Geovisualization - Assignment 3

Introduction

For our final project, we decided to work with Leaflet. With this project, we wanted to achieve the following goals: The first goal of ours was to create a flow map. One of the most interesting datasets for this kind of project is in our opinion migration data. This way we can showcase the movement of many people and it may also imply the relationship of people to other countries - for example you can see in the final outcome the strong relationship between Austria, Germany and Switzerland. For the task, we decided to use data of the immigrations and emigrations from and to Austria in the year of 2020. We decided to use Leaflet because we wanted to get to know the flow map plugin. Moreover, we preferred Leaflet because it is an open-source library and therefore we have full control over our code for the map. In general, we were keen to use as much open source software as possible.

The reasons *why* we decided to do this project are that we first of all wanted to work with a real-world dataset that showcases migration. Furthermore, we were very interested in where people came from and where they were going, regardless of their nationality. Finally, the leaflet plugin makes it possible to visualize how many people migrated - which we were keen to use.

Project Data and Geocoding

Data

We obtained the according data from [Statistik Austria](). Most available datasets we've found document which nationality the migrating people have. As this does not really tell how people migrated, we have put emphasis on the data showing exactly where people came from and where they went to. The dataset of Statistik Austria does meet this requirement.

	A	B	C	D	E	F	G	H	I	J
1	**Wanderungen mit dem Ausland (Außenwanderungen) 2020 nach Herkunfts- und Zielländern**									
2										
3		Insgesamt			Österreichische Staatsangehörige			Ausländische Staatsangehörige		
4	Herkunfts-/Zielland	Zuzüge aus dem Ausland	Wegzüge in das Ausland	Saldo	Zuzüge aus dem Ausland	Wegzüge in das Ausland	Saldo	Zuzüge aus dem Ausland	Wegzüge in das Ausland	Saldo
6	**Insgesamt**	**136.343**	**96.279**	**40.064**	**15.032**	**16.869**	**-1.837**	**121.311**	**79.410**	**41.901**
7	**EU-, EFTA-Staaten**	**89.026**	**59.797**	**29.229**	**3.993**	**6.227**	**-2.234**	**85.033**	**53.570**	**31.463**
8	**EU-Staaten (27)**	**84.940**	**55.852**	**29.088**	**3.112**	**4.428**	**-1.316**	**81.828**	**51.424**	**30.404**
9	**EU-Staaten vor 2004 (14)**	**35.738**	**23.264**	**12.474**	**2.646**	**3.768**	**-1.122**	**33.092**	**19.496**	**13.596**
10	Belgien	470	379	91	60	51	9	410	328	82
11	Dänemark	270	239	31	23	42	-19	247	197	50
12	Deutschland	23.147	13.646	9.501	1.924	2.824	-900	21.223	10.822	10.401
13	Finnland	292	343	-51	12	17	-5	280	326	-46
14	Frankreich	1.432	1.336	96	95	145	-50	1.337	1.191	146
15	Griechenland	1.102	822	280	42	46	-4	1.060	776	284
16	Irland	346	240	106	28	41	-13	318	199	119
17	Italien	4.516	2.883	1.633	160	177	-17	4.356	2.706	1.650
18	Luxemburg	195	117	78	16	14	2	179	103	76
19	Niederlande	1.129	914	215	69	139	-70	1.060	775	285
20	Portugal	570	500	70	22	44	-22	548	456	92
21	Schweden	422	425	-3	43	72	-29	379	353	26
22	Spanien	1.847	1.420	427	152	156	-4	1.695	1.264	431
23	**EU-Beitrittsstaaten ab 2004 (13)**	**49.202**	**32.588**	**16.614**	**466**	**660**	**-194**	**48.736**	**31.928**	**16.808**
24	Bulgarien	4.219	2.946	1.273	34	33	1	4.185	2.913	1.272
25	Estland	85	79	16	1	5	-4	84	74	20

Figure 1 - Screenshot of original migration data

As you can see in Figure 1, the original formatting of the data is not suitable - it would be hard (or impossible) for Leaflet to process it because there are enlarged cells that cover more than one “space”. This is possible in a xlsx-file, but a csv-file for example does not allow such kind of formatting. Additionally, the data contains unnecessary rows like the overall title, and summarizing rows, such as “Insgesamt” (engl.: Total).

Figure 2 - Work steps to customize the data

For the reasons mentioned above, the first step in cleaning our data is to remove unnecessary and disturbing rows.

