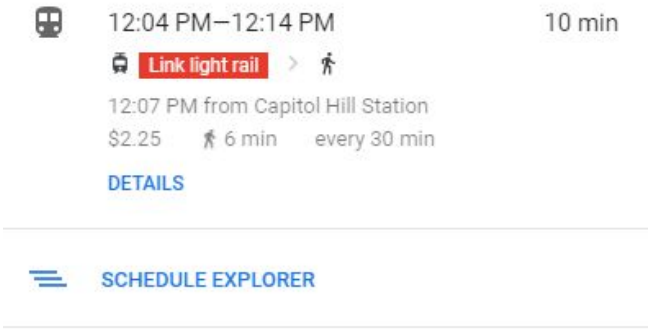
Where's My Bus At? (Transit Vis)

Zack Aemmer, Kelly Balmes, Alex Goldstein, Steven Wilson

Intro

- Bus delays have an element of chaos and are somewhat unpredictable, but aggregated data often reveals troublesome locations and routes
- Google's Real-Time General Transit Feed Specification (GTFS-RT) has created a standardized format built on top of the GTFS schedule specification for tracking location data in any bus system
- We have developed an interactive route-delay mapping tool for the King County Metro transit system that interfaces with the GTFS-RT data collected by OneBusAway
- Our tool is generalizable to any transit system operating on GTFS-RT, where the only prerequisite is a SQL database containing scraped trip location data

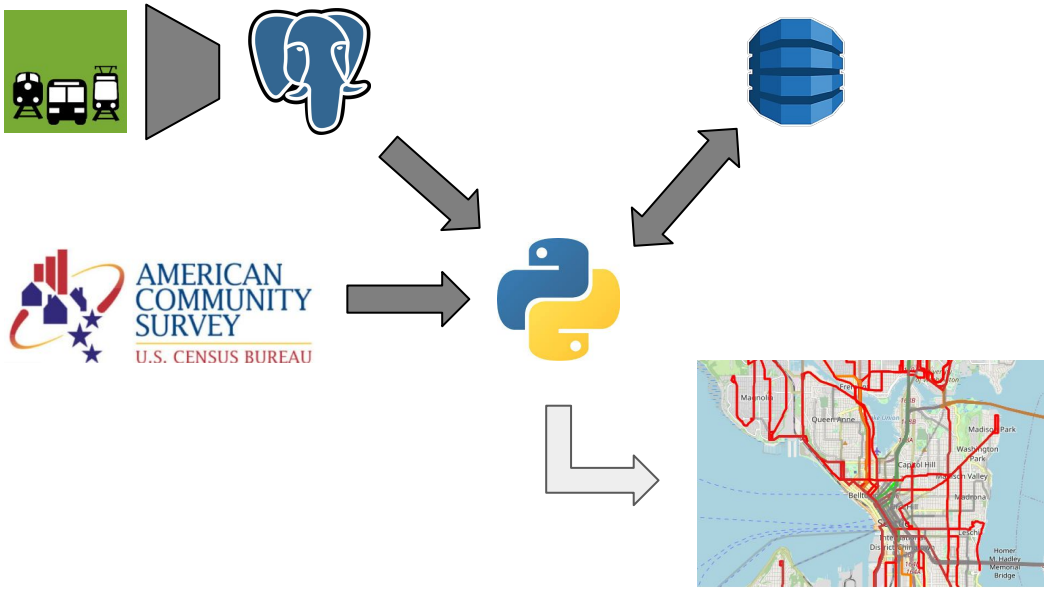


Users and Use Cases

- Planners, Engineers, Data Scientists
    - Identify locations/routes with low transit speeds
    - Aid in drawing conclusions on relationships between transit speeds and socioeconomic variables
    - Set up backend for automated aggregation of transit data
- Community members
    - Identify locations/routes to avoid when moving to a new neighborhood
    - Answer questions about network-wide transit reliability that officials don't always make available

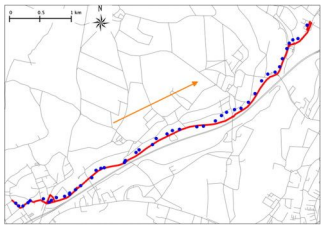
Functional Specification - Data Sources

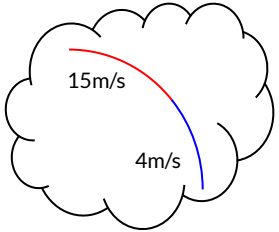
- Bus Delays
  - Relational database (RDS) filled with bus position data scraped from OneBusAway GTFS-RT
  - NoSQL database (DynamoDB) filled with daily summarized average speeds
- Socioeconomic Census Data
  - American Community Survey (ACS) data from:
    - *Table s0801: Commuting Characteristics*
    - *Table s1902: 12-Month Mean Income*
- Shapefiles
  - TIGER census tracts shapefile
  - King County Metro bus routes shapefile

