G2Viz: An Online Tool for Visualizing and Analyzing Public Transit System from GTFS Data

Sirapop Para[1], Thanachok Wirotsasithon[1], Thanisorn Jundee[1,2], Merkebe Getachew Demissie[3], Yoshihide Sekimoto[4], Filip Biljecki[5,6], Santi Phithakkitnukoon[1,2,*]

[1]City Context Lab, Department of Computer Engineering, Faculty of Engineering, Chiang Mai University, Thailand
[2]Excellence Center in Infrastructure Technology and Transportation Engineering (ExCITE), Chiang Mai University, Thailand
[3]Department of Civil Engineering, Schulich School of Engineering, University of Calgary, Canada
[4]Institute of Industrial Science, The University of Tokyo, Japan
[5]Department of Architecture, National University of Singapore, Singapore
[6]Department of Real Estate, National University of Singapore, Singapore

[*]Corresponding author: santi@eng.cmu.ac.th

Abstract

Public transit agencies have amassed substantial data through on-board and off-board sensors over the years. While data collection was the primary focus, there's now a shift towards deriving actionable insights from this wealth of information. As data-driven decision-making becomes increasingly vital, there's a growing need for effective ways to visualize and convey complex insights to decision-makers. This study addresses this need by introducing G2Viz, a visualizer for public transit operations. The development process of G2Viz spans requirement gathering, planning, and design, encompassing software architecture, data models, user interfaces, and system components. Rigorous implementation and testing ensure the tool's functionality and effectiveness. G2Viz, designed to dynamically visualize public transit operations using General Transit Feed Specification (GTFS) data, is a web application accessible globally via any web browser. Its open-source nature, robustness, and versatility facilitate communication among transit agencies, users, researchers, and city authorities. G2Viz empowers transit planners to make well-informed decisions about public transportation. (Access G2Viz at https://g2viz.citycontext.info)

Keywords

Data visualization, visualization tool, GTFS data, public transit operation, visual analytics, urban informatics

1. Introduction

Traffic data has been extensively explored for visualization purposes, involving the transformation of raw data into visual representations that offer valuable insights and enhance understanding and decision-making processes (Chen et al., 2015). Among the traffic data, trajectory data has gained significant attention in the visualization community (Andrienko et al., 2017). Prior studies have made notable advancements by utilizing trajectories of mobile phone users (M.G. Demissie et al., 2019; Deng et al., 2023; Phithakkitnukoon et al., 2022), GPS trajectories of taxis (M.G. Demissie et al., 2020), GPS trajectories of trucks (Merkebe Getachew Demissie & Kattan, 2022a; Kinjarapu et al., 2021), and GPS trajectories of shared electric scooter services (Phithakkitnukooon et al., 2021) to depict origin-destination flows.

Public transit agencies utilize on-board and off-board sensors to generate substantial volumes of big data. A review of the literature unveiled prominent data sources, including Automated Vehicle Location (AVL), Automated Passenger Counting (APC), Automated Fare Collection (AFC), and General Transit Feed Specification (GTFS) (Aemmer et al., 2022; Merkebe Getachew Demissie & Kattan, 2022b; Ge et al., 2021; Godfrid et al., 2022; Kunama et al., 2017; Prommaharaj et al., 2020). Assessing over previous studies, we evaluated the significance of these datasets in public transit planning and operations. Categorizing the studies, we identified various themes such as demand analysis (Li et al., 2011)(Ji et al., 2015), real-time tracking (Mazloumi et al., 2009), service optimization (Guido et al., 2016), access to jobs via transit service (Y. Kim et al., 2021, 2022), transit-based access to health services (J. Kim et al., 2023), and performance evaluation (Berkow et al., 2009; Glick et al., 2015; Ma & Wang, 2014; Mesbah et al., 2012).

Only a few studies have utilized transit big data sources to develop visualization tools for transit planning and operational decision-making processes. For example, Kurkcu et al. (Kurkcu et al., 2017) applied transit AVL data to create a web-based visualization tool that processed and displayed transit operations, including station and route selection and dwell time analysis modules. Another web-based application utilized AFC data to monitor and visualize the performance of bus fleets (Anwar et al., 2016). However, these previous visualization tools required heterogeneous data sources such as AFC, AVL, and incident data, leading to potential delays and needing external assistance to access and query stored data for transit agencies.

GTFS, a standardized format for sharing public transit data across various transportation systems, presents a valuable resource. Developing visualization tools based on GTFS presents several advantages, especially considering its widespread adoption by transit agencies and developers globally. These tools are more user-friendly and can simplify both transit planning and operational decision making by harnessing readily available standardized data. Therefore, the exploration of GTFS data as a foundation for visualization tools holds great potential for enhancing the management and efficiency of transit systems.

There are a few GTFS visualization tools available, each with its unique features and functionalities. Some tools, like gtfs-visualizations (Mueller, 2014) and GTFS-Viz (Kunama et al., 2017), were developed using Processing (Fry & Reas, 2023) but currently operate offline. Unfortunately, gtfs-visualizations, which focused on transit route visualization, is no longer functional due to outdated software dependencies (Processing 2.2.1). On the other hand, GTFS-Viz comprises two components; a data preprocessor and a visualizer – that collectively display transit vehicle locations on a map. Another Processing-based tool, PubtraVis (Prommaharaj et al., 2020), offers comprehensive visualization of mobility, speed, flow, density, and headway on a map. Its analysis mode generates detailed reports on prominent stations, route similarity, and route clustering.

In the realm of commercial products, ESRI ArcGIS Pro (ESRI, 2023) stands out as desktop GIS software capable of visualizing GTFS data. However, its usage involves a steep learning curve, which might pose a challenge for users unfamiliar with GIS software. Additionally, its licensing fees could limit accessibility, particularly for smaller organizations or individuals.

Furthermore, open-source Python-based tools like gtfspy-webviz (Kujala, 2020) offers online visualization of transit mobility, statistics, plots, segments, routes, and spreading. However, operating gtfspy-webviz necessitates a degree of technical proficiency in Python, making it not very user-friendly. Several Python packages cater specifically to GTFS data manipulation and visualization. For instance, gtfs-segments (Devunuri, 2024) excels in extracting stop spacing information from GTFS feeds, facilitating further in-depth analysis. Similarly, gtfs-functions (Toso & Oja, 2023) provides preprocessing capabilities for GTFS data, allowing visualization in Python environments through figures and plots.

Resources like GTFS Builder allow rural and tribal transit agencies to create fully valid GTFS for their bus routes (National RATP, 2024). Several GTFS-based R Packages have been introduced to enhance the value of GTFS data. For example, Pereira et al. (2023) have created gtfs2gps, an open-source R package that utilizes parallel computing to convert GTFS feeds from relational text files into a trajectory data table. In a similar method, Pereira et al. (2021) developed the r5r package, streamlining routing analysis by enabling the calculation of travel time matrices and accessibility. Furthermore, Herszenhut et al. (2023)

introduced gtfstools, designed for editing and analyzing transit feeds in GTFS format. Many features of this package are derived from functions found in other packages, such as tidytransit and gtfs2gps (Herszenhut et al., 2023).

Each tool and package offers its advantages and limitations, catering to diverse user needs and technical proficiencies in handling and visualizing GTFS data. Choosing the right tool often depends on the specific requirements, level of technical expertise, and the depth of analysis desired for the transit data.

In this study, we further unlock the potentials of GTFS data by developing an online visual analytics tool called G2Viz. The inspiration for G2Viz comes from the success of our previously developed tool, PubtraVis (Prommaharaj et al., 2020). During the development of PubtraVis, we conducted in-depth discussions with key stakeholders, including transit agencies, transit users, and academicians. Through these discussions, we gained valuable insights into the desired functionality of the visualization tool, user expectations, and technical requirements. Transit agencies expressed a need for a visualization tool that could display routes, transit stations, and schedules, while also providing basic visual analytics capabilities. By incorporating these valuable inputs from stakeholders, G2Viz is designed and developed to meet the specific needs of transit agencies and enhance the overall transit planning and operation decision-making processes. Unlike PubtraVis and other offline tools (i.e., GTFS-Viz, gtfs-visualizations, ESRI ArcGIS Pro), the G2Viz is a web application accessible globally through any web browser. Its cross-platform compatibility ensures usability on various devices, including desktops, laptops, and tablets, regardless of the operating system. By offering web-based access, G2Viz expands its utility to a wider audience, making it a versatile tool for transit agencies, researchers, and stakeholders in the public transportation field. When compared to its online visualization counterpart, gtfspy-webviz, G2Viz is more user-friendly. Its interface does not demand technical expertise from users for tasks such as importing GTFS data, tool execution, and deployment.

2. Methodology

2.1 GTFS data description

GTFS is a standardized format for public transit schedules and geographic data. Initially developed by Google in collaboration with transit agencies, it became an open data standard in 2006. GTFS includes text files with information about routes, schedules, stops, and geographic features. Table 1 summarizes the essential files in a GTFS feed, categorized as required, conditionally required, or optional. Figure 1 illustrates their interconnections.

Two types of GTFS data exist: GTFS Static and GTFS Realtime. GTFS Static comprises text files offering a snapshot of transit schedules and geography, updated periodically, and devoid of real-time data like delays. Conversely, GTFS Realtime offers real-time updates but employs a distinct data format and delivery via an API. Our current work centers on GTFS Static. The development of an online visual analytics tool for GTFS Realtime remains an intriguing avenue for future research.

Table 1. Most common files included in a GTFS feed.

