Interactive flow-map with an OD-matrix of regular movements in Tartu, Estonia

Interactive visualization of home-work commuting in Leaflet obtained from Call Detail Records (CDR)

Bryan R. Vallejo

Aug 22, 2021 • 12 min read

Interactive visualization of home-work commuting in Leaflet obtained from Call Detail Records (CDR)

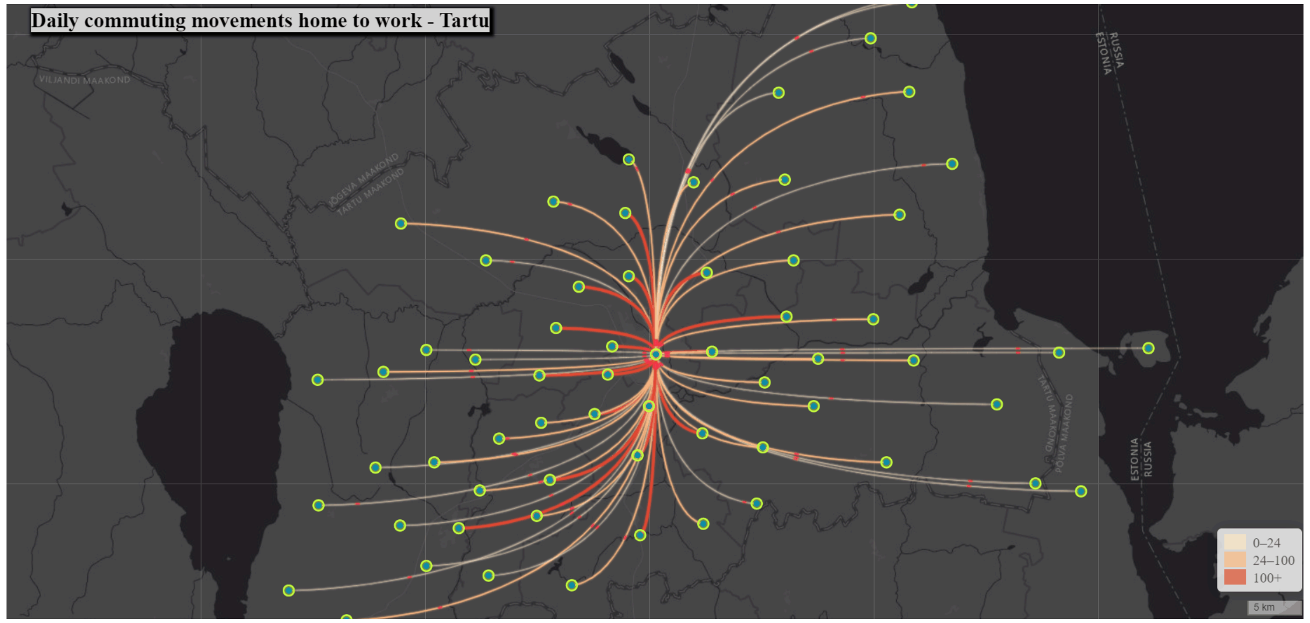


Image by the Author. Map animation of home-work commuting in Tartu County

Urban development has a strong tie with **Human Mobility**. Urban planners make decisions in the function of population dynamics. Let's say, traffic congestion is a clear fact that specific roads are needed to reach important locations of people (workplace), then urban planners may decide options to release the pressure in the road such as improvement in public transport. As we understand, the population is a key factor for modeling cities, the indicators in spatial dimension such as *population density, the spatial distribution of age groups, or elderly concentrations*, are supportive to make decisions to shape a liveable city.

Thanks to Information and Communication Technologies (ICT), the tracement of individuals (anonymously) can be possible. Worldwide, the mobile phone has strong dominance in communication, representing more than 6 000 million users in 2021 by Statista data. The mobile phone calls create a registry in the antennas network known as Call Detail Record (CDR) which lets create a dataset vast in spatial coverage and temporality (longitudinal). This mobile positioning data helps researchers, urban promoters, local government, private companies, or regional organizations (EU) to model cities.

> Final map animation [HERE!] (https://github.com/bryanvallejo16/movements-odmatrix-tartu) Repository HERE!
>
> **INTRODUCTION**

As a leading country in mobility studies, **Estonia** has been collecting continuously for 12-years time series of CDR data. The datasets have been analyzed from many perspectives by **The Mobility Lab** from the **University of Tartu** such as *tourism statistics, ethnic segregation, social networks, cross-border mobility (transnationalism), or spatial mobility*, just to name a few. The studies have given an understanding of – flows, meaningful locations, and spaces – of people. Mobile positioning data reveals meaningful locations of people such as **home or work** (anchor points/activity locations)[1], then the movement between locations can be aggregated and used to construct daily flows of people. The studies are vast in applications such as *long-term activity spaces, settlement hierarchies in societies, or transportation*, but for this visualization practice, we are going to be based on an OD-matrix which represents daily commuting between home to work. You can expand reading about CDR applications and the dataset generation on the **Mobility Lab (OD matrix) page** written by Estonian geographer Anto Aasa.

> **DATA**

The dataset [2] was collected and analyzed by phone operator Telia, **Mobility Lab** in the Department of Geography at the University of Tartu, and a spin-off company **Positium LBS**. Approximately 420.000 respondents per month were noted to whom the ***anchor point model*** was applied to identify the meaningful location at a monthly level. The dataset reveals the daily commuting of users between neighborhoods (spatial unit for aggregation).

The creation of this dataset is exceptional and fascinating, in a very high level of analysis and years of research at the Mobility Lab now is possible to comprehend the commuting of people in cities and provide feasible data to make decisions based on population dynamics.

I am glad I received permission from Anto Aasa to use this dataset for the creation of this educational material. The dataset can be used freely with proper reference and no commercial purposes.

> **Find the datasets HERE!**

The visualization of OD-matrix can be overwhelming due to the number of lines (movements) that are created. Here, you can visualize in a flow-map the daily commuting movements between the communities in Tartu county. As we can see, the map can reveal a higher number of commuters when it gets closer to the city center (Tartu city). But, the main objective of this practice is to make this flow map interactive (animated) so there is a better-granulated observation of movements. If you want to check more about OD-matrix visualization you can check the article ***"Bike***

[Sharing System movements in Helsinki: aggregation and visualization with an interactive flow-map"]

