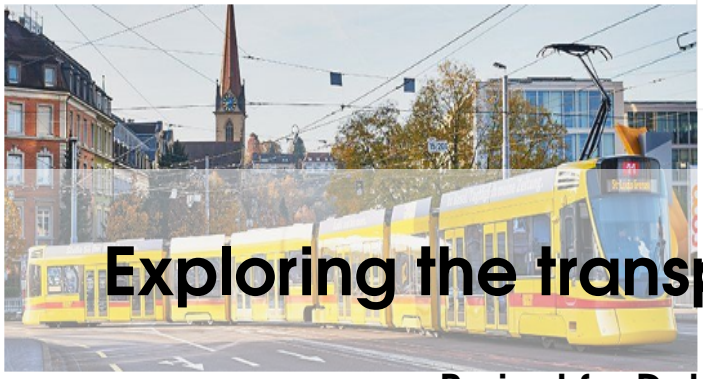



Exploring the transportation of Switzerland

Project for Data Visualisation course



Students:

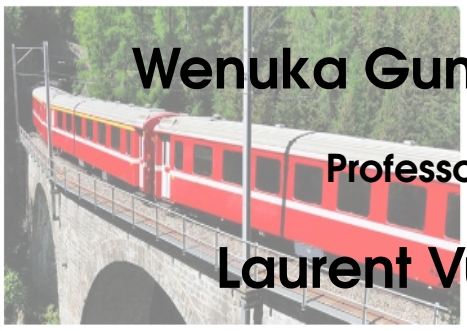
Mariam Hakobyan

Tianchi Yu

Wenuka Gunarathna

Professor:

Laurent Vuillon




PROJECT FOR COM-480 DATA VISUALIZATION COURSE, EPFL

Github - `https://github.com/com-480-data-visualization/com-480-project-story-tellers`
Web page - `https://threestorytellers.github.io/`

Spring 2020, COVID-19 time

Contents

1. Introduction

1.1 Overview

The public transportation system of Switzerland is playing a big role in mobility system. People use the transportation system for different purposes: work commute, sport, travel, outdoor or mountain activities, so on. This is where the **motivation** of the project comes from: "Visualize the time and space dynamic data of transportation system of Switzerland and find interesting insights".

The **inspiration** of the map comes from the ocean-ship map visualization by *Klin digital* team. We have found several life tracking applications for the transportation system, but more specifically the top view of the running different modes of vehicles in delta time has not yet implemented for Switzerland to our knowledge. To that end, we have visualized a Dynamic flow Map of moving vehicles in time and graphical interactive figures to get interesting insight about the system. The **overview** of the visualization is a process of interactive map in different map modes and graphical figures to answer several research questions. The **target audience** of the project is planned to include people who are interested in transportation system, network visualization, Switzerland and who are just curious about things in world. Will require some knowledge for the user to interact with the map functionalities and to understand the statistical figures.

1.2 Data

We have used **Open Data Platform of Swiss Public Transport** for all data we have processed in the visualisation.

- 2020 Timetable data [1] - data for initial analysis and source for the d3 js visualizations,
- GTFS format data [2] - public transportation feed used un the Interactive Map visualization,
- GEOJson of Switzerland map [4] -used for Swiss map layer
- Station information [3] - Overall spatial information of big train stations.

While processing the timetable data we have obtained 473233 schedules in a day out of which 78% share belongs to **bus** schedules: figure 1.1.

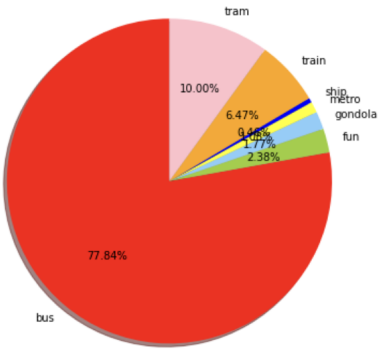


Figure 1.1: Transportation type distribution in schedules

The high percentage of bus schedules in day schedule is because of bus frequencies and distribution in almost all regions of the country. Because of memory limitation in the interactive map data hosting, we have excluded the data for bus vehicles and stations from the map visualizations.

