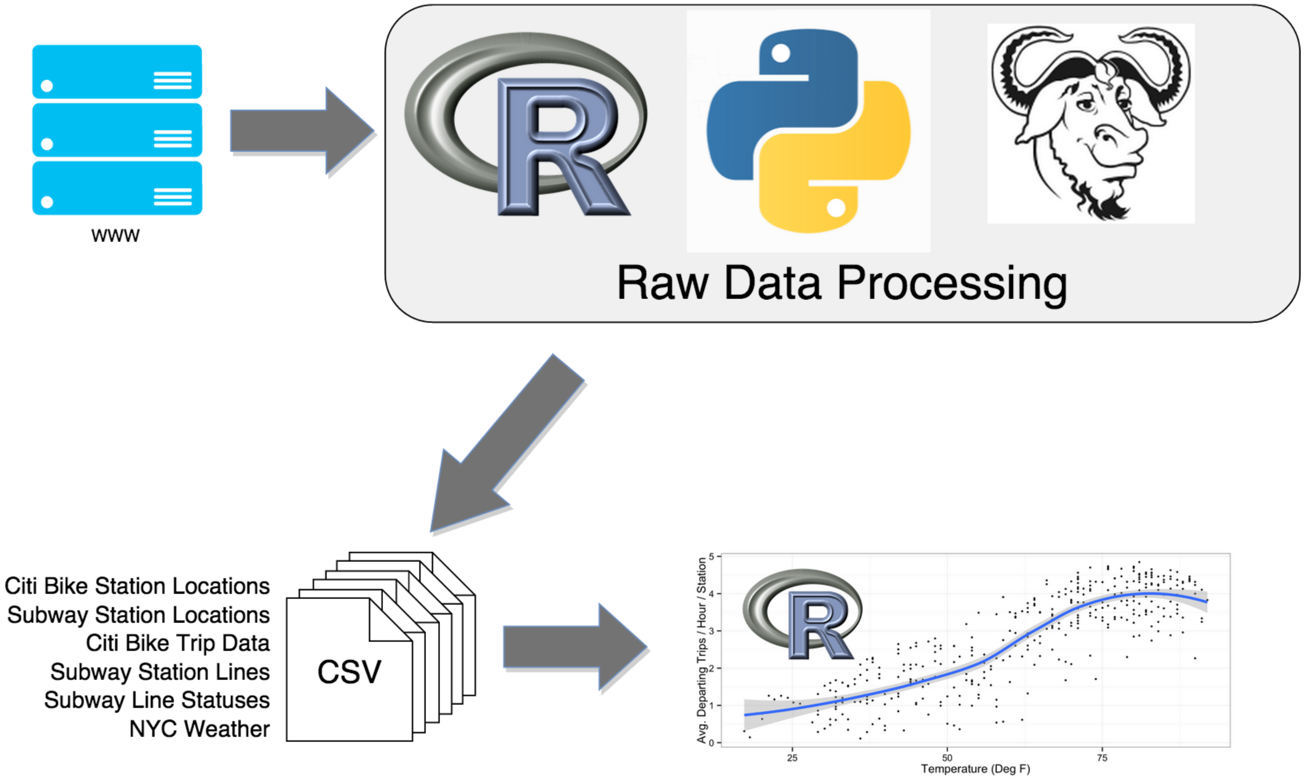
The Impact of Inclement Weather and Subway Outages on Citi Bike Ridership

Brian Weinstein
Andy Enkeboll
Ashutosh Nanda

https://github.com/BrianWeinstein/citibike-predictions

Overview



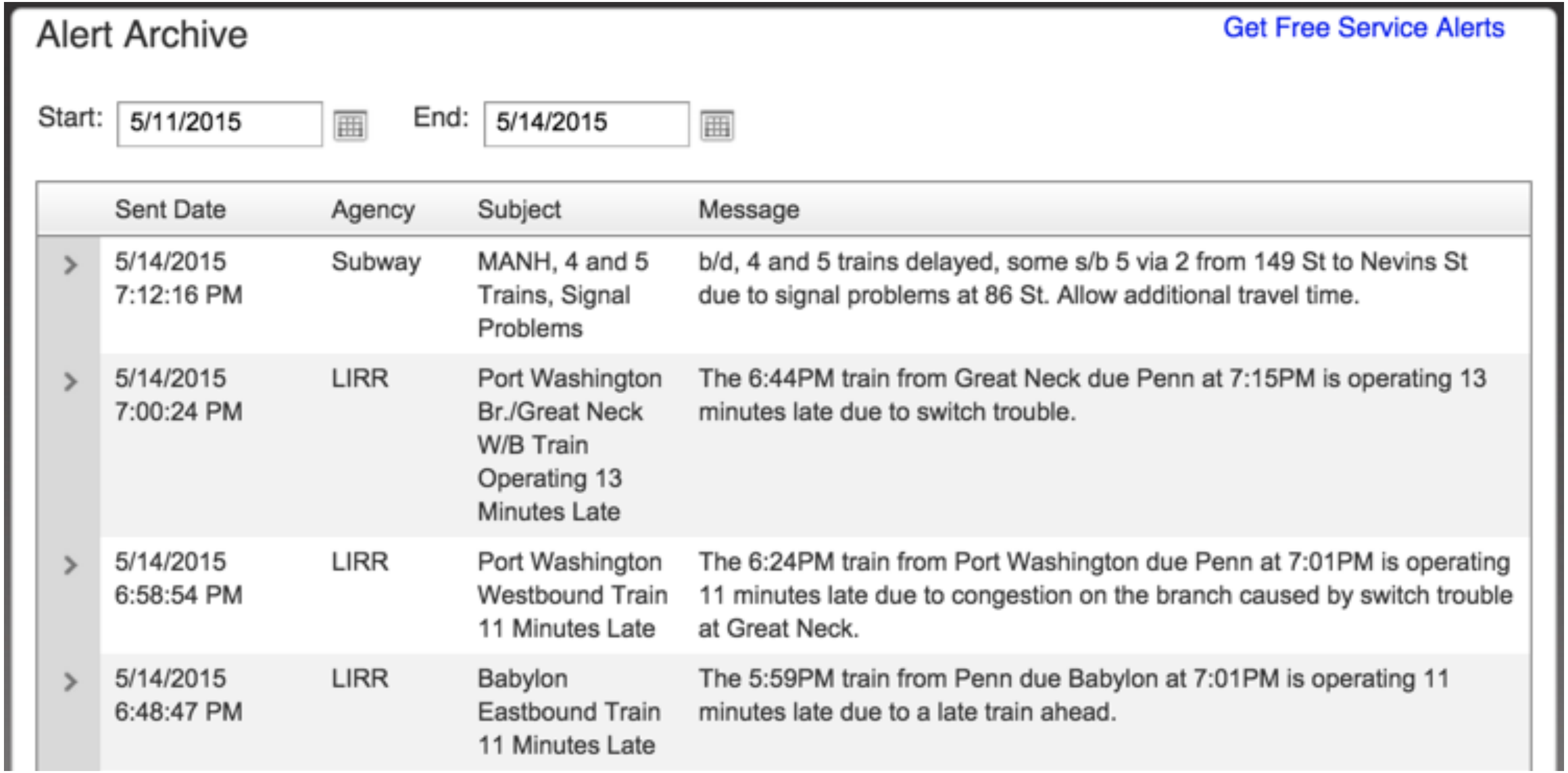
www

Raw Data Processing

Citi Bike Station Locations
Subway Station Locations
Citi Bike Trip Data
Subway Station Lines
Subway Line Statuses
NYC Weather

