EPFL

Data visualization
COM-480

Les vizionnaires
Process Book

by Matthias Wyss (SCIPER 329884)
Ursula El-Khoury (SCIPER 340994)
Sarah Badr (SCIPER 341366)

Contents

1 Overview of the Project

This project explores the spatial accessibility within the Swiss public transport network. More specifically, we study how easily different areas of Switzerland can reach major InterCity (IC) railway hubs, and how these accessibility patterns relate to socio-economic indicators such as wealth, employment, and population density.

Our main question is:

How accessible are major Swiss InterCity hubs by public transport across the country, and do differences in accessibility correlate with socio-economic patterns?

The scope of the project is limited to public transport accessibility. We focus on scheduled transport connections represented in GTFS data, with particular attention to the Swiss rail network and IC stations as central mobility hubs. Local and regional public transport connections are used as part of the accessibility calculation, but the main destination points remain the major IC hubs.
The final result is an interactive web dashboard composed of two complementary views:

- **The Spatial Dimension (Map View)**: An interactive Leaflet map that allows users to explore accessibility geographically, including travel-time regions, nearest IC stations, point-to-point routing, and socio-economic overlays.
- **The Statistical Dimension (Analysis View)**: A D3.js-based dashboard that uses statistical visualizations to investigate possible relationships between accessibility and socio-economic variables.

This report describes the full path from the initial concept to the final implementation. We first introduce all the datasets we have used for the project. We then explain how the interface evolved from the sketches and ideas developed during Milestones 1 and 2 and we analyze the data processing done for both the map and analysis views. After that, we describe the backend implementation, including data preprocessing, transformations, and the evolution of our routing approach. Finally, we conclude with a reflection on the project and a peer assessment describing each team member's contribution.

2 Data

In this section, we describe the various data sources integrated into this project to construct the routing network, define administrative boundaries, and provide the socio-economic context for our analysis. To ensure data reliability, the datasets were primarily sourced from official Swiss federal administration portals, alongside a community-driven repository.

Transport and Mobility Data

- **Transport Connections (GTFS):** To model the Swiss public transport network and accurately compute travel times to major InterCity (IC) hubs, we utilized General Transit Feed Specification (GTFS) timetables.

Geographic and Administrative Boundaries

- **swissBOUNDARIES3D:** To accurately render municipal, cantonal, and national borders within our spatial map, we used the official 3D topological landscape model provided by Swisstopo.
- **Official Commune Registry:** We utilized the FSO's official Swiss commune registry to consistently map, aggregate, and cross-reference municipal data with their respective cantons across all thematic layers.

Socio-Economic Indicators

- **Public Transport Usage:** Municipal-level data detailing modal splits and public transport adoption rates among commuters was sourced from the Swiss Federal Statistical Office (FSO). It is important to note that this dataset only covers the 180 largest Swiss cities, resulting in missing data for smaller municipalities. However, as no broader country-wide dataset of this nature currently exists, this remains the most reliable source available.
- **Real Estate Prices:** To evaluate the potential relationship between IC hub accessibility and housing costs, we incorporated the "Switzerland House Price Prediction" dataset sourced from Kaggle. This dataset only covers 356 municipalities. As there is currently no comprehensive official public dataset for real estate prices covering all Swiss municipalities, this was the best available option, though we acknowledge its limited geographic representativeness.
- **Population Density (STATPOP):** Demographic baselines, including total population counts and density per municipality, were extracted from the FSO's STATPOP population surveys.
- **Employment Distribution:** Data detailing the volume, concentration, and geographic distribution of jobs across municipalities was retrieved from the FSO's structural business statistics.
- **Wealth Proxy (Taxable Income):** We used municipal-level direct federal tax statistics provided by the Federal Tax Administration (ESTV) to extract the average taxable income, which serves as a reliable proxy for regional wealth in our analysis.

3 UI Evolution since Milestone 1 and 2

At Milestone 1, the project existed only as a concept and a dataset inventory. No visualization had been implemented, and the routing pipeline was not yet operational. The scope was broad and included several socioeconomic layers (real estate, employment, salary, adoption rate).

3.1 Key improvements from milestone 2:

3.1.1 Nearest IC Time filters

The initial prototype computed a single travel time matrix for one departure time (Friday 08:00) on a 3km grid. By Milestone 3, we extended the pipeline to cover four departure slots (08:00, 12:00, 17:00, 22:00) on both a weekday (Friday) and a weekend day (Saturday), giving users a more realistic picture of how accessibility varies across the day and between weekday and weekend schedules. The UI also changed: dropdown menus were replaced by filter chips for faster layer switching. Color harmony was refined throughout the interface: since the reachable IC hubs are already rendered in red, the selected origin marker needed a visually distinct treatment. We chose orange with a larger radius to ensure it stands out clearly from both the destination markers and the underlying accessibility heatmap.