1.3 Data for Interactive Map

Since the initial GTFS text files don't have *shapes.txt* files, we have used the *Precise OpenStreetMap (OSM)* map matching to public transport schedules API to get the exact trajectory of the route from point to point depending on what the mode of the a trip [6]. New *shapes.txt* file has been generated during conversion to characterize the real trajectory of a trip.

Then, we used the GTFS visualisation from GTFS text file API [11] to convert the text data to geojson files containing map trip related data to Sqlite format.

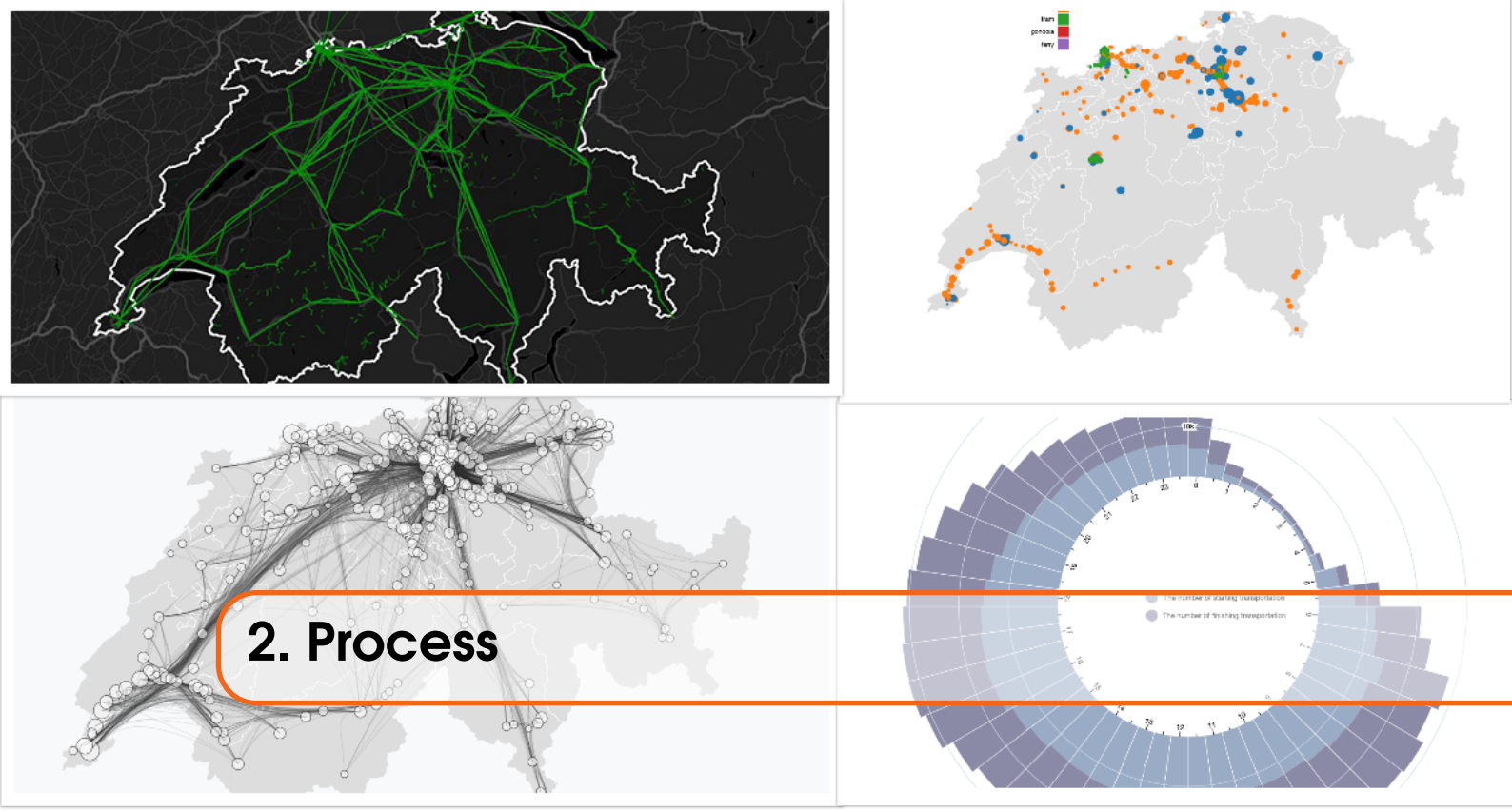
However, the Sqlite database was around *2GB*, which was impossible to load along with the web site, therefore we had to move that entire database to Mysql and hosted it on a third party hosting provider. The gtfs_shapes.geojson contained shapes for all trips, and it was over passing the size allowed on server, *github.io*. We had to rethink about key aspects of the visualization. When analyzing the data, around 97% data was due to busses and trains. Therefore due to server limitations we were forced to add less detailed shapes for trains (straight lines in between stations) and also to exclude all the busses. Further compression was done using a python script finally achieving a geojson file of 6.7MB for shapes which was a 99% reduction in size when compared to original file.