Geocoding

The next step is to geocode our data. The original data does not hold any geographic information besides the country name. For the flow map it is important to know the coordinates of the countries. To solve this issue, the QGIS-plugin MMQGIS was used. The original data, which has been cleaned as far as possible, was transferred to the plugin as a csv file. Based on the country data, the corresponding coordinates are now determined. Nominatim is used for this process of “coordinate assignment”. It is the tool with which OpenStreetMap geocodes, i.e. which finds the corresponding coordinates from the address that the user types into the search window.

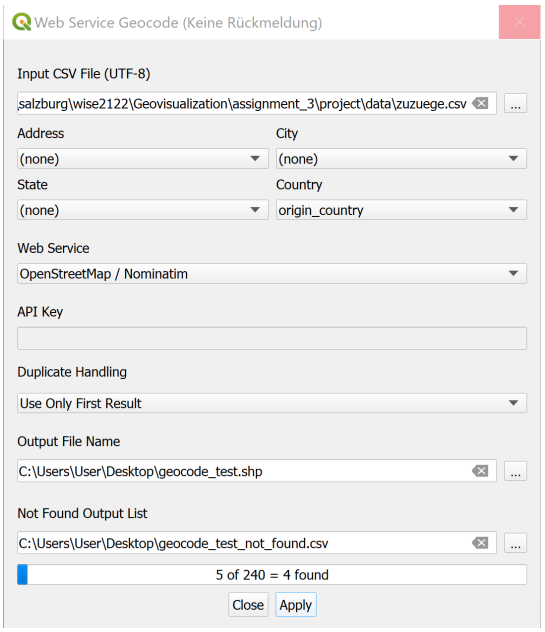

Figure 3 - Geocoding the data using MMQGIS

Of the total 240 records, 232 could be geocoded by MMQGIS. These were displayed in QGIS and they were also saved in a local shape-file. The 8 datasets that were not matched based on their country name were saved in a csv file. Our guess for the reason why these countries or regions could not be geocoded is the following: All countries that fall under this category were listed in the original data under rather unusual names - for example "Alderney,

Guernsey u. Sark [GBR]", "Sudan and Südsudan" (engl.: Sudan and South Sudan), or "Tansania - Vereinigte Republik" (engl.: Tanzania - United Republic). Although these names are a bit unusual, it is possible for human-kind to understand them. That's why we simply set the corresponding points in QGIS ourselves.
Unfortunately, this was not the only problem that geocoding brought us: Not all of the geocoded coordinates were correct, for example the entry "Syria - Arab Republic" was placed in Brussels on the embassy of Syria, or the entry "Sankt Martin (franz.) [FRA]" was identified as Franz Korbadits Street in Sankt Martin in der Wart. We removed all the wrongly geocoded points and put them to the right place. However, since we did this by hand, we can't be completely sure if we fixed all of these bugs. Especially because we are mostly familiar with Europe, but would be more likely to miss less obvious bugs concerning the other continents. Nevertheless, since the names of the countries are documented in German in the original data, it would make sense that most of the wrongly set points were put in the german-speaking countries - as there the names of places are similar to the names of foreign countries in the german language. An example for this conjecture would be Sankt Martin that was mentioned above.
In addition, the MMQGIS plugin adds, among other things, a relevant column to the dataset: the "type" column indicates to which type the added point belongs (see Figure 4). We checked the relevant entries that fall out of the grid (such as bus stops or motels) and saw no need for further action, since each of the points had already been corrected and "relocated" to the correct place on the world map.

	count bigint	type character varying (254)
1	220	administrative
2	8	[null]
3	1	land_area
4	2	motel
5	1	region
6	1	disputed
7	1	residential
8	2	diplomatic
9	1	bus_stop
10	3	island

Figure 4 - the OSM types of the locations that were set by MMQGIS

Dividing the data into immigration and emigration

Next, we wanted to be able to visualize the immigration and emigration in separate flows. Therefore, we have decided to split up the dataset in two. As you can see in Figure 1, the original data holds both information in one line. To adjust the data to our needs we have used PostgreSQL and its spatial extension PostGIS.
We imported the shape-file that was created by the plugin MMQGIS - and at this point of time already corrected - via QGIS toPostGIS.