(a) Old filters

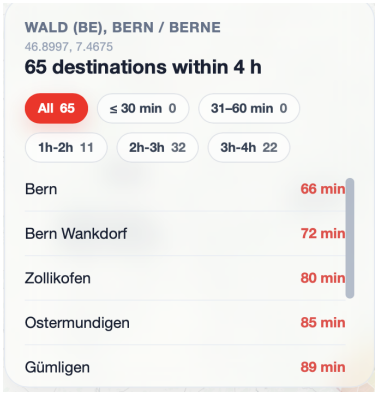
(b) New filters

Figure 1: Filters

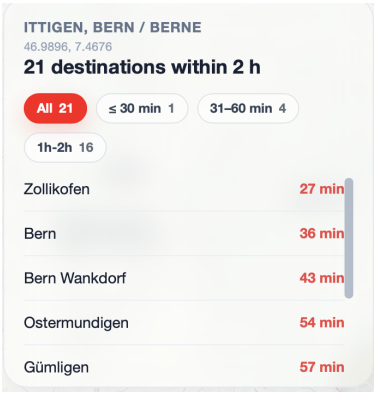
3.1.2 Reachable Destination Panel

What was initially a minimal tooltip was redesigned into a floating semi-transparent panel featuring filter chips for time thresholds and a scrollable ranked list of reachable IC hubs with travel times; moving from a raw data dump to a structured interactive detail view. The maximum travel time filter is fully dynamic: the list and chips update to show only IC hubs reachable within the selected threshold. Origin points, previously identified only by numeric IDs, now display the commune name, canton, and coordinates after spatially joining the grid against Swiss commune boundaries (GDENAME, KTNAME). This makes the map significantly more readable for non-expert users, and the same enrichment appears in hex cell popups throughout the interface

(a) Old version of the panel



(b) New panel with max travel time 4h

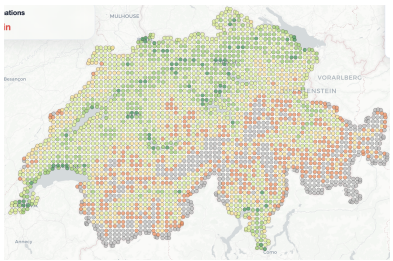


(c) New panel with max travel time 2h

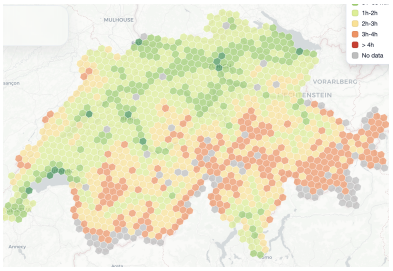
Figure 2: Panel versions

3.1.3 From Grid Points to Hexagonal Isochrones

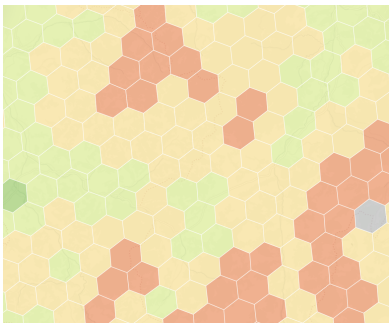
In Milestone 2, the map rendered raw data points on a fixed grid, which produced a pixelated and disconnected visual. In Milestone 3, we transitioned to hexagonal binning using the H3 spatial indexing library, with dynamic resolution based on zoom level, and introduced gap-filling interpolation for empty cells. The result is a fluid, continuous accessibility surface that better respects Switzerland's geographic constraints.



(a) Map with grid points



(b) Map with hexagonal binning



(c) Zoom on the hex

Figure 3: Map versions

.