CSV

Data Source: Subway Status v1

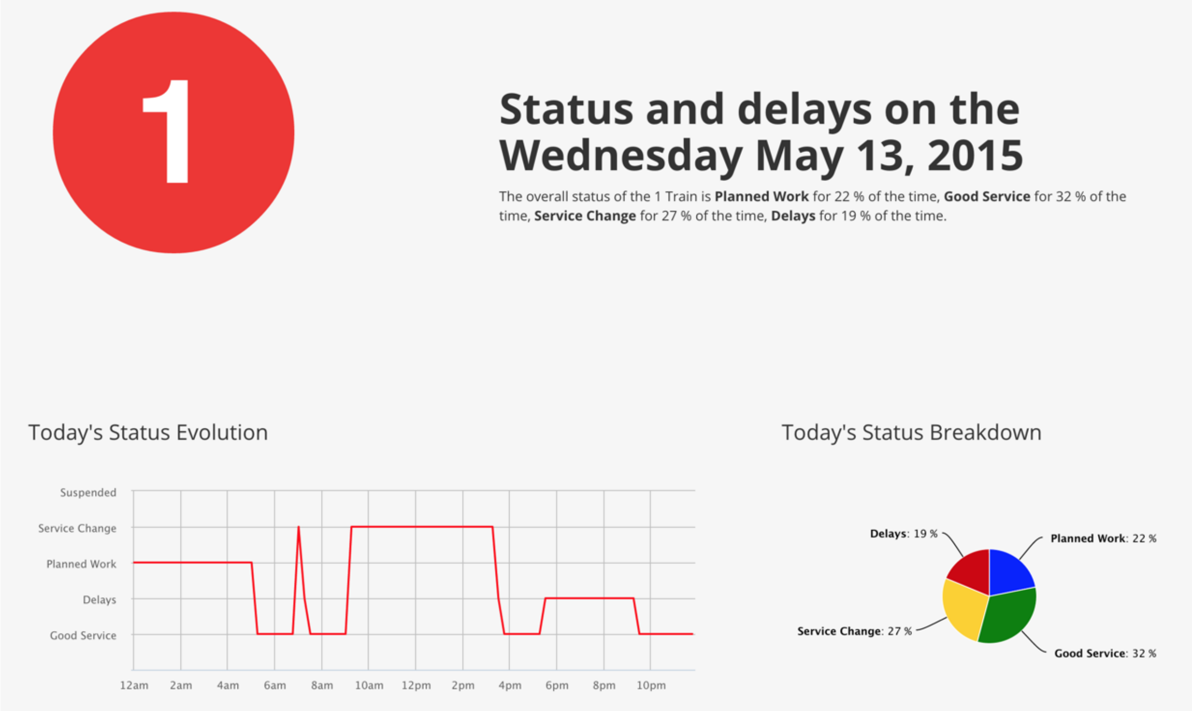


Alert Archive

Start: 5/11/2015 End: 5/14/2015

	Sent Date	Agency	Subject	Message
>	5/14/2015 7:12:16 PM	Subway	MANH, 4 and 5 Trains, Signal Problems	b/d, 4 and 5 trains delayed, some s/b 5 via 2 from 149 St to Nevins St due to signal problems at 86 St. Allow additional travel time.
>	5/14/2015 7:00:24 PM	LIRR	Port Washington Br./Great Neck W/B Train Operating 13 Minutes Late	The 6:44PM train from Great Neck due Penn at 7:15PM is operating 13 minutes late due to switch trouble.
>	5/14/2015 6:58:54 PM	LIRR	Port Washington Westbound Train 11 Minutes Late	The 6:24PM train from Port Washington due Penn at 7:01PM is operating 11 minutes late due to congestion on the branch caused by switch trouble at Great Neck.
>	5/14/2015 6:48:47 PM	LIRR	Babylon Eastbound Train 11 Minutes Late	The 5:59PM train from Penn due Babylon at 7:01PM is operating 11 minutes late due to a late train ahead.

Data Source: Subway Status v2

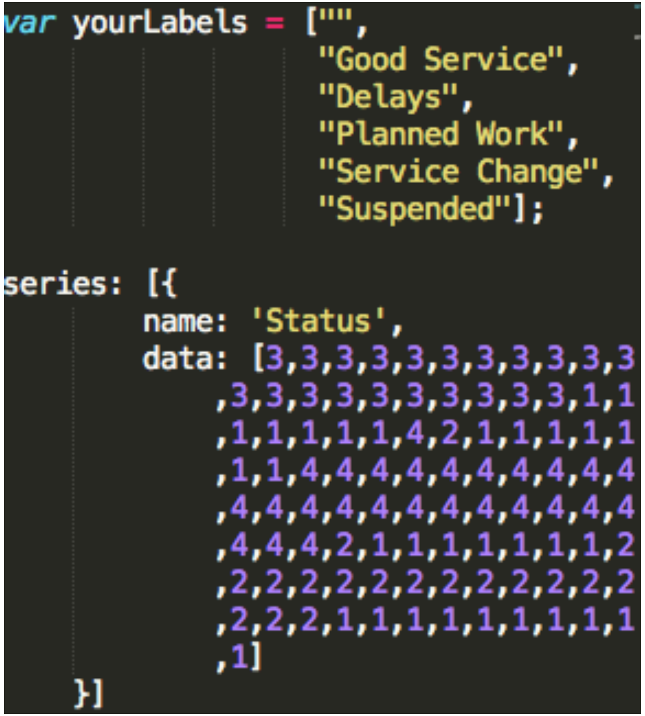


Status and delays on the Wednesday May 13, 2015

The overall status of the 1 Train is **Planned Work** for 22 % of the time, **Good Service** for 32 % of the time, **Service Change** for 27 % of the time, **Delays** for 19 % of the time.

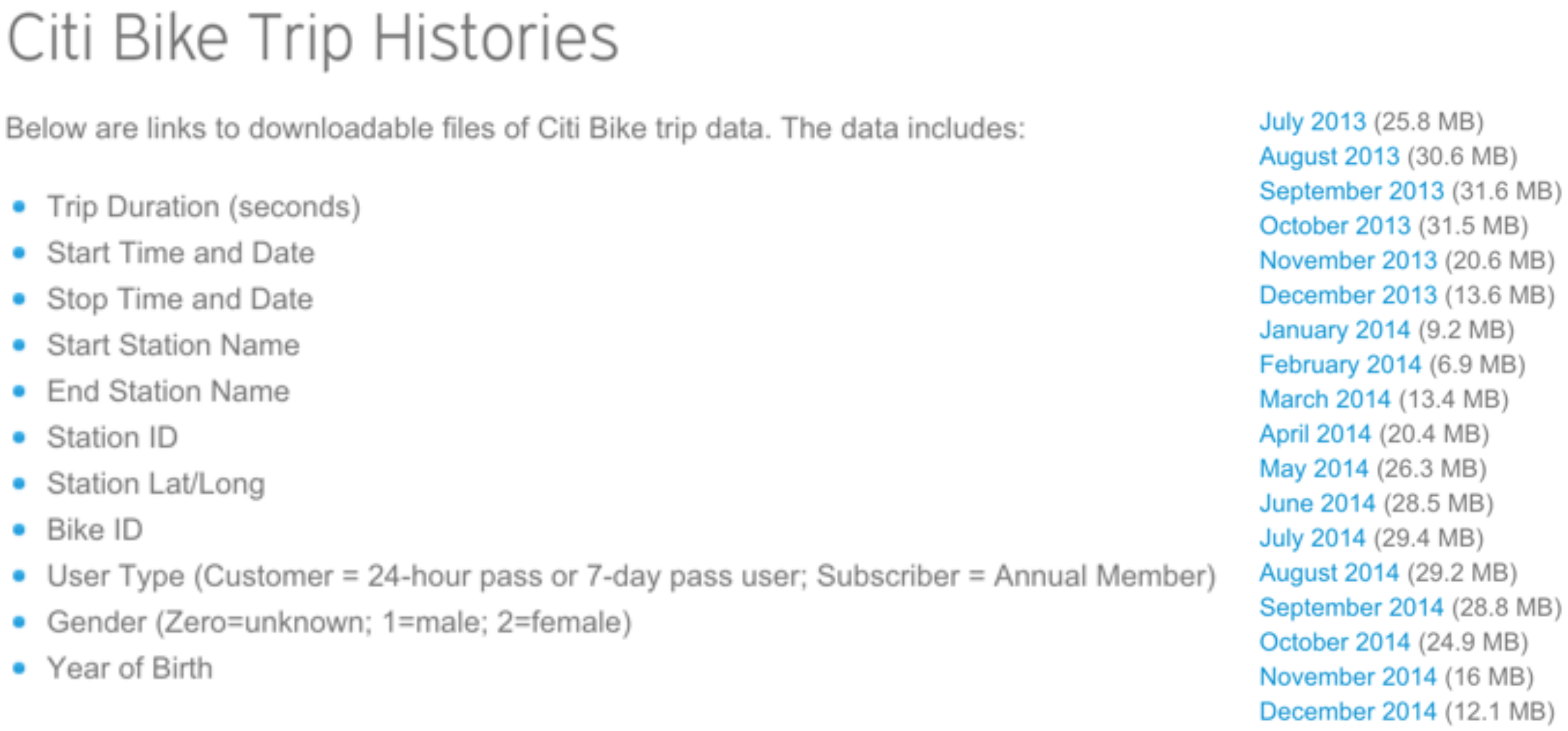
Today's Status Evolution

Today's Status Breakdown



```
var yourLabels = ["",
                  "Good Service",
                  "Delays",
                  "Planned Work",
                  "Service Change",
                  "Suspended"];

series: [{
        name: 'Status',
        data: [3,3,3,3,3,3,3,3,3,3,3
            ,3,3,3,3,3,3,3,3,3,3,1,1
            ,1,1,1,1,1,4,2,1,1,1,1,1
            ,1,1,4,4,4,4,4,4,4,4,4,4
            ,4,4,4,4,4,4,4,4,4,4,4,4
            ,4,4,4,2,1,1,1,1,1,1,1,2
            ,2,2,2,2,2,2,2,2,2,2,2,2
            ,2,2,2,1,1,1,1,1,1,1,1,1
            ,1]
    }]
```