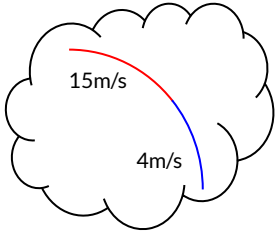


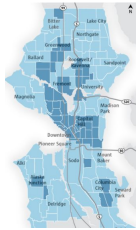
Functional Specification - Components

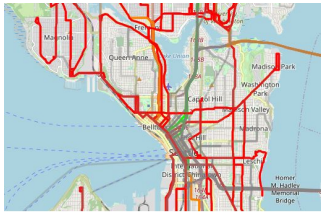
- Backend - Local/EC2
  a. Initialize_Dynamodb.py
     - Create database and upload KCM Routes
  b. Summarize_RDS.py
     - Query location data from RDS and calculate average speeds
     - Upload to database
- Frontend - Local
  a. Transit_Vis.py
     - Handles database I/O
     - Primary functions for drawing together data sources, creating Folium map, and displaying it
     - Interface for user to create the map from the terminal











Tour of Software - Project Structure

Repository Files:

- New from last time: moved src, test, data into new “transit_vis” directory

Included Files

```
transit_vis/
  |- README.md
  |- LICENSE.md
  |- widget_generate_transit_vis_map.ipynb
  |- transit_vis/
     |- src/
        |- initialize_dynamodb.py
        |- summarize_rds.py
        |- transit_vis.py
        |- create_gtfs_tables.sql
     |- tests/
        |- test_transit_vis.py
        |- test_backend_helpers.py
        |- data/
           |- kcm_routes.geojson
           |- ...
     |- data/
        |- kcm_routes.geojson
        |- s0801.csv
        |- s1902.csv
        |- seattle_census_tracts.geojson
  |- docs/
     |- component_specification.md
     |- functional_specification.md
```

Generated Files:

- New from last time: added descriptions in readme

Generated Files

Created in the data folder during tool operation:

- **kcm_routes_w_speeds_tmp.geojson:** A shapefile with added properties containing the speed data from the most recent run of the tool
- **seattle_census_tracts_2010_tmp.csv:** A data file containing the combined s0801 and s1902 census tables
- **google_transit.zip/google_transit:** A zip file and extracted folder containing the most up to date GTFS (tripids, routeids, stopids, etc.) information from King County Metro
- **kcm_routes_histogram.png:** An image file that shows the distribution of transit speeds for the entire network from the most recent run.

Created in the top-level folder during tool operation:

- **output_map.html:** The final result from the most recent run which can be viewed in any web browser.
- **output_map_widgets.html:** The final result from the most recent run of the jupyter notebook which can be viewed in any web browser.

Tour of Software - Setup

Planners, Engineers, Data Scientists

- If interested in collecting transit data:
  - python -m transit_vis.src.initialize_dynamodb
  - python -m transit_vis.src.summarize_rds
- Run main script: python -m transit_vis.src.transit_vis

```
transit_vis — -zsh — 103×24
(transit_vis_py385) kellybalmes@Kellys-MacBook-Pro transit_vis % python -m transit_vis.src.transit_vis
Modifying and writing census data...
Connecting to dynamodb...
Getting speed data from dynamodb...
Writing speed data to segments for visualization...
Generating map...
Saving map...
Map saved, please copy this file path into any browser: file:///Users/kellybalmes/Desktop/CSE_583/cse58
3/project/transit_vis/output_map.html
(transit_vis_py385) kellybalmes@Kellys-MacBook-Pro transit_vis %
```

Community members

- Obtain AWS credentials
- Run jupyter notebook widget_generate_transit_vis_map.ipynb

