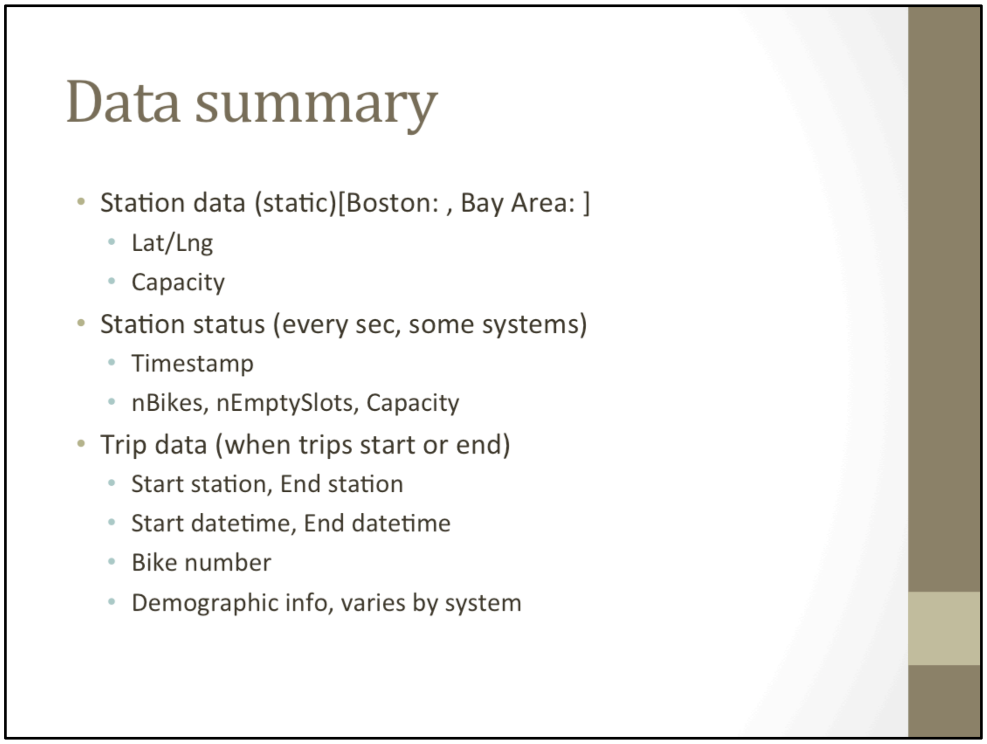
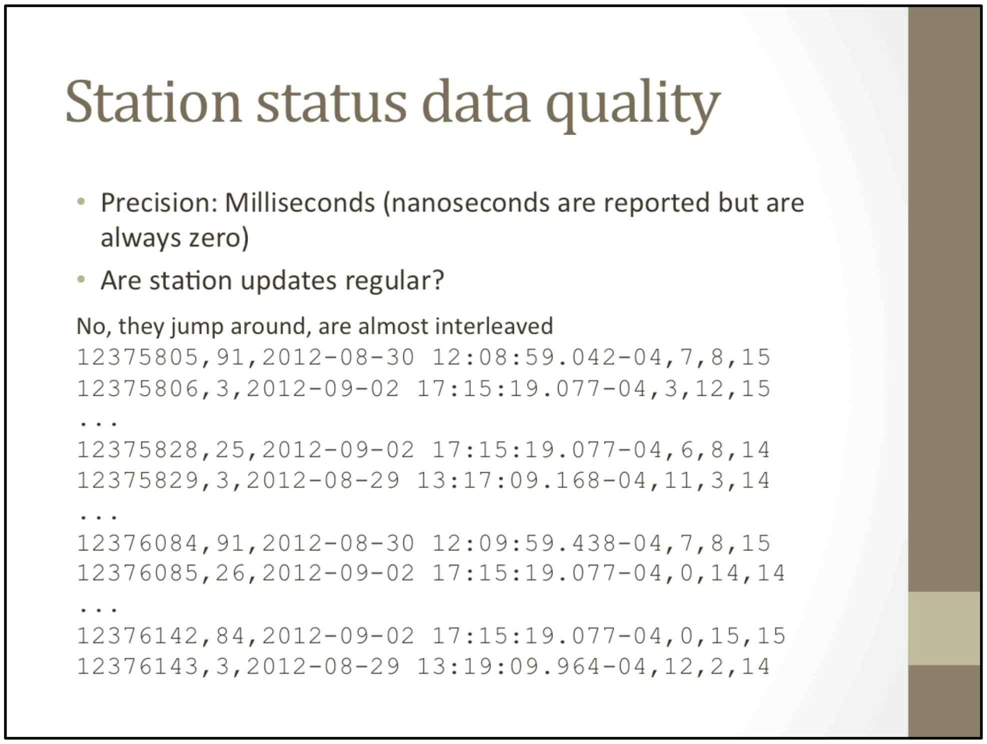
Demand patterns in bikeshare systems



Data summary

- Station data (static)[Boston: , Bay Area:]
  - Lat/Lng
  - Capacity
- Station status (every sec, some systems)
  - Timestamp
  - nBikes, nEmptySlots, Capacity
- Trip data (when trips start or end)
  - Start station, End station
  - Start datetime, End datetime
  - Bike number
  - Demographic info, varies by system

----- Meeting Notes (10/7/14 16:22) -----
Data summary: numbers for all of these

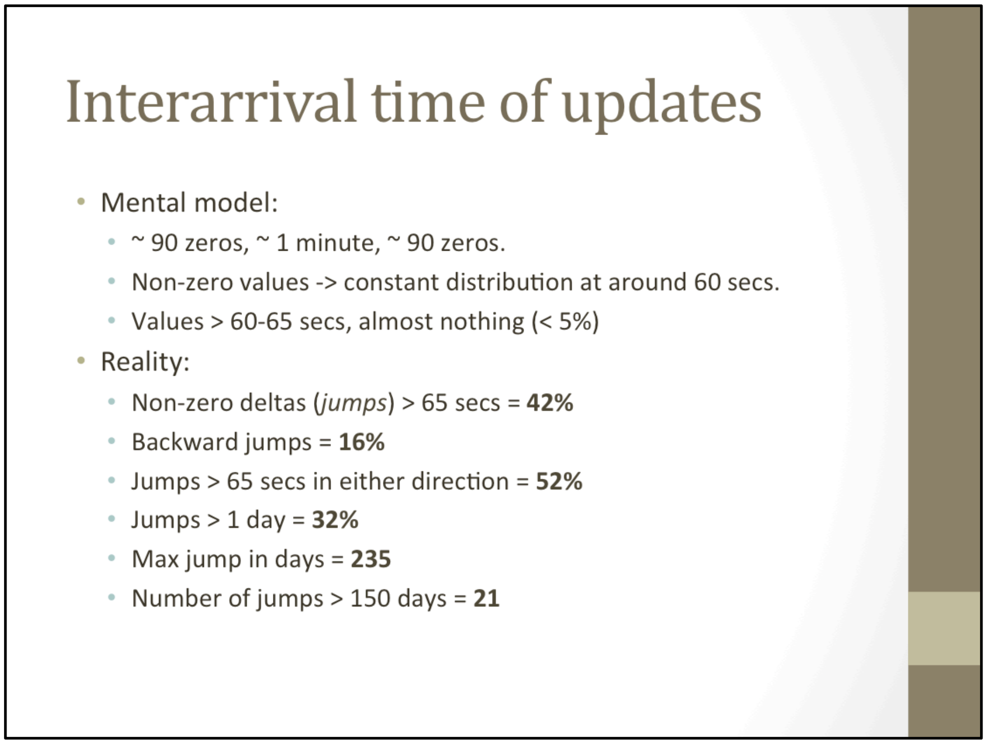


Station status data quality

- Precision: Milliseconds (nanoseconds are reported but are always zero)
- Are station updates regular?

No, they jump around, are almost interleaved

```
12375805,91,2012-08-30 12:08:59.042-04,7,8,15
12375806,3,2012-09-02 17:15:19.077-04,3,12,15
...
12375828,25,2012-09-02 17:15:19.077-04,6,8,14
12375829,3,2012-08-29 13:17:09.168-04,11,3,14
...
12376084,91,2012-08-30 12:09:59.438-04,7,8,15
12376085,26,2012-09-02 17:15:19.077-04,0,14,14
...
12376142,84,2012-09-02 17:15:19.077-04,0,15,15
12376143,3,2012-08-29 13:19:09.964-04,12,2,14
```

