Batch Analysis of Bike Share Trip Data Using OpenTripPlanner

James C. Wong, E.I.T.
OpenPlans

June 4, 2012

1 Introduction

The OpenTripPlanner (OTP) Project is an open source collaboration led by OpenPlans (www.openplans.org) to develop a multimodal trip planner for use by transit agencies, regional MPOs and other interested parties. As an open source project, the software is free to use and can be updated by anyone. OTP relies on two primary local data sources: a General Transit Feed Specification (GTFS) file and map data from Open Street Map (another open source project). While this tool has been developed primarily as a user-facing tool to assist travelers using transit, it has the potential to simultaneously provide a powerful analytic platform for planners.

This exercise is a proof of concept to demonstrate the ability to generate alternative transit/walk/bike itineraries for a series of trips that have already occurred on a particular mode. The Capital Bikeshare (CaBi) system in the Washington region provides a robust dataset of origins, destinations, departure time and trip duration for every trip taken. Using a current (May 2012) GTFS feed for the Washington Metropolitan Area Transit Authority (WMATA) and a dataset from Q1 of the same year, we developed a series of scripts to quickly generate the alternative statistics.

The overall framework consists of a raw CaBi trip dataset, a python script which queries an instance of OTP and some basic R code to manipulate and present the data. The python script is the core feature of this documentation as it is the mechanism used to generate and record metrics from alternative itineraries from each origin-destination pair in the CaBi dataset.

2 Methodology

2.1 Origin-Destination Data and OTP Dependencies

The origin-destination (O-D) dataset used for this exercise was from Washington, DC's CaBi system which publishes statistics about every trip taken on the system. Any O-D data can be used but it needs the fields shown here which are used in the sample code for this exercise:

- Duration (sec)[1]
- Start Date [2]
- Start Time
- StartLat
- StartLon
- EndLat
- EndLon

This method relies on having access to an instance of OTP for the region of interest. Instances of OpenTripPlanner can be generated online by using OTP Deployer in conjunction with a well-formed GTFS

[1]This is the only field that does not need to be in place to run the API calls, but is important for any comparison of planned trips to trips actually taken. Different types of analysis may not require this value.

[2]The start date and time can be combined into a single column with different transforms. See the Plan API for details.

feed for the region of interest. The python script makes repeated calls to the OTP API and records it in an output file.

2.2 Python Batch Processing

The Python script `batch.py` is located on github. In short, the script reads O-D information from the input file, formats and calls a series of queries to the OTP API, and records the resultant itinerary metrics in a new output file. Most of the code has instruction comments, but a few key pieces will be described here.

2.2.1 Weekdays

GTFS feeds are only good for a specific time period. In the event that the current GTFS feed is not applicable for the timeframe of the O-D trips, this code creates a current date that corresponds to the same day of the week. This is a simplifying assumption that transit today is the same as transit when the O-D data was generated. (This is reasonable for Washington, DC because the CaBi data is from 2012Q1 and the GTFS is just a few months later.)

```
from datetime import date, datetime, timedelta
weekday = date.today().weekday()
monday = date.today() - timedelta(days=weekday)
...
#for each row
  date_new = datetime.strptime(date, "%m/%d/%Y")
  date_new = monday + timedelta(days=date_new.weekday())
  date_new = date_new.strftime("%m/%d/%Y")
...
```

2.2.2 Format URL

This section formats a URL which will then be called according to the API documentation for the plan function of OTP. It uses some information directly from the input file (like the O-D coordinates) and other parameters set for the exercise. For example, mode is set to `WALK` and `TRANSIT` becaues we were comparing transit trips to the bikeshare trips. In `batch.py` there's actually a second iteration of this for each O-D pair to retrieve both a transit/walk itinerary and a bike itinerary. Printing it in the terminal is useful for debugging.

```
...
#for each O-D pair read from the input file
  arriveBy = False
  params = {'time' : '%s' % time_new,
      'fromPlace' : '%s,%s' % (o_lat, o_lon),
      'toPlace' :  '%s,%s' % (d_lat, d_lon),
      'maxWalkDistance' : 2000,  #Arbitrary
      'mode' : 'WALK,TRANSIT',   #See OTP API Documentation
      'date' : date_new,
      'numItineraries' : 1,
      'arriveBy' : 'true' if arriveBy else 'false' }
  url = URL + urllib.urlencode(params)  #Creates a POST link using params
  req = urllib2.Request(url)
  req.add_header('Accept', 'application/json')
  print url
```

```
...
```