Home: 47.653834, -122.307858

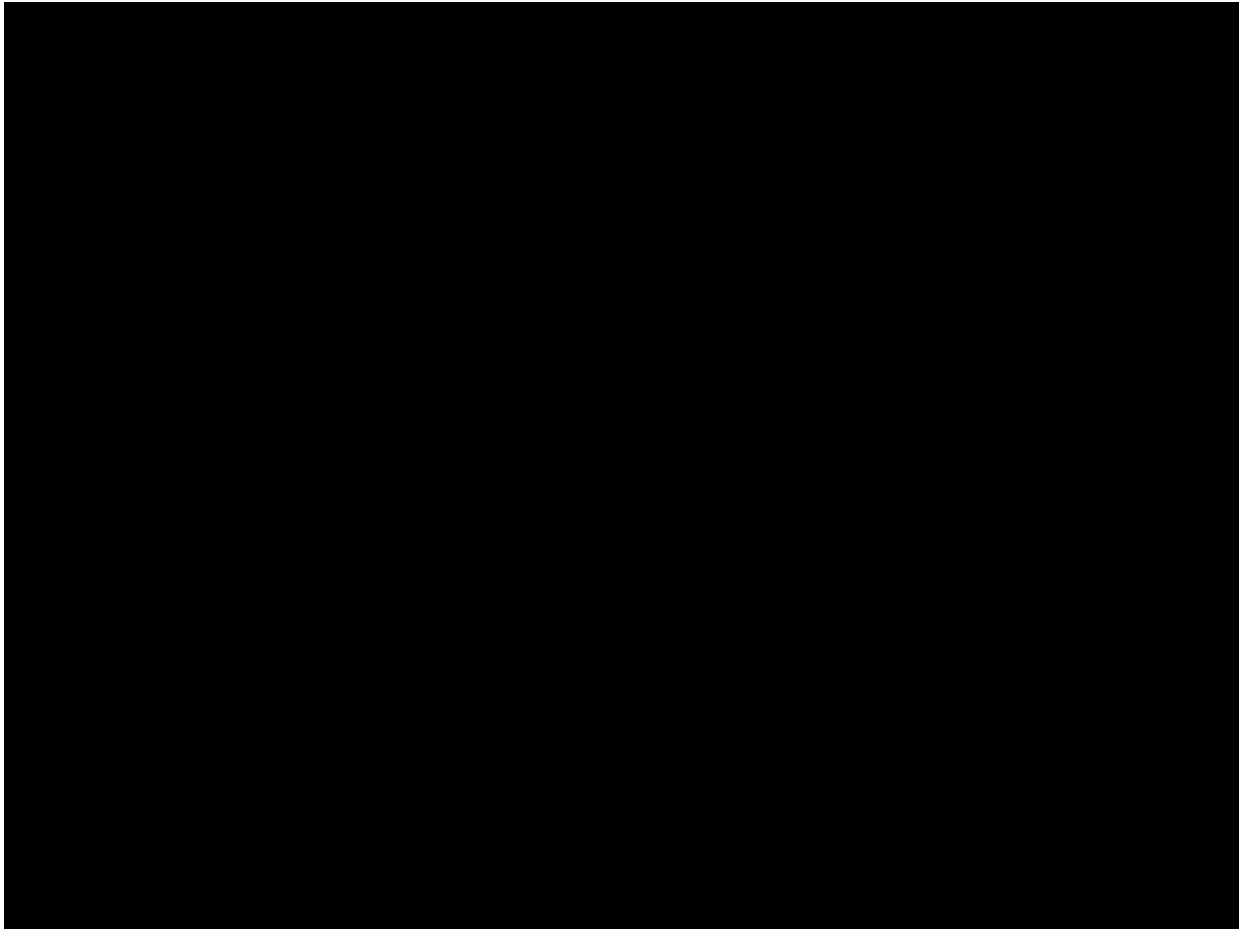
Destination: 47.606209, -122.332069

Income Minimum: Enter minimum yearly income ($)

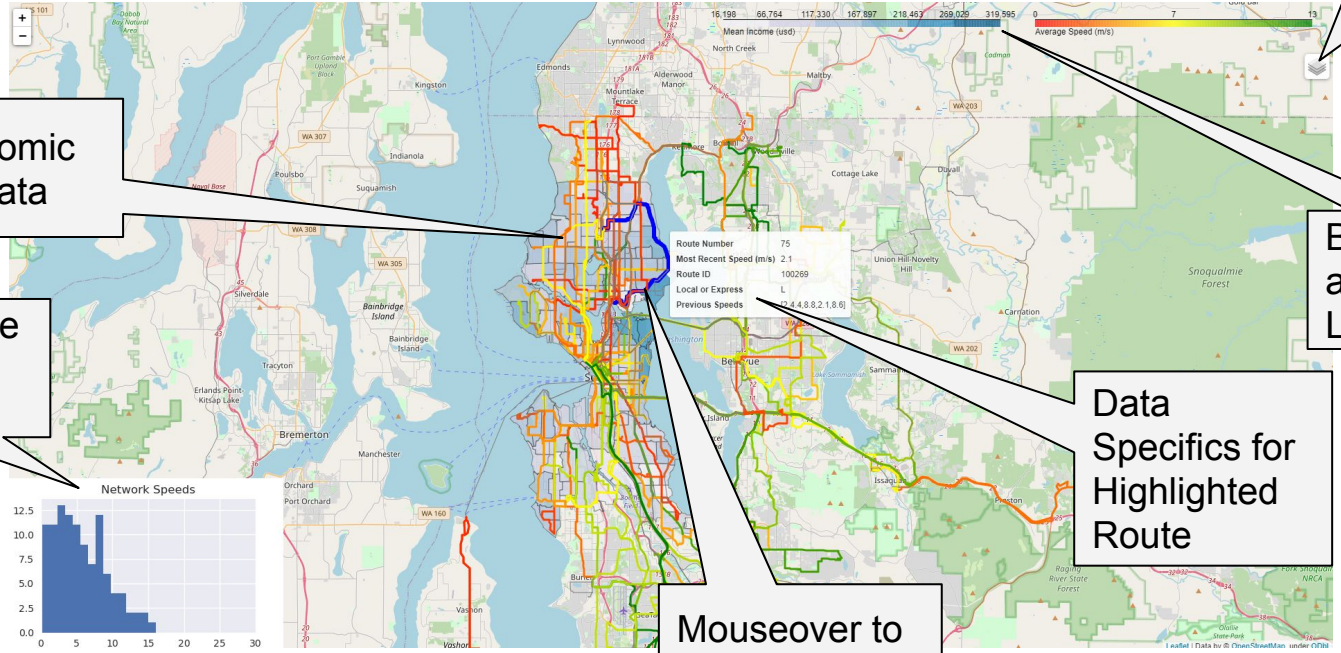
Income Maximum: Enter maximum yearly income ($)

