New York City Bike Share

Gary Miguel (garymm), James Kunz (jkunz), Everett Yip (everetty)

Background and Data:

Citi Bike is a public bicycle sharing system in New York City. It is the largest bike sharing program in the United States. The system administrators have made the following data freely available (http://www.citibikenyc.com/system-data) for every trip from 2013-07 to 2014-08 (10.4 million trips):

- Start time and date, end time and date.
- Start location, end location.
- Which bike the user is on.
- Type of user: subscriber or customer (one-off rental).
- Birth year of the user (only for subscribers).
- Gender of the user (only for subscribers).

We also use other freely available data on weather, events, holidays and geographic features.

Motivation and Goals:

We put ourselves in the mindset of an administrator of the bike share system, and use the data to better understand the system's users in order to provide better service. In particular we:

- Predict the rate of departures at a given station at a given time. If system administrators under-predict this, demand for bikes exceeds supply and customers will be unhappy. If they over-predict, they will waste money on excess capacity.
- Predict the total number of bikes being used throughout the entire system at any given time. Understanding when bikes will be in use will give administrators a high level view of general usage patterns, allowing them to plan maintenance and capacity increases more effectively.
- Find particular properties of different stations. For example, which stations have net outflows or inflows of bikes at certain times.

Preparing the data:

In addition to the Citi Bike ride data, we also included the following data:

- Weather data from NOAA (http://cdo.ncdc.noaa.gov/qclcd/QCLCD).
- List of state holidays from Wikipedia.
- Schedules of New York's MLB, NBA and NFL teams' games.
- Events collected by the New York Times (http://developer.nytimes.com/docs/events_api)

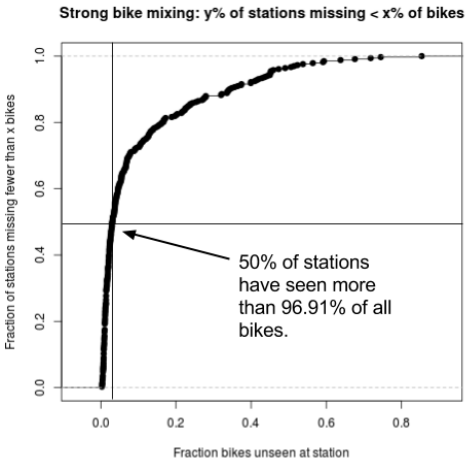
We converted the per-ride data to several aggregated measures, including:

- Counts of departures and arrivals at each station for each hour in the data.
- Combined measures for each hour of the day across all days in the data set (eg all departures between 07:00-08:00 summed across all days).
- Total bikes being used across the entire system for every hour.

This data was joined with the other data sources we collected. This involved extensive pre-processing as we encountered major challenges with data formatting and quality. Each data source has its own date and time format which we had to parse and join. For example, the New York Times events API returned date-time data in strings like "Dec. 3-15. Wednesday at 7:30 p.m.,

Thursday and Friday 8 p.m., Saturday at 2 and 8 p.m., Sunday at 2 p.m.", which we converted into a list of specific times. The weather data mixed numeric values with special alpha-numeric codes that we had to learn how to interpret.
We also found it useful to add in time-shifted features. For example, we have one feature for the outside temperature at a certain hour, and we added features for what the temperature would be 2, 4, and 6 hours in the future, and in the past. Depending on how far into the future predictions need to be made, we also found providing actual usage in the recent past as a feature to be incredibly useful.

Data Analysis:

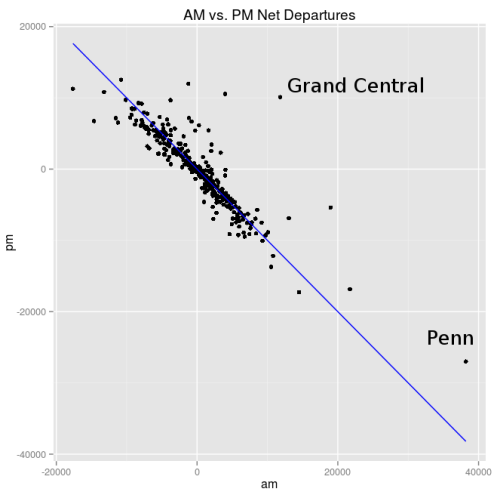


Bike mixing: How many stations do most bikes visit?

We looked at trying to cluster stations by the travel patterns of bikes. In particular, we were curious if we could learn associations between stations by the bikes that tend to travel between them. It turns out that almost all bikes visit all the stations; however, there are some AM/PM patterns that are interesting (see below).

Which stations typically lose or gain bikes in one day?