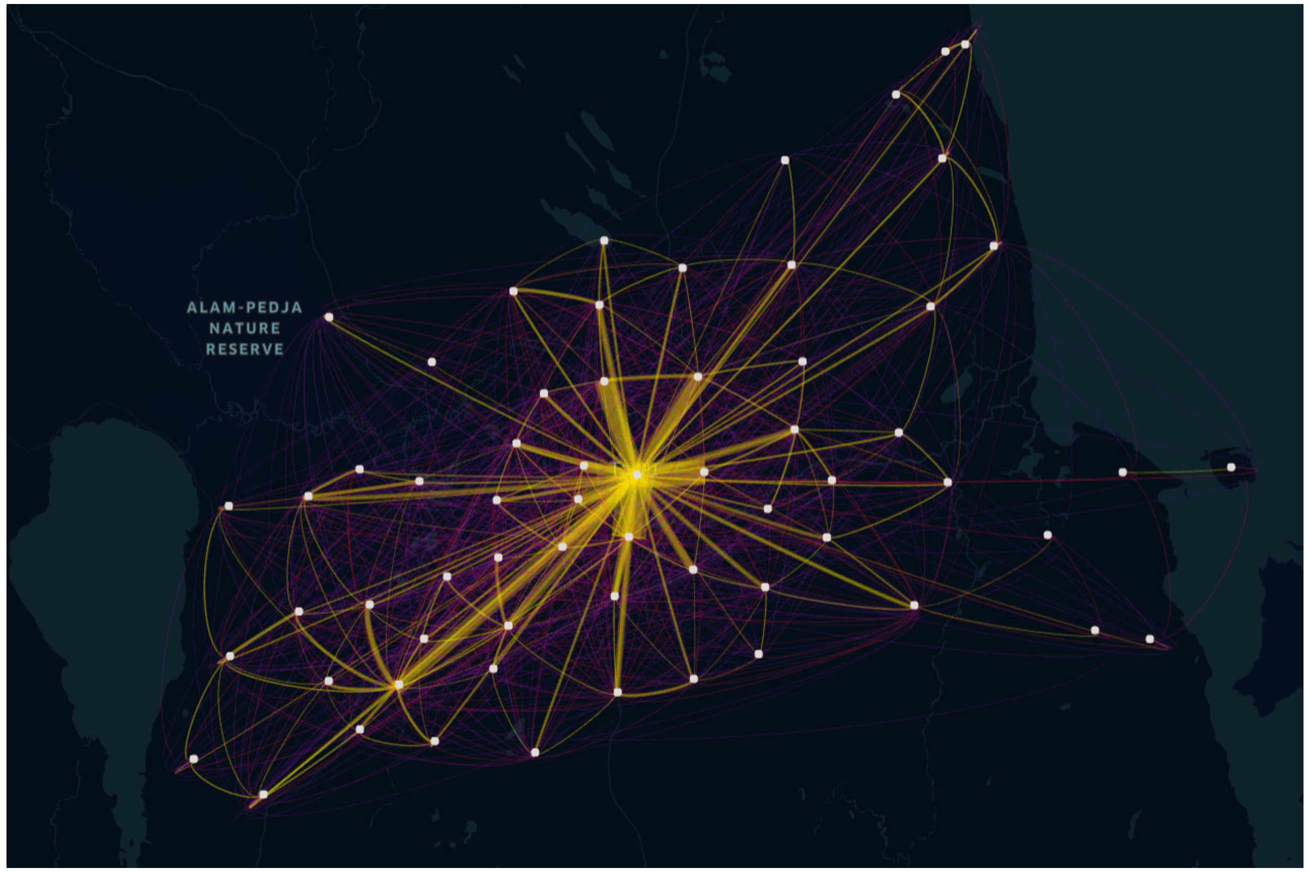


Image by the Author. Flow map with daily commuting home-work in Tarty county

> OBJECTIVE
>
> This visualization practice aims to create an interactive map animation (web map) in Leaflet that shows the movement between Home and Work of individuals (OD-Matrix) in Tartu, Estonia.
>
> **PRACTICE**

The visualization practice is divided into two sections: *1) Subset of movements based on Origin-Destination in Tartu County in **Python**, and 2) Interactive map animation of OD-Matrix in **Leaflet JavaScript**.*

The Repository contains only the subset of the general data. But you can re-use these codes for your own analysis. To clarify, the general dataset contains all neighborhoods as origins and all neighborhoods as destinations. In total there are 847 communities/neighborhoods and they can be seen on the next map:

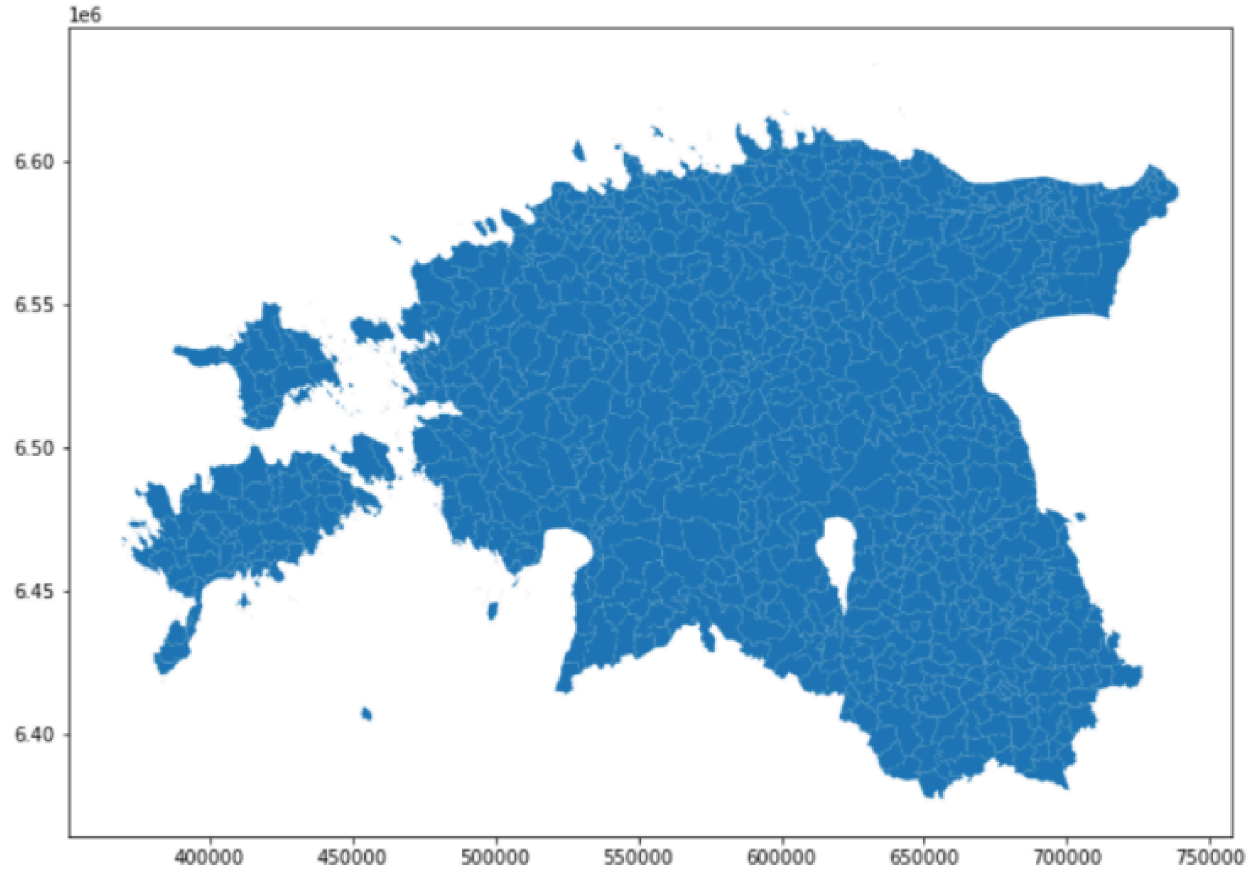


Image by the Author. Communities/neighborhoods of Estonia

> ***1) Defining Tartu County Origin-Destination dataset***

We want to visualize all trips that have **Tartu county as Origin** and all trips that have **Tartu as a Destination** at the national level.