File name	Description	Required
agency.txt	Contains information about the transit agency providing the data, such as name, URL, and time zone.	Required

stop.txt	Provides information about the stops where transit vehicles pick up and drop off passengers, including their names, geographic locations, and other details.	Required
routes.txt	Describes the routes that the transit agency operates, including their names, types (e.g., bus, subway), and the areas they serve.	Required
trips.txt	Describes the trips that transit vehicles take on a particular route at specific times, including the start and end times, and the sequence of stops.	Required
stop_times.txt	Provides the arrival and departure times for a transit vehicle at each stop along its route.	Required
calendar.txt	Specifies the dates and times when service is available for each route, as well as any exceptions to the normal schedule. This file is required unless all dates of service are defined in calendar_dates.txt.	Conditionally required
calendar_dates.txt	Provides exceptions to the service dates specified in the calendar.txt file, such as holidays or special events. If calendar.txt is omitted, then calendar_dates.txt is required and must contain all dates of service.	Conditionally required
feed_info.txt	Contains dataset metadata, including publisher, version, and expiration information.	Conditionally required
fair_attributes.txt	Provides information about the fares for each route, including prices and payment methods.	Optional
fare_rules.txt	Describes the rules for calculating fares, such as discounts and transfer policies.	Optional
shapes.txt	Describes the geographic shape of a route using a series of latitude and longitude coordinates.	Optional
frequencies.txt	Describes time between trips for or a compressed representation of fixed-schedule service.	Optional
transfers.txt	Describes rules for making connections at transfer points between routes.	Optional
pathways.txt	Provides information about pathways linking together locations within stations.	Optional

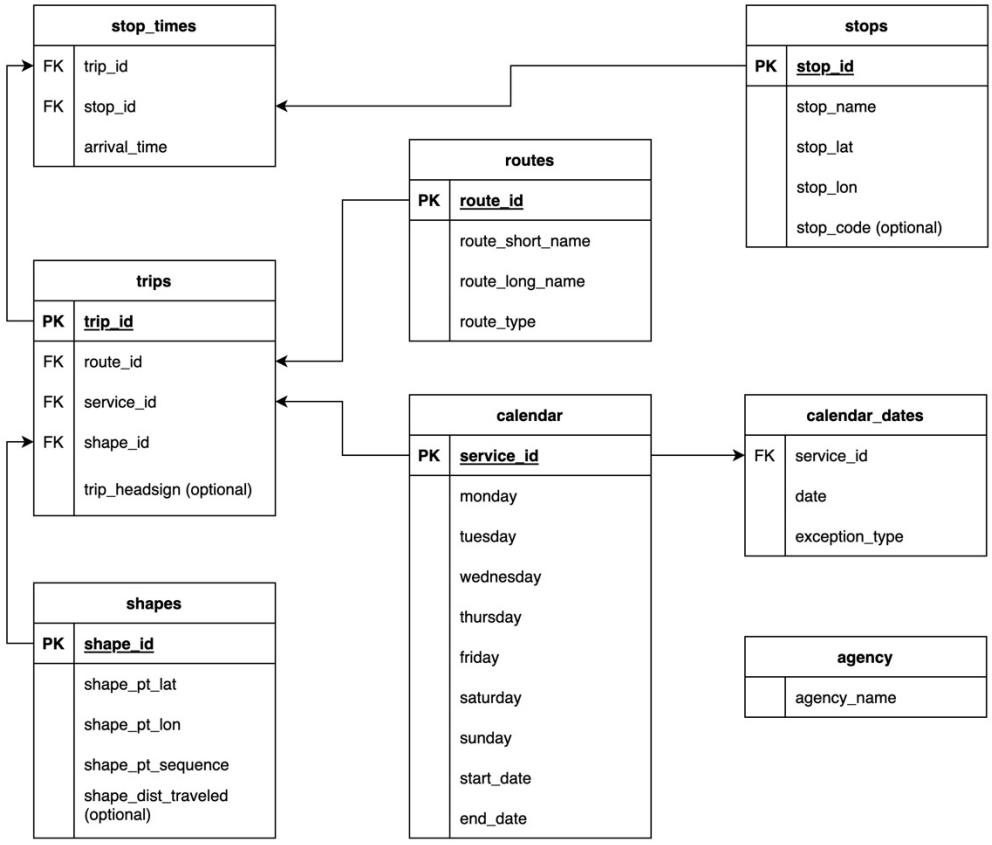


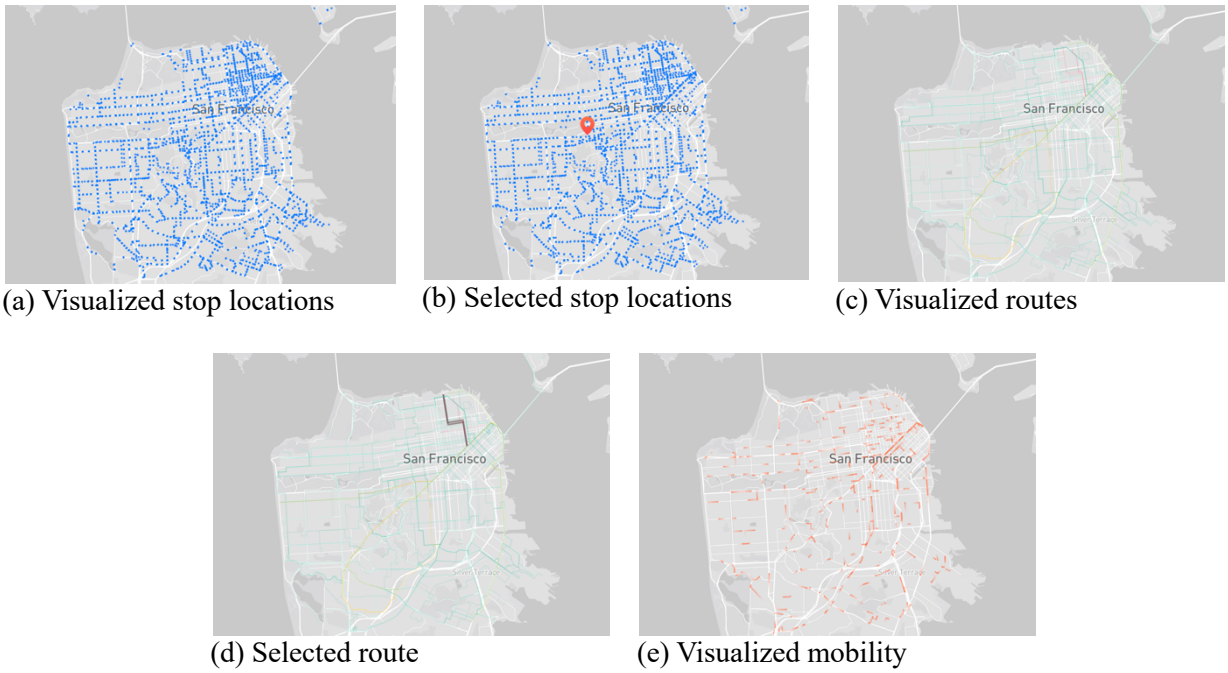
Figure 1. Relational model of text files included in a GTFS feed.

2.2 G2Viz overview

Inspired by the success of our prior tool, PubtraVis (Prommaharaj et al., 2020), G2Viz takes a step further. Unlike PubtraVis, G2Viz is a web app, accessible globally via web browsers on any device, regardless of the operating system. This cross-platform approach significantly broadens its accessibility.

G2Viz serves as a valuable tool for transit planners, enabling them to visualize and analyze general aspects of transit systems, aiding in service modification and strategic management (Berkow et al., 2009). It offers users insights into station locations, routes, transit vehicle movement, as well as transit performance metrics like service rate, speed, and headway.

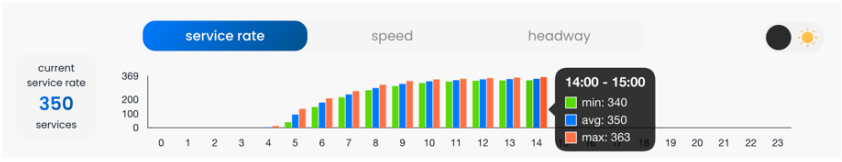
The user can upload a GTFS data for visualization in G2Viz or choose from provided samples. The tool visualizes transit aspects like station locations, routes, and vehicle movement on a map. The user can select a stop or route for more details. Fig. 2 illustrates this with examples from San Francisco GTFS data: (a) shows stop locations, (b) a selected stop, (c) visualized routes, (d) a selected route, and (e) vehicle movement. For transit performance, G2Viz uses bar charts in Figs. 3(a), 3(b), and 3(c) for service rate, speed, and headway, providing stats like minimum, maximum, and average values across daily hours.



(a) Visualized stop locations (b) Selected stop locations (c) Visualized routes

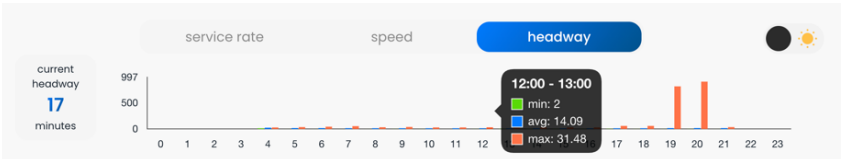
(d) Selected route (e) Visualized mobility

Figure 2. Snapshots of general aspects of transit that G2Viz provides, i.e., stop locations, routes, and mobility. (GTFS source: SFMTA, San Francisco)



(a) Service rate

(b) Speed



(c) Headway

Figure 3. Snapshots of transit performance information conveyed by a bar chart with minimum, maximum, and average values during daily service hours that the *G2Viz* provides, i.e., service rate, speed, and headway.

2.3 System architecture

G2Viz's system architecture, depicted in Fig. 3, follows the typical web app structure with two sections: front-end and back-end. The front-end hosts user interfaces for interactions, sending requests to the back-end. The back-end, or business logic section, provides services like data preprocessing, geospatial data processing, and transport performance data processing. After processing, the system sends results as JSON responses to the front-end for visualization.

G2Viz utilizes various technologies in both the front and back-ends. The front-end uses Vue.js[1] for building user interfaces, Deck.GL[2] for drawing a map and rendering geospatial information i.e., mobility, stops, and routes. Chart.js[3] is used for visualizing transit performance measures, i.e., service rate, speed, and headway, as clustered bar charts. For the back-end, FastAPI[4] is used for developing web APIs with Python, while handling the business logic is done with Python, and Pandas (McKinney, 2011) is used for data processing. For the deployment, Docker[5] is utilized to build, share, and run G2Viz.