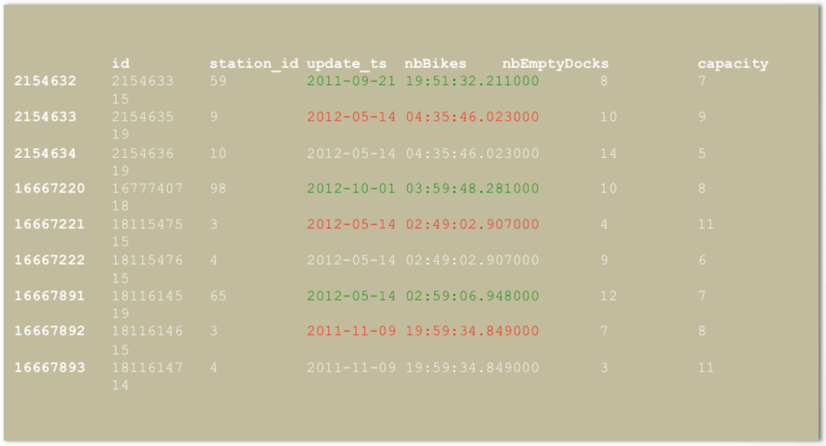


Interarrival time of updates

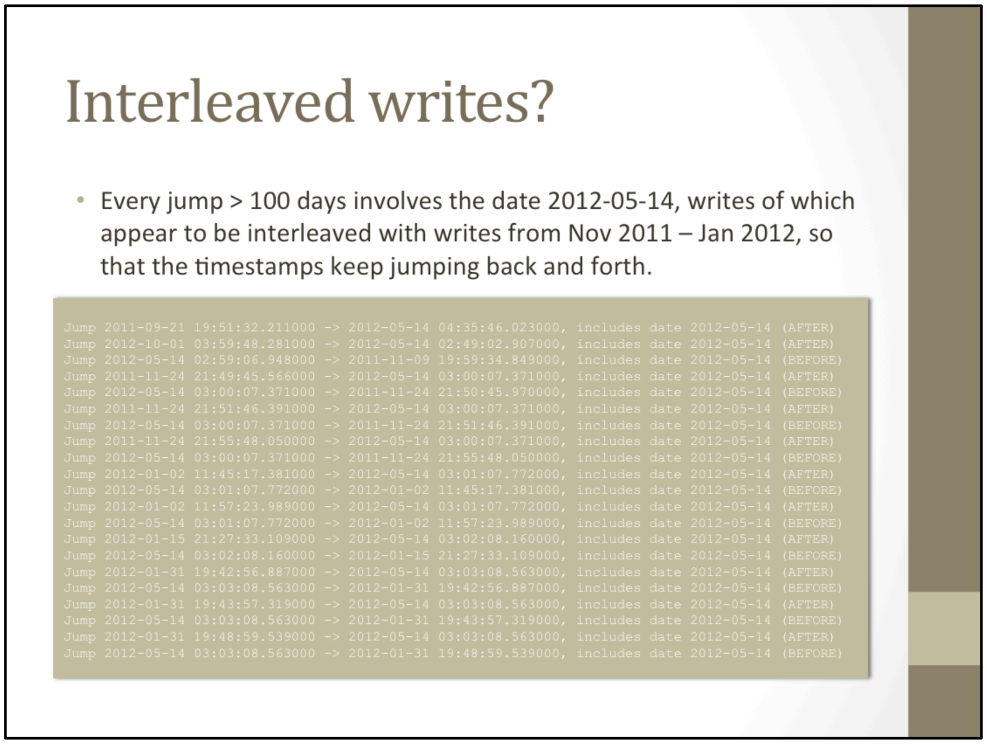
- Mental model:
  - ~ 90 zeros, ~ 1 minute, ~ 90 zeros.
  - Non-zero values -> constant distribution at around 60 secs.
  - Values > 60-65 secs, almost nothing (< 5%)
- Reality:
  - Non-zero deltas (*jumps*) > 65 secs = **42%**
  - Backward jumps = **16%**
  - Jumps > 65 secs in either direction = **52%**
  - Jumps > 1 day = **32%**
  - Max jump in days = **235**
  - Number of jumps > 150 days = **21**

Verify from the raw data

- Let's look at the raw data to confirm that our scripts are correct



```
            id          station_id update_ts  nbBikes    nbEmptyDocks          capacity
2154632     2154633     59          2011-09-21 19:51:32.211000        8            7
            15
2154633     2154635     9           2012-05-14 04:35:46.023000        10           9
            19
2154634     2154636     10          2012-05-14 04:35:46.023000        14           5
            19
16667220    16777407    98          2012-10-01 03:59:48.281000        10           8
            18
16667221    18115475    3           2012-05-14 02:49:02.907000        4            11
            15
16667222    18115476    4           2012-05-14 02:49:02.907000        9            6
            15
16667891    18116145    65          2012-05-14 02:59:06.948000        12           7
            19
16667892    18116146    3           2011-11-09 19:59:34.849000        7            8
            15
16667893    18116147    4           2011-11-09 19:59:34.849000        3            11
            14
```



Interleaved writes?

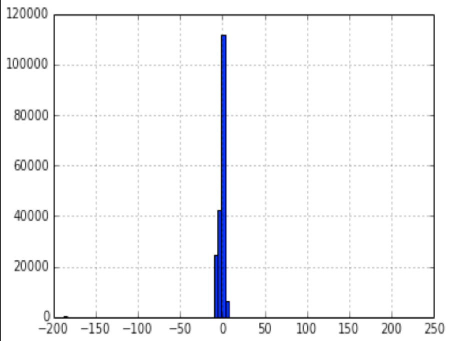
- Every jump > 100 days involves the date 2012-05-14, writes of which appear to be interleaved with writes from Nov 2011 – Jan 2012, so that the timestamps keep jumping back and forth.

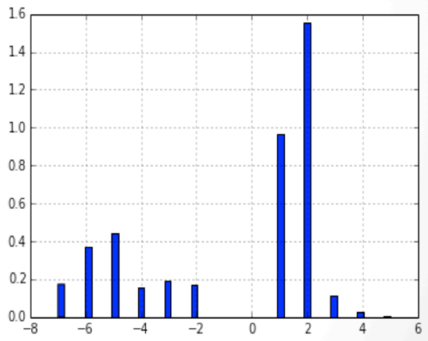
```
Jump 2011-09-21 19:51:32.211000 -> 2012-05-14 04:35:46.023000, includes date 2012-05-14 (AFTER)
Jump 2012-10-01 03:59:48.281000 -> 2012-05-14 02:49:02.907000, includes date 2012-05-14 (AFTER)
Jump 2012-05-14 02:59:06.948000 -> 2011-11-09 19:59:34.849000, includes date 2012-05-14 (BEFORE)
Jump 2011-11-24 21:49:45.566000 -> 2012-05-14 03:00:07.371000, includes date 2012-05-14 (AFTER)
Jump 2012-05-14 03:00:07.371000 -> 2011-11-24 21:50:45.970000, includes date 2012-05-14 (BEFORE)
Jump 2011-11-24 21:51:46.391000 -> 2012-05-14 03:00:07.371000, includes date 2012-05-14 (AFTER)
Jump 2012-05-14 03:00:07.371000 -> 2011-11-24 21:51:46.391000, includes date 2012-05-14 (BEFORE)
Jump 2011-11-24 21:55:48.050000 -> 2012-05-14 03:00:07.371000, includes date 2012-05-14 (AFTER)
Jump 2012-05-14 03:00:07.371000 -> 2011-11-24 21:55:48.050000, includes date 2012-05-14 (BEFORE)
Jump 2012-01-02 11:45:17.381000 -> 2012-05-14 03:01:07.772000, includes date 2012-05-14 (AFTER)
Jump 2012-05-14 03:01:07.772000 -> 2012-01-02 11:45:17.381000, includes date 2012-05-14 (BEFORE)
Jump 2012-01-02 11:57:23.989000 -> 2012-05-14 03:01:07.772000, includes date 2012-05-14 (AFTER)
Jump 2012-05-14 03:01:07.772000 -> 2012-01-02 11:57:23.989000, includes date 2012-05-14 (BEFORE)
Jump 2012-01-15 21:27:33.109000 -> 2012-05-14 03:02:08.160000, includes date 2012-05-14 (AFTER)
Jump 2012-05-14 03:02:08.160000 -> 2012-01-15 21:27:33.109000, includes date 2012-05-14 (BEFORE)
Jump 2012-01-31 19:42:56.887000 -> 2012-05-14 03:03:08.563000, includes date 2012-05-14 (AFTER)
Jump 2012-05-14 03:03:08.563000 -> 2012-01-31 19:42:56.887000, includes date 2012-05-14 (BEFORE)
Jump 2012-01-31 19:43:57.319000 -> 2012-05-14 03:03:08.563000, includes date 2012-05-14 (AFTER)
Jump 2012-05-14 03:03:08.563000 -> 2012-01-31 19:43:57.319000, includes date 2012-05-14 (BEFORE)
Jump 2012-01-31 19:48:59.539000 -> 2012-05-14 03:03:08.563000, includes date 2012-05-14 (AFTER)
Jump 2012-05-14 03:03:08.563000 -> 2012-01-31 19:48:59.539000, includes date 2012-05-14 (BEFORE)
```

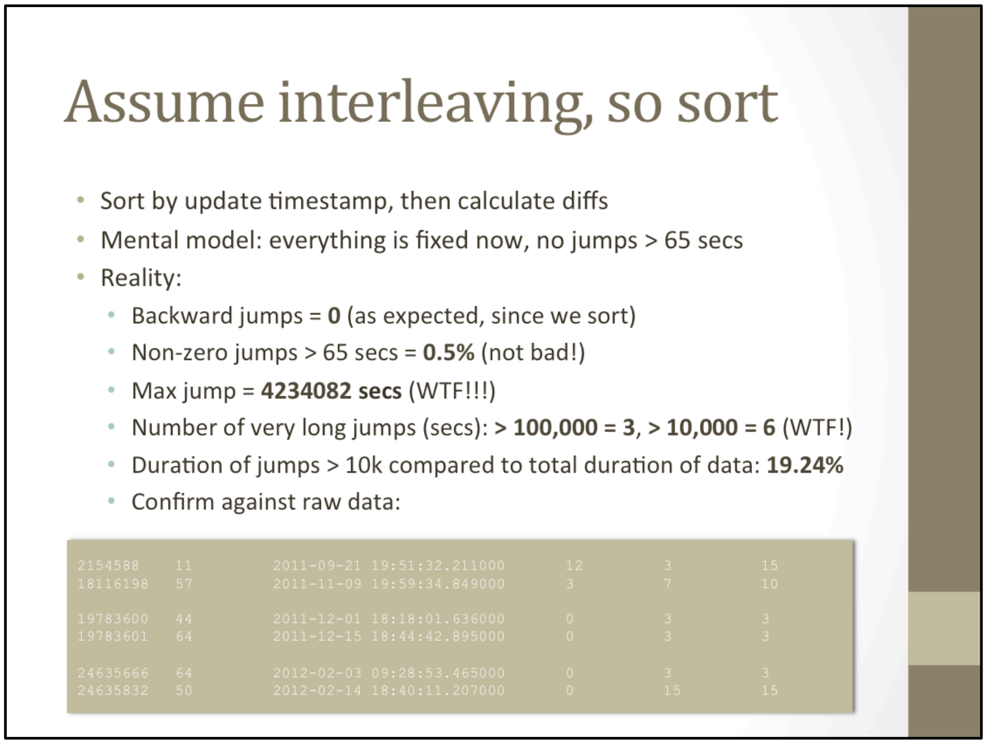
----- Meeting Notes (10/7/14 16:22) -----
Throw out bad data until you get to good data

Are all jumps interleaving?