1.1 Creating a list with Tartu County communities

We are going to read a layer with Tartu county communities and get a list with all the neighborhoods contained:

```
import geopandas as gpd
import pandas as pd
from pyproj import CRS
from shapely.geometry import Point

# tartu communities
fp = r'data/Communities_Tartu.shp'
geodata_tartu = gpd.read_file(fp)

# list with tartu communities for subset
tartu_communities = geodata_tartu['KANT_nk'].to_list()
```

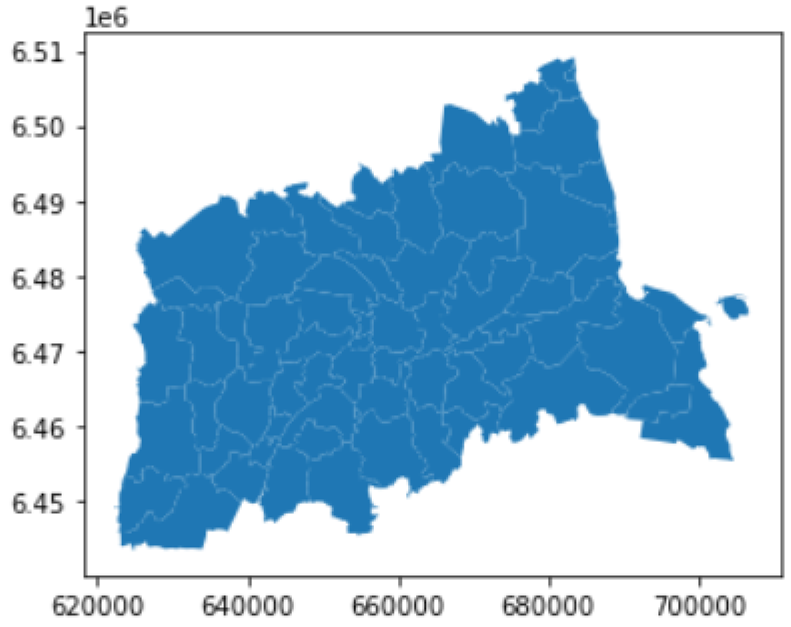


Image by Author. Neighborhoods/communities of Tartu County

1.2 Subset of Tartu County movements

Here, we are going to subset the origin and destinations only for Tartu County using the list `tartu_communities`. Keep in mind the whole dataset contains the movements at the national level.

```
# reading movement OD data
fp = r'data/OD_matrix_201708.csv'
movements = gpd.read_file(fp, encoding='Latin')

# selecting all origins from Tartu County
origins = movements[movements['KANT_start'].isin(tartu_communities)]

# selecting all destination from Tartu County origins
movements_tartu =  origins[origins['KANT_end'].isin(tartu_communities)]
```

We read it in *geopandas* because we need to geometry columns in the next step.

1.3 Coordinate transformation to WGS84 for Leaflet

In Leaflet, the dataset must be shown in WGS84 coordinates. Here, I am using a trick. I will project the dataset twice. The first time is to obtain the latitude and longitude in WGS84 of Origin, and the second time is to obtain coordinates for Destination. I project twice because by creating a geometry it releases in Estonian projection, then I project in WGS84, then I get the latitude and longitude. Like this:

```
# create a geometry column of ORIGIN
movements_tartu = movements_tartu.copy()

xorigins = movements_tartu['X_start'].to_list()
yorigins = movements_tartu['Y_start'].to_list()

movements_tartu['geometry'] = [Point(float(xcoor), float(ycoor)) for xcoor,
# defining data in Estonian coordinate system
movements_tartu.crs = CRS.from_epsg(3301)

# reprojecting to wgs84
movements_tartu = movements_tartu.to_crs(4326)

# add coordinates in wgs82 for ORIGIN
movements_tartu['x_origin'] = [coordinate.x for coordinate in movements_tart
movements_tartu['y_origin'] = [coordinate.y for coordinate in movements_tart

# update geometry for DESTINATION

# adding geometry with ending point just to obtain coordinates
xdest = movements_tartu['X_end'].to_list()
ydest = movements_tartu['Y_end'].to_list()

movements_tartu['geometry'] = [Point(float(xcoor), float(ycoor)) for xcoor,

# defining data in Estonian coordinate system
movements_tartu.crs = CRS.from_epsg(3301)

# reprojecting to wgs84
movements_tartu = movements_tartu.to_crs(4326)

# add coordinates in wgs82 DESTINATION
movements_tartu['x_dest'] = [coordinate.x for coordinate in movements_tartu[
movements_tartu['y_dest'] = [coordinate.y for coordinate in movements_tartu[

# update geometry for origin
movements_tartu['geometry'] = [Point(float(xcoor), float(ycoor)) for xcoor,

# defining data in Estonian coordinate system
movements_tartu.crs = CRS.from_epsg(3301)
```

```
# reprojecting to wgs84
movements_tartu = movements_tartu.to_crs(4326)
```