3.1.4 Point-to-Point Feature

The original scope covered only aggregate accessibility metrics. We added an interactive A to B mode allowing users to select any two grid points and retrieve the actual travel time between them, bridging the gap between nationwide heatmaps and individual journey queries. When clicking on a zoomed-out

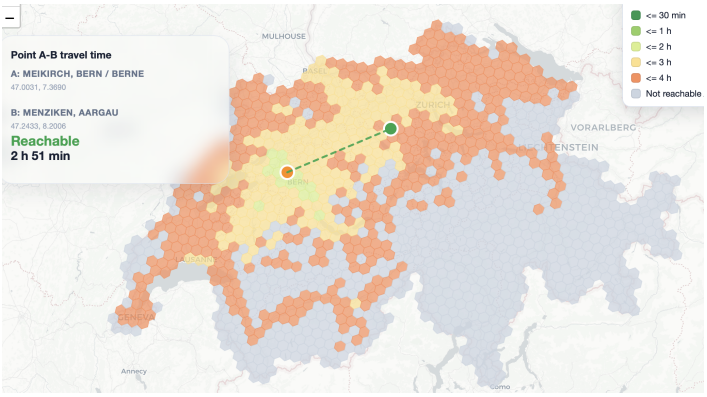


Figure 4: Point to point feature

hex cell, the panel displays the average travel time across all grid points inside it, along with the list of municipalities it covers: for example, a single hex may contain grid points across various communes. As in the Nearest IC mode, the selected point displays its commune name, canton, and coordinates for full spatial context.

3.1.5 Integration of social layers

Since Milestone 2, we integrated five thematic map layers alongside the transport accessibility view: public transportation usage, population density, employment volume, average taxable income, and real estate prices. Each layer lets users explore how demographics, wealth, and employment are distributed across Switzerland, revealing spatial patterns and disparities that shed light on regional inequalities and social structure.

3.1.6 Analysis view

The analysis view, completed after Milestone 2, unifies all thematic layers to directly examine how transport accessibility relates to regional socioeconomic characteristics. It is organised around four guiding questions, whether major employment centres enjoy privileged access to railway hubs, whether prosperity correlates with better transport access, whether stronger connections systematically drive up prices, and whether population density guarantees or prevents isolation, each investigated through dedicated charts and statistical comparisons. Together, these analyses lead to a central insight: Swiss rail infrastructure functions not just as a mobility system but as a structural backbone that shapes corporate clustering and regional wealth distribution, even as local housing markets continue to follow highly localized dynamics.

4 Backend Implementation

The backend data pipeline was one of the most challenging and important parts of the project. Since the website itself is static, all expensive transport computations had to be done in advance. Our backend therefore acts as a preprocessing pipeline: it downloads and cleans the transport data, computes travel-time matrices with `r5py`, transforms the results into lighter files, and exports them in formats that the frontend can load directly.

We used `r5py` as the core routing engine for the project. `r5py` is a Python interface to the R5 transport routing engine, which is designed to compute travel times over multimodal transport networks. In our case, it combines GTFS public transport schedules with OpenStreetMap street data, allowing us to estimate how long it takes to travel from one point to another using walking and scheduled public transport. This made it well suited for our project because we needed realistic public transport accessibility calculations at the scale of Switzerland.

4.1 From the First r5py Notebook to a Precomputed Pipeline

Our first implementation was built in a basic `r5py` notebook. The goal was to compute travel times from a grid of points covering Switzerland to all major InterCity stations. We used two main data sources:

- Swiss GTFS timetable data, which describes scheduled public transport services.
- OpenStreetMap data, used by `r5py` to connect stops to the walking and street network.

The routing was configured with public transport and walking modes:

```
transport_modes=[r5py.TransportMode.WALK, r5py.TransportMode.TRANSIT]
```

We also used a one-hour departure time window and the median travel time percentile:

```
departure_time_window=timedelta(minutes=60)
percentiles=[50]
```

This was a design decision to avoid making the visualization depend on a single exact departure minute. Instead, each result represents a more stable estimate of accessibility around a given time of day.

The first major technical challenge came from the raw GTFS feed. Some route types, especially taxi like services with `route_type = 1500`, were not handled correctly by the routing engine and caused problems during network construction. To solve this, we filtered the GTFS tables before building the `r5py` network. We removed unsupported routes, then removed the trips, stop times, stops and transfers that depended on those routes. This produced a cleaner file that could be reused by all later notebooks.

Another early issue was the definition of the IC destinations. At first, destinations were extracted at platform level. This created routing failures because platforms are often represented as child stops and are not always well connected to the pedestrian network. We therefore aggregated IC stops to the station level using the `parent_station` field. This made the destination set more stable and improved routing success because users conceptually travel to a station, not to an individual platform.