As a system administrator, if bikes taken from station X in the morning are returned to the same station X in the evening, then no work needs to be done to keep the number of bikes at that station stable, assuming there are sufficient bikes at the station to satisfy a day's worth of demand. Only about 2.6% of trips start and end at the same station, so in order to analyze this, we have to look at net departures and arrivals per day to see how much the system administrators have to manually correct the distribution of bikes.

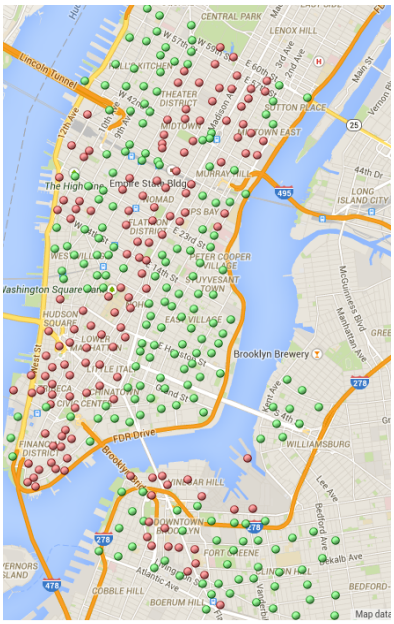


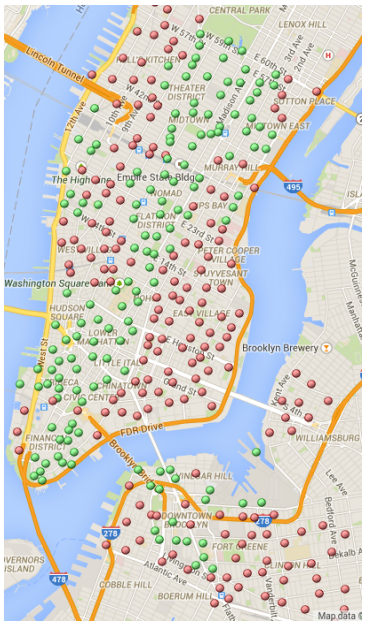
We found:

- Almost all stations are in balance within the day.
- While stations balance over a day, they often do so in different ways. In particular, certain stations lose bikes in the morning and gain them in the evening, while others see the opposite pattern.

Each point in the above plot represents a station. The x and y values are the net number of bikes lost in AM and PM, respectively (negative loss = a gain). Being above the y=-x line means a station loses bikes on an average day.

In the following two plots, green dots represent the net sources. We can clearly see the rush into the city core in the morning and out in the evening. As can be seen in the previous plot, a handful of stations are always net sources or net sinks.

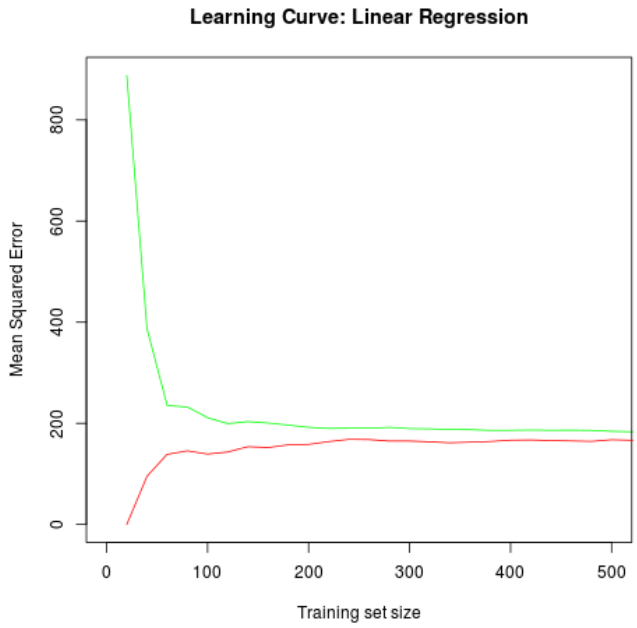




Morning Commute Evening Commute

Predicting total system usage, by hour

We have explored a few algorithms and feature sets to predict the total number of system users at any given time. We have data for 10,233 hours. Aggregating the departure and arrivals across trips, we can compute an estimate of the total number of bikes currently signed out of the system over the course of any particular hour. We built a linear regression model based on the features mentioned earlier. The learning curve is to the left. It is a classic case of high bias: test set performance is similar to training and both are woefully bad. In fact, it performed so poorly (MSE: 177) it was nearly outperformed by simply predicting the mean (MSE: 208.4).



We built two additional models on this data, first using a GLM (specifically poisson), given that it is the canonical distribution for modelling this sort of problem, but it too underfit. Secondly, we built a random forest using R's randomForest library. It achieved considerably better (26.8 MSE on the training set and 30.3 on the test set).

How many bikes will leave a particular station at a particular time?