Data Source: CitiBike Trips



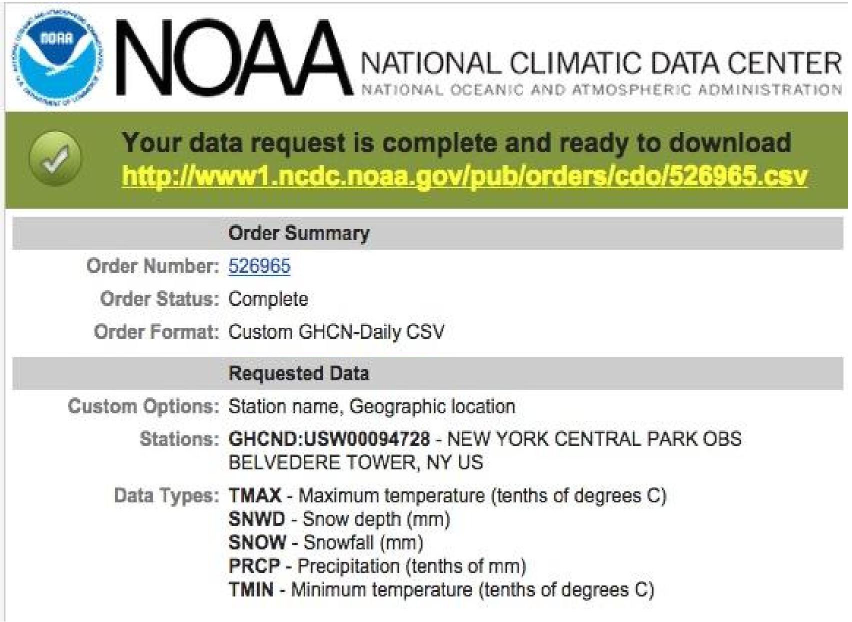
Citi Bike Trip Histories

Below are links to downloadable files of Citi Bike trip data. The data includes:

- Trip Duration (seconds)
- Start Time and Date
- Stop Time and Date
- Start Station Name
- End Station Name
- Station ID
- Station Lat/Long
- Bike ID
- User Type (Customer = 24-hour pass or 7-day pass user; Subscriber = Annual Member)
- Gender (Zero=unknown; 1=male; 2=female)
- Year of Birth

July 2013 (25.8 MB)
August 2013 (30.6 MB)
September 2013 (31.6 MB)
October 2013 (31.5 MB)
November 2013 (20.6 MB)
December 2013 (13.6 MB)
January 2014 (9.2 MB)
February 2014 (6.9 MB)
March 2014 (13.4 MB)
April 2014 (20.4 MB)
May 2014 (26.3 MB)
June 2014 (28.5 MB)
July 2014 (29.4 MB)
August 2014 (29.2 MB)
September 2014 (28.8 MB)
October 2014 (24.9 MB)
November 2014 (16 MB)
December 2014 (12.1 MB)

Data Source: New York Weather



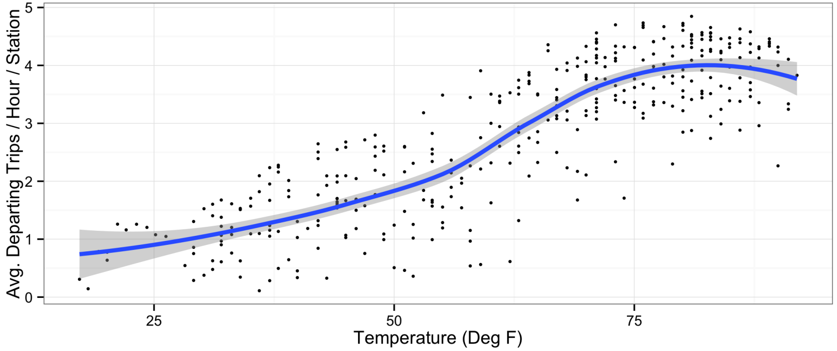
NOAA NATIONAL CLIMATIC DATA CENTER
NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION

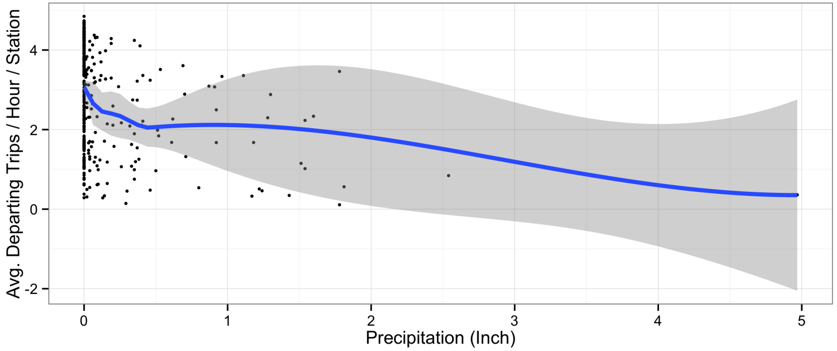
Your data request is complete and ready to download
http://www1.ncdc.noaa.gov/pub/orders/cdo/526965.csv

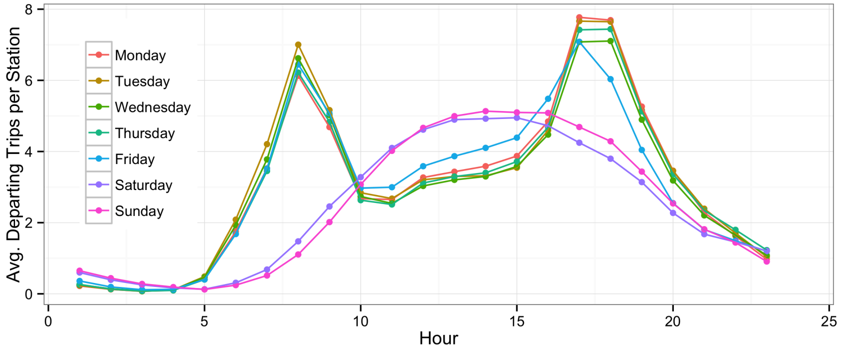
	Order Summary
Order Number:	526965
Order Status:	Complete
Order Format:	Custom GHCN-Daily CSV
	Requested Data
Custom Options:	Station name, Geographic location
Stations:	**GHCND:USW00094728** - NEW YORK CENTRAL PARK OBS BELVEDERE TOWER, NY US
Data Types:	**TMAX** - Maximum temperature (tenths of degrees C) **SNWD** - Snow depth (mm) **SNOW** - Snowfall (mm) **PRCP** - Precipitation (tenths of mm) **TMIN** - Minimum temperature (tenths of degrees C)