1.4 Data preparation for d3.js graphs

From the timetable data, we have extracted the number of Initiating and Terminating transports in different time frequency: {hourly, half hourly, quarter hourly}. This data later on reflects in the Radial Bar chart in "hour" frequency.

Using the stations information data we could extract the top 300 train stations in Switzerland in terms of number of boarding passengers. Then from timetable data we have build a graph for those 300 stations where we the nodes are connected if there is direct train between the node and the weights correspond to the number of trains in a day between two nodes. This information is later used in Edge bundling graph on the map.

We have also extracted the number of entering transportation in any type of stations(except bus stations, because of aforementioned reason) in time, which later on reflects in the Time interactive map graph. We have used python to extract data for d3 js graphs, and saved into the */github_web/data* folder.



2. Process

2.1 Interactive Map

The one of the main milestones of the project was to create an interactive flow-map for the transportation system, which depicts the flow of the vehicles in time and space.

We have started the interactive map by referring to the Transit map project [5] which uses Google API to construct the map. On top of the map we have incorporated layers with Geojson and Json data files[12], to draw the the points as moving vehicles in delta time, points as stations with coordinates and lines as traffic roads. We have also implemented functionalities for the map to obtain information about stations: Geo-positions and operating transport types, traffic routes defined by the trajectory of the trips. This functionalities are available to user in the form of the Figure 2.1 UI.

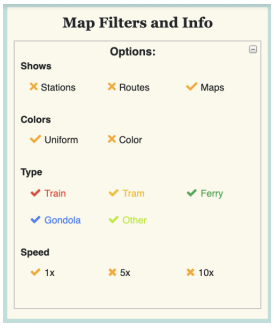


Figure 2.1: Map filters and Info

The most challenging task of the interactive map was to show the most important data within the server limitations. We had to spent a significant time creating an API and also to reduce the existing data files. Locally we could host the data files and see significant amount of points on the map, but we didn't have chance to apply the full data into the server.

Figure 2.2: The Map on the local server

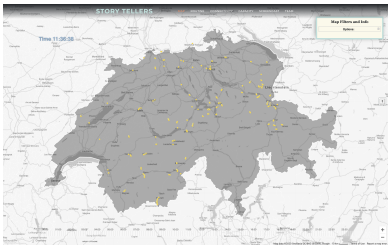


Figure 2.3: The Map on the Github server

Filters and Functionalities

Several filters and functionalities were implemented on the map for the user experience and information exchange. In the Map filters and Info Window section, filters are added to show/hide stations, routes and the map. We add a color scheme to each type of stations along with a filtering process for each type of vehicles. The speed set button was designed to animate vehicles at different speeds while the time bar was used to navigate the user to any preferred time in the current date. Detailed

Vehicles: To realise the flow of each vehicle, we need to know large enough number of coordinates on its trajectory. So we constructed the line-Pools, which consists of the trajectories of vehicles where we picked the coordinates as the positions of vehicles. Moreover, we used the schedule of vehicles to build the connection between the real time and the movements. We used icons as nodes for vehicles so that the movement of vehicles to be visually clear.

We can get information about the vehicle by hovering on it and if we click on the point we will see the trip plan on the Map Filters and info part.