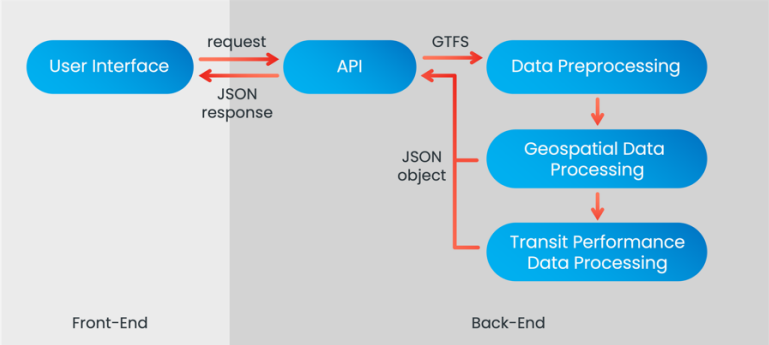


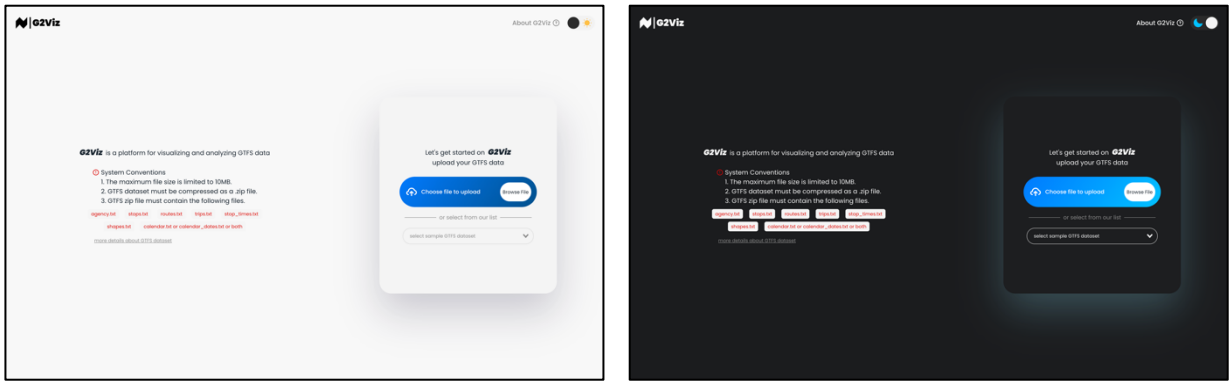
Figure 3. System architecture of *G2Viz*.

2.4 User interfaces

Regarding the user interfaces, there are five main pages that allow user to visualize and interact with the processed GTFS data, i.e., homepage, file uploading, mobility visualization, stop visualization, and route visualization.

Homepage. The homepage is where the user begins. At the top, there's a logo that redirects users to the homepage when clicked. Users can toggle between dark and white themes for display (Figs. 4(a) and 4(b)). The left side has G2Viz info and a link to GTFS references below. On the right, users find a file upload button and a dropdown menu for sample GTFS datasets. This allows users to visualize their own or try the tool with sample data.

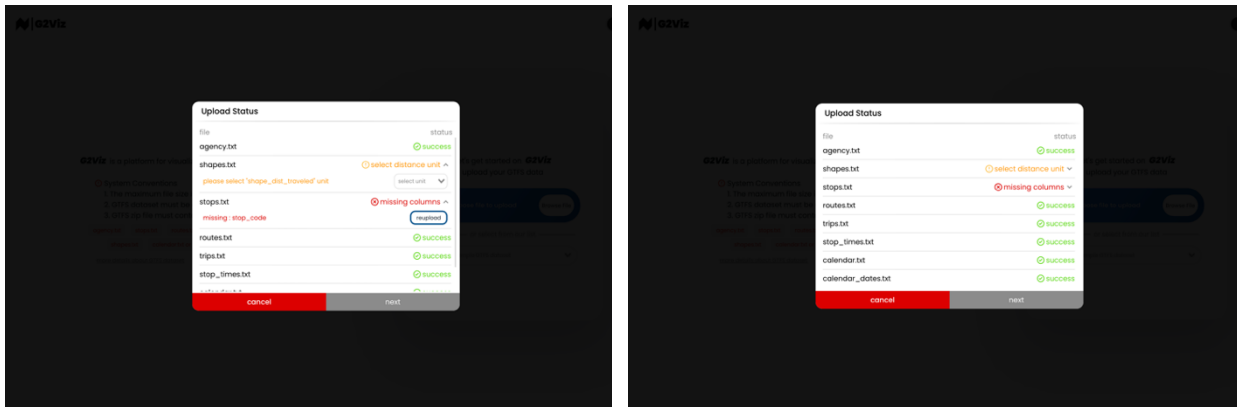
[1] https://vuejs.org
[2] https://deck.gl
[3] https://www.chartjs.org
[4] https://fastapi.tiangolo.com
[5] https://www.docker.com



(a) White mode theme (b) Dark mode theme

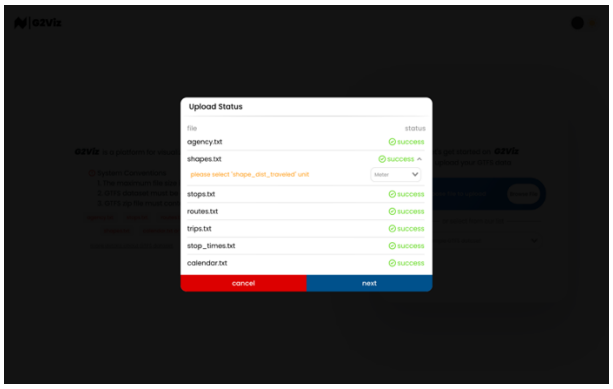
Figure 4. Homepage interface of G2Viz.

File Upload. After uploading GTFS data successfully, a modal appears (Fig. 5(a)). It displays file status: green for success, orange for warning, and red for error. Clicking on warnings or errors expands details (Fig. 5(b)). A re-upload button on the right lets users upload only incomplete files without redoing the whole GTFS dataset. When all files validate (Fig. 5(c)), the 'next' button appears, allowing users to move forward.



(a) The corresponding details when the user clicks on a warning or error message.

(b) File uploading modal interface.



(c) The stage when all files are successfully validated and ready to proceed to the next stage.

Figure 5. File upload interfaces.

Mobility Visualization. After processing GTFS data, the website redirects to the mobility visualization page (Fig. 6) with three sections: control panel (left), chart (top), and map (center). The control panel includes agency name, mode selector (Mobility, Stops, Routes), calendar, peak period info, running timestamp, time slider, mobility speed slider, and contact info with a user feedback link. The chart section, using Chart.js, displays transport performance as bar charts (service rate, speed, headway), with average values for each hour. The map section dynamically shows transit vehicle movement based on the timestamp.

Stop Visualization. When switching to stop visualization mode by clicking "stops," the control panel and map change (Fig. 7). The control panel includes mode selector, calendar, and stop dropdown for mode, date, and stop selection, along with a top ten list of crowded stations. The map transforms into a scatter plot visualizing all stops. The user can interact by clicking on a stop, which displays stop details below the dropdown. These details include stop name, code, service routes, and trip info like route short name, headsign, arrival time, and headway (Fig. 8).

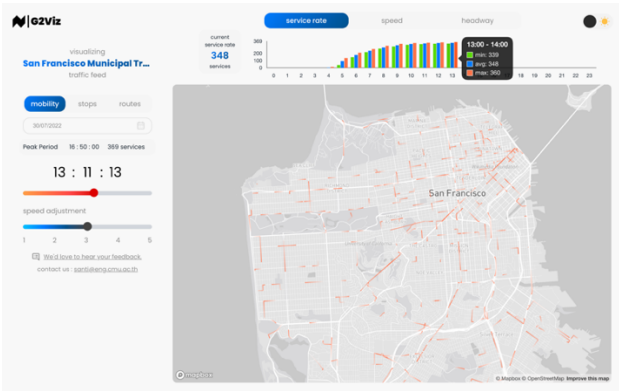


Figure 6. Mobility visualization interface.

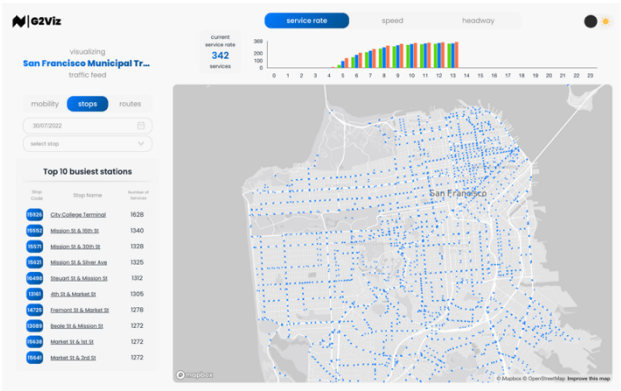


Figure 7. Stops visualization interface.

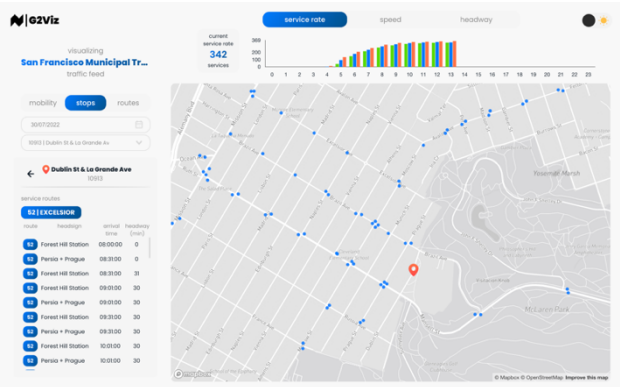


Figure 8. Stop details are showed in the control panel and the

Route Visualization. To explore transit routes on the map, the user clicks the "routes" button to enter route visualization mode (Fig. 9). Here, the control panel offers mode selection, calendar, and a route comparison feature for analyzing and planning routes. The comparison feature helps with route analysis, expansion, system design, and accessibility studies. The user can choose specific routes for comparison via the route dropdown, and hovering over route colors reveals route types. Below this, the user will find a top ten list of the busiest routes (i.e., the routes with the greatest number of trips in the day). Routes are displayed as path layers for interaction. When selecting a route, the control panel shows route details like short name, long name, service days, route type, and stop info (Fig. 10).

When enabling the comparison feature, the route dropdown becomes a second calendar to compare two routes using different dates (Fig. 11). Route colors change accordingly, making the comparison clear. The user can toggle the view of each compared route. While the user can still interact with the map by selecting a route, stop details are not displayed in this mode (Fig. 12).