	id [PK] integer	geom geometry	herkunfts character varying (254)	zuzug integer	wegzug integer	saldo integer	display_na character varying (254)	type character varying (254)	latlong character varying (254)
1	1	0101000020110F000...	Belgien	470	379	91	België / Belgique / Belgien	administrative	50.6402809,4.6667145
2	2	0101000020110F000...	Dänemark	270	239	31	Danmark	administrative	55.670249,10.3333283
3	3	0101000020110F000...	Deutschland	23147	13646	9501	Deutschland	administrative	51.0834196,10.4234469
4	4	0101000020110F000...	Finnland	292	343	-51	Suomi / Finland	administrative	63.2467777,25.9209164
5	5	0101000020110F000...	Frankreich	1432	1336	96	France	administrative	46.603354,1.8883335
6	6	0101000020110F000...	Griechenland	1102	822	280	Ελλάς	administrative	38.9953683,21.9877132
7	7	0101000020110F000...	Irland	346	240	106	Éire / Ireland	administrative	52.865196,-7.9794599
8	8	0101000020110F000...	Italien	4516	2883	1633	Italia	administrative	42.6384261,12.674297
9	9	0101000020110F000...	Luxemburg	195	117	78	Lëtzebuerg	administrative	49.8158683,6.1296751
10	10	0101000020110F000...	Niederlande	1129	914	215	Nederland	administrative	52.15517,5.38721
11	11	0101000020110F000...	Portugal	570	500	70	Portugal	administrative	40.0332629,-7.8896263
12	12	0101000020110F000...	Schweden	422	425	-3	Sverige	administrative	59.6749712,14.5208584
13	13	0101000020110F000...	Spanien	1847	1420	427	España	administrative	39.3260685,-4.8379791
14	14	0101000020110F000...	Bulgarien	4219	2946	1273	България	administrative	42.6073975,25.4856617
15	15	0101000020110F000...	Estland	95	79	16	Eesti	administrative	58.71971295,24.5075441067...
16	16	0101000020110F000...	Kroatien	6020	2195	3825	Hrvatska	administrative	45.5643442,17.0118954

Figure 5 - Geocoded data in PostGIS

You can see in Figure 5 that not all columns of the original data were transferred. For example, we did not attach any importance to whether the migrants are Austrian citizens or not, which is why we did not include the corresponding information. However, some additional information that was helpful to us was included in the table, either added by MMQGIS (display_na and type) or generated using QGIS (latlong column). This table was then divided into the two tables geovisualization_wegzuege and geovisualization_zuzuege.

The first step was to create these tables and transfer the information. This was done by executing the following SQL statement (exemplary for the immigrations):

```
CREATE TABLE geovisualization_zuzuege AS (
    SELECT id AS origin_id, herkunfts AS origin_country,
        ST_X(ST_Transform(geom,4326)) AS origin_lon,
        ST_Y(ST_Transform(geom,4326)) AS origin_lat,
        zuzug AS count, saldo, geom
    FROM geocoded_countries
);
```

Code 1 - SQL statement to create the table geovisualization_zuzuege

The Flowmap plugin for Leaflet has very special requirements as far as the data structure is concerned. For this reason, additional columns have to be added which address the destination of the respective flows. In the case of immigrations (to Austria) this is of course always Austria. Therefore the following SQL statement (code 2) could be applied to each row without a restrictive WHERE clause (in the UPDATE statement).

```
ALTER TABLE test_geovisualization_zuzuege
ADD COLUMN destination_id INT,
ADD COLUMN destination_country VARCHAR,
ADD COLUMN destination_lon DOUBLE PRECISION,
ADD COLUMN destination_lat DOUBLE PRECISION;

UPDATE test_geovisualization_zuzuege
SET destination_id = 999,
    destination_country = 'Österreich',
    destination_lon = 14.550072,
    destination_lat = 47.516232;
```

Code 2 - Adding destination information to the table

Approximately the same procedure is performed for the geovisualization_wegzuege table, but of course the SQL scripts had to be adapted accordingly.