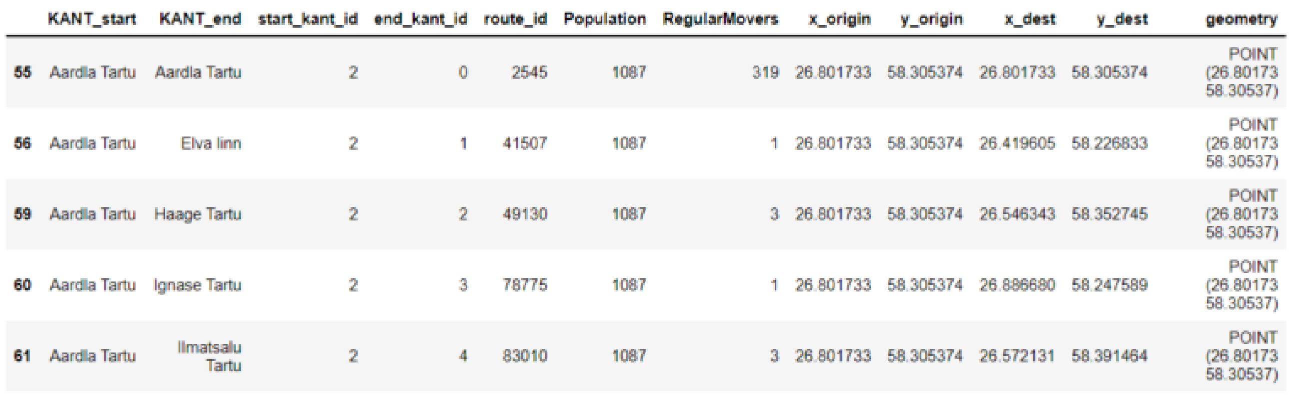
Now, I update codes for **OD visualization** and subset the needed columns with WGS84 coordinates.

```
# updating code of destinations
movements_tartu['end_kant_id'] = list(range(len(movements_tartu)))

movements_tartu['start_kant_id'] = movements_tartu['start_kant_id'].astype(i
movements_tartu['end_kant_id'] = movements_tartu['end_kant_id'].astype(int)

# getting the needed columns
movements_tartu = movements_tartu[['KANT_start', 'KANT_end', 'start_kant_id'
                                   'Population', 'RegularMovers', 'x_origin'

movements_tartu.to_file(r'data/movements_tartu.geojson', driver='GeoJSON')
movements_tartu.head()
```

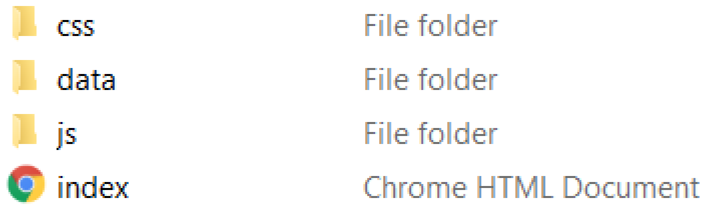


	KANT_start	KANT_end	start_kant_id	end_kant_id	route_id	Population	RegularMovers	x_origin	y_origin	x_dest	y_dest	geometry
55	Aardla Tartu	Aardla Tartu	2	0	2545	1087	319	26.801733	58.305374	26.801733	58.305374	POINT (26.80173 58.30537)
56	Aardla Tartu	Elva linn	2	1	41507	1087	1	26.801733	58.305374	26.419605	58.226833	POINT (26.80173 58.30537)
59	Aardla Tartu	Haage Tartu	2	2	49130	1087	3	26.801733	58.305374	26.546343	58.352745	POINT (26.80173 58.30537)
60	Aardla Tartu	Ignase Tartu	2	3	78775	1087	1	26.801733	58.305374	26.886680	58.247589	POINT (26.80173 58.30537)
61	Aardla Tartu	Ilmatsalu Tartu	2	4	83010	1087	3	26.801733	58.305374	26.572131	58.391464	POINT (26.80173 58.30537)

Image by the Author. The final structure of the Tartu County movements dataset

> ***2) Interactive map animation of daily commuting in Leaflet***

Now, we have to create a repository containing the necessary files to make the map animation work. We add a folder called `css` and another folder called `js.` Also, we add an HTML file called `index.html` You can obtain these files by cloning the repository in your local disk. So your local folder may look like these:



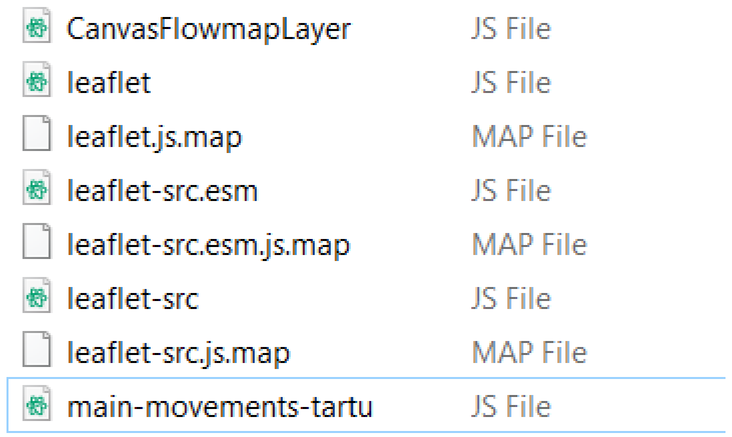
css	File folder
data	File folder
js	File folder
index	Chrome HTML Document

Image by the Author. Repository structure.

Be aware that inside the *data folder* there is already a `movements_tartu.geojson` file which is the results of step 1)

To work with the files for web mapping, I recommend using **Atom** or simply you can use **Notepad++**.

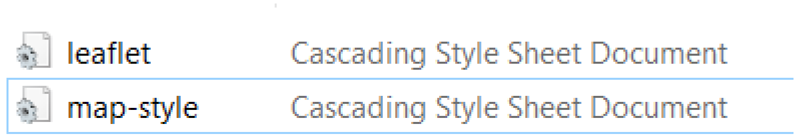
First, we are going to download and copy the **Leaflet API** for web mapping in the folder `js.` Additionally, we add an empty JS file named in this case `main-movements-tartu.js` Finally, we include the `CanvasFlowmapLayer.js` file which is included in the repository but can also be found in **jwalsilgeo Github**. The folder must look like this:



CanvasFlowmapLayer	JS File
leaflet	JS File
leaflet.js.map	MAP File
leaflet-src.esm	JS File
leaflet-src.esm.js.map	MAP File
leaflet-src	JS File
leaflet-src.js.map	MAP File
main-movements-tartu	JS File

Image by the Author. Structure of js folder

Then, in the folder, `css` we add the CSS file that comes from Leaflet and an empty file that we will call in this case `map-style.css` CSS folder looks like this:



leaflet Cascading Style Sheet Document
map-style Cascading Style Sheet Document

Image by the Author. Structure of CSS folder

2.1) Loading files to index.html

We are going to open the *index.html* file with Atom or Notepad++ and we start loading the files. It will contain a **head** and a **body**. In the body section, is where we include the main files that run the map animation, also the Leaflet files, and the data file. It includes the base map from ESRI.

```
<!DOCTYPE HTML>

<html>
 <head>
  <meta charset="utf-8">
  <title>Daily commuting movements home to work - Tartu</title>

<!--link to stylesheet-->
  <link rel="stylesheet" href="css/map-style.css">

<!-- link to leaflet stylesheet-->
  <link rel="stylesheet" href="css/leaflet.css">
 </head>

<body>
  <!--the sequence of elements called matters!-->
  <!-- title of your map-->
  <h1> Daily commuting movements home to work - Tartu </h1>

<!-- division div for the map -->
  <div id="map"></div>

<!-- link to leaflet javascript library-->
  <script src="js/leaflet-src.js"></script>

<!-- load Esri Leaflet because we want to use an Esri basemap -->
  <script src="https://unpkg.com/esri-leaflet@2.3/dist/esri-leaflet.js"></sc

<!-- Load animation tweening lib requirement for CanvasFlowMapLayer -->
  <script src="https://unpkg.com/@tweenjs/tween.js@18.5.0/dist/tween.umd.js'

<!-- then load CanvasFlowMapLayer -->
  <script src="js/CanvasFlowmapLayer.js"></script>

<!--link to the files that contains geoJson data-->
  <script src="data/movements_tartu.geojson" > </script>

<!-- link to main javascript file -->
  <script src="js/main-movements-tartu.js"></script>

</body>
<html>
```