Users can also track the vehicle by zooming in a specific vehicle and recenter into the initial map after observing the route.

Finally, we add an option of showing/hiding the single route of a specific vehicle clicked by the user. By this, the trajectory of that vehicle will be properly shown.

Stations: Besides vehicles it would be informative for the user to see the stations and their locations. Initially the stations layer is hidden for avoiding of confusion, but the user can opt to see the stations by activating the stations in the Options section in the window.

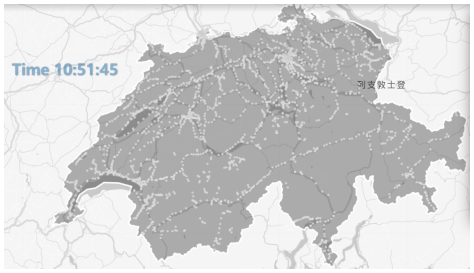


Figure 2.4: Uniformly colored Stations

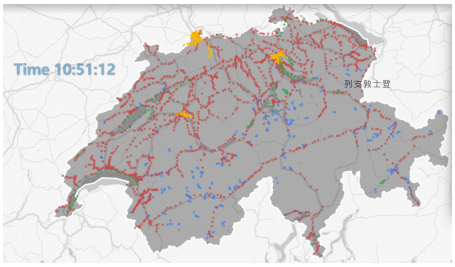


Figure 2.5: Multi coloured Stations

What do we want to show with this function, is the distribution of the stations in Switzerland and the traffic intensity. But without classification of transportation type, we can not get significant information from it. So, we have involved color differentiation into the stations characterizing the operating transportation type(bus station are excluded as mentioned before).To make the map more visual and distinct, we chose the color palette of Google brands 2.5. User can also choose what type of stations to see and what to hide in the Station type part of the Options section.

Routes: By obtaining the nodes, the “edges” are ready to come out. We want to take the vehicles as the “flowing blood” and roots as "blood vessels" for the vehicles, which will depict the whole picture of the entire Swiss transportation network in a vivid way.

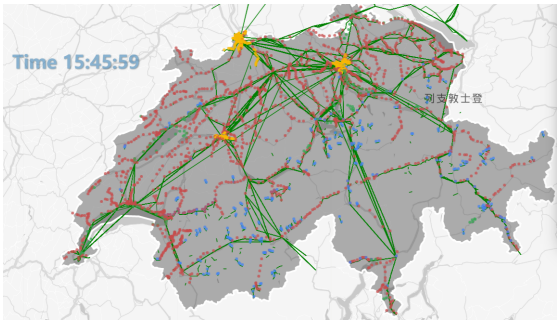


Figure 2.6: traffic roads

Time: Time bar and time is another important function for the user to see information in a free way. Users don’t have to wait several minutes or hours to see the flow of the transportation on their desired time. That is why we have involved the interactive time bar for every 10 minutes on the bottom of the map and time information on the map. Usually, the duration of trips are quite long, so we have decided to speedup the velocity of the flow-vehicles by showing the time-lapse of the flow. Initially, the time starts from the user-current hour and can be changed if the user opts to see the speed-up version of the vehicles’ flow. Thus, we have created three level speed-up function: $1x$, $5x$ and $10x$(i.e. $1s_{real} = 10s_{flow}$).

2.2 Radial bar chart

Besides the Interactive map we were curious to answer several research questions on the transportation data. Fist, we were interested to visualize the **"Day routine of transportation schedule"**. We were keen to see the night and day shifts and the pick hours of the schedules. Initially, we planned the graph to be Bar chart in flat axis, but the fact that we were dealing with time periodic data where $23:55$ is very close to $00:00$ we came up with the *Radial stacked bar chart* representation of the number of initiating and terminating transports in each **half hour**. Figure 2.7.

