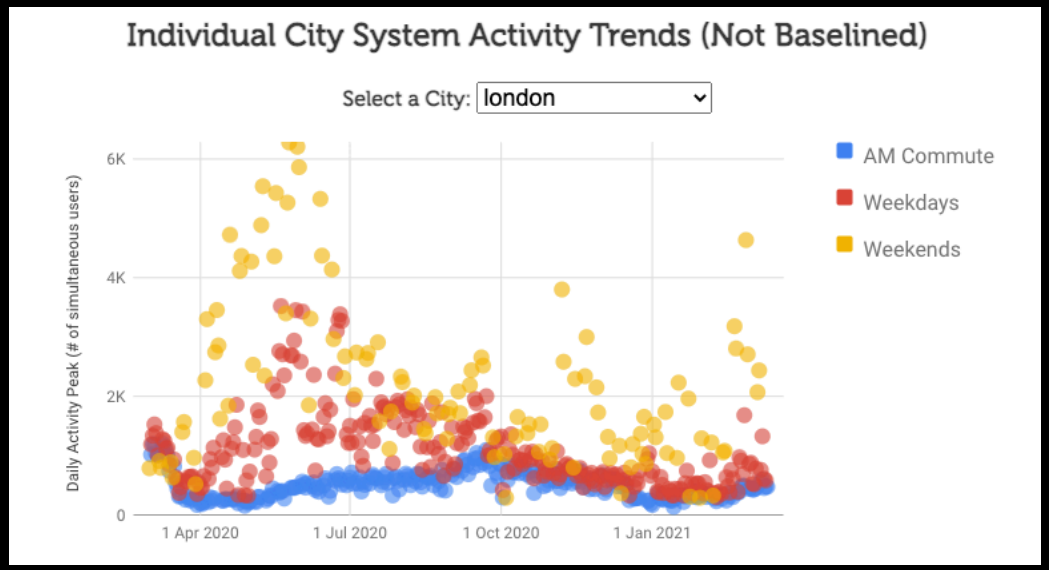
Google Charts + OpenLayers



https://tubecreature.com/

Google Charts





https://apps.cdrc.ac.uk/bikeshareindex/

Consumer Data Research Centre

An ESRC Data Investment

End

Thanks to:

- Prof James Cheshire
- Dr Duncan Smith (CASA at UCL)

Oliver O'Brien
o.obrien@ucl.ac.uk
@oobr