Building CSV

Last but not least, our data preparation includes the creation of the corresponding CSV files. For the sake of simplicity, QGIS is used for this: The two tables are loaded into a QGIS project via the connection to PostGIS. Then they are exported in CSV format. You can view the results of the data preparation in Figures 6 (Emigration) and 7 (Immigration).

origin_id	origin_country	origin_lon	origin_lat	destination_id	destination_country	destination_lon	destination_lat	count	saldo
999	Österreich	14.550072	47.516232	1	Belgien	4.6667145	50.6402809	379	91
999	Österreich	14.550072	47.516232	2	Dänemark	10.3333283	55.670249	239	31
999	Österreich	14.550072	47.516232	3	Deutschland	10.4234469	51.0834196	13646	9501
999	Österreich	14.550072	47.516232	4	Finnland	25.9209164	63.2467777	343	-51
999	Österreich	14.550072	47.516232	5	Frankreich	1.8883335	46.603354	1336	96
999	Österreich	14.550072	47.516232	6	Griechenland	21.9877132	38.9953683	822	280
999	Österreich	14.550072	47.516232	7	Irland	-7.9794599	52.865196	240	106
999	Österreich	14.550072	47.516232	8	Italien	12.674297	42.6384261	2883	1633
999	Österreich	14.550072	47.516232	9	Luxemburg	6.1296751	49.8158683	117	78
999	Österreich	14.550072	47.516232	10	Niederlande	5.38721	52.15517	914	215
999	Österreich	14.550072	47.516232	11	Portugal	-7.8896263	40.0332629	500	70
999	Österreich	14.550072	47.516232	12	Schweden	14.5208584	59.6749712	425	-3
999	Österreich	14.550072	47.516232	13	Spanien	-4.8379791	39.3260685	1420	427
999	Österreich	14.550072	47.516232	14	Bulgarien	25.4856617	42.6073975	2946	1273
999	Österreich	14.550072	47.516232	15	Estland	24.5075441067899	58.71971295	79	16
999	Österreich	14.550072	47.516232	16	Kroatien	17.0118954	45.5643442	2195	3825

Figure 6 - Extract of the processed emigration data in csv format

origin_id	origin_country	origin_lon	origin_lat	destination_id	destination_country	destination_lon	destination_lat	count	saldo
1	Belgien	4.6667145	50.6402809	999	Österreich	14.550072	47.516232	470	91
2	Dänemark	10.3333283	55.670249	999	Österreich	14.550072	47.516232	270	31
3	Deutschland	10.4234469	51.0834196	999	Österreich	14.550072	47.516232	23147	9501
4	Finnland	25.9209164	63.2467777	999	Österreich	14.550072	47.516232	292	-51
5	Frankreich	1.8883335	46.603354	999	Österreich	14.550072	47.516232	1432	96
6	Griechenland	21.9877132	38.9953683	999	Österreich	14.550072	47.516232	1102	280
7	Irland	-7.9794599	52.865196	999	Österreich	14.550072	47.516232	346	106
8	Italien	12.674297	42.6384261	999	Österreich	14.550072	47.516232	4516	1633
9	Luxemburg	6.1296751	49.8158683	999	Österreich	14.550072	47.516232	195	78
10	Niederlande	5.38721	52.15517	999	Österreich	14.550072	47.516232	1129	215
11	Portugal	-7.8896263	40.0332629	999	Österreich	14.550072	47.516232	570	70
12	Schweden	14.5208584	59.6749712	999	Österreich	14.550072	47.516232	422	-3
13	Spanien	-4.8379791	39.3260685	999	Österreich	14.550072	47.516232	1847	427
14	Bulgarien	25.4856617	42.6073975	999	Österreich	14.550072	47.516232	4219	1273
15	Estland	24.5075441067899	58.71971295	999	Österreich	14.550072	47.516232	95	16
16	Kroatien	17.0118954	45.5643442	999	Österreich	14.550072	47.516232	6020	3825
17	Lettland	24.7537645	56.8406494	999	Österreich	14.550072	47.516232	243	63

Figure 7 - Extract of the processed emigration data in csv format

Leaflet Map with Flow Maps

We worked with Leaflet and one example from GitHub called Canvas-Flowmap-Layer which is a custom layer plugin for Leaflet JavaScript (Wasilkowski, J., 2017). It was developed to map the flow of objects from an origin point to a destination point by using a Bezier curve. GeoJSON point feature coordinates are translated to pixel space so that rendering for the points and curves can be mapped to a Canvas Element in HTML. One of the code examples from this GitHub was used for our project.
Our web map uses the open-source JavaScript library for mobile-friendly interactive maps called Leaflet (Leafletjs.com. 2022). With a css-stylesheet the look of the website is provided and with different scripts the map works as intended.