- Mental model
  - Equivalent numbers of positive and negative jumps
- Reality (has a positive bias ☺)





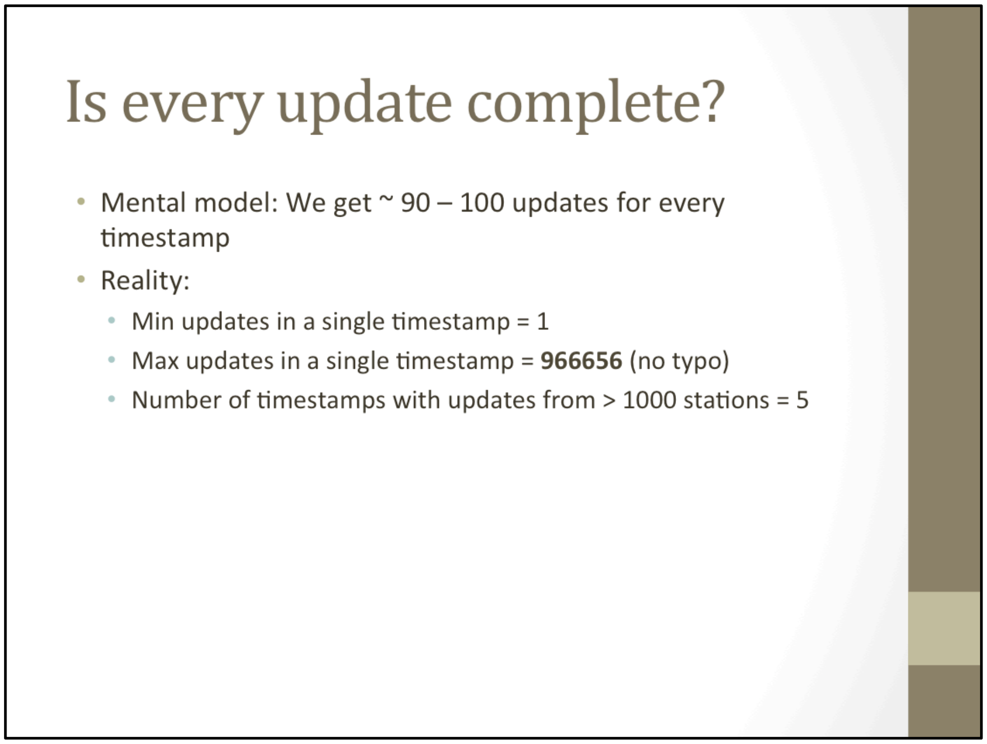


Assume interleaving, so sort

- Sort by update timestamp, then calculate diffs
- Mental model: everything is fixed now, no jumps > 65 secs
- Reality:
  - Backward jumps = **0** (as expected, since we sort)
  - Non-zero jumps > 65 secs = **0.5%** (not bad!)
  - Max jump = **4234082 secs** (WTF!!!)
  - Number of very long jumps (secs): **> 100,000 = 3**, **> 10,000 = 6** (WTF!)
  - Duration of jumps > 10k compared to total duration of data: **19.24%**
  - Confirm against raw data:

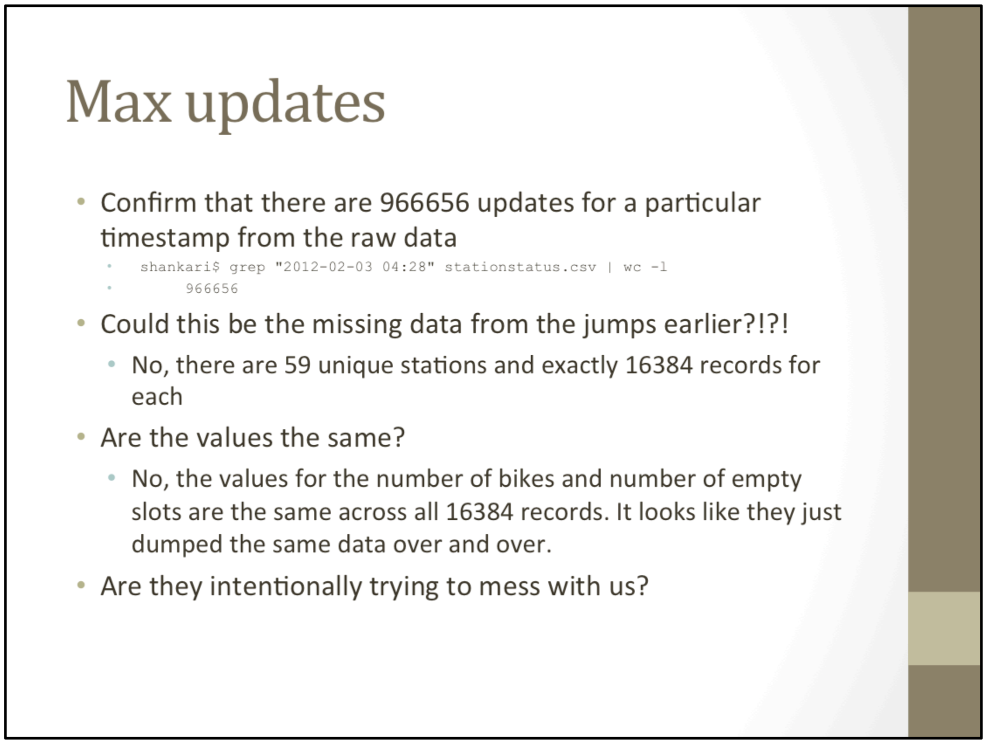
```
2154588    11          2011-09-21 19:51:32.211000      12          3           15
18116198   57          2011-11-09 19:59:34.849000      3           7           10

19783600   44          2011-12-01 18:18:01.636000      0           3           3
19783601   64          2011-12-15 18:44:42.895000      0           3           3

24635666   64          2012-02-03 09:28:53.465000      0           3           3
24635832   50          2012-02-14 18:40:11.207000      0           15          15
```

----- Meeting Notes (10/7/14 16:22) -----
Go back to data provider
Check if ids were missing



Is every update complete?

- Mental model: We get ~ 90 – 100 updates for every timestamp
- Reality:
  - Min updates in a single timestamp = 1
  - Max updates in a single timestamp = **966656** (no typo)
  - Number of timestamps with updates from > 1000 stations = 5



Max updates

- Confirm that there are 966656 updates for a particular timestamp from the raw data

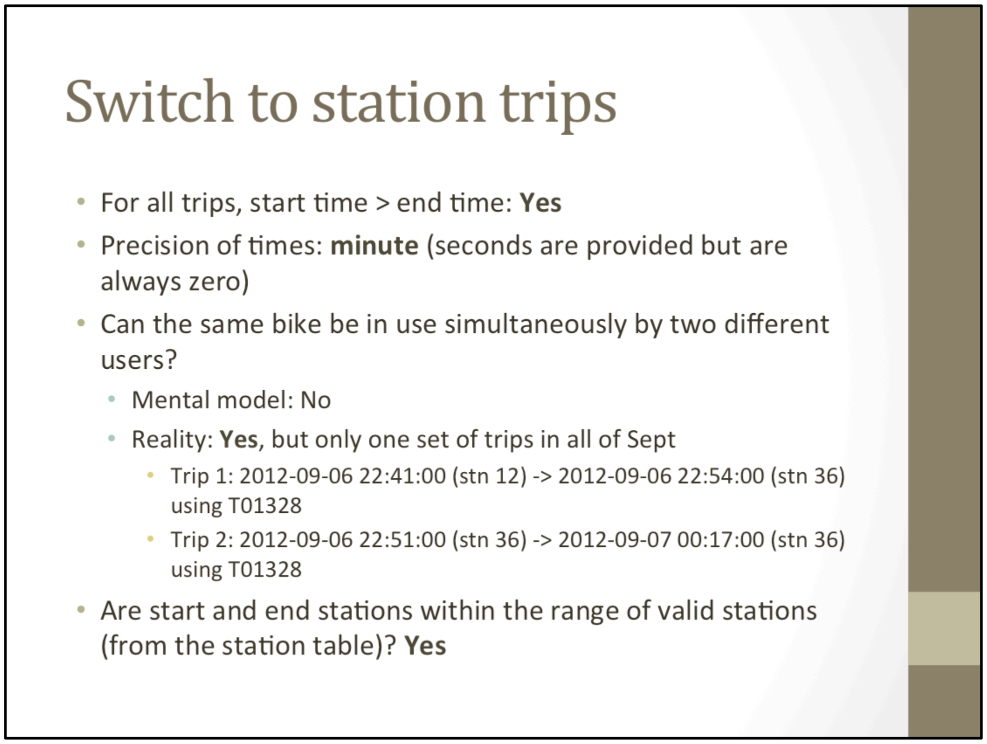
```
shankari$ grep "2012-02-03 04:28" stationstatus.csv | wc -l
      966656
```

- Could this be the missing data from the jumps earlier?!?!
  - No, there are 59 unique stations and exactly 16384 records for each
- Are the values the same?
  - No, the values for the number of bikes and number of empty slots are the same across all 16384 records. It looks like they just dumped the same data over and over.
- Are they intentionally trying to mess with us?