2.2) Parameters in map-style.css

We have to style the HTML file and simply style the objects: *title, body, legend, and map*. Open the `map-style.css` and include the next code:

```
h1{
 position: fixed;
 font-family: "Times New Roman", Times, serif;
 font-size: 24px;
  box-shadow: 2px 2px 3px 3px black;
 background: lightgray;
    color: black;
    margin-left:5%;
    margin-top: 0.6%;
    z-index: 2;
}
body{
 width: 100%;
 height: 100%;
 margin: 0px;
 font-family: "Times New Roman";
}
.info {
    padding: 6px 8px;
    font-family: "Times New Roman", Times, sans-serif;
    font-size: 16px;
    background: white;
    background: rgba(255,255,255,0.8);
    box-shadow: 0 0 15px rgba(0,0,0,0.2);
    border-radius: 5px;
}
.info h3 {
  font-size: 18px;
  font-family:  "Times New Roman", Times, serif;
  text-decoration: underline;
  text-shadow: 2px 2px 5px gray;
    margin: 0 0 5px;
    color: #282825   ;
}
.legend {
    line-height: 20px;
    color: #555;
}
.legend i {
    width: 25px;
    height: 18px;
    float: left;
    margin-right: 8px;
    opacity: 0.7;
}
#map {
 height:100%;
 width:100%;
 left:0%;
 overflow:hidden;
 position:fixed;
 border:1px #444 solid;
}
```

If you open the index.html in the browser you may look at an empty canvas like this:

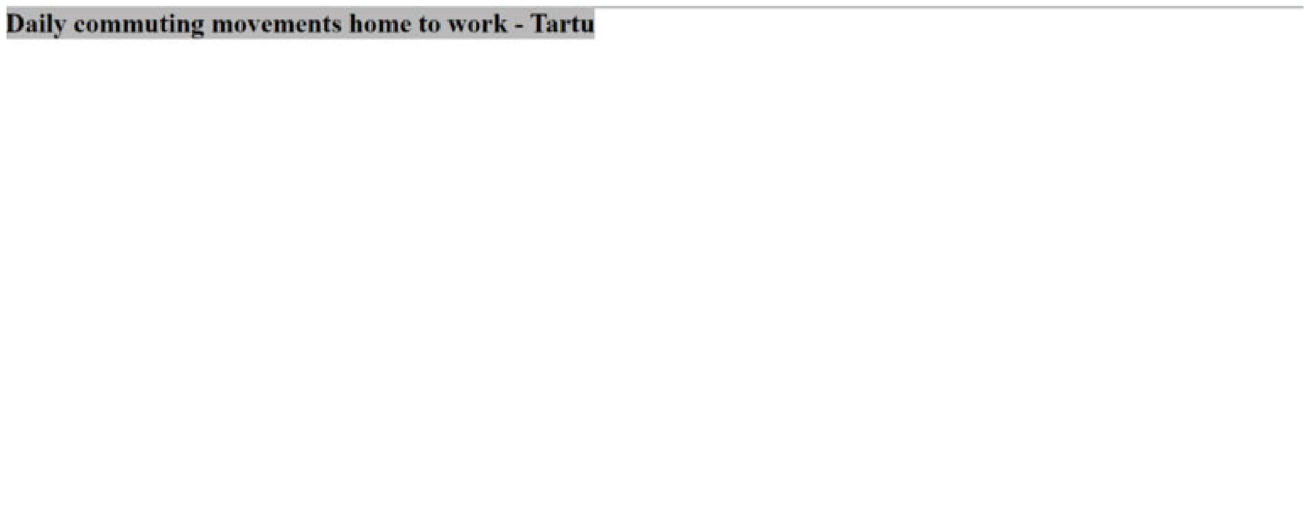
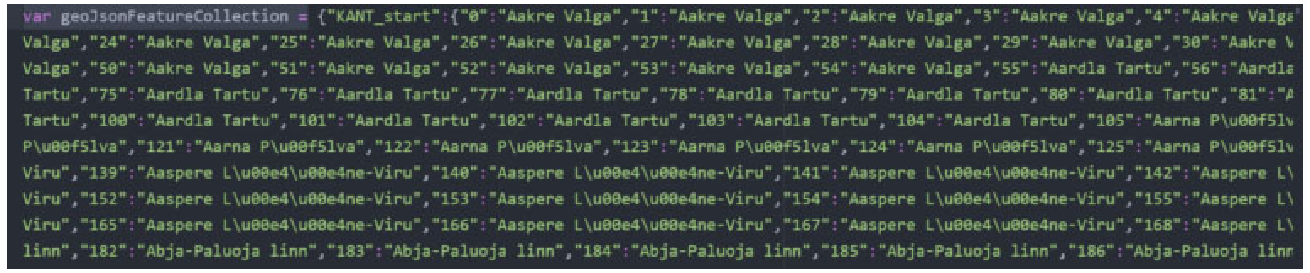