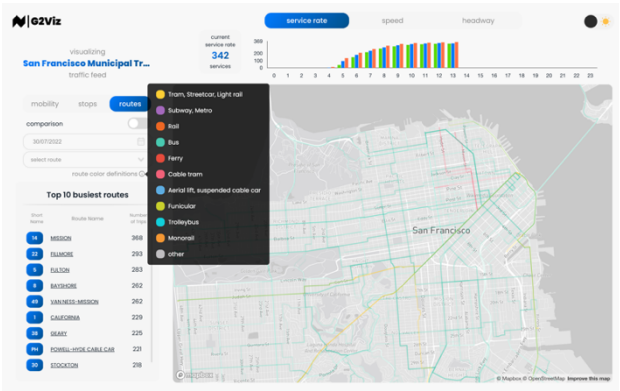


Figure 9. Route visualization interface with a route legend displayed when hovering the cursor or mouse over.

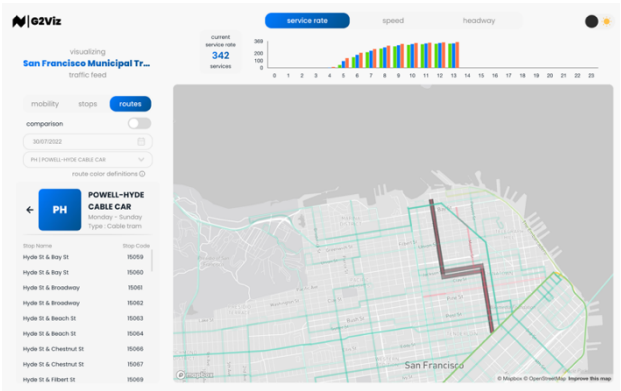


Figure 10. When route is selected, its detail is showed in the control panel and the selected route is highlighted on the map.

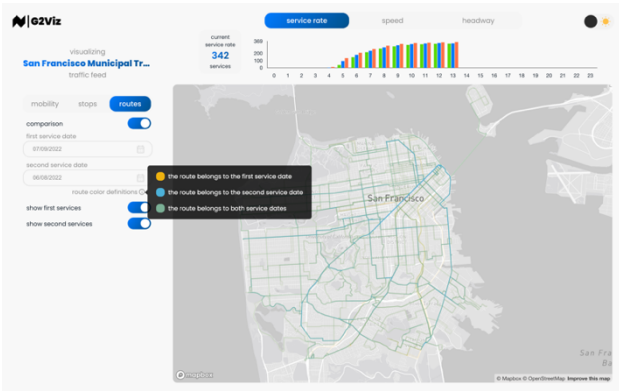


Figure 11. User can select two service dates to be compared using the comparison feature. Different colors represent routes that belong to each comparing service date as well as routes that belong to both service dates.

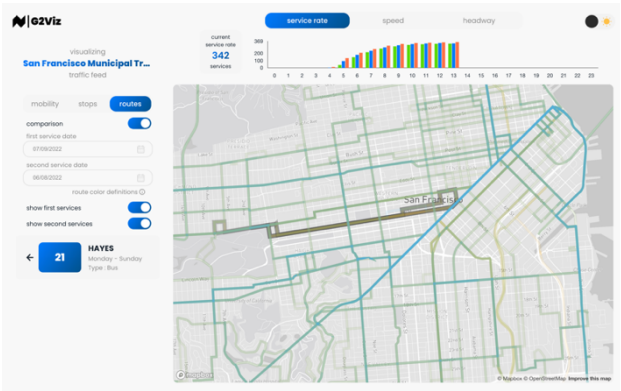


Figure 12. Route details are still available while using the route comparison feature.

2.5 API

In general, the API (Application Program Interface) provides the users with seamless integration and programmable access to the application's features and data by serving as a bridge between different software applications, allowing them to communicate and interact with each other. In the context of G2Viz, the API services, developed using FastAPI, enable the front-end to retrieve and exchange data with the back-end. Table 2 contains a list of G2Viz's endpoints that enable the developers to request the services that suit specific needs by using different HTTP request methods; GET for retrieving data, POST for creating or uploading new data.

Table 2. *G2Viz's* API endpoints

Method	Endpoint	Description
POST	/upload/chunk	Provides a service for uploading file to the server.
POST	/reupload/chunk	Provides a service for re-uploading file to the server.
GET	/upload/status/{file_name}/{file_path}	Provides a service for retrieving an uploading status.
GET	/reupload/status/{file_name}/{file_path}	Provides a service for retrieving a re-uploading status.
GET	/mobility/{file_path}	Provides a service for processing mobility data and retrieving initial mobility data.
GET	/mobility/chunk_duration /{file_path}/{date}/{time}/{duration}	Provides a service for retrieving mobility data by a specific period of time.
GET	/analytics/service_rate/{file_path}/{date}	Provides a service for processing and retrieving service rate data by a specific date.
GET	/analytics/speed/{file_path} /{date}	Provides a service for processing and retrieving speed data by a specific date.
GET	/analytics/headway/{file_path}/{date}	Provides a service for processing and retrieving headway data by a specific date.
GET	/stops/{file_path}/{date}	Provides a service for processing and retrieving all stops data by a specific date.
GET	/stops/{file_path}/{date} /{stop_id}	Provides a service for processing and retrieving a specific stop data by a specific date and stop ID.
GET	/routes/{file_path}/{date}	Provides a service for processing and retrieving all routes data by a specific date.
GET	/routes/{file_path}/{date} /{route_id}	Provides a service for processing and retrieving a specific route data by a specific date and route ID.

2.6 Data preprocessing

In order to prepare the GTFS dataset for visualization and ensure data accuracy, *G2Viz* performs a two-step data preprocessing approach consisting of data cleaning and data transformation.

Data Cleaning. GTFS data often contains errors, such as incorrect or missing values. To minimize the risk of errors and ensure data integrity, G2Viz identifies and removes records with incorrect values by comparing their data types with the GTFS reference and subsequently removes the corresponding records. For the missing values, G2Viz utilizes the “dropna” function from the Pandas library, incorporating parameter adjustments beyond default values to handle NULL values. Two parameters, "how" and "subset", are adjusted in this process.

1. “how” Parameter: Set to “any”, this parameter dictates that a row will be discarded if any values within the subset are NULL.
2. “subset” Parameter: Dynamically assigned based on the “required” and “Conditionally Required” fields of the dataset, the “subset” parameter is essential for the calculation of each functionality in G2Viz. For instance, the subset for the “trips.txt” file encompasses fields such as “route_id”, “service_id”, “trip_id”, and “shape_id”.

Furthermore, the structure of GTFS as a relational dataset pushes G2Viz to do more than just remove rows with NULL values. It generates corresponding lists of dropped primary keys, as shown in Fig. 1, including “dropped_stop_id”, “dropped_route_id”, “dropped_trip_id”, “dropped_service_id”, and “dropped_shape_id” in order to keep track of the eliminated values associated with primary keys. For example, if there are any missing values in the “routes.txt”, the row will be removed and the corresponding “route_id” will be added to the “dropped_route_id”. After processing all dataset files, G2Viz takes an additional step by removing relevant NULL values from related files. For example, it removes entries in “trips.txt” with “route_ id” found in the list of “dropped_ route _id”.

Data Transformation. GTFS data can be diverse and contain numerous attributes that are unnecessary for this tool’s functionality, such as zone_id, stop_url, location_type, parent_station, stop_timezone, wheelchair_boarding, level_id, and platform_code in “stops.txt” file, as well as direction_id, block_id, wheelchair_accessible, and bikes_allowed in “trips.txt” file. For optimizing performance and reducing the size of processed data, *G2Viz* eliminates all unnecessary attributes from the data by implementing an array of essential columns for each file and utilizing the power of the “usecols” parameter within the Pandas library’s “read_csv” API. By employing this approach, G2Viz reads only the required columns, optimizing performance and reducing the data size effectively.

Through these data cleaning and transformation, *G2Viz* prepares a well-suited GTFS dataset for the next stage. This process removes errors, such as incorrect or missing values, and eliminates unnecessary attributes. The result is an optimized dataset that enhances the accuracy and efficiency of subsequent data processing stages.

2.7 Geospatial data processing

As Deck.GL is employed for our geospatial data visualization i.e., mobility, stops, and routes, the preprocessed data then needs to be put into an appropriate and corresponding form.

Mobility. A table containing all required data attributes for visualizing the mobility is generated, as shown in Table 3. Here we explain how each attribute is obtained from the data processing’s point of view.

Table 3. Mobility attribute table.

Attribute	Type	Description
start_date	Date	The first service date from calendar.txt or calendar_dates.txt file.
end_date	Date	The last service date from calendar.txt or calendar_dates.txt file.
initial_lon	Number	An average longitude for initializing the background map.
initial_lat	Number	An average latitude for initializing the background map.
agency_name	Text	An agency name for displaying in the control panel.

path	Array of objects (shape_id and coordinates)	An array containing all path details.
shape_id	Number	A shape ID from shapes.txt file.
coordinates	Array of numbers	A shape coordinates from shapes.txt file.
date	Date	A current visualization date.
time	Number	A current visualization time.
path_timestamps	Array of objects (shape_id and timestamps)	A collection of mobility timestamps associated with shape_id.
timestamps	Array of numbers	A collection of all timestamps.

The control panel displays the agency name extracted from the GTFS feed's agency.txt file in the upper left corner. For the calendar component, "start_date" and "end_date" attributes are sourced from calendar.txt or calendar_dates.txt files to provide valid dates for selection. "Date" and "time" attributes are used for front-end validation.

For the background map section, the "initial_lon" and "initial_lat" values are calculated based on the average longitude and latitude values from the shapes.txt file, which are then used by Deck.GL to set the initial position of the map. In order to display mobility on the map using Deck.GL, two essential attributes are required; path_timestamps and paths. The "coordinates" attribute is a set of arrays containing longitude and latitude values that are used to locate the transit vehicle's location on the map, which can be extracted directly from "shape_pt_lat" and "shape_pt_lon" fields in "shapes.txt" file. Consequently, the "path" attribute is simply a set of arrays that contain "coordinates" and "shape_id" as a reference field. The "timestamps" attribute is used to indicate the visualization time of each coordinate, which is not originally provided in the GTFS data and the "path_timestamps" attribute is simply a set of "timestamps" along with associated "shape_id" as a reference value.