2.2.3 Record Itinerary Metrics

Once the URL is encoded, it is called and the result from OTP is an XML object filled with itineraries. We have limited the response to one itinerary but you could store and analyze up to three from using current OTP settings (can be increased in source code). See the documentation for details on what kinds of output metrics you can save for each response. These are stored in the row with the input data to finalize the dataset with both the incoming O-D pairs and the resulting information about the planned itineraries.

```
...
#after creating the URL each time
  content = response.read()    #Store response from OTP
  objs = json.loads(content)
  itineraries = objs['plan']['itineraries']
...
```

3 Analysis and Results

3.1 Loading Data in R

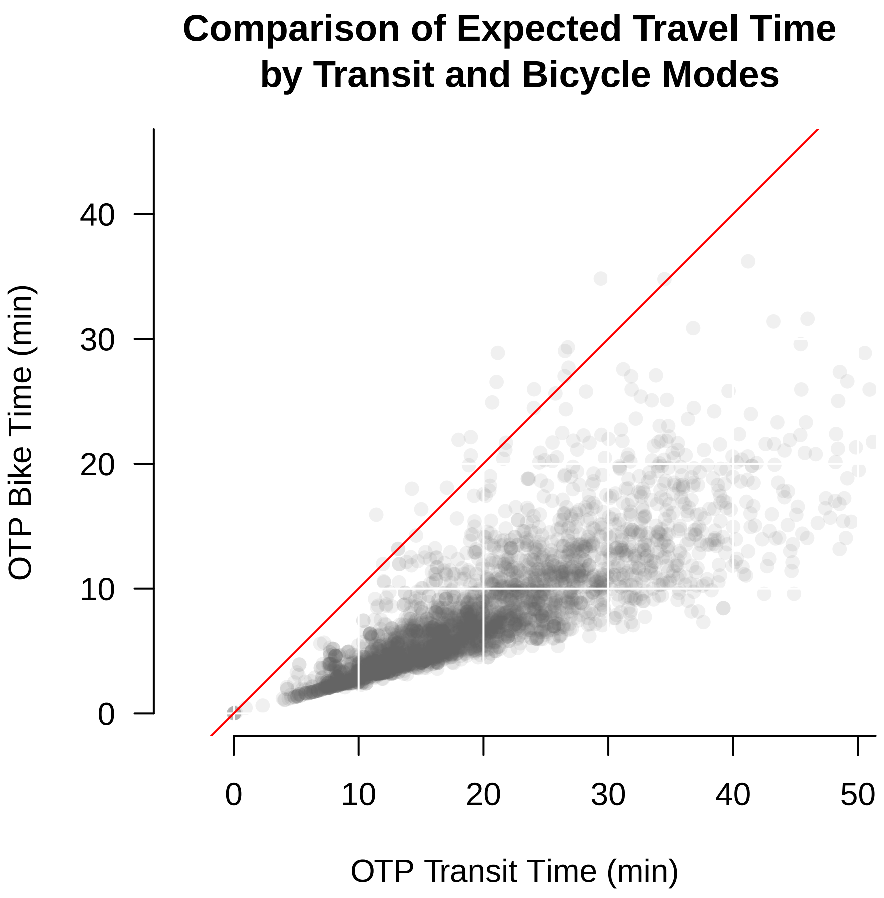
The Python code in the previous section produces an output file called `bikeBatchOut.csv` which we use for analysis within R. Begin by reading in the output file in R and calculate more useful columns based on the data for easier use in analysis. Most of the R functions shown here manipulate and reogranize the data for use in various graphics.

```
cabi <- read.csv(file = "bikeBatchOut-test.csv", header = TRUE)

cabi = transform(cabi, TotalTransitTime = WalkTime + TransitTime +
    WaitingTime)
cabi = transform(cabi, TotalTransitTime_mins = TotalTransitTime/60)
cabi = transform(cabi, CaBiTime_mins = CaBiTripTime/60)
cabi = transform(cabi, WalkTime_mins = WalkTime/60, TransitTime_mins = TransitTime/60,
    WaitingTime_mins = WaitingTime/60, PlannedBikeTime_mins = PlannedBikeTime/60,
    TotalTransitTripDist = TotalTransitTripDist/1609.344, PlannedBikeDist = PlannedBikeDist/1609.344)

cabi = transform(cabi, CaBi_mph = PlannedBikeDist/(CaBiTime_mins/60),
    PlannedBike_mph = PlannedBikeDist/(PlannedBikeTime_mins/60))

cabi = transform(cabi, BikeTimeDiff_percent = (PlannedBikeTime_mins -
    CaBiTime_mins)/PlannedBikeTime_mins, BikeTimeDiff_abs = (PlannedBikeTime_mins -
    CaBiTime_mins))

keep <- c("CaBiTripTime", "TotalTransitTripDist", "PlannedBikeTime",
    "PlannedBikeDist", "CaBiTime_mins", "TotalTransitTime_mins", "WalkTime_mins",
    "TransitTime_mins", "WaitingTime_mins", "PlannedBikeTime_mins", "BikeTimeDiff_percent")
names(subset(cabi, select = keep))

##  [1] "CaBiTripTime"          "TotalTransitTripDist"
##  [3] "PlannedBikeTime"       "PlannedBikeDist"
##  [5] "CaBiTime_mins"         "TotalTransitTime_mins"
##  [7] "WalkTime_mins"         "TransitTime_mins"
##  [9] "WaitingTime_mins"      "PlannedBikeTime_mins"
## [11] "BikeTimeDiff_percent"
```