Image by the Author. Empty canvas

2.3) Defining a variable in data movements-tartu.js

To include the movement dataset in Leaflet, you may need to define it as a variable. You need to open the data file, and include `var geoJsonFeatureCollection =` So, the data file must look like this:



```
var geoJsonFeatureCollection = {"KANT_start":{"0":"Aakre Valga","1":"Aakre Valga","2":"Aakre Valga","3":"Aakre Valga","4":"Aakre Valga
Valga","24":"Aakre Valga","25":"Aakre Valga","26":"Aakre Valga","27":"Aakre Valga","28":"Aakre Valga","29":"Aakre Valga","30":"Aakre V
Valga","50":"Aakre Valga","51":"Aakre Valga","52":"Aakre Valga","53":"Aakre Valga","54":"Aakre Valga","55":"Aardla Tartu","56":"Aardla
Tartu","75":"Aardla Tartu","76":"Aardla Tartu","77":"Aardla Tartu","78":"Aardla Tartu","79":"Aardla Tartu","80":"Aardla Tartu","81":"A
Tartu","100":"Aardla Tartu","101":"Aardla Tartu","102":"Aardla Tartu","103":"Aardla Tartu","104":"Aardla Tartu","105":"Aarna P\u00f5lv
P\u00f5lva","121":"Aarna P\u00f5lva","122":"Aarna P\u00f5lva","123":"Aarna P\u00f5lva","124":"Aarna P\u00f5lva","125":"Aarna P\u00f5lv
Viru","139":"Aaspere L\u00e4\u00e4ne-Viru","140":"Aaspere L\u00e4\u00e4ne-Viru","141":"Aaspere L\u00e4\u00e4ne-Viru","142":"Aaspere L\
Viru","152":"Aaspere L\u00e4\u00e4ne-Viru","153":"Aaspere L\u00e4\u00e4ne-Viru","154":"Aaspere L\u00e4\u00e4ne-Viru","155":"Aaspere L\
Viru","165":"Aaspere L\u00e4\u00e4ne-Viru","166":"Aaspere L\u00e4\u00e4ne-Viru","167":"Aaspere L\u00e4\u00e4ne-Viru","168":"Aaspere L\
linn","182":"Abja-Paluoja linn","183":"Abja-Paluoja linn","184":"Abja-Paluoja linn","185":"Abja-Paluoja linn","186":"Abja-Paluoja linn
```

Image by the Author. Variable definition for data

2.4) Creating a map animation in main-movement-tartu.js

Here, we are going to start creating the map with Leaflet and animate the movements in Tartu County neighborhoods. Let's go step by step. You can take a look at the ***Leaflet Interactive Choropleth Map*** example where I took some help for establishing the map variable and defining the colors for legend.

To start, we add a variable for the map, defining the zoom level and center. Then, we add an ESRI base map.

```
//--- PART 1: ADDING BASE MAPS AND SCALE BAR ---

// variable for the map
var map = L.map('map', {
 center: [58.57, 25.5],
 zoom: 8
});

var esri = L.esri.basemapLayer('DarkGray');
 esri.addTo(map);

L.control.scale({imperial:false, position:'bottomright'}).addTo(map);
```

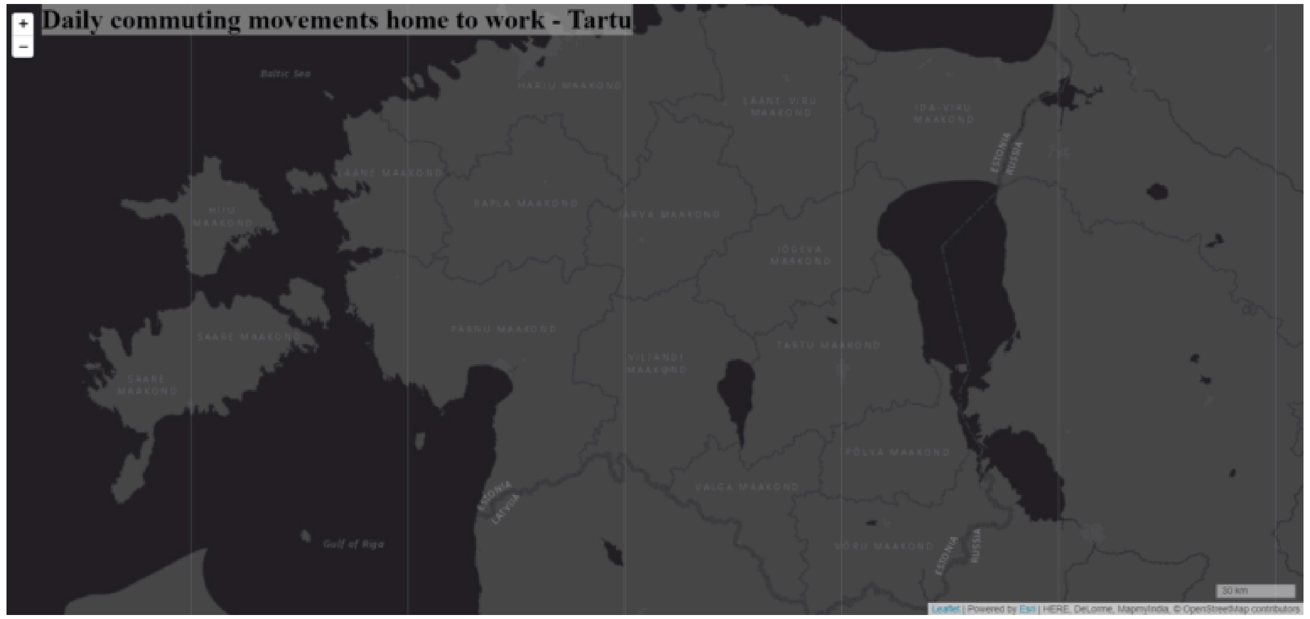


Image by Author. Map object added from Leaflet and ESRI base map

Now, we add the movements data with the animation function.

```
// - - PART 2: ADDING ANIMATION OF OD MATRIX WITH VARIABLE geoJsonFeatureCol
```

```
var oneToManyFlowmapLayer = L.canvasFlowmapLayer(geoJsonFeatureCollection, {
 originAndDestinationFieldIds: {
 originUniqueIdField: 'start_kant_id',
 originGeometry: {
 x: 'x_origin',
 y: 'y_origin'
 },
 destinationUniqueIdField: 'end_kant_id',
 destinationGeometry: {
 x: 'x_dest',
 y: 'y_dest'
 }
 },
 canvasBezierStyle: {
 type: 'classBreaks',
 field: 'RegularMovers',
 classBreakInfos: [{
 classMinValue: 0,
 classMaxValue: 24,
 symbol: {
 strokeStyle: '#fee8c8',
 lineWidth: 0.5,
 lineCap: 'round',
 shadowColor: '#fee8c8',
 shadowBlur: 2.0
 }
 }, {
 classMinValue: 25,
 classMaxValue: 100,
 symbol: {
```

```
 strokeStyle: '#fdbb84',
 lineWidth: 1.5,
 lineCap: 'round',
 shadowColor: '#fdbb84',
 shadowBlur: 2.0
 }
 }, {
 classMinValue: 101,
 classMaxValue: 10000000,
 symbol: {
 strokeStyle: '#e34a33',
 lineWidth: 3,
 lineCap: 'round',
 shadowColor: '#e34a33',
 shadowBlur: 2.0
 }
 }],
 defaultSymbol: {
 strokeStyle: '#e7e1ef',
 lineWidth: 0.5,
 lineCap: 'round',
 shadowColor: '#e7e1ef',
 shadowBlur: 1.5
 },
 },
 pathDisplayMode: 'selection',
 animationStarted: true
}).addTo(map);
```

```
// Selection for dispaly
oneToManyFlowmapLayer.on('mouseover', function(e) {
 if (e.sharedOriginFeatures.length) {
 oneToManyFlowmapLayer.selectFeaturesForPathDisplay(e.sharedOriginFeatures,
 }
 if (e.sharedDestinationFeatures.length) {
 oneToManyFlowmapLayer.selectFeaturesForPathDisplay(e.sharedDestinationFeatu
 }
});

oneToManyFlowmapLayer.selectFeaturesForPathDisplayById('start_kant_id', 673,
```

If you update `index.html` you already can visualize the animation.