Min updates

- There are 7679 timestamps that have less than 10 updates.
- There are 38 timestamps that have a single update.
- Mental model: Maybe the updates are just trickling in more slowly due to network delays. Let's look at the raw data.
  - Reality: Initially, all the updates are from the same two stations (44 and 41). Then 41 stops.
  - We are getting the same 1 minute interval updates as always, but they contain information about only one station.
  - We get an update from station 44, and only station 44, for an entire two hours during the afternoon commute on 1st Dec 2011.
- Abandon station status -> focus on station trips!

```
2011-12-01 16:50:28.822000    19783569  44         0          3          3
2011-12-01 16:51:29.153000    19783570  44         0          3          3
2011-12-01 16:52:29.532000    19783571  44         0          3          3
…………..
2011-12-01 18:15:00.603000    19783597  44         0          3          3
2011-12-01 18:16:00.960000    19783598  44         0          3          3
2011-12-01 18:17:01.287000    19783599  44         0          3          3
```

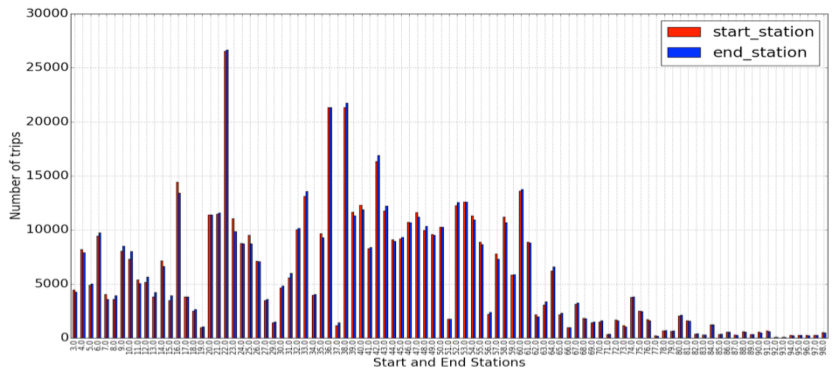


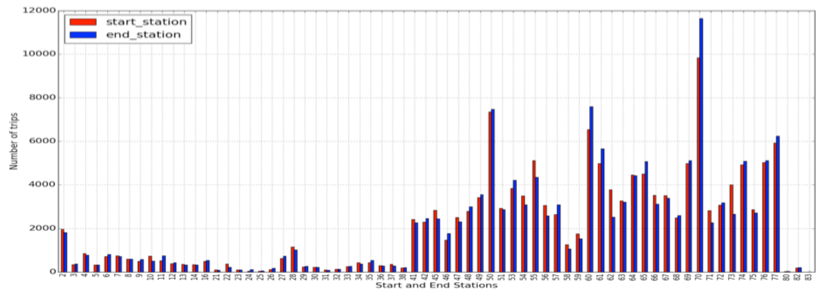
Switch to station trips

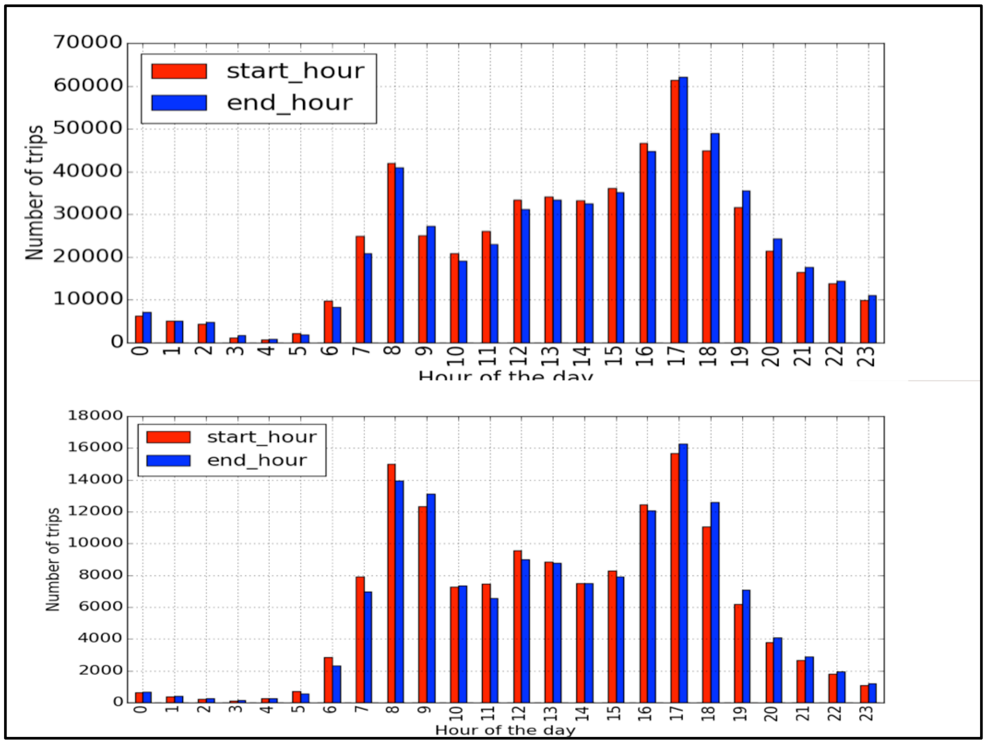
- For all trips, start time > end time: **Yes**
- Precision of times: **minute** (seconds are provided but are always zero)
- Can the same bike be in use simultaneously by two different users?
  - Mental model: No
  - Reality: **Yes**, but only one set of trips in all of Sept
    - Trip 1: 2012-09-06 22:41:00 (stn 12) -> 2012-09-06 22:54:00 (stn 36) using T01328
    - Trip 2: 2012-09-06 22:51:00 (stn 36) -> 2012-09-07 00:17:00 (stn 36) using T01328
- Are start and end stations within the range of valid stations (from the station table)? **Yes**

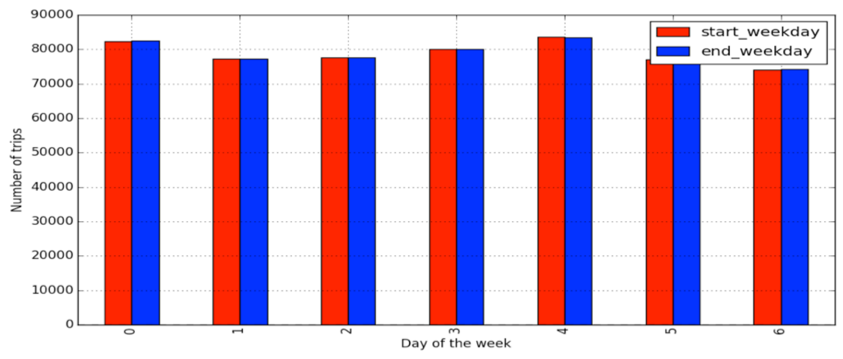
----- Meeting Notes (10/7/14 16:22) -----
Are trip durations reasonable?
end station is empty
bike numbers W343435 v/s 0x343
weather
how often do people forget to return trips
Model trips with long durations separately