Generate Map

Tour of Software - Operation Demo



Tour of Software - Map Components



Layer Toggle

Socioeconomic and Bus Data

Bus Speed and Income Legends

Network-Wide Speed Distribution

Data Specifics for Highlighted Route

Mouseover to Highlight a Specific Route

Tour of Software - ipynb User Interface

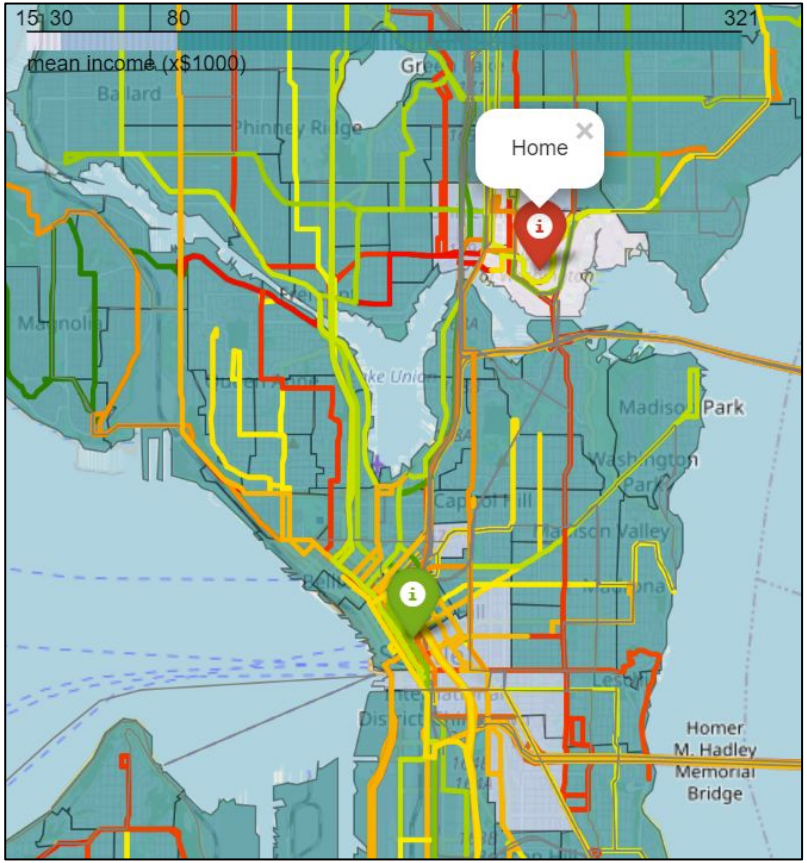
Home: 47.653834, -122.307858

Destination: 47.606209, -122.332069

Income Minimum: Enter minimum yearly income ($)

Income Maximum: Enter maximum yearly income ($)

Generate Map



- Made with Ipywidget package
- Allows creation of objects for filtering from inputs
- Generates a folium map with markers for user's home and destination of interest; changes the tile colors based off income upper and lower limits

Unit Tests

From terminal: coverage run -m unittest discover

- Smoke, One-Shot, and Edge tests for all functions that do not include database I/O

- Coverage for:
  - Constructing map elements
  - Drawing together data and generating map
  - Helper functions in backend modules

- Separate data folder for simulating local file I/O while testing

Lessons Learned

- Github Security (AWS Keys)
- Package Capability/Simplicity Tradeoff (Folium Widgets/Interactivity/Secondary software installs)
- OS Compatibility (Mac vs PC)
- Writing Tests for File I/O (Separate Testing Data Files)
- Limitations in Processing Large Amounts of Data with Free-Tier Cloud Hardware (Run Locally)
- Github Upload Limits (Keep Data in Cloud)