PlugIn Canvas-FlowMap-Layer

One of those scripts is the plugin Canvas-Flowmap-Layer. With this plugin, different data relationships are possible to visualize. It supports "one-to-many", "many-to-one" and "one-to-one" origin-to-destination relationships. The right data format is expected for the plugin. Therefore, a correct geojson must be provided. Both origin and destination coordinates and attributes need to be available for each point feature.

Example for a point feature in a feature collection from Austria to Belgium:

{ "type": "FeatureCollection",
"name": "wegzuege",
"crs": { "type": "name", "properties": { "name": "urn:ogc:def:crs:OGC:1.3:CRS84" } },
"features": [
{ "type": "Feature", "properties": { "origin_id": 999, "origin_country": "Österreich", "origin_lon": 14.550072, "origin_lat": 47.516232, "destination_id": 1, "destination_country": "Belgien", "destination_lon": 4.6667145, "destination_lat": 50.6402809, "count": 379, "saldo": 91 }, "geometry": { "type": "Point", "coordinates": [4.6667145, 50.6402809] } }
}

With the help of the documentation in GitHub we changed the code to our needs. After the creation of the map element and the input of the link to the csv files we use one of the functions from the plugin Canvas-Flowmap-Layer called createCanvasFlowmapLayer. In Code 3 our code snippet for this function is displayed.

```
function createCanvasFlowmapLayer(csvFilePath, customLayerId, doOneTimeDemoSetup) {
   Papa.parse(csvFilePath,  {
   download: true,
   header: true,
   dynamicTyping: true,
   skipEmptyLines: true,
   complete: function(results) {
     var geoJsonFeatureCollection = {
       type: 'FeatureCollection',
       features: results.data.map(function(datum) {
         return {
           type: 'Feature',
           geometry: {
             type: 'Point',
             coordinates: [datum.origin_lon, datum.origin_lat]
           },
           properties: datum
         }
       })
     };
```

Code 3 - Parser for csv to geojson

This function needs a parser to change the csv-file into the correct geojson. The used parser is Papa Parse, an in-browser csv parser (Holt, M., 2019.). It uses a XMLHttpRequest to download remote files. This is the reason why we need to save our data in GitHub and use the link to the raw data. The XMLHttpRequest expects a URL, a link beginning with http or https.
Then the required properties for this plugin are determined. For the origin points and the destination points the attributes are specified (Code 4).

```
var demoLayer = L.canvasFlowmapLayer(geoJsonFeatureCollection, {
  originAndDestinationFieldIds: {
    originUniqueIdField: 'origin_id',
    originGeometry: {
      x: 'origin_lon',
      y: 'origin_lat'
    },
    destinationUniqueIdField: 'destination_id',
    destinationGeometry: {
      x: 'destination_lon',
      y: 'destination_lat'
    }
  },
```

Code 4 - Specification of the point properties

Afterwards we included class breaks for the amount of immigration or emigration for each flow. Code 5 shows an excerpt of our classes in the canvasBezierStyle. The class breaks are 0-25, 25-100, 100-500, 500-5000, 5000-10000, and 10000-25000. These breaks were determined by the distribution of the data.

```
canvasBezierStyle: {
  type: 'classBreaks',
  field: 'count',
  classBreakInfos: [{
    classMinValue: 0,
    classMaxValue: 25,
    symbol: {
      strokeStyle: '#5fded1',
      lineWidth: 0.3,
      lineCap: 'round',
      shadowColor: '#5fded1',
      shadowBlur: 1.5
    }
  }, {
    classMinValue: 26,
    classMaxValue: 100,
    symbol: {
      strokeStyle: '#50bfb3',
      lineWidth: 1.5,
      lineCap: 'round',
      shadowColor: '#50bfb3',
      shadowBlur: 1.5
    }
```

Code 5 - Class breaks example

For the user interface, we took the already existing design and modified it to our desire. The buttons got new names and different events, the colors of the whole web map got changed, and the missing parts got inserted.
The buttons change the visualisation between emigration and immigration. Other user input is the click or mouseover function with which one can choose a specific point to show the flow to or from Austria. The click or mouseover can be changed to new, add or subtract the point from the selection. The last buttons either show all or none of the flows.
For the design of the web map the base map was changed. It is now the grey design from ESRI. The colors of the flows were changed to an unassuming color and width, the points are now big enough to click easily, and the legend was included.

Our finished wep map is displayed with screenshots on the next page. Figure 8 showcases the web map for all emigration from Austria. The legend is in the left down corner and the box for user interaction is in the right down corner. The yellow point is the origin point and the smaller blue points the destination points.