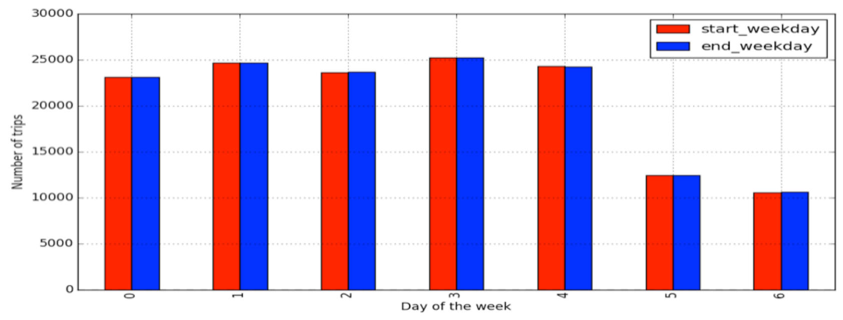
BASIC STATS

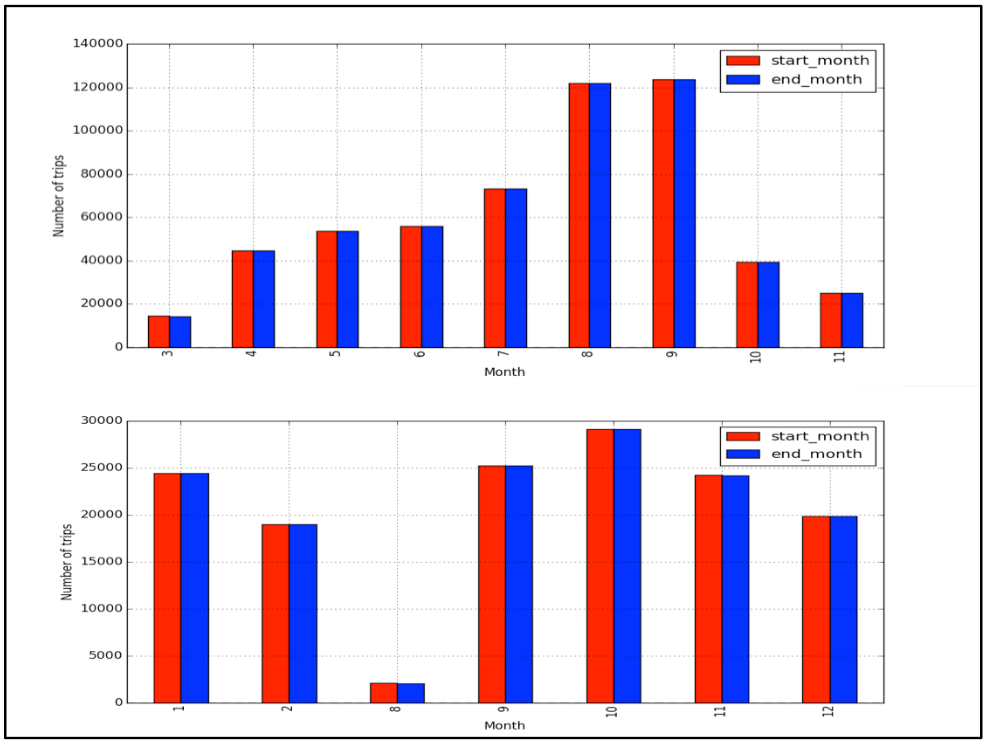

start_station
end_station
Number of trips
Start and End Stations


start_station
end_station
Number of trips
Start and End Stations

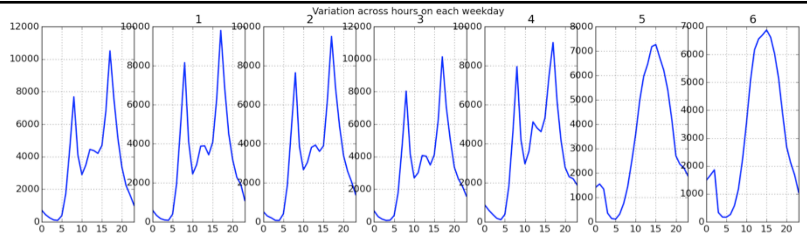

start_hour
end_hour
Number of trips
Hour of the day

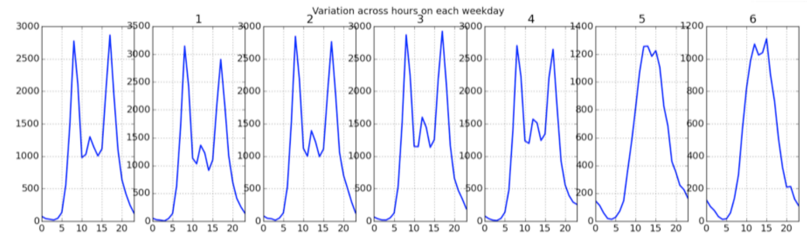

90000
80000
70000
60000
50000
40000
30000
20000
10000
0
Number of trips
start_weekday
end_weekday
0
1
2
3
4
5
6
Day of the week


30000
25000
20000
15000
10000
5000
0
Number of trips
start_weekday
end_weekday
0
1
2
3
4
5
6
Day of the week

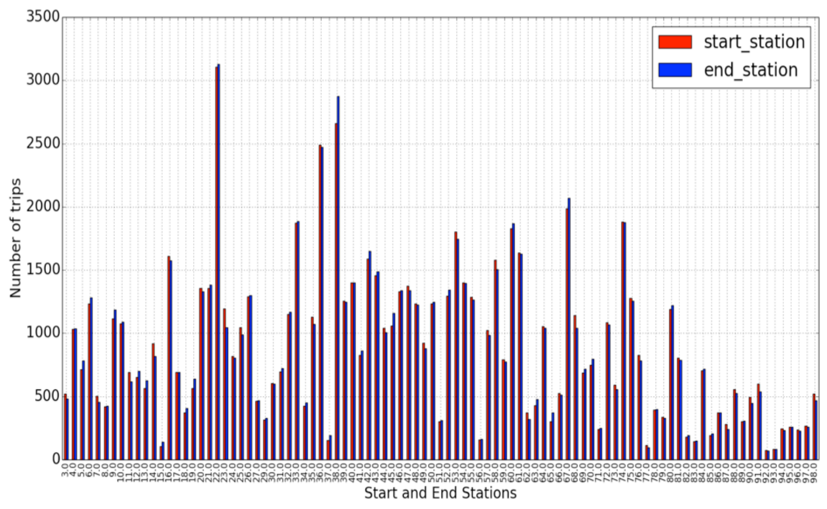


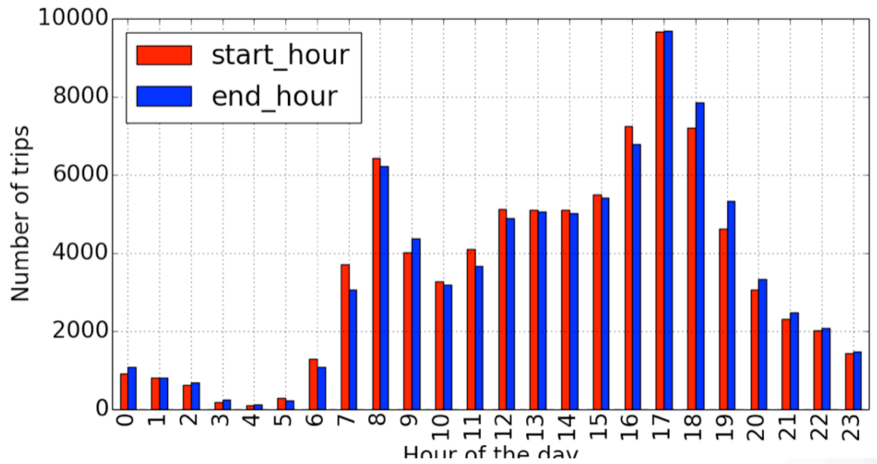
----- Meeting Notes (10/7/14 16:22) -----
reorder