4.2 Spatial Grid and Granularity Decisions

The first prototype used a coarse grid over Switzerland. This made the computation easier, but the visual result was too sparse and did not show local accessibility differences clearly enough. We then moved to a finer spatial grid built in the Swiss projected coordinate system `EPSG:2025`, because it uses meters and therefore allows us to create regular cells such as 2 km or 3 km squares. The centroids of these cells were then converted to `ESPG:4326` for routing and web display.

For the nearest-IC map, we improved the granularity to a 2 km grid. This gave much more detailed spatial coverage while remaining computationally possible because the destinations were limited to IC stations. To avoid wasting computation on irrelevant areas, we added a spatial filter: only grid points within 5 km of a transit stop were kept. This removed many cells in lakes, mountains, and uninhabited areas while keeping public transport accessibility is meaningful.

However, this finer grid also increased computation time significantly. This led to an important design decision: we used different grid resolutions depending on the task. The nearest-IC computation could use the 2 km grid, but the point-to-point feature required an all-to-all matrix between grid cells. Since all-to-all routing grows quadratically, the point-to-point pipeline uses a 3 km grid as a compromise between precision and feasability.

4.3 Optimizing the Travel-Time Computations

Initially, we considered computing one `r5py` matrix for every computation of departure time and maximum travel-time threshold. For example, this would mean recomputing the network for 09:00 with 30 minutes, 09:00 with 60 minutes, and so on. This was too expensive.

The key optimization was to compute one full matrix per departure time with a maximum travel time of 240 minutes. Then, all shorter thresholds were derived by filtering the resulting matrix in `pandas`.

For each departure time, we kept the full 240-minute result as a Parquet file, then generated the 30, 60, 90, 120, 150, 180, 210, and 240 minute versions from it. This kept the frontend file structure simple while avoiding repeated routing computations.

The backend exports three main types of files:

- `travel_times_...parquet`: full or filtered travel-time matrices.
- `od_pairs_...parquet`: valid origin-destination pairs within a given threshold.
- `accessibility_summary_...parquet`: one row per origin, including the minimum travel time to an IC hub and the number of reachable destinations.

4.4 Nearest IC Pipeline

The nearest-IC pipeline computes accessibility from each grid point to the set of IC stations. For each day type, departure time, and maximum travel time, we generate two frontend files:

- `accessibility.geojson`, containing one feature per grid cell with properties such as municipality, canton, minimum travel time to an IC station, and number of reachable IC destinations.
- `reachable.json`, containing the list of IC stations reachable from each origin point.

These files are organized as `docs/data/nearestIC/weekday,weekend/hour/maxTime/`
The weekday data uses services from Friday, March 20, while the weekend data uses services from Saturday, March 21.

This folder structure was a frontend-oriented decision. Instead of loading one vey large dataset, the map can load only the files that correspond to the user's current filters, for example weekday, 17:00, 120 minutes. This makes the interface more responsive and avoids unnecessary browser memory usage.

4.5 Point A to Point B Pipeline

The point-to-point feature required a different backend strategy. Instead of routing from every grid point to IC stations, we needed to route from every grid point to every other grid point. This is much heavier: with 4626 points in the 3 km grid, one departure time represents more than 21 million possible origin-destination pairs before `r5py` pruning.

To make this feasible, we split the origins into chunks of 250 points. Each chunk is computed separately and saved as a Parquet file. We also generated two index files:

- `manifest.parquet`, which lists every chunk file and its metadata.
- `origin_chunk_index.parquet`, which tells us which chunk contains the data for a given origin.

For the website, these Parquets files are converted into JSON chunks. The frontend first loads a manifest, then only downloads the chunk corresponding to the selected point A. This prevents the browser from loading a complete all-to-all matrix and makes the interaction possible in a static web application.

4.6 Transforming Backend Outputs for the Website

Because the final dashboard runs in the browser, the raw backend outputs had to be transformed into web-readable formats. We used Parquet during computation because it is efficient for large tabular data, but converted selected outputs into GeoJSON and JSON for the frontend.

GeoJSON is used for spatial layers because Leaflet can render it directly. JSON is used for lookup tables, such as reachable IC stations or point-to-point travel times. We also added manifest files so that the frontend can dynamically locate the correct dataset based on the selected mode, departure time, day type, and travel-time threshold.

This transformation step was essential since it separates heavy computation from interaction. All routing is done offline in Python, while the website only performs lighweight loading, filtering and rendering.

5 Conclusion

This project allowed us to explore Swiss public transport accessibility from both a spacial and analytical perspective. Starting from a relatively simple idea of mapping travel times to major InterCity hubs, the project gradually evolved into a more complete interactive dashboard combining routing computation, map-based exploration, and socio-economic analysis.