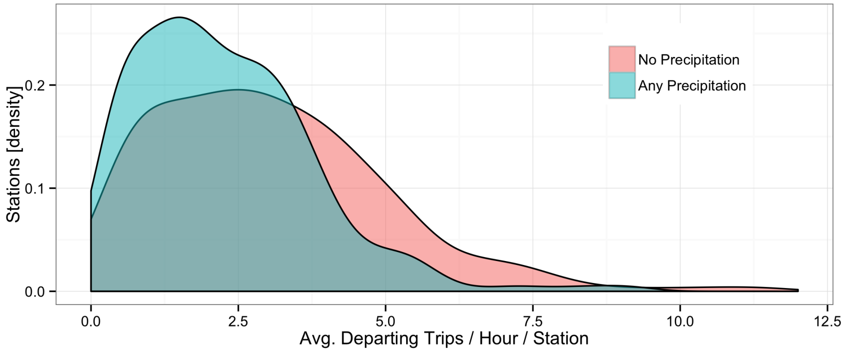
West New York
Union City
MANHATTAN
A
Broadway
UPPER EAST SIDE
Lincoln Tunnel
HELL'S KITCHEN
5th Ave
Park Ave
3rd Ave
1st Avenue
9th Ave
MIDTOWN
9A
Empire State Building
oken
8th Ave
6th Ave
LONG ISLAND CIT
1st Avenue
78
EAST VILLAGE
SOHO
GREEN
Grand

Data Exploration









Prediction Model

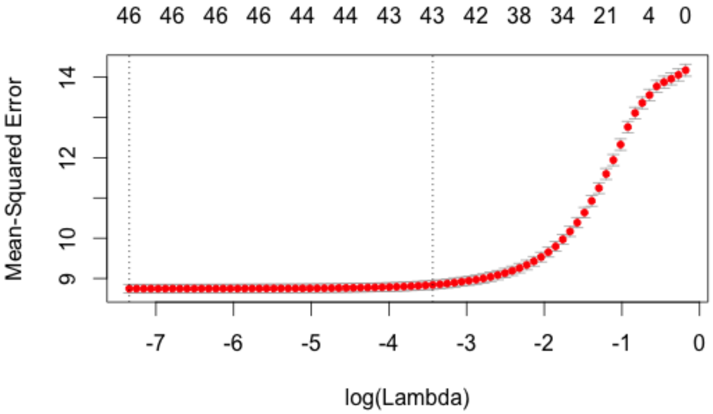
- Tested 4 regression models on two subsets of predictors:

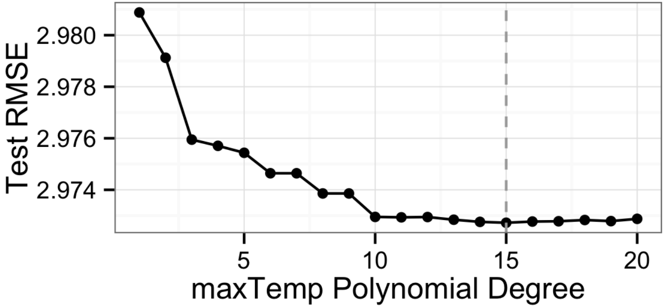
  Subset 1: `trips ~ hour + weekday + nearestSubStationDist + avgSubStationStatus + anyPrecip + maxTemp`

  Subset 2: `trips ~ hour + weekday + citiStationID +           avgSubStationStatus + anyPrecip + maxTemp`

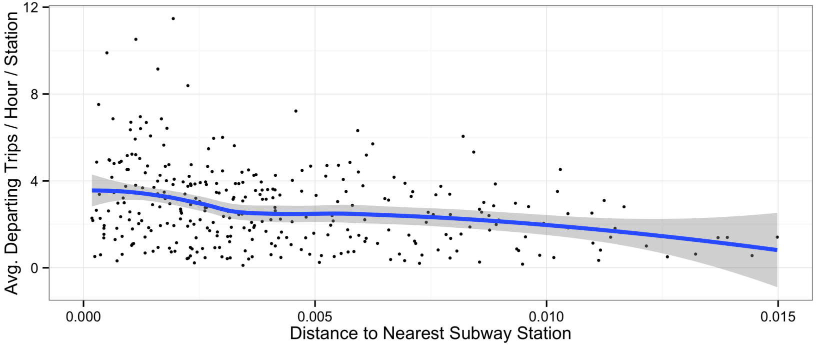
-

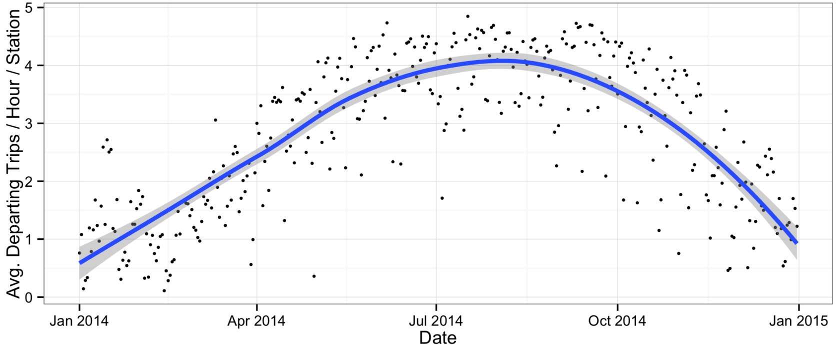
Test RMSE	Ridge	Lasso	Linear Least Squares	Polynomial Least Squares
Subset 1	3.2910	3.2885	3.2884	3.2811
Subset 2	2.9843	2.9808	2.9809	2.9727

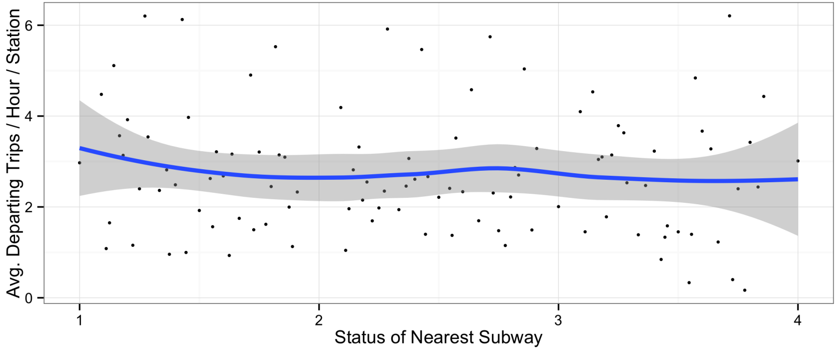


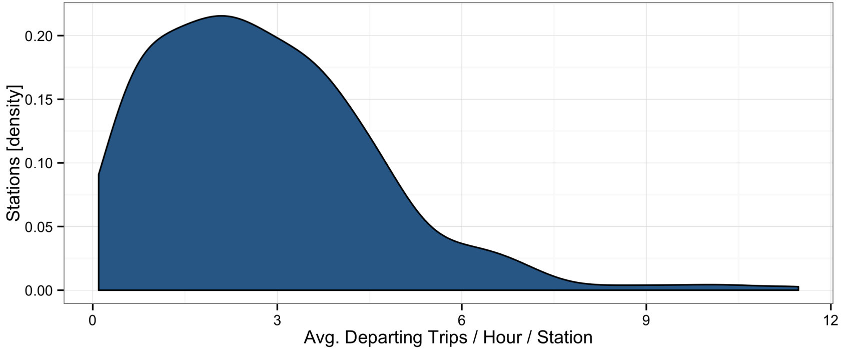


Additional: Data Exploration







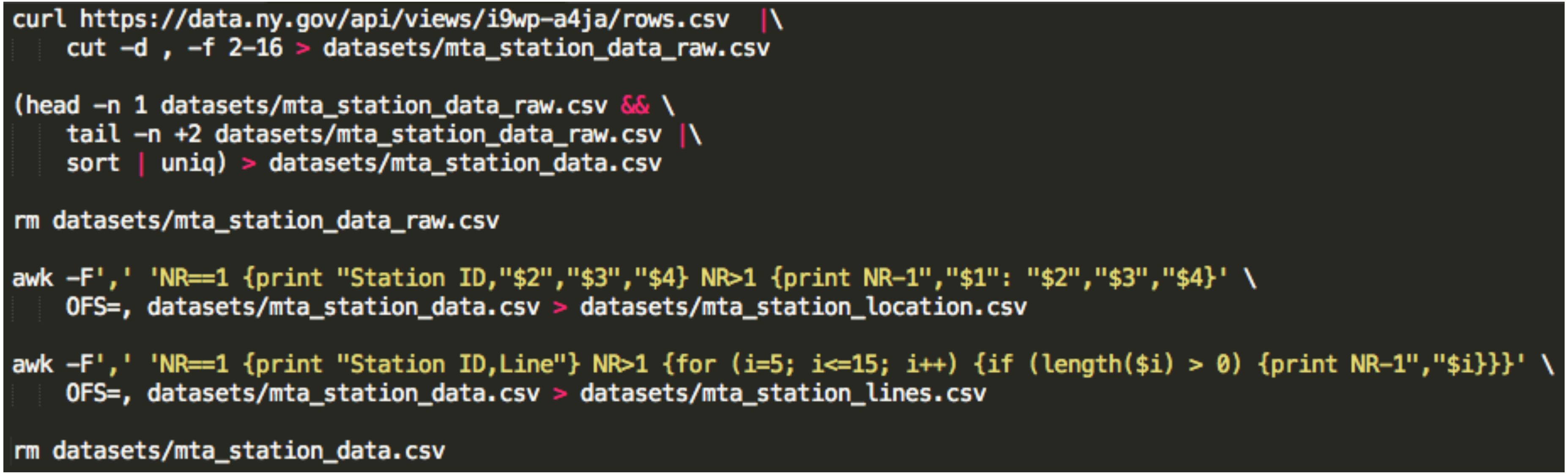


Additional: Optimization

When using `set.seed(12)`, Lasso didn’t shrink any of the coefficients to 0, although it does with other seeds.