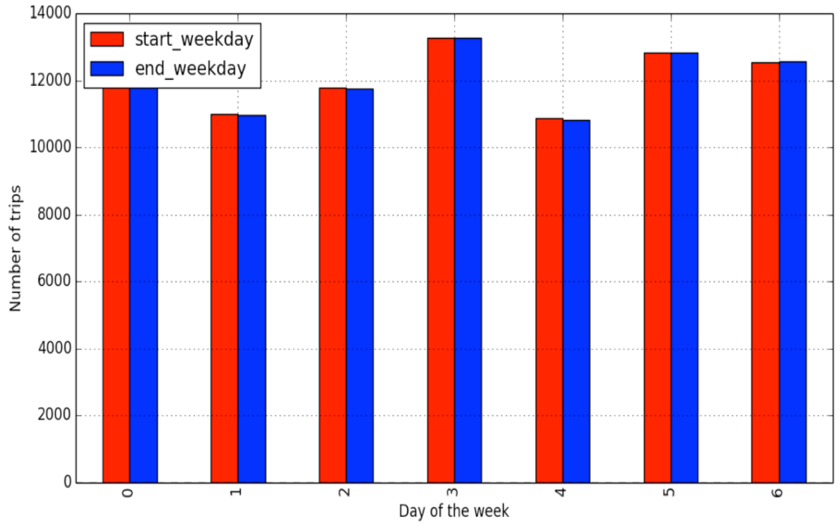

Variation across hours on each weekday
1
2
3
4
5
6


Variation across hours on each weekday
1
2
3
4
5
6

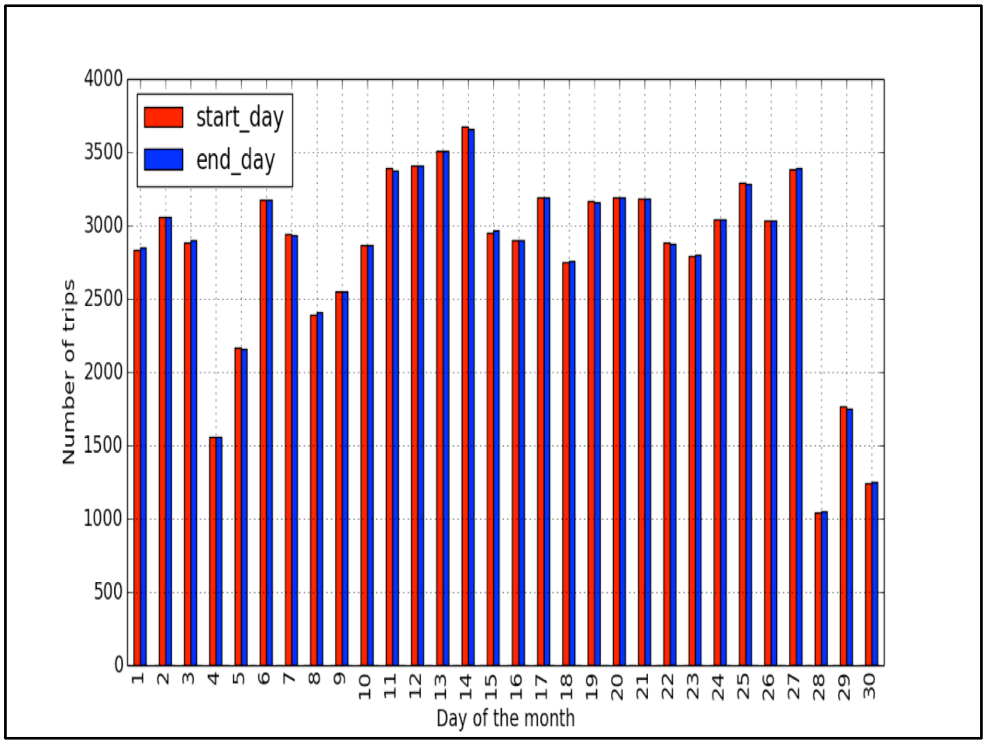
Basic trip characteristics

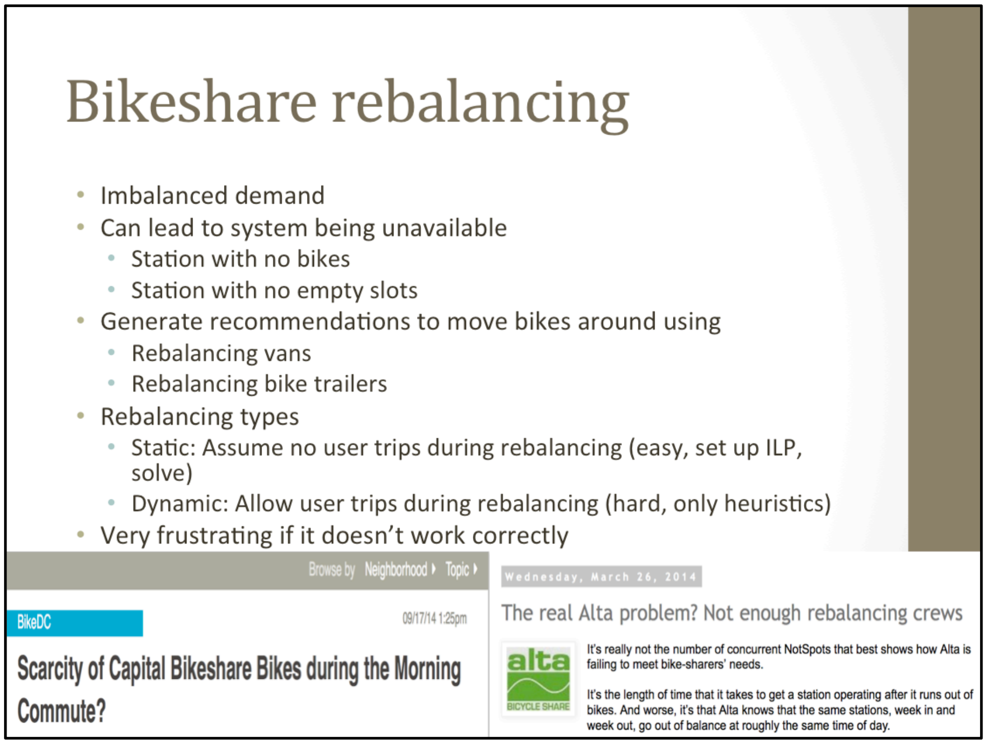



start_hour
end_hour
Number of trips
Hour of the day
0
2000
4000
6000
8000
10000
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23



start_weekday
end_weekday
Number of trips
Day of the week
0
2000
4000
6000
8000
10000
12000
14000
0
1
2
3
4
5
6


start_day
end_day
Number of trips
Day of the month
0
500
1000
1500
2000
2500
3000
3500
4000
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30



Bikeshare rebalancing

- Imbalanced demand
- Can lead to system being unavailable
  - Station with no bikes
  - Station with no empty slots
- Generate recommendations to move bikes around using
  - Rebalancing vans
  - Rebalancing bike trailers
- Rebalancing types
  - Static: Assume no user trips during rebalancing (easy, set up ILP, solve)
  - Dynamic: Allow user trips during rebalancing (hard, only heuristics)
- Very frustrating if it doesn't work correctly

Browse by Neighborhood ▸ Topic ▸

BikeDC 09/17/14 1:25pm

Scarcity of Capital Bikeshare Bikes during the Morning Commute?

Wednesday, March 26, 2014

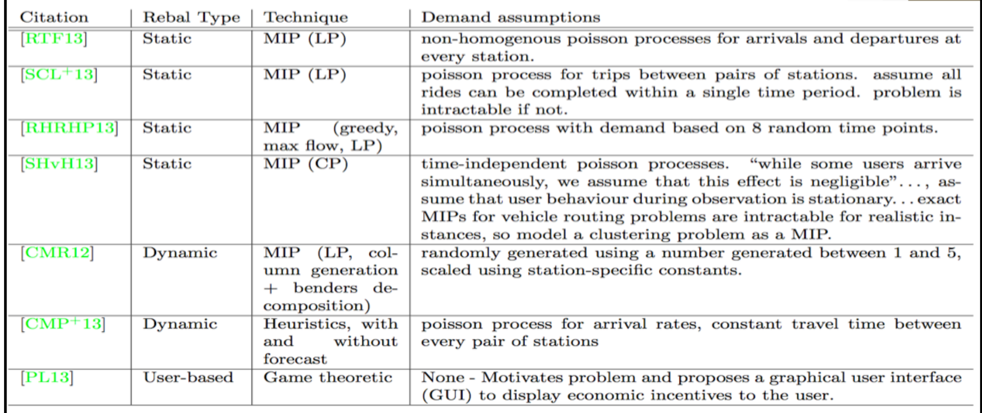
The real Alta problem? Not enough rebalancing crews

It's really not the number of concurrent NotSpots that best shows how Alta is failing to meet bike-sharers' needs.

It's the length of time that it takes to get a station operating after it runs out of bikes. And worse, it's that Alta knows that the same stations, week in and week out, go out of balance at roughly the same time of day.

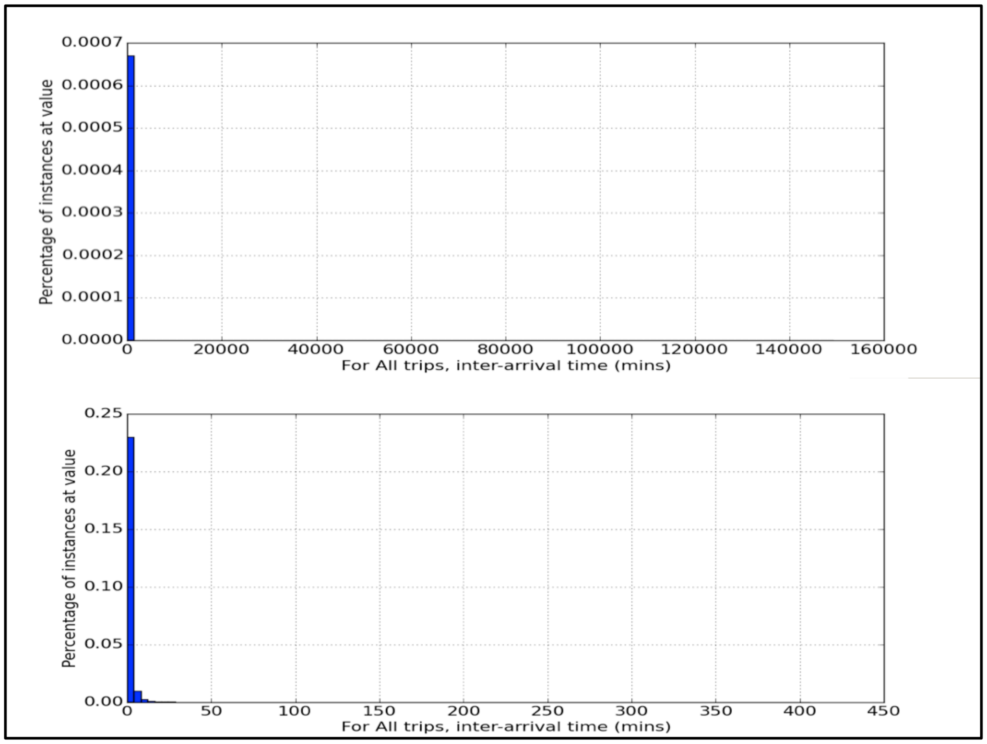
Are the arrival rates poisson?

- Existing techniques are based on package pick up and delivery
  - Assume poisson arrival rates, set up constraints, use ILP, CP or similar heuristics to solve

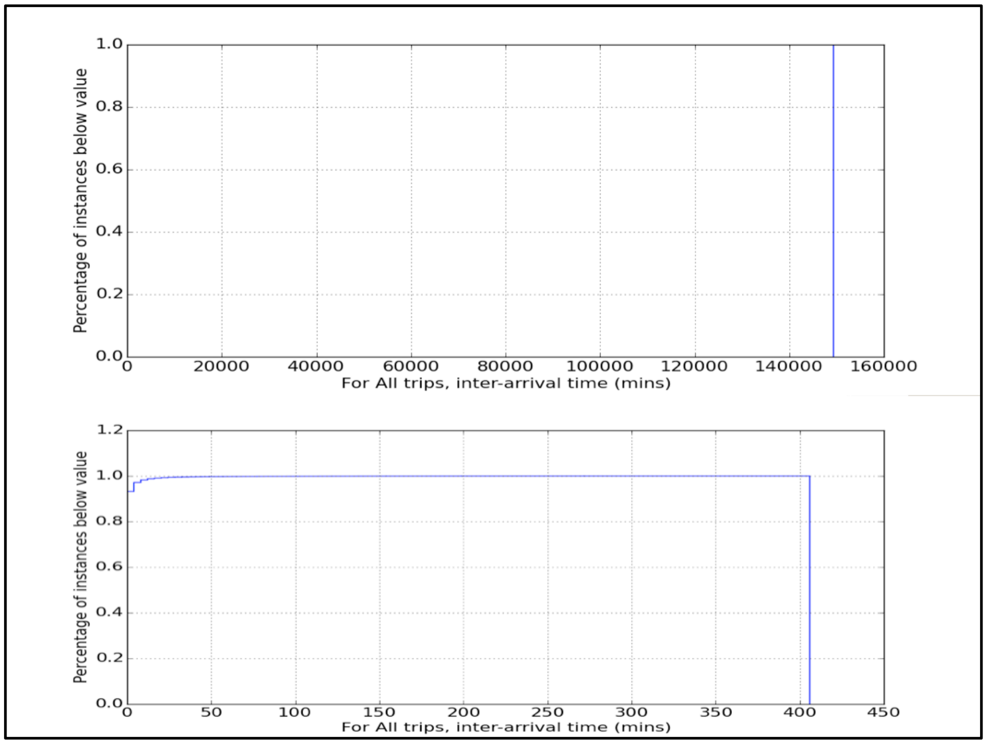


Citation	Rebal Type	Technique	Demand assumptions
[RTF13]	Static	MIP (LP)	non-homogenous poisson processes for arrivals and departures at every station.
[SCL+13]	Static	MIP (LP)	poisson process for trips between pairs of stations. assume all rides can be completed within a single time period. problem is intractable if not.
[RHRHP13]	Static	MIP (greedy, max flow, LP)	poisson process with demand based on 8 random time points.
[SHvH13]	Static	MIP (CP)	time-independent poisson processes. "while some users arrive simultaneously, we assume that this effect is negligible"..., assume that user behaviour during observation is stationary... exact MIPs for vehicle routing problems are intractable for realistic instances, so model a clustering problem as a MIP.
[CMR12]	Dynamic	MIP (LP, column generation + benders decomposition)	randomly generated using a number generated between 1 and 5, scaled using station-specific constants.
[CMP+13]	Dynamic	Heuristics, with and without forecast	poisson process for arrival rates, constant travel time between every pair of stations
[PL13]	User-based	Game theoretic	None - Motivates problem and proposes a graphical user interface (GUI) to display economic incentives to the user.