3.2 Comparing Bike Share Trips with Transit Alternatives

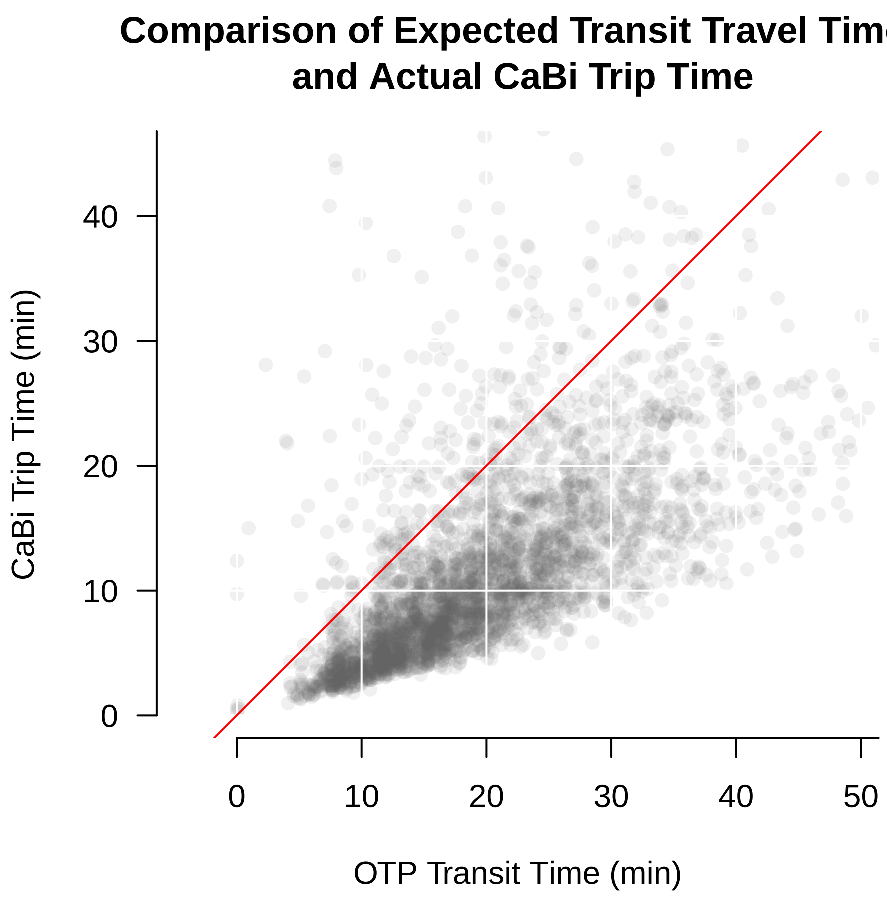
One of the primary questions that this exercise sought to explore was whether or not CaBi users are using bike share as an alternative to transit. We suspect this would largely occur for very short trips where the marginal time cost of accessing transit (walking to a station and waiting for a vehicle) would outweigh the speed of a transit vehicle over bike share. The first chart shown below helps to describe this relationship. A one percent sample of the CaBi O-D data was used to plan trips using transit and bicycle modes. If the predicted travel times were equivalent, they would fall along the red diagonal line. When a bike trip is predicted to be faster than its equivalent transit trip, it will fall below the line and vice versa. Based on the plot, we can see that most of the trips taken by CaBi riders are more efficient on bicycle than by transit. The distribution appears to favor those trips which have a more significant travel time difference; that is, fewer people take trips on CaBi that can be made by transit in about the same time. This relationship is particularly evident for trips that are expected to take between 5 and 20 minutes.

```r
with(cabi, plot(TotalTransitTime_mins, PlannedBikeTime_mins, xlim = c(0,
    45), ylim = c(0, 45), xlab = "OTP Transit Time (min)", ylab = "OTP Bike Time (min)",
    main = "Comparison of Expected Travel Time \n by Transit and Bicycle Modes",
    col = rgb(100, 100, 100, 25, maxColorValue = 255), pch = 16, asp = 1, xaxt = "n",
    yaxt = "n", bty = "n"))
axis(2, at = 10 * (0:6), tck = 1, col = "white", lwd = 1, labels = FALSE)
axis(2, at = 10 * (0:6), las = 2)
axis(1, at = 10 * (0:6), tck = 1, col = "white", lwd = 1, labels = FALSE)
axis(1, at = 10 * (0:6))
abline(a = 0, b = 1, col = "red", lwd = 1)
```