To generate the "timestamps" attribute, the travel time and the number of coordinates along each path are required. The travel time is calculated as the difference between the first and last "arrival_time" fields in the "stop_times.txt" file for a specific "trip_id". For example, "trip_id: 224060", where the first "arrival_time" is 15:35:00 and the last "arrival_time" is 15:55:00, as shown in Fig. 13, has a travel time of 1,200 seconds or 20 minutes. For the number of coordinates for each path, it is obtained by counting the records associated with each "shape_id". For example, the "shape_id: 9202" contains 86 coordinates, as depicted in Fig. 14. Having obtained the travel time and the number of coordinates for each path, we create the "timestamps" attribute by merging the "shapes.txt", "trips.txt", and "stop_times.txt" files. Then, we create an array with a size equal to the number of coordinates and assign the first and last values as the "arrival_times" of the first and last records of that path. Subsequently, we calculate each timestamp in the array using a summation series. We divide the travel time by the number of coordinates for that specific path. In the case of "trip_id: 224060", the timestamps array is generated by creating an arithmetic sequence with a common difference calculated as the quotient of 1,200 divided by 86, resulting in a value of 13.95, as shown in Fig. 15. This ensures that the timestamps are evenly distributed along the path.

trip_id	arrival_time	departure_time	stop_id	stop_sequence	stop_headsign	pickup_type	drop_off_type	shape_dist_traveled	timepoint
224060	**15:35:00**	15:35:00	100799	1		0	0	0	1
224060	15:36:51	15:36:51	3925	2		1	0	0.904117	0
224060	15:39:04	15:39:04	6975	3		1	0	1.480276	0
224060	15:40:47	15:40:47	3880	4		1	0	1.925972	0
224060	15:41:50	15:41:50	3829	5		1	0	2.200333	0
224060	15:43:36	15:43:36	100441	6		1	0	2.662899	0
224060	15:46:50	15:46:50	3703	7		1	0	3.536706	0
224060	15:48:04	15:48:04	3627	8		1	0	3.858911	0
224060	15:48:42	15:48:42	3599	9		1	0	4.027227	0
224060	15:49:58	15:49:58	3527	10		1	0	4.356887	0
224060	15:51:13	15:51:13	3470	11		1	0	4.682072	0
224060	15:52:48	15:52:48	3457	12		1	0	5.123398	0
224060	**15:55:00**	15:55:00	3476	999		0	0	5.656377	1

Figure 13. The first and last arrival_times of "trip_id: 224060".

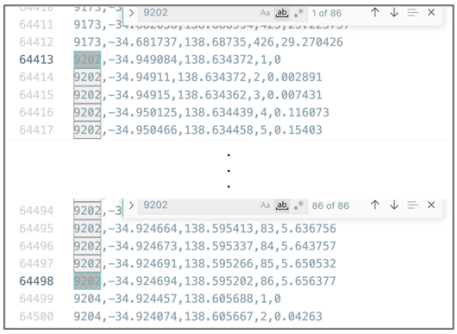


Figure 14. The number of coordinates along the "shape_id: 9202".


```
{
    "trip_id": 224060,
    "shape_id": 9202,
    "timestamps": [
        56100.0,
        56113.95,
        56127.91,
        56141.86,
        56155.81,
        56169.77,
```


Figure 15. The timestamps of "trip_id: 224060".

Stops. Likewise, a table containing all required data attributes for visualizing stops is generated, as shown in Table 4. Here, we will elaborate on how each attribute is derived from a data processing perspective.

Table 4. Stop attribute table.

Attribute	Type	Description
top_stop_list	Array of objects (stop_id, stop_code, stop_name, and count)	A list of top ten stops with the greatest number of services of that day.
stops_data	Array of objects (stop_id, stop_code, stop_name, stop_lat and stop_lon)	An array of all stop data.
stop_data	Object (stop_code, stop_name and an array of routes_data)	An object of stop data with a specific "stop_id".
routes_data	Array of objects (route_short_name and route_long_name)	An array of all route data associated with the stop.
services_data	Array of objects (route_short_name, trip_headsign, arrival_time and headway)	An array of all service data associated with the stop.

Switching to "stops" mode transforms the interface with features: a stop dropdown, top ten crowded stations, and map markers. Data for these features comes from "stops.txt," using "stop_lat" and "stop_lon" for Deck.GL mapping. The "stop_id" attribute links location to stop info, while "stop_code" and "stop_name" create dropdown options tied to "stop_id" for seamless interaction. Additionally, "top_stop_list" contains attributes for the top ten crowded stations, as shown in Fig. 9.

When the user selects a stop from the map, the dropdown, or top ten list, the "stop_data" will provide essential information such as the "stop_code" and "stop_name" attributes, which are extracted directly from the "stops.txt". Moreover, the "routes_data" includes the "route_long_name" and "route_short_name" attributes that correspond to the selected stop, sourced from the "routes.txt" file. These attributes are utilized to display the stop information in the control panel, as shown in Fig. 10. Furthermore, the "services_data"

is employed to display all trip data associated with the selected stop, providing the user with comprehensive information about the services associated with that station.

Routes. All required data attributes for visualizing routes are gathered in a generated table shown in Table 5. Like mobility and stops, here we outline the process of obtaining each attribute from a data processing perspective.

Table 5. Route attribute table.

Attribute	Type	Description
top_route_list	Array of objects (route_id, route_short_name, route_long_name, and count)	A top ten list of routes with the greatest number of services of that day.
routes_data	Array of objects (route_id, route_short_name, route_long_name, route_type, and path)	An array of all routes data.
route_data	Object (route_short_name, route_long_name, and route_type)	An object of route data with a specific "route_id".
service_day	Array of numbers	The array of service days of the week (Monday thru Sunday), with "0" indicating that the route has no service on that day, while "1" implying that the route has service on that day.
stops_data	Array of objects (stop_code and stop_name)	An array of all stops data associated with the route.

When the "routes" mode is selected, it activates route visualization, including a route dropdown, a top ten list of busy routes, and routes displayed on the map. The data required for this comes from "routes_data," mostly directly from "routes.txt." However, to extract the "route" attribute, the data is merged from "routes.txt," "trips.txt," and "shapes.txt." Deck.GL uses "route" and "route_type" for map display and coloring, while "route_id" links map routes to their data. Additionally, "route_short_name" and "route_long_name" populate the route dropdown. Each dropdown option is linked to a "route_id" for user interaction. The "top_route_list" handles the top ten crowded routes, as shown in Fig. 11.

When a route is selected by the user, "route_data" provides crucial information such as "route_short_name," "route_long_name," and "route_type," all extracted directly from "routes.txt." Additionally, "service_day" indicates the schedule for that route, derived by merging data from "calendar.txt," "trips.txt," and "routes.txt," specifically considering the days from "Monday" to "Sunday" in "calendar.txt." These attributes populate route information in the control panel, as seen in Fig. 12. Moreover, "stops_data" is employed to display all stop details linked to the selected route, offering comprehensive stop information for that route.

2.8 Transit performance data processing

As Graph.js is employed for our transit performance visualization i.e., service rate, speed, and headway, the preprocessed data then needs to be put into an appropriate and corresponding form similar to the geospatial data processing requirements.

Service rate. Table 6 displays a comprehensive set of data attributes necessary for visualizing the service rate. The following explanation outlines the process of acquiring each attribute from a data processing perspective.

Table 6. Service rate attribute table.

Attribute	Type	Description
initial_data	Object (min_service_rate, max_service_rate, peak_time, and peak_number)	A set of data for initializing the visualization of service rate.
hourly_service_rate_data	Array of objects (hour, min, max, and average)	A collection of hourly statistical data for service rates.
service_rate_data	Array of objects (timestamp and count)	A collection of service rate data for chart plotting.

The "initial_data" utilizes "min_service_rate" and "max_service_rate" to set the initial chart axis scale. The "peak_time" and "peak_number" are used to inform the peak service rate duration in the control panel, as shown in Fig. 8.

As the visualization time progresses, the chart dynamically updates the current service rate using "service_rate_data" by comparing the visualization time with the corresponding "timestamp" attribute and updating the chart value based on the "count" attribute. To visualize data from the past hour, the chart relies on the "hourly_service_rate_data," which contains statistics like minimum, maximum, and average service rates for each hour.

Creating "service_rate_data" and "hourly_service_rate_data" involves merging "trips.txt," "calendar.txt," "calendar_dates.txt," and "stop_times.txt." Then, filtering trips based on the selected visualization date by comparing dates with associated "service_id" in "calendar.txt" or "calendar_dates.txt." Service rates are calculated by counting trips for each time period. To do this, "arrival_time" is converted into a numerical format, and trips are grouped by "trip_id." Start and end timestamps for every trip are obtained by analyzing the "arrival_time" of the first and last sequences in each trip. Then, transit performance data is calculated at 60-second intervals using "service_rate_data," with timestamps generated as an arithmetic sequence with a 60-second interval. Service rates are determined by comparing these timestamps with trip start and end times. If a timestamp falls within a trip's service window, it indicates service during that period, and the count is updated accordingly.

Speed. Table 7 exhibits a compilation of essential data attributes for visualizing speed. In the subsequent section, we outline the process of acquiring each attribute from the standpoint of data processing.

Table 7. Service rate attribute table.

Attribute	Type	Description
initial_data	Object (min_speed and max_speed)	A set of initial data for initializing the visualization of speed.
hourly_speed_data	Array of objects (hour, min, max, and average)	A collection of hourly statistical data for speed.
speed_data	Array of objects (timestamp, min, max, and average)	A collection of speed data for chart plotting.

The "initial_data" relies on "min_speed" and "max_speed" attributes to set the initial chart axis scale. The "speed_data" updates the current speed statistics, while "hourly_speed_data" displays statistical speed data for the past hour on the chart.

Creating “speed_data” and “hourly_speed_data” involves merging “trips.txt,” “calendar.txt,” and “calendar_dates.txt” files. All trips for the current visualization date are filtered based on the selected date and the corresponding “service_id” in “calendar.txt” or “calendar_dates.txt.” Speed calculation requires trip distance and travel time data. First, “shapes.txt” is merged into the dataset. To determine trip distances, the dataset is grouped by “trip_id,” and distances are extracted from the “shape_dist_travel” attribute with the highest “shape_pt_sequence” value, indicating the trip’s end. Then, the dataset is merged with “stop_times.txt” to calculate travel times. To compute travel times, “stop_time.txt” data is grouped by “trip_id,” and “arrival_time” is converted to numerical format. Travel time is calculated by subtracting the “arrival_time” values of the first and last “stop_sequence” attributes for each trip. Finally, speed is computed by dividing distance by travel time for each trip.