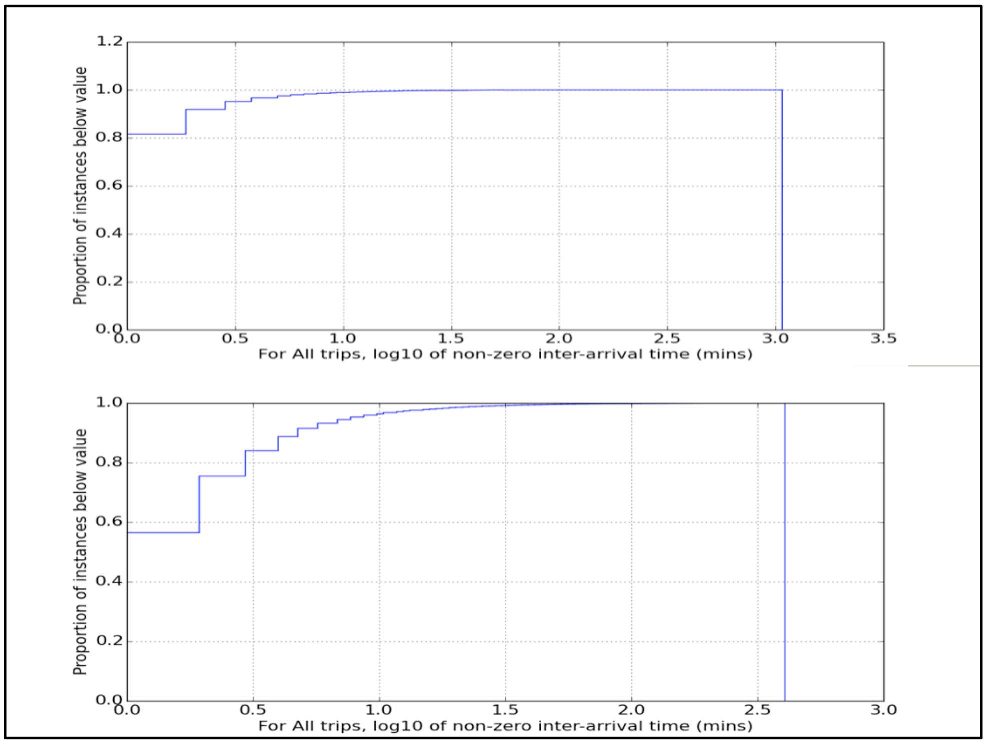
Table 2: Summary of prior work



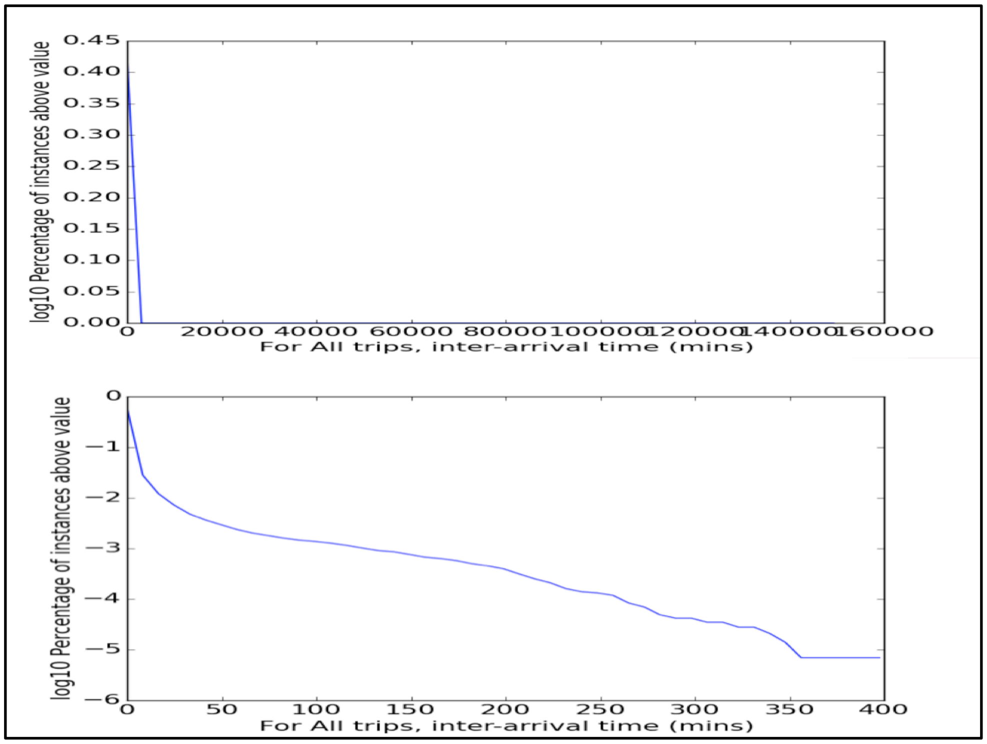
As we can see, there is a giant spike around 0, which drowns almost everything out, but there is a really long tail where the max inter-arrival times are really long. Part of the reason that max time is so high in the Boston case is because the entire system is shut down during the winter months.
This combination of high peak around zero and long tail indicates a strong peak load pattern – during “rush hours”, there are a lot of requests to the system, and during the night, there are long stretches where there are no trips. We can see this from the graphs in slide 18 as well.



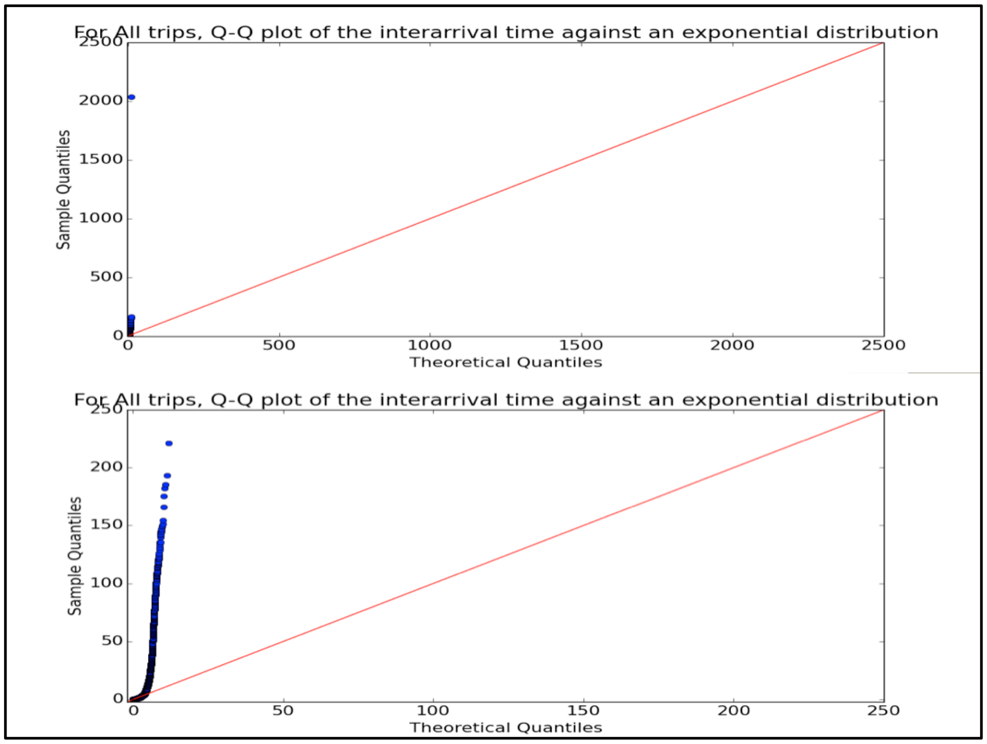
Here are the cdfs for the same data.
This is pretty useless, so let's switch to log scales. Unfortunately, as we can see from the pdf, a lot of the inter-arrival times are zero, so we need to strip them out first.
Question: is that OK? How do we address zero values in a log plot?