Figure 9 displays the immigration to Austria. The yellow color is again indicating the origin points and the blue color the destination point.

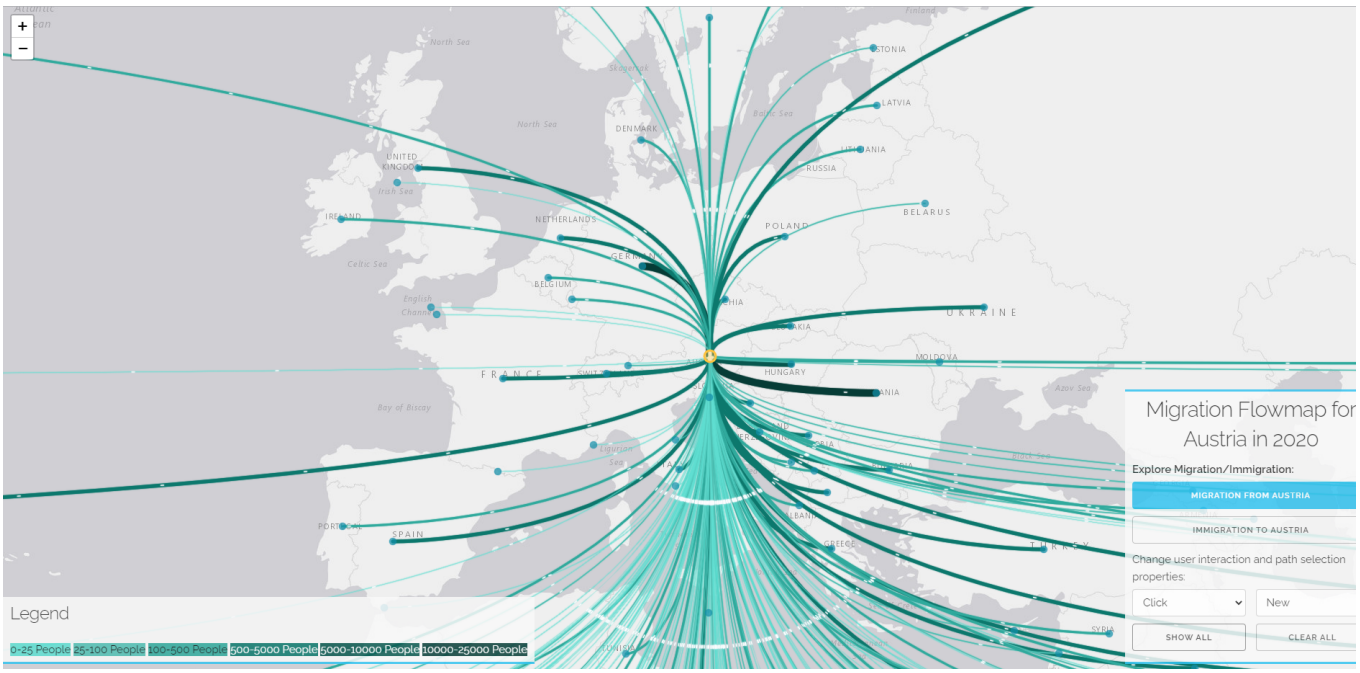

Figure 8 - Emigration from Austria

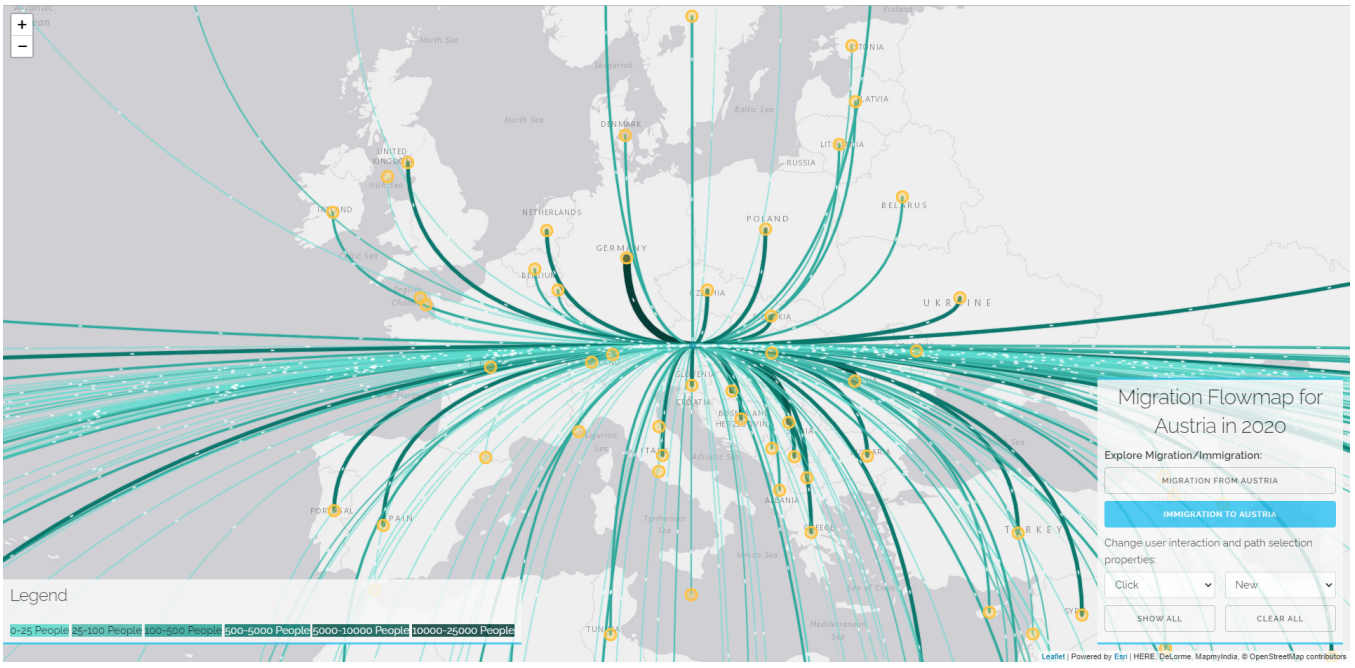

Figure 9 - Immigration to Austria

Conclusion

All in all the plugin Canvas-Flowmap-Layer is a satisfying inclusion to a web map. It fit our need perfectly to show the flow of migration from Austria. The inclusion of csv data into the web map was after a short research easily done and the changes to the existing code can be effortlessly included. The documentation in GitHub was valuable to understand the plugin and the examples were a delight to figure out the code.

Encountered Problems

Throughout the project we encountered a few problems. It was quite the effort to get the suitable data, as most data sets document the nationality of the migrants and not the origin/start point of migration. Geocoding was hard work, as many plugins in QGIS only work with one single location and not a whole csv file. And ArcGIS did not produce satisfying results and is not Open-Source. The parser we use to change the csv to the right geojson format needs the data from a file that starts with http or https – therefore we had to upload our data to Github and publish our project accordingly.

Possible Improvements