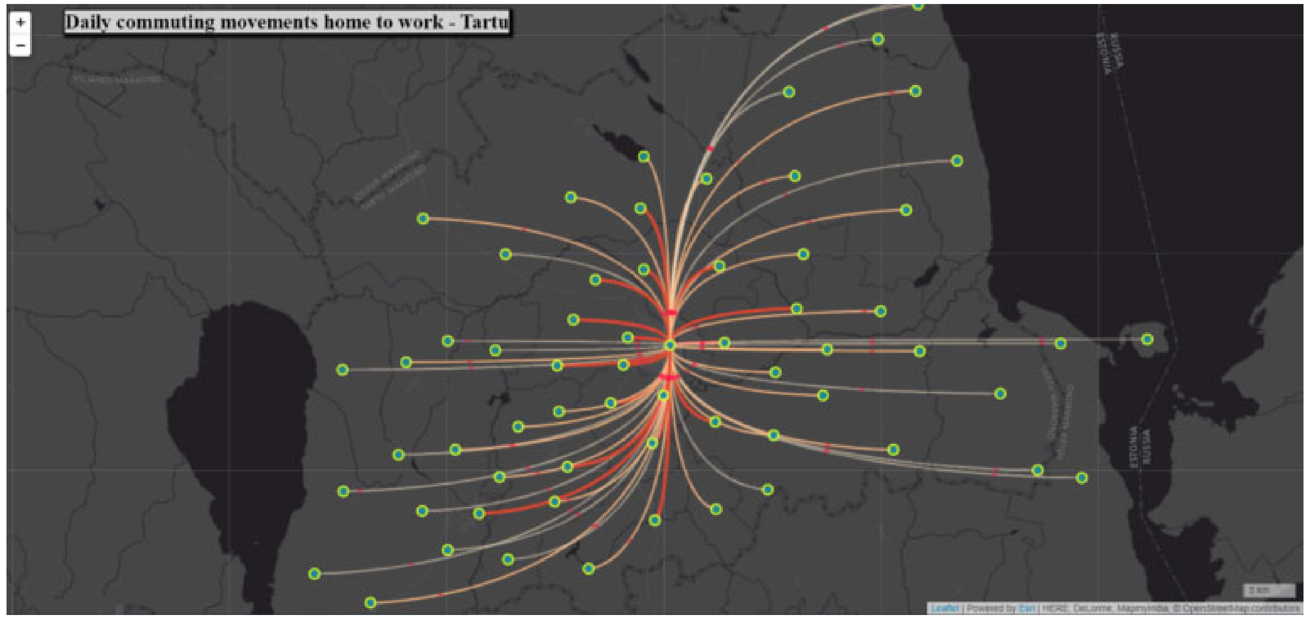


Image by the Author. Interactive flow map with daily commuting home-work in Tarty county

We have defined that the movement will be visualized with `mouseover` action. So, the interaction with the map animation is only by hovering the mouse around the elements. Also, we defined 3 classes and the coordinates for Origin and Destination.

Finally, we add the legend with the colors, in case you need support by selecting color you can use **ColorBrewer for maps**:

```
//PART 3. ADDING A LEGEND WITH COLORS

function getColor(d) {
return d > 100  ? '#e34a33' :
       d > 24   ? '#fdbb84' :
                '#fee8c8' ;
}

var legendcolor = L.control({position: 'bottomright'});

legendcolor.onAdd = function (map) {
  var div = L.DomUtil.create('div', 'info legend'),
    grades = [0, 24, 100],
    labels = [];
    // loop through our density intervals and generate a label with a colore
        for (var i = 0; i < grades.length; i++) {
            div.innerHTML +=
            '<i class ="line" style="background:' + getColor(grades[i] + 1) 
            grades[i] + (grades[i + 1] ? '&amp;ndash;' + grades[i + 1] + '<b

}    return div;
       };
legendcolor.addTo(map);
```

With legend looks like this.

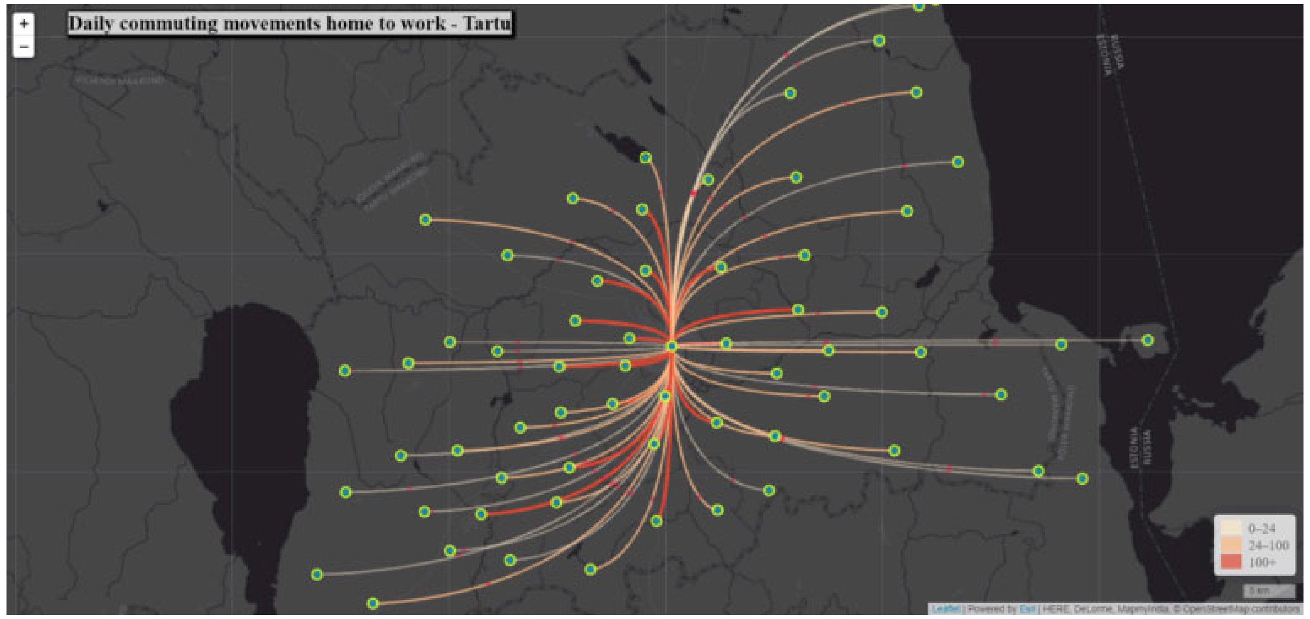


Image by the Author. Final map with legend

RECOMMENDATION

If you are experienced with Leaflet-JavaScript for Web Mapping. You can add the communities as background, or an info box with the name of the neighborhoods. I leave this freely for users. But now, you have the base of an Origin-Destination Matrix, for Tartu County.