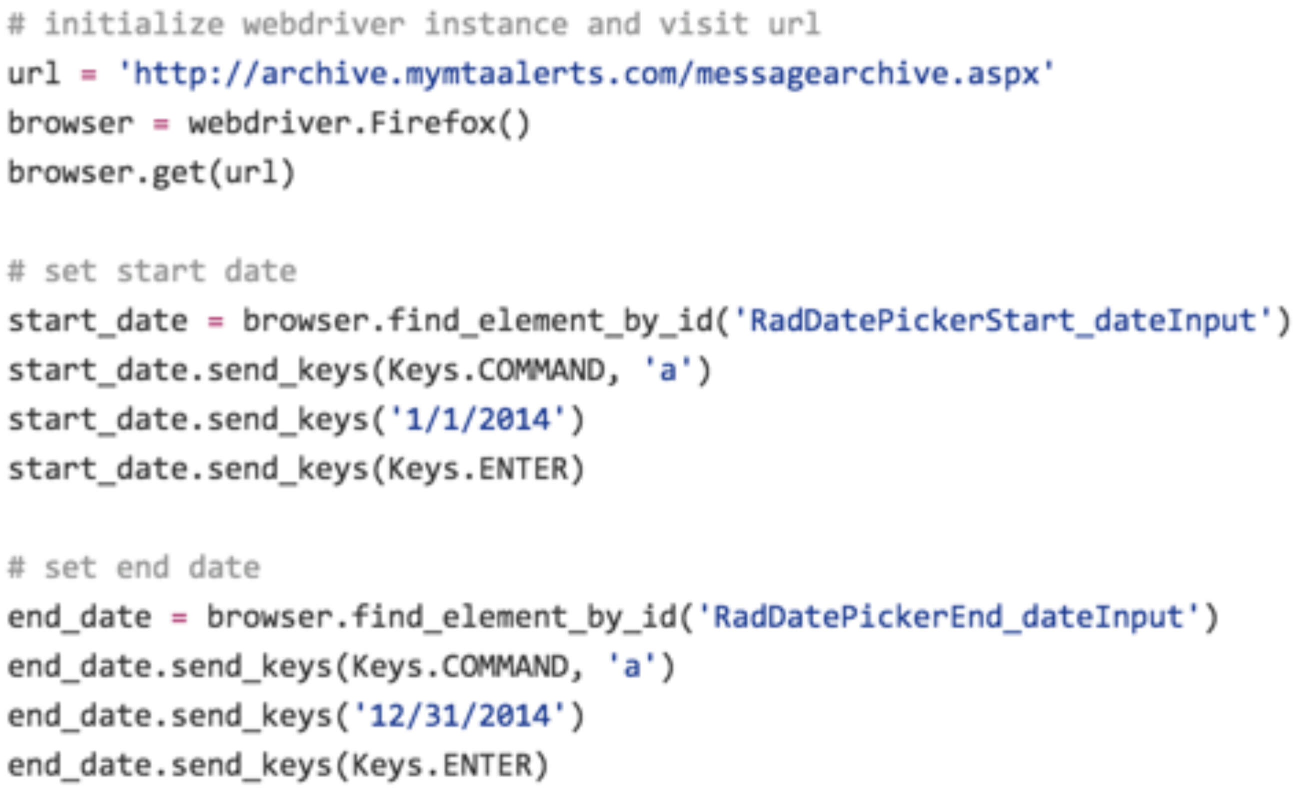
```
>coef(lasso.model, s=lasso.bestlambda
(Intercept)             -3.29118        weekday1                0.67272
hour1                   -0.34314        citiStationID127        3.56446
hour2                   -0.43761        citiStationID143       -0.19339
hour3                   -0.48536        citiStationID167        2.74610
hour4                   -0.47186        citiStationID228        1.33829
hour5                   -0.40206        citiStationID254        0.27684
hour6                    0.22949        citiStationID259        0.85215
hour7                    1.31776        citiStationID294        3.27361
hour8                    3.65633        citiStationID298       -0.70295
hour9                    3.08644        citiStationID331        0.13698
hour10                   1.89055        citiStationID332        0.27645
hour11                   2.08708        citiStationID339       -0.58213
hour12                   2.42439        citiStationID352        2.22051
hour13                   2.56377        citiStationID357        1.69681
hour14                   2.78327        citiStationID389       -0.41492
hour15                   2.82930        citiStationID430        0.81722
hour16                   3.38743        citiStationID431       -0.81171
hour17                   4.56809        citiStationID485        1.70873
hour18                   4.35563        citiStationID494        3.09902
hour19                   2.80921        citiStationID536        2.79051
hour20                   1.68588        avgSubStationStatus -0.11149
hour21                   0.91142        anyPrecip1             -0.63762
hour22                   0.57315        maxTemp                 0.04402
hour23                   0.21723
```

Data Source: MTA Subway Locations



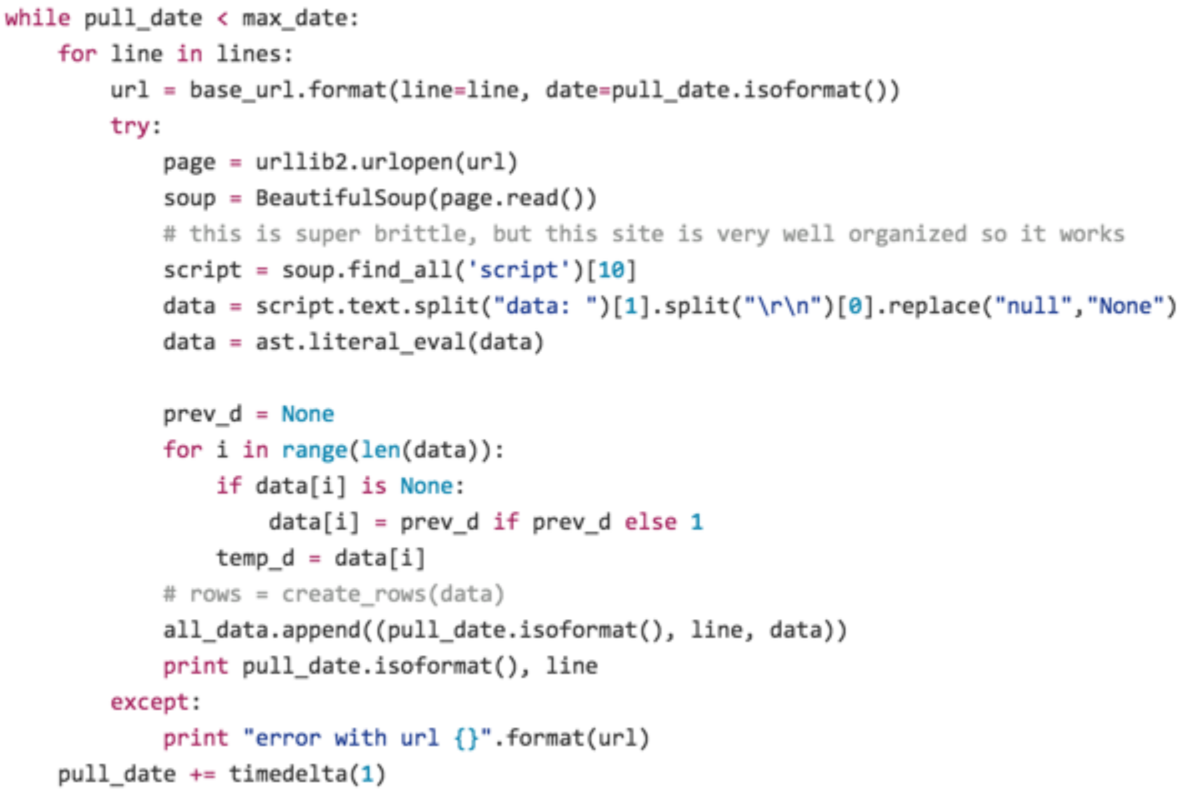
```
curl https://data.ny.gov/api/views/i9wp-a4ja/rows.csv  |\
    cut -d , -f 2-16 > datasets/mta_station_data_raw.csv

(head -n 1 datasets/mta_station_data_raw.csv && \
    tail -n +2 datasets/mta_station_data_raw.csv |\
    sort | uniq) > datasets/mta_station_data.csv

rm datasets/mta_station_data_raw.csv

awk -F',' 'NR==1 {print "Station ID,"$2","$3","$4} NR>1 {print NR-1","$1": "$2","$3","$4}' \
    OFS=, datasets/mta_station_data.csv > datasets/mta_station_location.csv

awk -F',' 'NR==1 {print "Station ID,Line"} NR>1 {for (i=5; i<=15; i++) {if (length($i) > 0) {print NR-1","$i}}}' \
    OFS=, datasets/mta_station_data.csv > datasets/mta_station_lines.csv

rm datasets/mta_station_data.csv
```