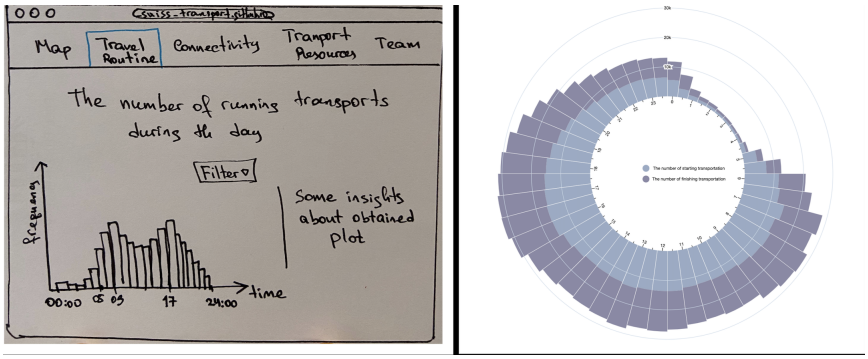


Figure 2.7: Routine graph: Sketch and Realization

We treat the **time** as discrete categorical variable to plot the Radial Stacked Bar Chart. The reason of visualizing also the number of terminating transportation is to see the transformation from pick to non-pick time and visa versa. The user can hover around the graph and get the number of transportation for each class and hour.

2.3 Map bundling

We were interested to see visualization that depicts: **"How connected are the big train stations in different parts of the country?"**. The term connected here means direct trip from a stop to stop. Initially, we had planned to represent this information as a force graph where nodes would be the stations, colored according to canton group and the edges will be simulated by physical forces characterizing the frequency of trains for connected nodes.The implementation of the graph came out not very visually appealing because we had large number of nodes and edges which created visual clutter. We had to choose the number of nodes and threshold the edge power in a way that the network would be connected, so that the user would have the best experience with this graph. But in that case we may appear to be biased in data selection.

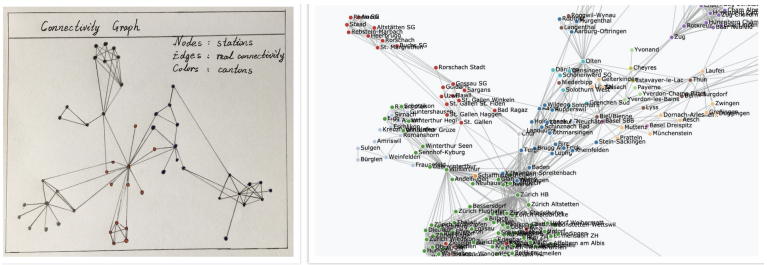


Figure 2.8: Force graph: Sketch and Realization

In the process of brainstorming solutions to the challenge of visualizing the connectivity graph, we realized the fact that stations had GEO spatial information {latitude, longitude } which can not be forced in the force graph as the locations of nodes are defined by physical force, but can be used to position the nodes properly as in the map. The edge bundling was considered as a solution, and the example of Flight Paths Edge Bundling of US airports appeared to be starting point for map, data and interaction [8]. We use Swiss boundaries topojson, stations and trips data processed and located in $/github_web/data$ folder for defining the graph and edge weights. The advantage of edge bundling visualization on the map is that the stations are fixed in the projected Geo locations and there is no need for hierarchy. The source code uses d3 geo Voronoi-arc to draw the diagrams and edges. The result of visualization is the following Figure 2.9.

Figure 2.9: Edge bundling on Map

The user can over around the stations and see the most powerful connections of the nodes/stations highlighted with red edges. As well as the tool tip will give the station's name and canton information of hovered node.

2.4 Capacity

We were curious about the amount of transport resources by type in each canton initially. So the stacked bar chart was the initial plan for the visualization where the groups would be the cantons and the classes the transportation type. While visualizing this bar chart we have noticed that the classes are unevenly distributed(buses and trains are dominating) and the graph wasn't visually appealing. Figure2.10

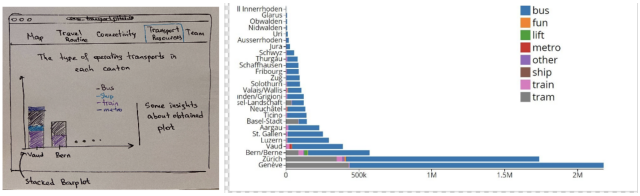


Figure 2.10: Initial Sketch of stacked bar chart and Realization

The solution to this challenge again came in the form of transforming the visualization into space/map and time domain. The key information to the user is the *coloring pallet* of different types of the stations and the*circle radius* characterizing the amount of transportation in specific time period. To give more flexibility to the user we have added the Timeline bar where the user can choose the specific time period and length for seeing the amount of incoming vehicles in each station. We can hover around the stations and get name and count information with the help of tool tip.Figure 2.11