Possible Improvements could be to use more detailed data and add more interface items. The data for Austria could be extended for more years to visualize the shift in time. It could also be possible to integrate a larger data set for immigration and emigration of the whole world. The user interface can also have an extension with a search function or popups on the flows.

References

GitHub. 2022. *GitHub - MadeByMel/geovisualisation*. [online] Available at: <https://github.com/MadeByMel/geovisualisation> [Accessed 1 February 2022].

Holt, M., 2019. *Papa Parse - Powerful CSV Parser for JavaScript*. [online] Papaparse.com. Available at: <https://www.papaparse.com/> [Accessed 4 February 2022].

Leafletjs.com. 2022. *Leaflet — an open-source JavaScript library for interactive maps*. [online] Available at: <https://leafletjs.com/> [Accessed 1 February 2022].

Pgadmin.org. 2022. *pgAdmin - PostgreSQL Tools*. [online] Available at: <https://www.pgadmin.org/> [Accessed 1 February 2022].

Qgis.org. 2022. *QGIS Project*. [online] Available at: <https://qgis.org/de/site/> [Accessed 1 February 2022].

STATISTIK AUSTRIA, 2021. *Statistik Austria*. [online] Statistik.at. Available at: <http://www.statistik.at/web_de/statistiken/index.html> [Accessed 1 February 2022].

Wasilkowski, J., 2017. *GitHub - jwasilgeo/Leaflet.Canvas-Flowmap-Layer: A LeafletJS custom map layer for mapping the flow of objects, ideas, people, etc. with Bezier curves rendered on the HTML canvas.*. [online] GitHub. Available at: <https://github.com/jwasilgeo/Leaflet.Canvas-Flowmap-Layer> [Accessed 1 February 2022].