Data Source: Subway Status v1



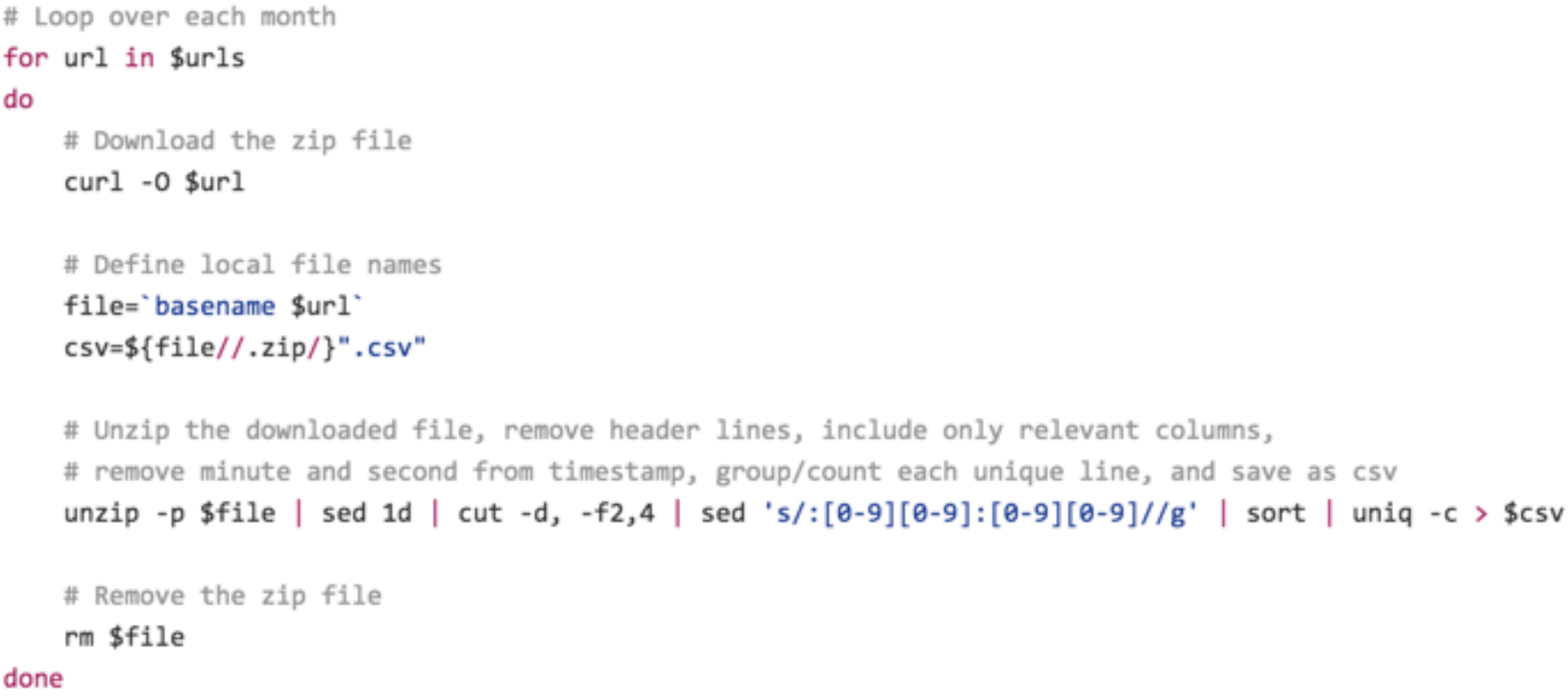
```
# initialize webdriver instance and visit url
url = 'http://archive.mymtaalerts.com/messagearchive.aspx'
browser = webdriver.Firefox()
browser.get(url)

# set start date
start_date = browser.find_element_by_id('RadDatePickerStart_dateInput')
start_date.send_keys(Keys.COMMAND, 'a')
start_date.send_keys('1/1/2014')
start_date.send_keys(Keys.ENTER)

# set end date
end_date = browser.find_element_by_id('RadDatePickerEnd_dateInput')
end_date.send_keys(Keys.COMMAND, 'a')
end_date.send_keys('12/31/2014')
end_date.send_keys(Keys.ENTER)
```

Data Source: Subway Status v2



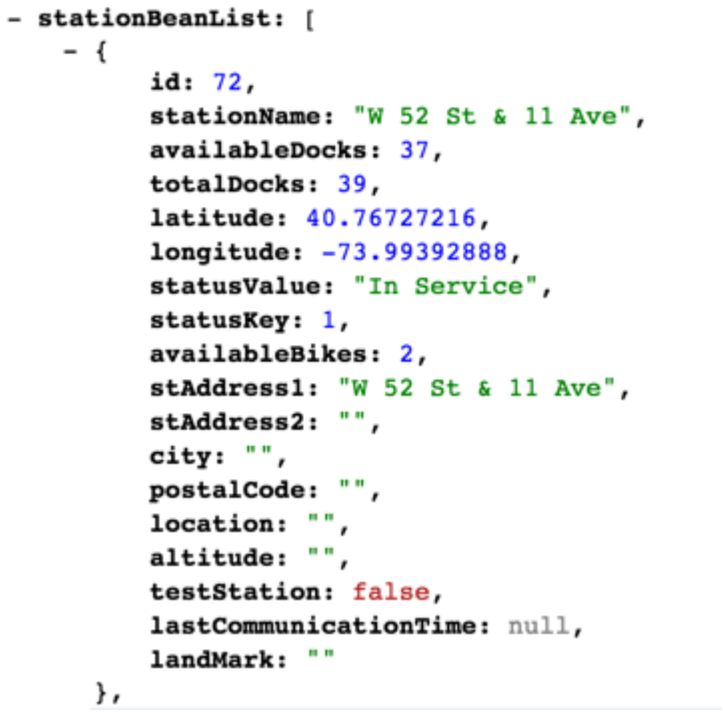
```
while pull_date < max_date:
    for line in lines:
        url = base_url.format(line=line, date=pull_date.isoformat())
        try:
            page = urllib2.urlopen(url)
            soup = BeautifulSoup(page.read())
            # this is super brittle, but this site is very well organized so it works
            script = soup.find_all('script')[10]
            data = script.text.split("data: ")[1].split("\r\n")[0].replace("null","None")
            data = ast.literal_eval(data)

            prev_d = None
            for i in range(len(data)):
                if data[i] is None:
                    data[i] = prev_d if prev_d else 1
                temp_d = data[i]
            # rows = create_rows(data)
            all_data.append((pull_date.isoformat(), line, data))
            print pull_date.isoformat(), line
        except:
            print "error with url {}".format(url)
    pull_date += timedelta(1)
```