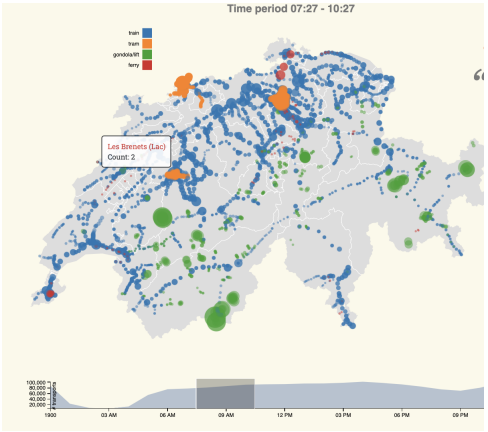


Figure 2.11: Initial Sketch of stacked bar chart and Realization

3. Terminus

3.1 Project success evaluation

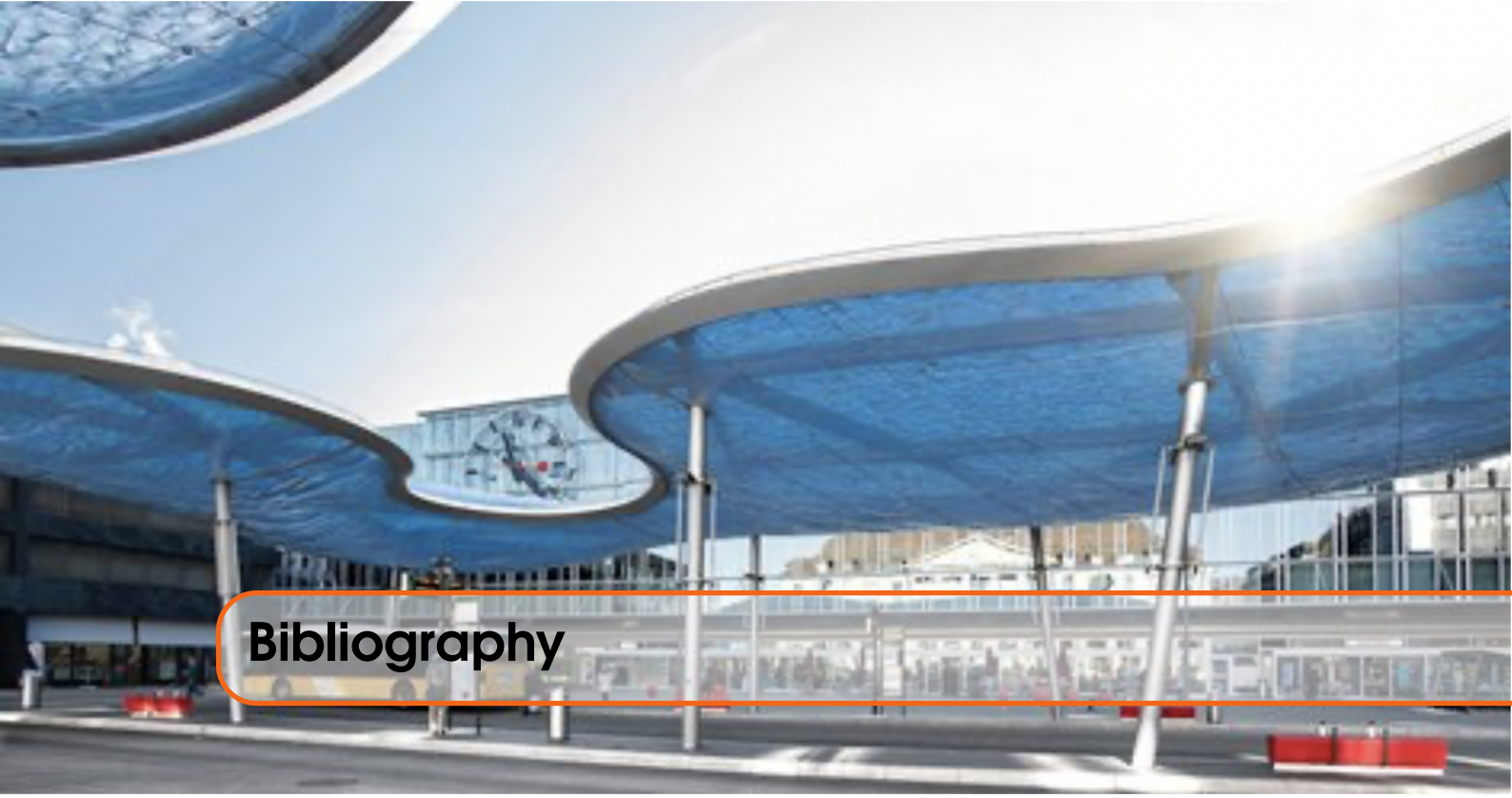
The have implemented almost all the steps that we have defined in the the beginning of the project for visual data story of transportation system. We have put a lot emphasizes on having informative and user friendly visualizations. The only challenge that we haven't thought about in the beginning was the size of the data and lack of web hosting resources in terms of memory. We have built the visualization framework which was tested to be working locally wit all the data, and online as well with small amount of data, but we couldn't host all the data we have planned to. Thus, we had to filter out several routes and trips from the database to reduce the load of the memory. We have evaluated that we don't lose a lot information by removing the bus routes because the bus schedules are quite uniform in time and space(across country) so there was no diversity to depict with bus transportation modes. After filtering the data we again end up having files which exceed the max limitation of github. Thus, we put very small sample of the data in the working web and to see all the vehicles movement, we would ask to run the web locally and download the files from this link to *githubio_web.api.gtfs_data* file.