Similar to the service rate calculation, speed data is computed at 60-second intervals. It’s grouped by corresponding timestamps using the same conditions as for the service rate. Utilizing Pandas library’s DataFrame API, we apply the min, max, and mean functions to each group to calculate statistical data for each timestamp. These results are then assigned to the “min,” “max,” and “average” attributes within “speed_data”. For “hourly_speed_data,” we further group the data from “speed_data” by hour based on their “timestamp” values. Once again, we apply the min, max, and mean functions to compute statistical data for each hour. The outcomes for each hour are assigned to the “min,” “max,” and “average” attributes within “hourly_speed_data”.

Headway. The generated table includes all the necessary data attributes for visualizing headway, as displayed in Table 8. Let us delve into the explanation of how each attribute is obtained from the viewpoint of data processing.

Table 8. Headway attribute table.

Attribute	Type	Description
initial_data	Object (min_headway and max_headway)	A set of initial data for initializing the visualization of headway.
hourly_headway_data	Array of objects (hour, min, max, and average)	A collection of hourly statistical data for headway.
headway_data	Array of objects (timestamp, min, max, and average)	A collection of speed data for chart plotting.

The “initial_data” employs the “min_headway” and “max_headway” attributes to establish the initial chart axis scale, consistent with the methodology from previous sections. The “headway_data” is responsible for updating the real-time headway statistics, while “hourly_headway_data” displays the headway statistics for the past hour on the chart.

To compute “headway_data,” the system requires data from “trips.txt,” “stop_times.txt,” “calendar.txt,” and “calendar_dates.txt” files. Initially, “trips.txt” is combined with “calendar.txt” and/or “calendar_dates.txt” files to filter trips providing service on the chosen date. This filtering is based on date comparisons with the “service_id” for each trip. Once trips available for the current visualization date are identified, they are linked with “stop_times.txt” to extract “arrival_time” values.

Subsequently, trips are grouped by “trip_id” and “trip_headsign,” grouping sub-trips with the same headsign together. This arrangement facilitates headway calculation by considering differences in “arrival_time” values within each group. After computing headways, the dataset is grouped based on “arrival_time” values, coinciding with 60-second interval timestamps. The headway data within each timestamp undergoes analysis using the DataFrame API to calculate statistical data, mirroring the approach used in speed visualization. The results are assigned to the “headway_data” attribute and serve as the basis for computing “hourly_headway_data”.

For “hourly_headway_data”, the procedure parallels previous subsections. Initially, “headway_data” is grouped by hour, considering “timestamps” values. Subsequently, min, max, and mean functions are applied to each hourly group to derive hourly statistical data, which is then assigned to the “hourly_headway_data” attribute.

3. Demo

G2Viz has been fully developed and operates as detailed in the preceding sections. It is accessible to the public online via https://g2viz.citycontext.info. Additionally, a demonstration video illustrating G2Viz’s functionality is accessible at https://www.youtube.com/watch?v=6BhUVtMDmyQ.

4. Results

To evaluate G2Viz’s user experience, we conducted a comprehensive assessment that included examining its processing speed, usability study, and comparing transit performance between two different cities’ systems to showcase G2Viz’s potential applications.

4.1 User experience study

Processing time. A web app’s processing time directly impacts user experience. Delays in tasks or information loading can frustrate users. Given G2Viz’s data handling, processing time is crucial. We assessed it with 20 diverse GTFS datasets of various sizes. Three separate tests were conducted for each dataset, and the resulting processing times are detailed in Table 9. The longest processing time recorded was 62.19 seconds during the third run of the Adelaide Metro dataset, which had a file size of 10.00 megabytes, the maximum allowed by our server. This dataset also exhibited the highest average processing time of 59.50 seconds. In contrast, the quickest processing time of 3.42 seconds was achieved during the third run of the Negaibus dataset, which was the smallest at 0.30 megabytes among the datasets examined. Statistically, there’s a correlation between processing time and input GTFS file size, as illustrated in Fig. 16. This relationship can be characterized by a strong linear regression described by Eq. (1) with an R^2 value of 0.82.

$$\mathbf{y = 4.94x + 4.13}, \tag{1}$$

where x is the GTFS file size in megabyte and y is the processing times in second. This result suggests that the average waiting time for the user should not exceed one minute, regardless of the size of the input file, provided that the maximum allowed size is 10 megabytes.

Processing time isn’t solely determined by file size. The number of trips in the “trips.txt” file is a significant factor. As a result, larger files can sometimes process faster for various reasons. This could be due to extra agency files unrelated to our tool or the dataset’s largest file not being “trips.txt” and going unused in the data processing section.

Table 9. The result of processing time study.

GTFS dataset	File Size (megabytes)	The processing time of the first attempt (seconds)	The processing time of the second attempt (seconds)	The processing time of the third attempt (seconds)	The average processing time (seconds)
Adelaide Metro, Australia	10.00	57.61	58.69	62.19	59.50

Metro Transit, Minneapolis, USA	9.80	44.58	43.77	46.88	45.08
BC Transit, Canada	9.50	35.97	42.91	38.67	39.17
Regione Piemonte, Italy	9.10	52.17	55.40	53.87	53.81
GO Transit, Toronto, Canada	7.70	48.14	46.01	46.25	46.80
Santiago DPTM, Chile	7.30	53.37	57.66	55.74	55.59
Poznań, Poland	6.60	32.08	34.87	35.21	34.05
AMB Mobilitat, Spain	5.90	22.94	24.89	22.01	23.28
Auckland Transport, New Zealand	5.60	25.99	28.46	24.58	26.34
CDTA, Albany, NY, USA	4.70	29.20	27.79	27.82	28.27
Autolinee Varesine, Varese VA, Italy	4.30	26.87	31.84	31.86	30.19
Action Buses, Canberra, Australia	3.70	36.79	36.71	37.40	36.97
Golden Gate Transit, USA	3.00	12.89	14.21	13.06	13.39
Metz, France	2.70	13.06	13.14	13.22	13.14
LSL, Finland	2.40	9.93	12.38	10.60	10.97
Tuvisa-EuskoTran, Spain	2.10	11.41	11.37	11.09	11.29
Bibus, Brest, France	1.60	17.44	17.48	18.32	17.75
Brampton Transit, Ontario, Canada	1.10	9.06	9.24	9.08	9.13
Bogor Angkots, Indonesia	0.90	9.77	9.74	9.50	9.67
Nagaibus, Japan	0.30	3.51	3.72	3.42	3.55

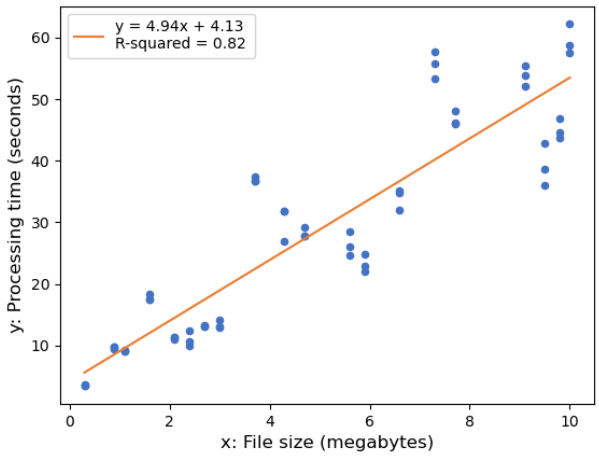


Figure 16. Relationship between the processing time and the input GTFS file size is characterized by a linear regression, $y = 4.9x + 4.13$ with a correlation, $R^2 = 0.82$.

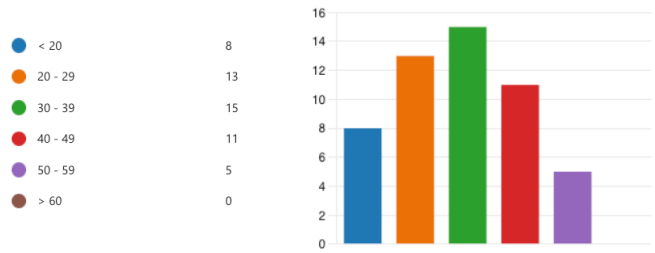
Usability. To assess G2Viz's usability, we conducted a user study involving 52 participants, including students, academics, and professionals in transportation, urban planning, and IT. We utilized an online survey accessible through the control panel section. After using the tool, each participant completed a questionnaire based on the Theory of Four Elements of User Experience (Guo, 2012). The questionnaire included four statements about the user's experience with the tool, accompanied by an open question for suggestions and comments. Participants rated their level of agreement on a 5-point Likert scale for each statement:

1. It is easy to use.
2. It is useful.
3. It is easy to start using.
4. It is fun and engaging.

The study included a diverse group of participants comprising 29 males and 23 females, representing various age and occupation groups. Distributions of age and occupation are shown in Figs. 17(a) and 17(b) respectively. Rating result is shown in Fig. 17(c), where being 'fun and engaging' received the highest rating (4.56) followed by 'useful' (4.50) and 'easy to use' (4.13), while 'easy to start using' received the lowest rating (3.48).

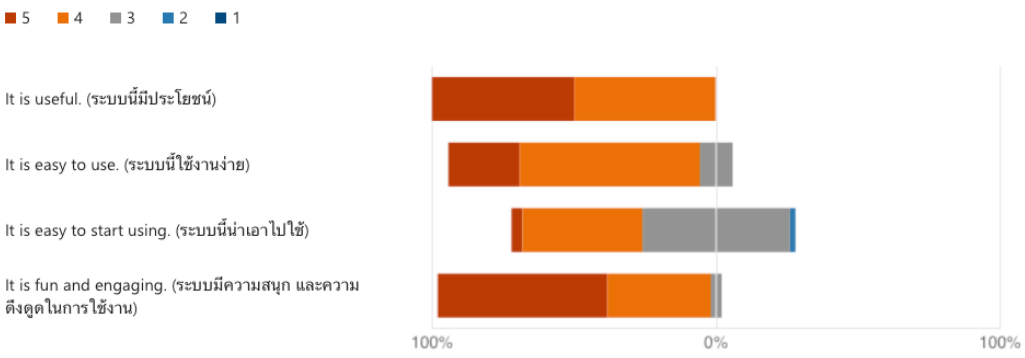
Users provided valuable feedback on the tool, acknowledging its usefulness and graphical representation. Many users expressed the need for a user manual or a read-me page explaining how to use the tool effectively. Their comments also included suggestions for additional features, such as exporting numerical data, comparing different transit planning models, generating GTFS files, conducting cause-and-effect analyses, supporting urban system optimization, and offering real-time transit performance monitoring.