Data Source: CitiBike Trips

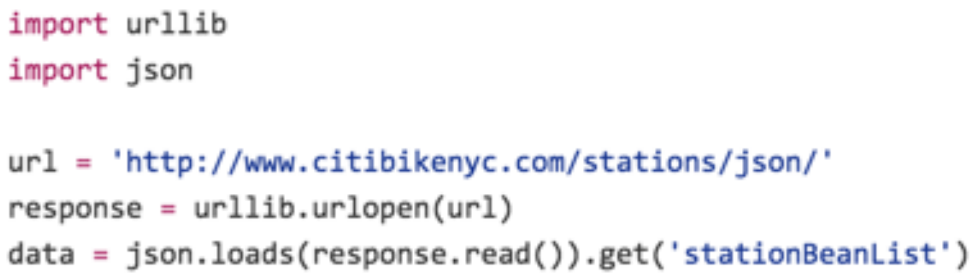


```
# Loop over each month
for url in $urls
do
    # Download the zip file
    curl -O $url

    # Define local file names
    file=`basename $url`
    csv=${file//.zip/}".csv"

    # Unzip the downloaded file, remove header lines, include only relevant columns,
    # remove minute and second from timestamp, group/count each unique line, and save as csv
    unzip -p $file | sed 1d | cut -d, -f2,4 | sed 's/:[0-9][0-9]:[0-9][0-9]//g' | sort | uniq -c > $csv

    # Remove the zip file
    rm $file
done
```

Data Source: CitiBike Locations



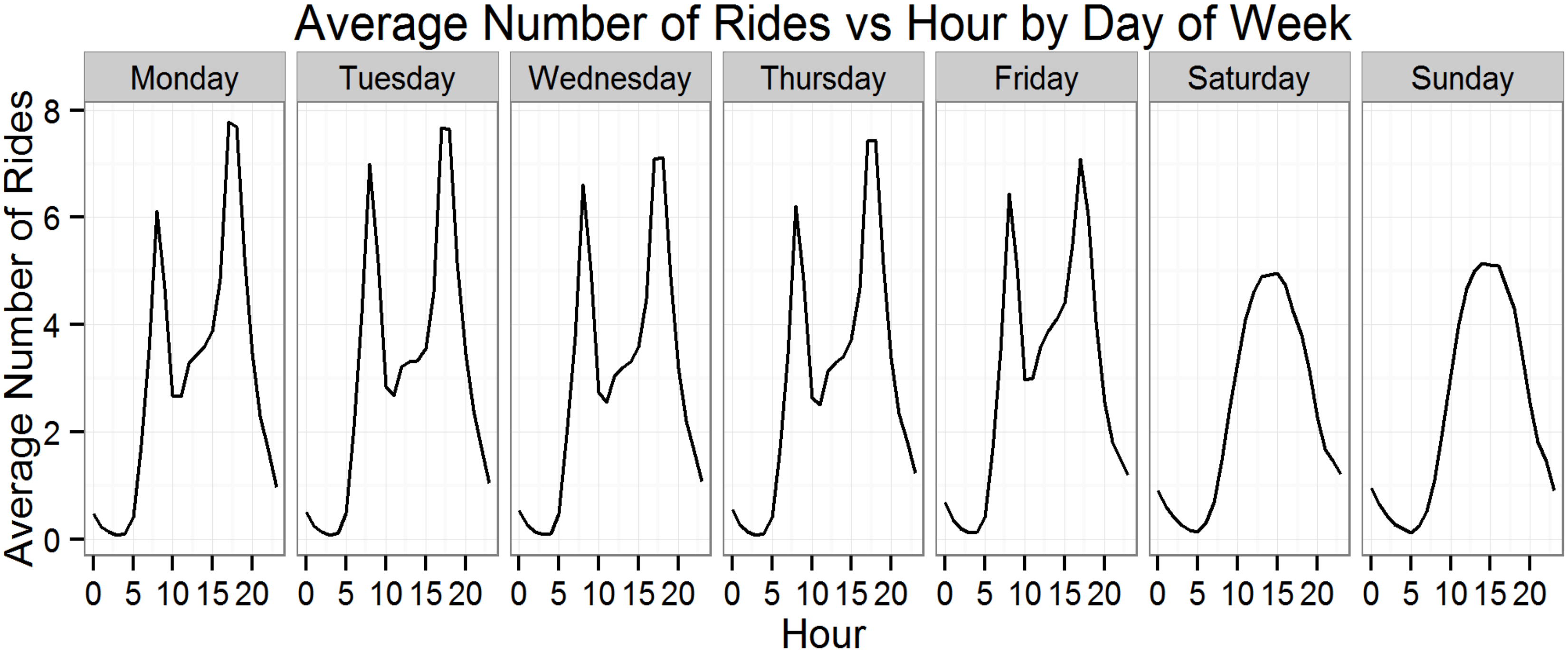
```
- stationBeanList: [
   - {
          id: 72,
          stationName: "W 52 St & 11 Ave",
          availableDocks: 37,
          totalDocks: 39,
          latitude: 40.76727216,
          longitude: -73.99392888,
          statusValue: "In Service",
          statusKey: 1,
          availableBikes: 2,
          stAddress1: "W 52 St & 11 Ave",
          stAddress2: "",
          city: "",
          postalCode: "",
          location: "",
          altitude: "",
          testStation: false,
          lastCommunicationTime: null,
          landMark: ""
     },
```



```python
import urllib
import json

url = 'http://www.citibikenyc.com/stations/json/'
response = urllib.urlopen(url)
data = json.loads(response.read()).get('stationBeanList')
```

Other Analysis

Analyzing the Graph of Citibike Stations



Difference Between Graph Structure

Metrics

- Average In Degree, Average Out Degree
- Average Path Length Between Stations
- Eigenvector Centrality (Bonacich, 1972, Journal of Mathematical Sociology)
  - Original basis for Google's PageRank
  - Calculate eigenvector for largest eigenvalue of adjacency matrix
  - Each entry in this vector is a centrality measure for a node
  - Measure is guaranteed unique by Perron-Frobenius theorem
    - All nodes having positive centrality measures requires greatest eigenvalue

$$x_v = \frac{1}{\lambda_{\max}} \sum_{t \in N(v)} x_t = \frac{1}{\lambda_{\max}} \sum_{t \in G} a_{v,t} x_t$$

Simple Metrics - In Degree

Weekday

Ending Station ID	In Degree
473	332
151	331
412	331
497	331
293	330
312	330

Weekend

Ending Station ID	In Degree
151	321
293	320
387	317
312	316
263	315
428	315

Simple Metrics - Out Degree

Weekday

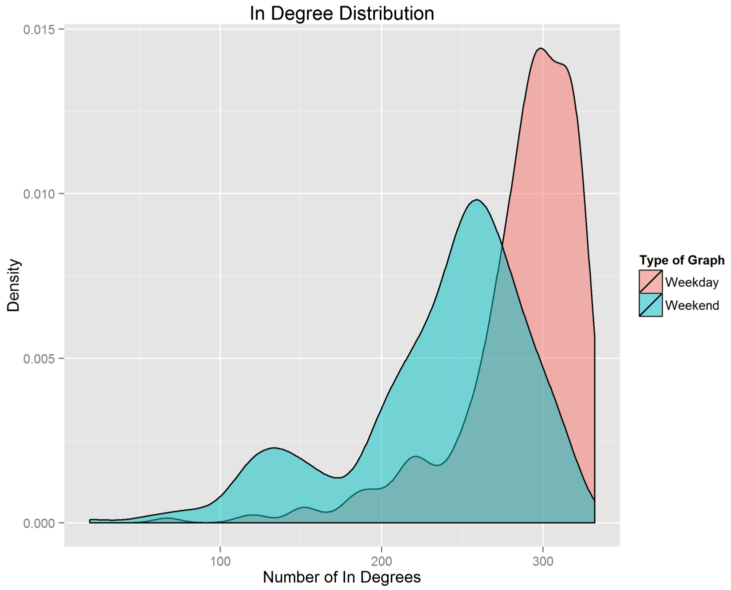
Starting Station ID	Out Degree
151	331
497	331
265	329
387	329
236	328
250	328