3.2 Task distribution and Peer assessment

We quite managed to implement initially defined things without even one face to face meeting because of COVID19 confinement. The tasks and team members working on them are the following:

- Data processing - Initially Mariam and Tianchi processed data and found insights, Wenuka worked on google map API. GTFS data was pre-processed by Mariam, while Wenuka converted them to Geojson and SQL format and later compressed.
- Data hosting - Wenuka worked to put the large datasets into a remote server and connect to main web on github.io.
- Interactive Map visualization - Ideas have been discussed all together
  - Backend - Api to read time table was mostly done by Wenuka
  - Frontend - UI parts and interactive web design was mostly done by Tianchi also with some help from Mariam and Wenuka

- The d3.js graphs - Ideas have been discussed all together. Routine, connectivity and capacity graphs are done by Mariam, while Tianchi and Wenuka were ready to help whenever needed.
- Overall design - Mostly Wenuka, Tianchi and Mariam have also given a hand,
- Process book - all together,
- Screencast - Mariam



Bibliography

REF

[1] Timetable data - https://opentransportdata.swiss/en/dataset/timetable-2020-hrdf

[2] GTFS data - https://opentransportdata.swiss/en/dataset/timetable-2020-gtfs/resource/559278ea-fd2a-4d93-a462-f9c6d7b69dae

[3] Passengers boarding and alighting for train stations- https://opentransportdata.swiss/en/dataset/einundaus

[4] GEO json data for Switzerland bounderies - https://github.com/interactivethings/swiss-maps

[5] Transit map Api - https://github.com/vasile/transit-map

[6] Pfaedle - https://github.com/ad-freiburg/pfaedle

[7] Radial Stacked Bar Chart - https://bl.ocks.org/KoGor/9f3932d3c7154f8ab3ea2078b2aca113

[8] Flight Paths Edge Bundling - https://bl.ocks.org/sjengle/2e58e83685f6d854aa40c7bc546aeb24

[9] Interactive map in d3 js - https://bl.ocks.org/domhorvath/dd850c5e97d4022fbf0f11611a0cf528

[10] Google GTFS reference - https://developers.google.com/transit/gtfs/reference

[11] GTFS vizualisation from GTFS text files - https://github.com/vasile/GTFS-viz

[12] Google Layer building - https://developers.google.com/maps/documentation/javascript/layers