To narrow down which features and techniques would be useful for this problem, we started off by focusing on just one particular station (E 40 St & 5 Ave) and just focusing on subscriber departures.

For each algorithm we used the scikit-learn implementation, with grid-search and 5-fold cross-validation to find meta-parameters. For each hour, we tried to predict log(number of departures + 1) (see last section for rationale). We trained on 0.7 of the data, tested on the remainder. Results of various algorithms and feature types on our test set:

Description	Mean Squared Error(log(x + 1))
Ridge (α=2.07)	0.572
Lasso (α=.0297)	0.572
Lasso (α=.0297), scaled features	0.550
Lasso (α=.0297), scaled polynomial features	2.331
SVM, RBF kernel (C=27)	0.700
SVM, RBF kernel (C=27), scaled features	0.423
SVM, RBF kernel (C=27), scaled polynomial features	0.747
Random Forest (50 trees)	0.257

Note we also tried elastic net models, but cross validation found the optimal l1 ratio was 1.0, meaning it was exactly the same model as Lasso.
Having determined that a random forest is the best type of model from the ones we tried, we then applied it to some related prediction tasks.
For each station, we built a random forest, and predicted hourly departures for it, sorted the stations by MSE and then looked at 10th, 50th and 90th percentile MSE.
The first task was to try to predict how many subscribers would depart each station at each hour. We tried two approaches: predicting this directly for each station, and predicting it as a fraction of the total system usage * predicted totFor non-political activities and programs onlyal system usage. Our guess was that we have much more data to build a better model of total system usage, so this method might be better. In fact it was not.

Description	10th % MSE	50th % MSE	90th % MSE
subscribers for each station	0.744	4.431	10.013
subscribers for each station, calculated as (predicted fraction of total system departures) * (predicted system departures)	0.758	4.477	10.188

After this we started predicting log(hourly departures). See note at the end for why we think this is a better thing to predict. We tried two approaches to predicting total departures: predicting it directly, and building one model for customers, another for subscribers, and summing their predictions. The intuition is that these two types of users behave very differently, and the separate models did give different features importance. For example, for our test station of choice, our random forests had the following feature importances (where the importance across all features must sum to 1):

Subscriber top feature importances	Customer top feature importances
Hour of day: 0.387	Hour of day: 0.128
Day of week: 0.107	Current temperature: 0.078

Current temperature: 0.048	Temperature 2 hours ago: 0.065
Temperature 2 hours ago: 0.032	Day of week: 0.060

The differences in feature importances are clear, but the two approaches perform very similarly.

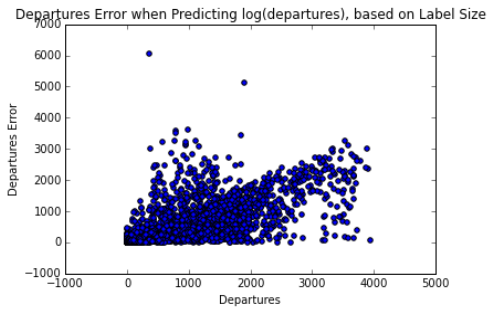
log(all departures for each station)	0.185	0.241	0.280
log(all departures for each station), predicted as prediction for customers + prediction for subscribers	0.186	0.241	0.279

And we tried two ways of predicting the subscriber departures, summed over all stations in the entire system. Again, we thought building separate models for separate stations might be beneficial, but it seems we just don't have enough features to accurately predict the per-station

Description	**MSE**
log(system-wide subscribers)	0.750
log(system-wide subscribers), predicted as sum of predictions for each station	1.067

Technical aside: Log()s & Heteroskedastic data

In class and PS1 we saw the importance of a fixed variance in the error term when performing linear regression. It is unreasonable for our problem to assume that this exists. Instead, we expect a fixed variance in terms of some %age around the expected value. We capture this by predicting the log of the value instead of the value itself, we minimize sum of $(\log(\text{predict}) - \log(\text{actual}))^2$, or, equivalently $(\log(\text{predict} / \text{actual}))^2$. As one would expect with this sort of formulation, we see the learning algorithms accept larger (absolute) errors around larger target numbers than small ones (see plot showing error distribution, each point is an hour in the test set). The % error tolerated in this data remains relatively constant similar to how you'd expect the absolute errors to remain constant in a homoskedastic problem. This intuition carries over into the practical application of the data.



Future work:

More features: We know data exists on transit delays, transit ridership, more data exists on local events, illnesses, crime, economic trends, bike rides with privately owned bicycles.

More sophisticated graph clustering algorithms of the sort introduced in CS224W may have yielded interesting unsupervised learning results. Studying the flow of traffic between pairs of stations might increase the understanding of bike flow in the system. Perhaps it would illuminate why a number of stations seem to be net sources or sinks regardless of the time of day.