Weekend

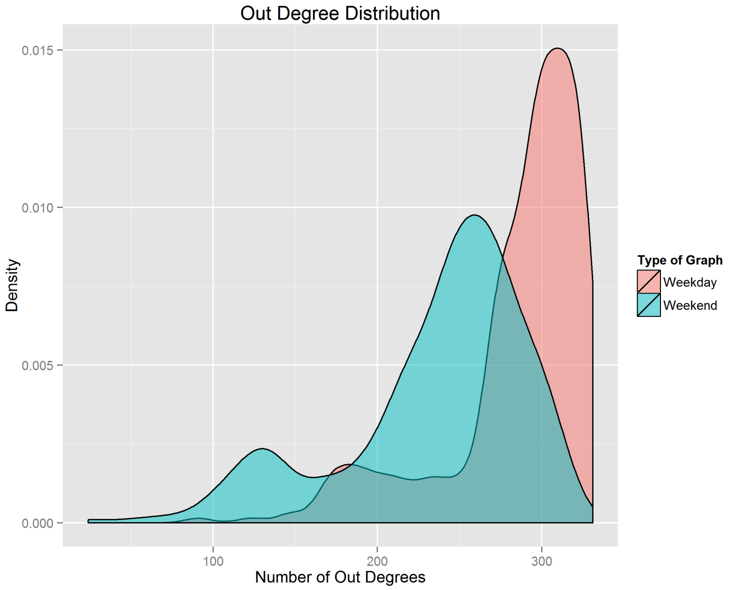
Starting Station ID	Out Degree
312	318
151	316
251	314
237	313
236	312
387	312

Simple Metrics - Degree Distribution

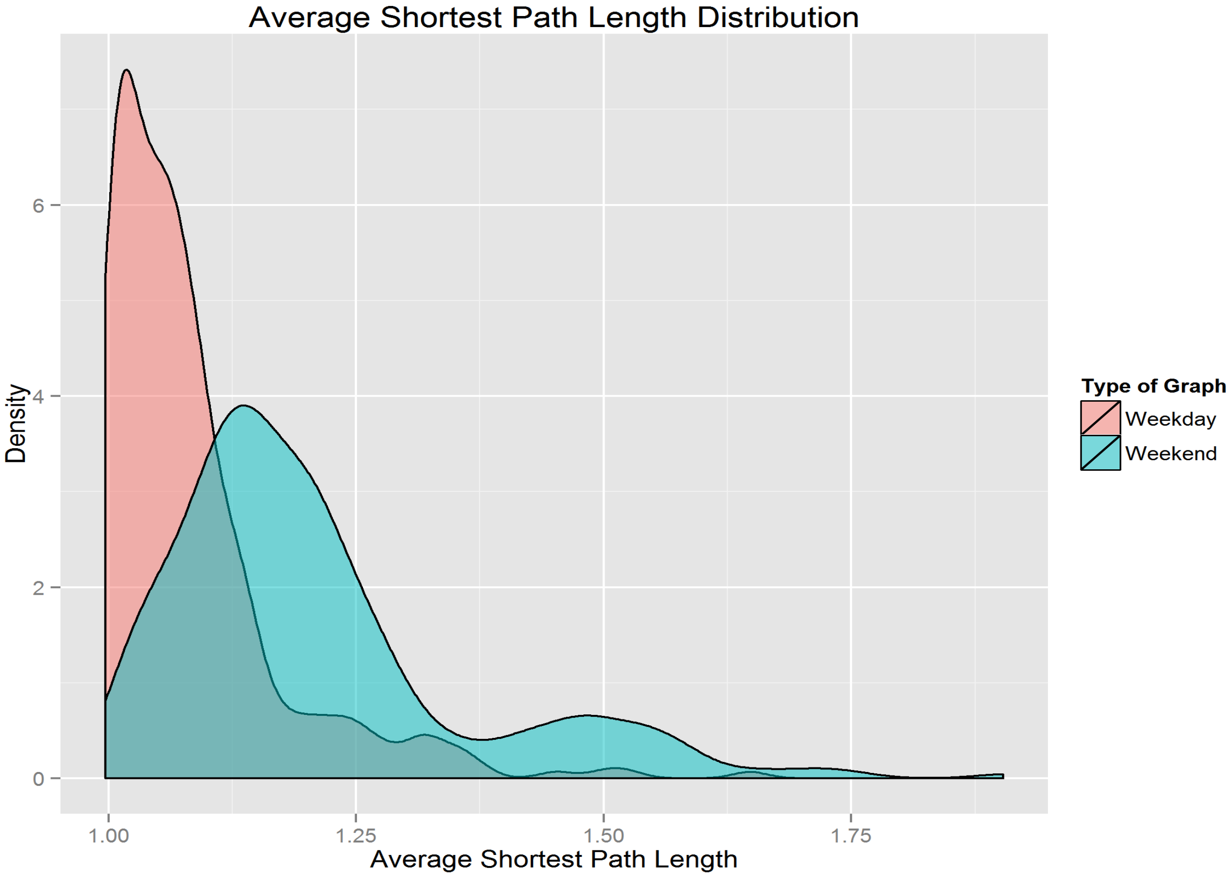
In Degree



Out Degree



Less Simple Metrics - Average Path Length Between Stations



Complicated Metrics - Eigencentrality

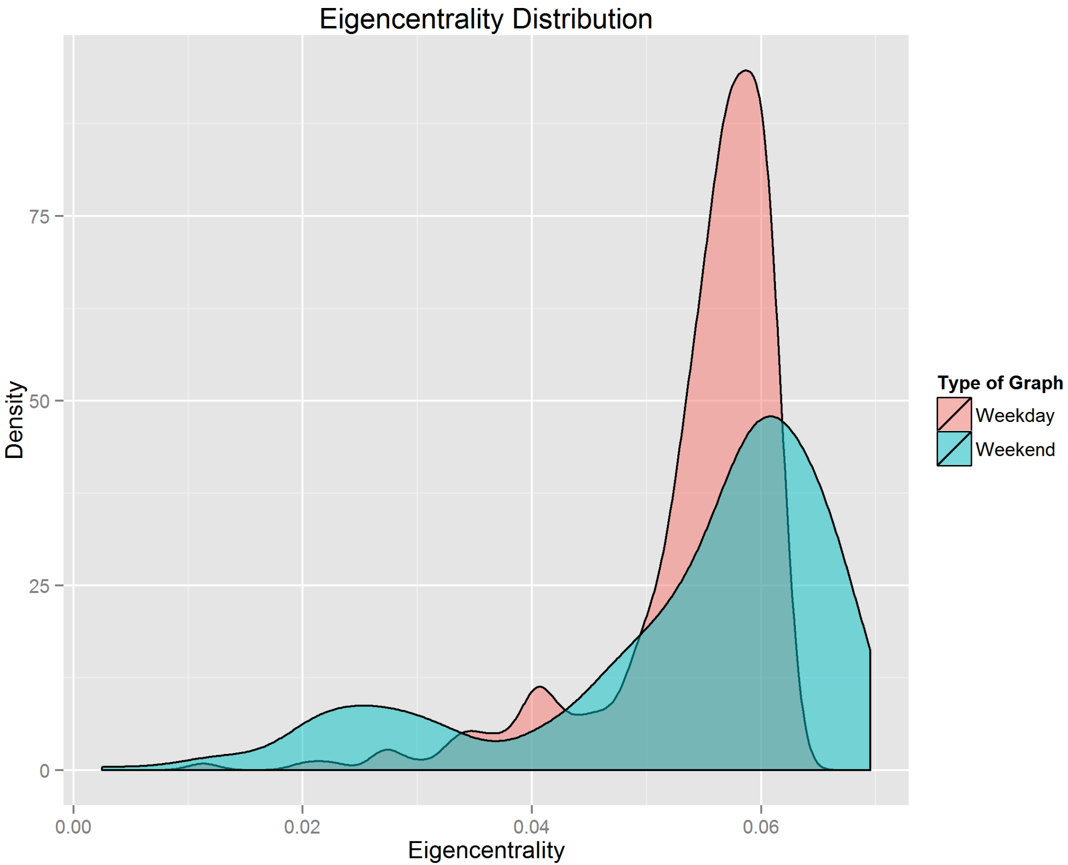
Weekday

Station ID	
473	0.0619
151	0.0618
497	0.0618
412	0.0618
293	0.0617
312	0.0617

Weekend

Station ID	Out Degree
151	0.0695
293	0.0695
312	0.0690
250	0.0686
251	0.0686
428	0.0685

Complicated Metrics - Eigencentrality


Eigencentrality Distribution
Density
Eigencentrality
Type of Graph
Weekday
Weekend