These comments collectively highlight the tool's effectiveness in transportation and urban planning while indicating areas for improvement. Addressing the need for a user manual and incorporating features related to urban planning and transit system optimization are essential steps for enhancement. Additionally, exploring the possibility of a feature or system for direct editing of GTFS feeds or their generation from the design stage presents potential avenues for G2Viz's future development.



(a) Age group distribution

(b) Occupation distribution



(c) Rating result

Figure 17. Usability study results.

4.2 G2Viz use case: transit performance comparison

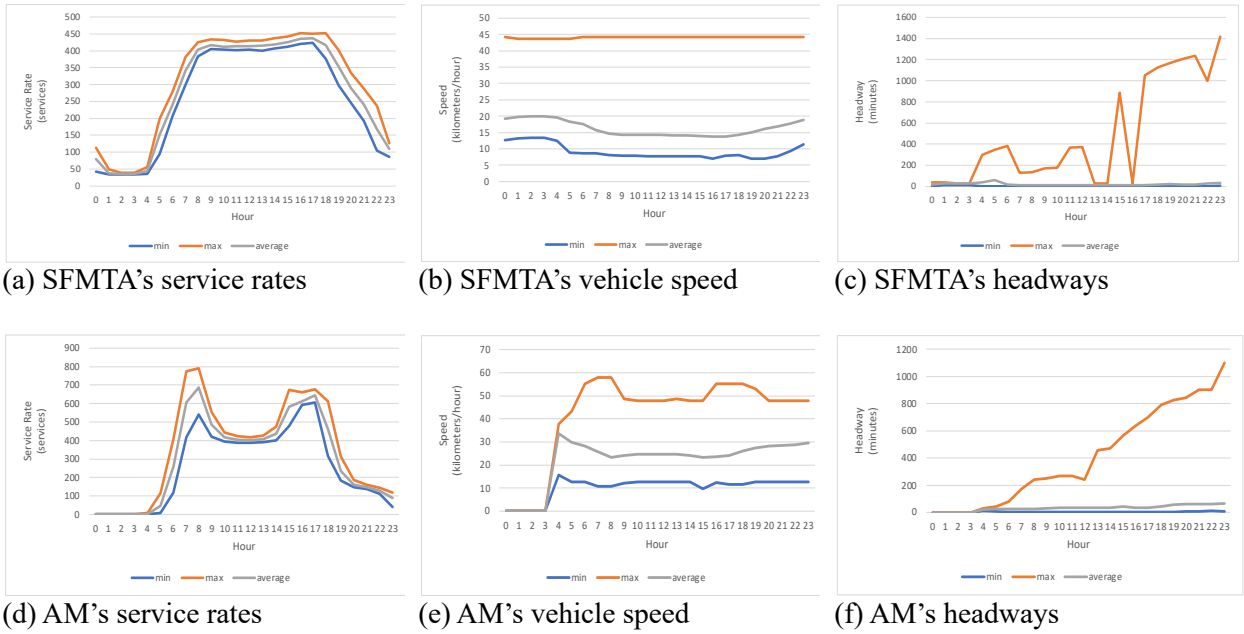
In addition to visualizing and analyzing performance of a particular city's transit system, the user may use G2Viz to conduct a simple comparative analysis between cities. Comparing transit performance allows cities to benchmark their performance against other cities and identify areas for improvement. Comparative analysis helps promote continuous improvement, foster innovation, make informed decisions, and enhance the overall quality and effectiveness of transit systems.

As an example, a comparative analysis was conducted between Adelaide Metro (AM) in Australia and the San Francisco Municipal Transportation Agency (SFMTA) in the United States. The analysis focused on the services provided on Monday, August 1, 2022. The dataset for AM was extensive, measuring 10.5 megabytes, and included 7,756 stops, 578 routes, covering an approximate area of 4,838.57 km^2. On the other hand, the dataset for SFMTA was 6.9 megabytes, encompassing 3,273 stops, 64 routes, and spanning around 211.97 km^2. This comparative analysis aimed to examine and compare different aspects of the transit services provided by these two transportation agencies.

Notably, Fig. 18 uncovers distinct performance characteristics between the two transit systems, as the results illustrate. Figures 18(a) and 18(d) present strikingly different service rate patterns between SFMTA and AM. For AM, there emerge two clear peak periods. Between 5:00 and 8:00, the service rate undergoes a rapid surge, leaping from 48 to 688 services within a mere three hours. The peak is evident at 8:10, registering 790 services. Subsequently, from 18:00 to 20:00, the service rate sharply declines, dropping from 464 to 162 services, followed by a gradual decrease. This trend aligns with the expected pattern of higher passenger numbers during peak commuting hours, followed by a decline in the evening. In contrast, SFMTA's service rate ascends from the early morning hours and maintains a consistently high level throughout the day, reaching its zenith at 16:00 with 790 services. This pattern is trailed by a decline in the evening. One plausible explanation for this observation could be attributed to tourism. San Francisco's global renown, iconic landmarks, cultural offerings, and its status as a major global tech and business hub make it a sought-after destination for international visitors. As a result, the sustained high service rates throughout the day for SFMTA may be linked to tourism, setting it apart from the peak commute hours observed in AM.

When examining speed (as seen in Figs. 18(b) and 18(e)), AM displays two peak hours during which top speeds reach around the mid-50s km/hr. However, its average and minimum speeds decrease during these peak hours, likely due to substantial traffic congestion. In contrast, SFMTA maintains lower yet more consistent speed levels throughout the day. One plausible explanation for this contrast is the presence of dedicated transit lanes and speed regulations in San Francisco, enabling transit vehicles to avoid general traffic and resulting in more uniform speed levels. Furthermore, SFMTA's maximum speed closely aligns

with the typical speed limits in San Francisco, around 25 miles per hour or approximately 40.23 kilometers per hour. While AM also offers transit lanes for buses, taxis, cyclists, and emergency vehicles, the maximum speed, as shown in Fig. 18(e), follows a distinct pattern. This suggests that AM experiences two speed peaks during commuting hours, while the maximum speed data during off-peak hours align closely with the speed limit in urban areas, typically limited to 50 kilometers per hour. This trend may be attributed to the increased number of services operating during peak hours. Consequently, certain routes providing services outside urban areas, where the speed limit is 100 kilometers per hour, may need to accelerate to accommodate the heightened demand for services during these hours.



(a) SFMTA's service rates (b) SFMTA's vehicle speed (c) SFMTA's headways

(d) AM's service rates (e) AM's vehicle speed (f) AM's headways

Figure 18. Transit performance comparison between the San Francisco Municipal Transportation Agency (SFMTA) and the Adelaide Metro (AM) in terms of the service rate ((a) and (d)), speed ((b) and (e)), and headway ((c) and (f)).

When examining the headway aspect, AM, as shown in Fig. 18(f), demonstrates a continuous increase in maximum headway, starting from 30 minutes in the morning and extending to a substantial 1,101 minutes in the evening. This indicates longer waiting times for passengers or the availability of extended journeys throughout the day, reflecting the broader service coverage compared to SFMTA. On the contrary, SFMTA, depicted in Fig. 18(c), exhibits a fluctuating pattern in its maximum headway. Notably, it presents shorter headway durations during peak commuting hours, aligning with the increased service rate observed between 5:00 and 7:00 in the morning. Subsequently, the headway tends to remain elevated after working hours, corresponding to the decrease in service rate after 18:00. Moreover, there is a disparity in the headway distribution between the two cities. AM demonstrates a continuous rise in headway, ranging from 22 to 66.84 minutes. Conversely, SFMTA maintains a more consistent headway, spanning from 12 to 40 minutes throughout the day, even though it experiences an increase to 62.44 minutes at 5:00. This suggests that SFMTA may uphold a more efficient and uniform schedule compared to AM.

In summary, this comparative analysis effectively highlights the advantages of our G2Viz transit performance analytics in comprehending and contrasting the inherent attributes of transit systems through their GTFS feeds. This analytical tool streamlines the planning and evaluation of transit systems, providing valuable insights into transit behaviors and facilitating well-informed decision-making.

5. Conclusion

This paper introduces G2Viz, a public transit operation visualizer, and its development process. The development stages encompass requirement gathering, planning, and design, which include aspects such as software architecture, data models, user interfaces, and system components. Subsequently, rigorous implementation and testing were conducted to ensure the tool's functionality and effectiveness.

G2Viz is specifically designed to measure and dynamically visualize public transit operations by utilizing General Transit Feed Specification (GTFS) data. As a web application, it offers universal accessibility to users worldwide through any web browser, ensuring cross-platform compatibility across various devices and operating systems. Serving as a versatile tool, G2Viz fosters seamless communication among transit agencies, users, researchers, and city authorities. Its capabilities empower transit planners to make well-informed decisions regarding public transportation.

G2Viz offers transit agencies insights into past performance for better resource allocation. Policy makers benefit from evidence-based decisions and long-term planning. Citizens gain understanding of historical transit trends, fostering accountability and aiding in planning. This tool enables agencies to optimize services, helps policy makers justify investments, and empowers citizens with knowledge for informed engagement, enhancing overall transit planning and accountability.

The development and implementation of our tool were not without challenges. Due to the complexity of processing and visualizing large datasets, there may be performance limitations when using this tool with very large transit datasets. The user may experience longer processing time or slower rendering speed when working with extensive datasets. To ensure optimal performance, the maximum file size is limited to 10 megabytes. The user is encouraged to use datasets within this size limit for the best experience. While efforts have been made to optimize the tool for mobile devices, it is important to note that the tool is not fully responsive and may not provide an optimal user experience on smaller screens. The tool's functionality and complexity may lead to a congested user interface on mobile devices. It is recommended to use the tool on larger screens or desktop devices for a more seamless experience.

While the current version of the tool has made notable strides in visualizing and analyzing transit data, there remain several areas that offer potential for exploration and enhancement. These include improving performance optimization, ensuring responsive design for mobile compatibility, incorporating advanced filtering and customization options, enabling data merging capabilities, and integrating real-time data. These areas represent promising directions for further improvement and development of G2Viz.