This perspective, however, assumes that all CaBi riders take a direct and efficient path to their destination. The nature of CaBi system, however, invites not only commuters, but also tourists who are unfamiliar with the city and leisure riders who are not making direct trips. In addition to the O-D pairs, the original dataset provided actual trip duration. Using this value in lieu of OTP predicted bike time provides insight into those other users.

```
with(cabi, plot(TotalTransitTime_mins, CaBiTime_mins, xlim = c(0,
    45), ylim = c(0, 45), xlab = "OTP Transit Time (min)", ylab = "CaBi Trip Time (min)",
    main = "Comparison of Expected Transit Travel Time \n and Actual CaBi Trip Time",
    col = rgb(100, 100, 100, 25, maxColorValue = 255), pch = 16, asp = 1, xaxt = "n",
    yaxt = "n", bty = "n"))
axis(2, at = 10 * (0:6), tck = 1, col = "white", lwd = 1, labels = FALSE)
axis(2, at = 10 * (0:6), las = 2)
axis(1, at = 10 * (0:6), tck = 1, col = "white", lwd = 1, labels = FALSE)
axis(1, at = 10 * (0:6))
abline(a = 0, b = 1, col = "red", lwd = 1)
```



Notice in the above figure that many trips took significantly longer by bicycle than they would have taken by transit (shown above the red diagonal). This is indicative of the type of user described earlier who makes an indirect trip. Furthermore, the agglomeration of points is not as concentrated as the OTP predictions. The OTP prediction is based on averages and is a deterministic model; it does not simulate a distribution of bicycle user speeds. As a tool designed for travelers, average speed is useful so users receive consistent results. For analysis purposes, however, it may be more helpful to simulate the distribution of speeds for users when planning trips. The following section discusses this further.

3.3 Assessing OTP Bike Routing Speed

OpenTripPlanner uses a single walking and bicycling speed for use in travel time calculations. Travel time is calculated using a combination of the travel speed and the expected number of stops due to intersections and other impediments. The results is a distribution of average speeds which will all be less than the assumed en-route speed. Currently, the speed set in OTP for bicyclists is 11 mph. The summary statistics shown in the table below highlights the narrow band of average speeds that results from OTP (low standard deviation). This yields predicted average speeds for trips between 9 and 11 mph. The actual average CaBi speed, however, has much more distributed profile and is slower. The average actual speed for CaBi riders is just under 8 mph with a standard deviation of 2.5 mph.

```
## Loading required package: boot
```

	PlannedBike_mph	CaBi_mph
median	10.72	7.96
mean	10.64	7.89
SE.mean	0.01	0.05
CI.mean.0.95	0.02	0.09
var	0.33	7.58
std.dev	0.58	2.75
coef.var	0.05	0.35

Because of the unique characteristics of CaBi riders, we are not yet sure that this discrepancy in average speed deserves to be rectified to a slower bicycle travel speed.

4 Conclusion

The purpose of this exercise was to introduce users to the analytical opportunities of batch-processing origin-destination information and finding uses for the aggregate or disaggregate outputs. In just a few commands using R or another statistical package, one can easily identify trends and patterns in a customized output dataset. OTP can provide simple summary metrics such as planned time spent traveling, speed of planned trips using different modes, most popular routes or other specifics. Multi-dimensional databases of information can also be generated from the hierarchical format of the different itineraries that come from the OTP API. There are also oportunities for application in more mainstream transportation analyses such as travel demand modeling which often has difficulty handling trip characteristics at which OTP operates. Hopefully this will spark interest in applying this methodology for transportation planinng purposes.