Overall, the project shows that accessibility is not only a geographic question, but also a socio-economic one. The visualizations reveal strong spatial differences between well-connected urban areas, regional centers, and more isolated municipalities. They also show that accessibility does not always correlate in a simple way with wealth, density, or employment, which made the interactive approach especially useful.

5.1 Limitations and improvements

Despite these results, the project has several limitations. First, the routing data is precomputed for specific dates, departure times, and maximum travel-time thresholds. This means that the dashboard does not represent live conditions, delays, disruptions, seasonal timetable changes, or user-defined departure times.

Second, the spatial representation is based on grid points rather than exact addresses. The nearest-IC pipeline uses a 2 km grid, while the point-to-point feature uses a 3 km grid for performance reasons. This improves computation feasibility, but it also means that travel times are approximations based on grid cell centroids rather than precise door-to-door routes.

Another limitation concerns the socio-economic analysis. The datasets used do not always have the same spatial granularity, year, or collection method. Some values had to be joined at the municipality level, which can hide variation inside larger municipalities.

Finally, some technical compromises were necessary to make the project usable in a browser. Large routing outputs had to be simplified, chunked, and converted into frontend-friendly formats. These decisions improved performance, but they also limited the level of detail that could be shown interactively.

Future improvements could include live timetable updates, user-selected departure dates, finer grid resolutions, additional transport modes, and deeper statistical modeling. Even with these limitations, the final dashboard provides a meaningful way to explore how public transport accessibility is distributed across Switzerland and how it intersects with socio-economic geography.

6 Peer assessment

While the project was a highly collaborative effort, the following breakdown outlines the main parts of the project completed by each team member.

Sarah Badr:

- Designed the UI skeleton for both the map and analysis views, including the Leaflet map, side panel, initial analysis layouts, and the interaction logic connecting the different components.
- Continuously integrated and refined the UI: Nearest IC travel-time mode with better filters, improved reachable data panel and pop-ups with more user friendly information (color, names, text size) and overall color harmony.
- Implemented the hexagonal isochrone workflow from first implementation to zoom-level refinement using the H3 library.
- Include commune and canton names in the UI by updating the original parquet data, propagating those changes through the processing pipeline, and integrating the attributes into the frontend.
- Optimized travel-time computations by running r5py once per departure to compute a full travel-time matrix (capped at 240 minutes), then deriving all shorter thresholds via in-memory pandas filtering instead of rerunning r5py for each departure–time combination.
- UI integration: built a script `scripts/build_nearestic_2000m.py` to convert parquet points into GeoJSON points for the Nearest IC mode, and integrated the resulting GeoJSON layers into the UI to support combinations of travel times, departure times, and day types.
- Contributed to writing and structuring the final report.

Ursula El-Khoury:

- Engineered the accessibility computation pipeline using `r5py`, Swiss GTFS data, and OpenStreetMap data.
- Filtered the GTFS feed by removing unsupported transport modes, specifically taxi services.
- Improved routing reliability by aggregating IC destinations from platform-level stops to station-level locations.
- Designed and optimized the spatial grid used for routing, including the 2 km grid for nearest-IC accessibility and spatial filtering of origins within 5 km of public transport stops.
- Developed the nearest-IC backend outputs, including accessibility summaries, reachable-destination files, and frontend-ready GeoJSON/JSON datasets.
- Developed the Point A to Point B routing pipeline using a 3 km grid, chunked all-to-all travel-time matrices, and manifest files for efficient frontend loading.
- Implemented the Point A to Point B interface on the website.
- Contributed to writing and structuring the final report.
- Filmed and voiced the final project demonstration.

Matthias Wyss:

- Started the core Accessibility Computation Pipeline utilizing `r5py` and GTFS datasets.
- Researched, selected, and validated the socio-economic datasets used in the Map View and Analysis View, including public transport usage, population density, employment, income, and real estate indicators.

- Architected the `map_views_generation.ipynb` notebook to clean, compute, and export all socio-economic GeoJSON layers (Public Transport usage, Density, Employment, Income, Real Estate) for the Map view.
- Created the `correlation_analysis.ipynb` pipeline to process and generate the specific GeoJSON datasets required for the Analysis view.
- Refactored the front-end architecture to integrate the socio-economic layers into the Map View, and developed the entire D3.js Analysis View. This included modularizing the analysis UI with one JavaScript file per plot (four `viz-*.js` files).
- Contributed to writing and structuring the final report.