Pedaling Towards Progress

Analyzing Washington, DC's Bikesharing system using Open-source tools

Maxwell Lindsay

Van Oord

June 23, 2023

Introduction to CaBi

- Memberships are available for $95 a year for unlimited trips under 45 minutes [1]. Casual users can pay $1 to unlock the bike, then $0.05 per minute.
- Primarily intended for short trips
- Docked bikeshare
- 700+ docks, 35 million trips (so far)

* recently, non-docked E-bikes that have been introduced

The data

Capital Bikeshare publishes the following data about each trip on a monthly basis:

- start time
- end time
- start station index
- end station index
- Membership status of the rider (Member or Non-member?)

These data are available as csv files per month or year from Capital Bikeshare at https://ride.capitalbikeshare.com/system-data

My goal

- Analyze all trips up to the present day
- Street-level estimates of bikeshare use
- Inspired by A blog post from Daniel J patterson from 2020

Overview of my approach

1. Download CSVs of every trip
2. Parse and normalize the CSVs using Pandas
3. Find The number of trips between unique pairs of stations
4. Build Valhalla routing tiles
5. Find a route between each station pair using Valhalla
6. Aggregate the trip statistics across every single route

Download CSVs of every trip

Parse and normalize the CSVs using Pandas

- Longer than a 4 hours
  This is an arbitrary threshold. However, the CaBi system is primarily intended for short trips, and the pricing reflects this. The bikes can be used for leisure and tourism, but they are priced to encourage users to change bikes regularly. Also, for the purposes of this project, long trips are less likely to have
- Starting and ending at the same station
  For obvious reasons, cannot easily be routed
- Stations with an invalid start or end station
  A number of trips in the dataset are missing a value for the start or ending point

Find unique trips between two stations

Building a Valhalla Routing network

- Download OSM tiles
- Build Valhalla routing network
- Use Pandas the start and end point of every unique trip

Build PostGIS topology at the same time

- Using PLpgSQL triggers, add a corresponding entry to the topogeometry table for every new entry to the routes tables

Sum up the trips on every topological element

- Using an SQL query, we can sum the trips on every unique street

Lessons learned

- Routing is *much* faster if you run it locally as binaries rather than a web api from a container
- Valhalla allows a like of parameters for bike routing and is lightning fast

Shoutouts

- GIS-OPS for their thoughtful and intuitive Valhalla Docker containers
-