CONCLUSION

The map animation of home-work movements in Tartu county helps urban planners to understand the population dynamics (spatial mobility). With this, transportation can be improved especially for travel times. Of course, people prefer to go fast to work. Also, local investment can be a long-term option, to invest in real estate for offices. If people are moving for work for sure it is a potential place to grow economically.

If you need support in OD matrix visualization or coding maps with Python you can find me in my LinkedIn profile or just leave a comment in this article.

REFERENCES

[1] Ahas, R., Silm, S., Järv, O., Saluveer, E., Tiru, M. (2010). ***"Using Mobile Positioning Data to Model Locations Meaningful to Users of Mobile Phones".*** Volume 17. DOI https://doi.org/10.1080/10630731003597306

[2] Aasa, A. (2019). ***OD-matrices of daily regular movements in Estonia [Data set]***. University of Tartu, Mobility Lab. https://doi.org/10.23659/UTMOBLAB-1

WRITTEN BY

Bryan R. Vallejo

See all from Bryan R. Vallejo

Animation Estonia Geospatial Maps Mobility

Share This Article

Write for TDS

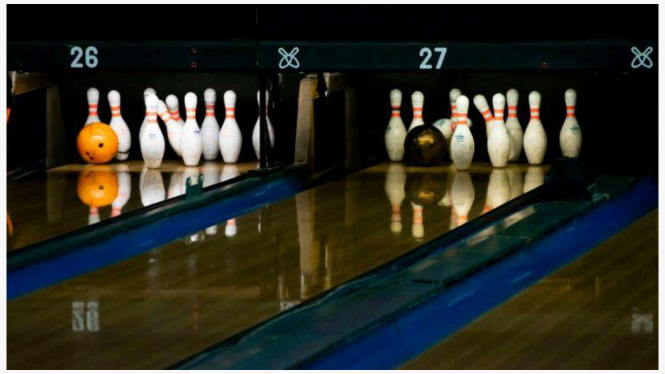
Related Articles

Spotting Spatiotemporal Patterns in Earthquake Data

Use density-based clustering and survival analysis to estimate when earthquakes occur

Elliot Humphrey

January 10, 2024 • 10 min read



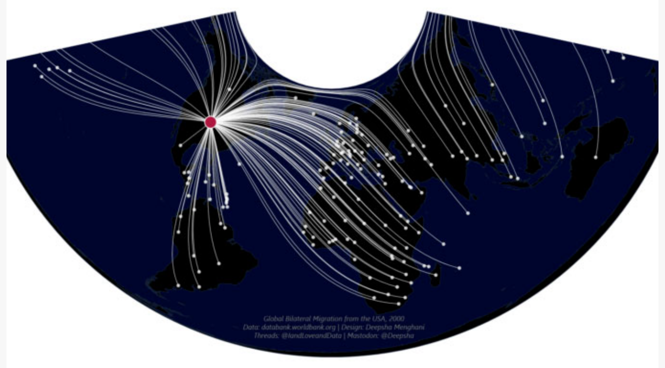
DATA SCIENCE

Exploring Real and Virtual Spaces with Data

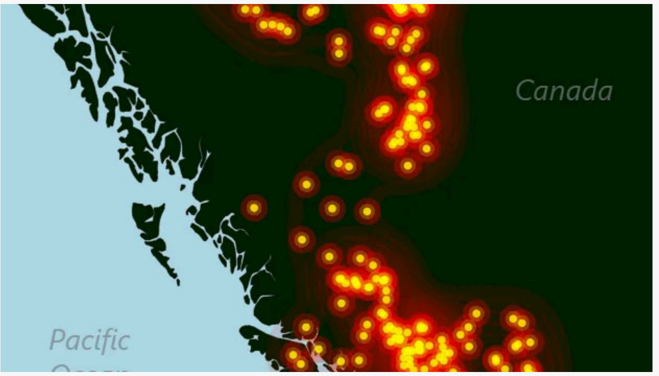
Our weekly selection of must-read Editors' Picks and original features

TDS Editors

April 4, 2024 • 3 min read



JAVASCRIPT



JAVASCRIPT

Charting the Final Frontier: Completing the #30DayMapChallenge Odyssey

From Urban Collisions to Global Connections: Unveiling the Full Spectrum of Geo-Visual Storytelling with Observable…

Deepsha Menghani

November 29, 2023 • 6 min read

DATA SCIENCE

Exploring a Global Wildlife GIS database

Using Python to characterize the International Union for Conservation of Nature (IUCN)'s geospatial data base.

Milan Janosov

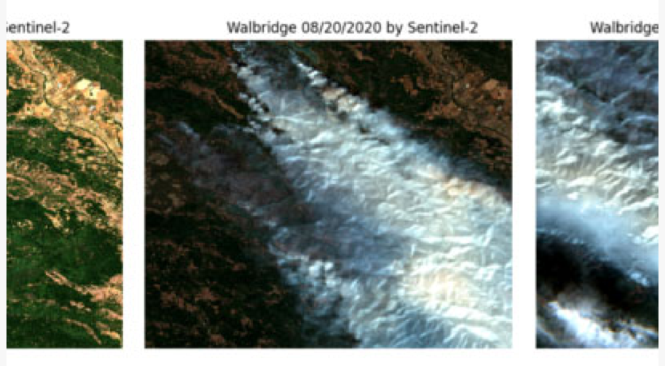
October 19, 2023 • 13 min read

DATA SCIENCE

Navigating the Cartographic Challenge: Halfway Through the #30DayMapChallenge

From Haunted Locales to Political Patterns: A Mapping Journey Through Data and Design with Observable…

Deepsha Menghani



CLIMATE CHANGE

Quantifying Burned Areas from Wildfires Using Satellite Imagery

Determining the burned area in forests due to wildfires using Sentinel-2 images with Python in…

Mahyar Aboutalebi, Ph.D. 🎓

July 22, 2024 • 9 min read

Finding Patterns in Convenience Store Locations with Geospatial Association Rule Mining

Understanding spatial trends in the location of Tokyo convenience stores

Elliot Humphrey

April 1, 2023 • 7 min read

Subscribe to Our Newsletter

WRITE FOR TDS · ABOUT · ADVERTISE · PRIVACY POLICY · TERMS OF USE

COOKIES SETTINGS