Funding

This work is funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grants (RGPIN/03037-2022). This work was supported in part by Chiang Mai University.

References

Aemmer, Z., Ranjbari, A., & MacKenzie, D. (2022). Measurement and classification of transit delays using GTFS-RT data. *Public Transport*, *14*, 263–285. https://doi.org/10.1007/s12469-022-00291-7

Andrienko, G., Andrienko, N., Chen, W., Maciejewski, R., & Zhao, Y. (2017). Visual analytics of mobility and transportation: State of the art and further research directions. *IEEE Transactions on Intelligent Transportation Systems*, *18*(8), 2232–2249. https://doi.org/10.1109/TITS.2017.2683539

Anwar, A., Odoni, A., & Toh, N. (2016). BusViz: Big Data for Bus Fleets. *Transportation Research Record: Journal of the Transportation Research Board*, *2544*(1), 102–109. https://doi.org/10.3141/2544-12

Berkow, M., El-Geneidy, A. M., Bertini, R. L., & Crout, D. (2009). Beyond Generating Transit Performance Measures: Visualizations and Statistical Analysis Using Historical Data.

Transportation Research Record: Journal of the Transportation Research Board, *2111*, 158–168.
Chen, W., Guo, F., & Wang, F. Y. (2015). A Survey of Traffic Data Visualization. *IEEE Transactions on Intelligent Transportation Systems*, *16*, 2970–2984. https://doi.org/10.1109/TITS.2015.2436897
Demissie, M.G., Kattan, L., Phithakkitnukoon, S., Homem de Almeida Correia, G., Veloso, M., & Bento, C. (2020). Modeling Location Choice of Taxi Drivers for Passenger Pick-Up Using GPS Data. *IEEE Intelligent Transportation Systems Magazine*, *13*, 70–90. https://doi.org/10.1109/MITS.2020.3014099
Demissie, M.G., Phithakkitnukoon, S., & Kattan, L. (2019). Trip Distribution Modeling Using Mobile Phone Data: Emphasis on Intra-Zonal Trips. *IEEE Transactions on Intelligent Transportation Systems*, *20*(7). https://doi.org/10.1109/TITS.2018.2868468
Demissie, Merkebe Getachew, & Kattan, L. (2022a). Estimation of truck origin-destination flows using GPS data. *Transportation Research Part E: Logistics and Transportation Review*, *159*(102621). https://doi.org/https://doi.org/10.1016/j.tre.2022.102621
Demissie, Merkebe Getachew, & Kattan, L. (2022b). Understanding the temporal and spatial interactions between transit ridership and urban land-use patterns: an exploratory study. *Public Transport*, *14*, 385–417. https://doi.org/https://doi.org/10.1007/s12469-022-00296-2
Deng, X., Chen, W., Zhou, Q., Zheng, Y., Li, H., Liao, S., & Biljecki, F. (2023). Exploring spatiotemporal pattern and agglomeration of road CO2 emissions in Guangdong, China. *Science of the Total Environment*, *871*(May), 162134. https://doi.org/10.1016/j.scitotenv.2023.162134
Devunuri, S. (2024). *gtfs-segments* (2.1.1). GitHub. https://pypi.org/project/gtfs-segments
ESRI. (2023). *ArcGIS Pro*. Environmental Systems Research Institute. https://pro.arcgis.com/en/pro-app/latest/get-started/get-started.htm
Fry, B., & Reas, C. (2023). *Processing*. GitHub. https://github.com/benfry/processing4/
Ge, L., Sarhani, M., Voß, S., & Xie, L. (2021). Review of transit data sources: Potentials, challenges and complementarity. *Sustainability (Switzerland)*, *13*(20), 11450. https://doi.org/10.3390/su132011450
Glick, T. B., Feng, W., Bertini, R. L., & Figliozzi, M. A. (2015). Exploring Applications of Second-Generation Archived Transit Data for Estimating Performance Measures and Arterial Travel Speeds. *Transportation Research Record: Journal of the Transportation Research Board*, *2538*, 44–52. https://doi.org/10.3141/2538-06
Godfrid, J., Radnic, P., Vaisman, A., & Zimányi, E. (2022). Analyzing public transport in the city of Buenos Aires with MobilityDB. *Public Transport*, *14*, 287–321. https://doi.org/10.1007/s12469-022-00290-8
Guido, G., Vitale, A., & Rogano, D. (2016). Assessing public transport reliability of services connecting the major airport of a low density region by using AVL and GIS technologies. *International Conference on Environment and Electrical Engineering (EEEIC 2016)*, 1–5. https://doi.org/10.1109/EEEIC.2016.7555483
Guo, F. (2012). *More than usability: The four elements of user experience, part IV*. http://www.uxmatters.com/mt/archives/2012/04/more-than-usability-the-four-elements-of-user-experience-part-i.php
Herszenhut, D., Pereira, R. H. M., Andrade, P. R., & Joao Bazzo, I. (2023). *Introduction to gtfstools*. https://cran.r-project.org/web/packages/gtfstools/vignettes/gtfstools.html
Ji, Y., Mishalani, R. G., & McCord, M. R. (2015). Transit passenger origin-destination flow estimation: Efficiently combining onboard survey and large automatic passenger count datasets. *Transportation Research Part C: Emerging Technologies*, *58*(B), 178–192. https://doi.org/10.1016/j.trc.2015.04.021
Kim, J., Rapuri, S., Chuluunbaatar, E., Sumiyasuren, E., Lkhagvasuren, B., Budhathoki, N. R., & Laituri, M. (2023). Developing and evaluating transit-based healthcare accessibility in a low- and middle-income country: A case study in Ulaanbaatar, Mongolia. *Habitat International*, *131*, 102729.
Kim, Y., Kim, J., Ha, H. J., Nakajima, N., & Lee, J. (2022). Job Accessibility as a Lens for Understanding the Urban Structure of Colonial Cities: A Digital Humanities Study of the Colonial Seoul in the 1930s Using GIS. *ISPRS International Journal of Geo-Information*, *11*(12), 614.

https://doi.org/10.3390/ijgi11120614
Kim, Y., Lee, J., Kim, J., & Nakajima, N. (2021). The Disparity in Transit Travel Time between Koreans and Japanese in 1930s Colonial Seoul. *Findings*, *July*. https://doi.org/10.32866/001c.25226
Kinjarapu, A., Demissie, M. G., Kattan, L., & Duckworth, R. (2021). Applications of Passive GPS Data to Characterize the Movement of Freight Trucks: A Case Study in the Calgary Region of Canada. *IEEE Transactions on Intelligent Transportation Systems*, *23*, 9210–9225. https://doi.org/10.1109/tits.2021.3093061
Kujala, R. (2020). *gtfspy-webviz*. GitHub. https://github.com/CxAalto/gtfspy-webviz
Kunama, N., Worapan, M., Phithakkitnukoon, S., & Demissie, M. (2017). GTFS-VIZ: Tool for preprocessing and visualizing GTFS data. *Adjunct Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers (UbiComp/ISWC 2017)*, 388–396. https://doi.org/10.1145/3123024.3124415
Kurkcu, A., Miranda, F., Ozbay, K., & Silva, C. T. (2017). Data visualization tool for monitoring transit operation and performance. *5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS 2017)*, 598–603. https://doi.org/10.1109/MTITS.2017.8005584
Li, D., Lin, Y., Zhao, X., Song, H., & Zou, N. (2011). Estimating a Transit Passenger Trip Origin-Destination Matrix Using Automatic Fare Collection System. *Database Systems for Adanced Applications (DASFAA 2011)*, 502–513. https://doi.org/10.1007/978-3-642-20244-5_48
Ma, X., & Wang, Y. (2014). Development of a Data-Driven Platform for Transit Performance Measures Using Smart Card and GPS Data. *Journal of Transportation Engineering*, *140*(12), 04014063. https://doi.org/10.1061/(ASCE)TE.1943-5436.0000714
Mazloumi, E., Currie, G., & Rose, G. (2009). Using GPS Data to Gain Insight into Public Transport Travel Time Variability. *Journal of Transportation Engineering*, *136*(7), 623–631. https://doi.org/10.1061/(asce)te.1943-5436.0000126
McKinney, W. (2011). pandas: a Foundational Python Library for Data Analysis and Statistics. *Python for High Performance and Scientific Computing*, *14*(9), 1–9.
Mesbah, M., Currie, G., Lennon, C., & Northcott, T. (2012). Spatial and temporal visualization of transit operations performance data at a network level. *Journal of Transport Geography*, *25*, 15–26. https://doi.org/10.1016/j.jtrangeo.2012.07.005
Mueller, M. (2014). *gtfs-visualizations*. GitHub. https://github.com/cmichi/gtfs-visualizations
National RATP. (2024). *GTFS Builder Guidebook*.
Pereira, R. H. M., Andrade, P. R., & Vieira, J. P. B. (2023). Exploring the time geography of public transport networks with the gtfs2gps package. *Journal of Geographical Systems*, *25*, 453–466. https://doi.org/10.1007/s10109-022-00400-x
Pereira, R. H. M., Saraiva, M., Herszenhut, D., Braga, C. K. V., & Conway, M. W. (2021). R5r: Rapid Realistic Routing on Multimodal Transport Networks with R5 in R. *Transport Findings*, *March*. https://doi.org/doi.org/10.32866/001c.21262
Phithakkitnukoon, S., Hankaew, S., Demissie, M. G., Smoreda, Z., & Ratti, C. (2022). Temporary Migration Flow Inference and Analysis From Perspective of Mobile Phone Network Data. *IEEE Access*, *10*, 23248–23258. https://doi.org/10.1109/ACCESS.2022.3154485
Phithakkitnukooon, S., Patanukhom, K., & Demissie, M. G. (2021). Predicting spatiotemporal demand of dockless e-scooter sharing services with a masked fully convolutional network. *ISPRS International Journal of Geo-Information*, *10*(11), 773. https://doi.org/10.3390/ijgi10110773
Prommaharaj, P., Phithakkitnukoon, S., Demissie, M. G., Kattan, L., & Ratti, C. (2020). Visualizing public transit system operation with GTFS data: A case study of Calgary, Canada. *Heliyon*, *6*(4), e03729. https://doi.org/10.1016/j.heliyon.2020.e03729
Toso, S., & Oja, R. (2023). *gtfs_functions*. GitHub. https://github.com/Bondify/gtfs_functions