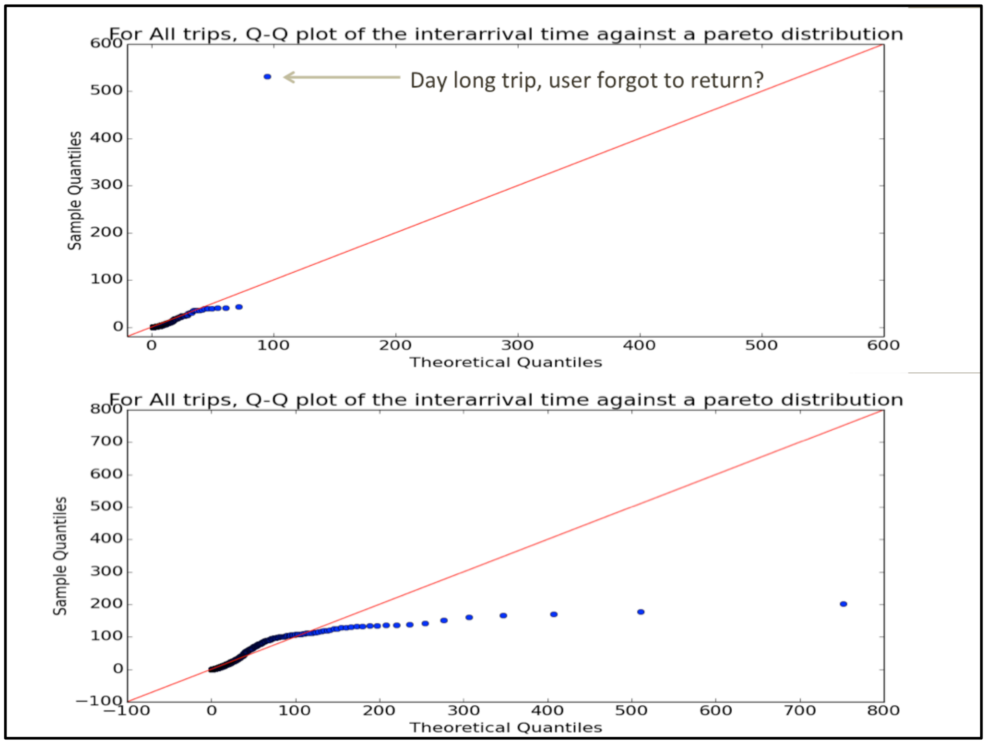
That at least shows some differentiation, but the majority of trips are still around 0, which corresponds to an inter-arrival time of 1.



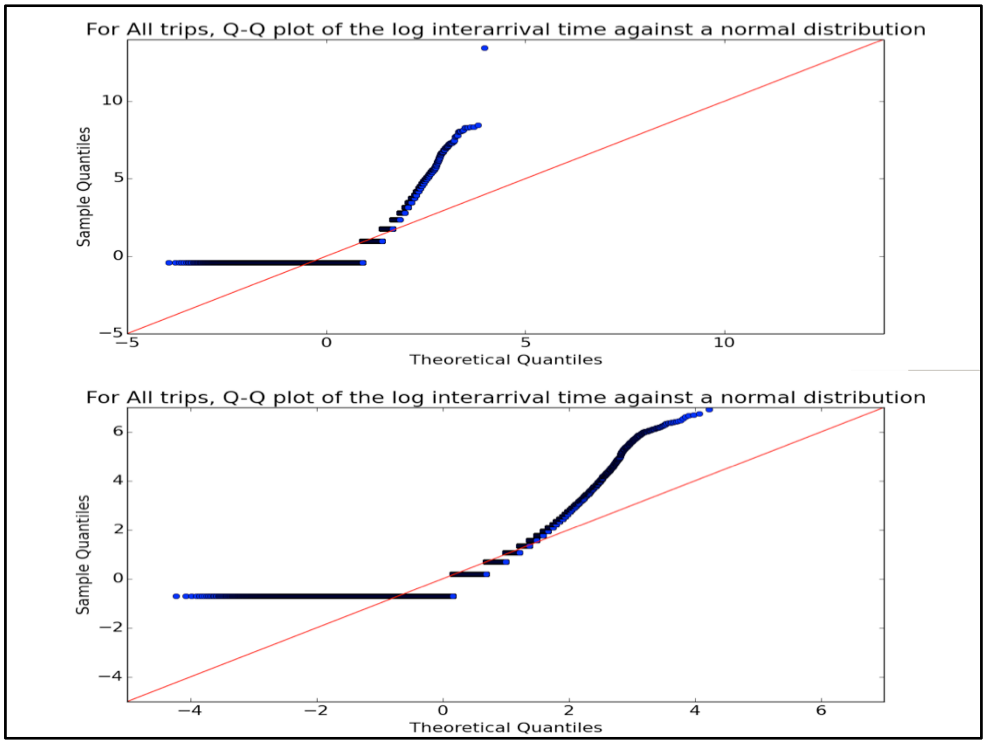
This is the complementary log plot. Per vern: This plots the *complementary* CDF (1-F(x)) with the Y axis log-scaled. For that, an exponential distribution will be a straight line. As we can see, for the Boston case, the outlier effect is so strong that the plot is almost a straight vertical line. How do we deal with this? log-log plot?
The Bay Area case has a nicer curve, but still really concentrated around short interarrival times.



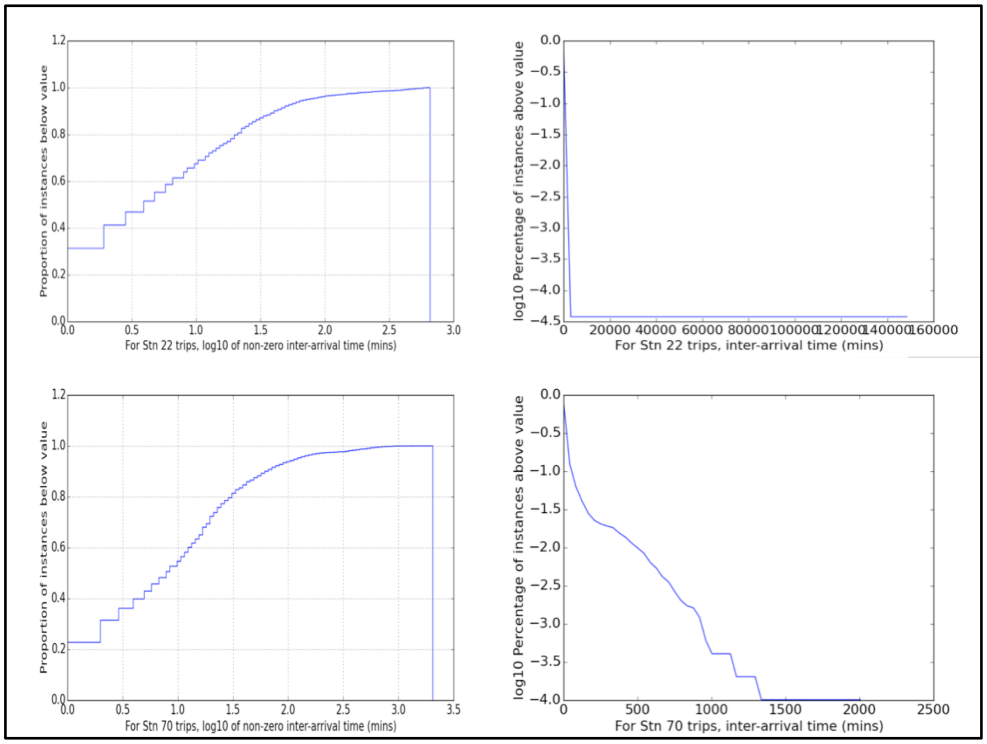
If it is poisson, we would expect the Q-Q plot of the inter-arrival times to match the exponential distribution. As we can see, we are not even close.



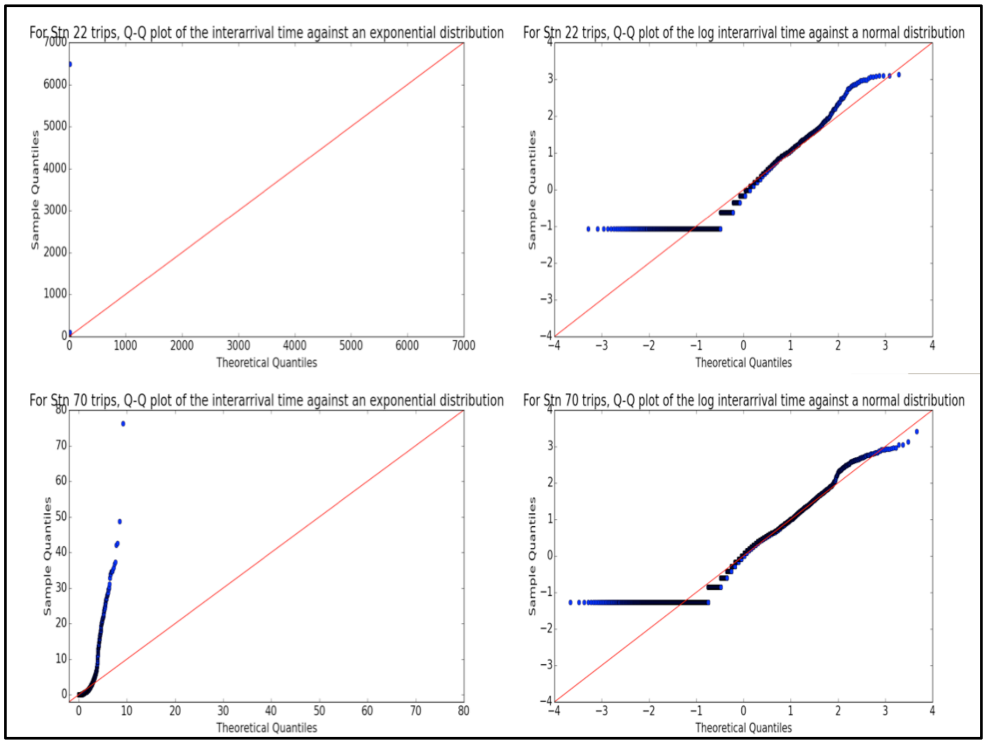
There is a much better match for the pareto distribution, except at the tails. The comparison is also distorted by a couple of extreme outliers. I could replot after removing them, but they appear to be legitimate trips, so I decided to move to station specific plots anyway.



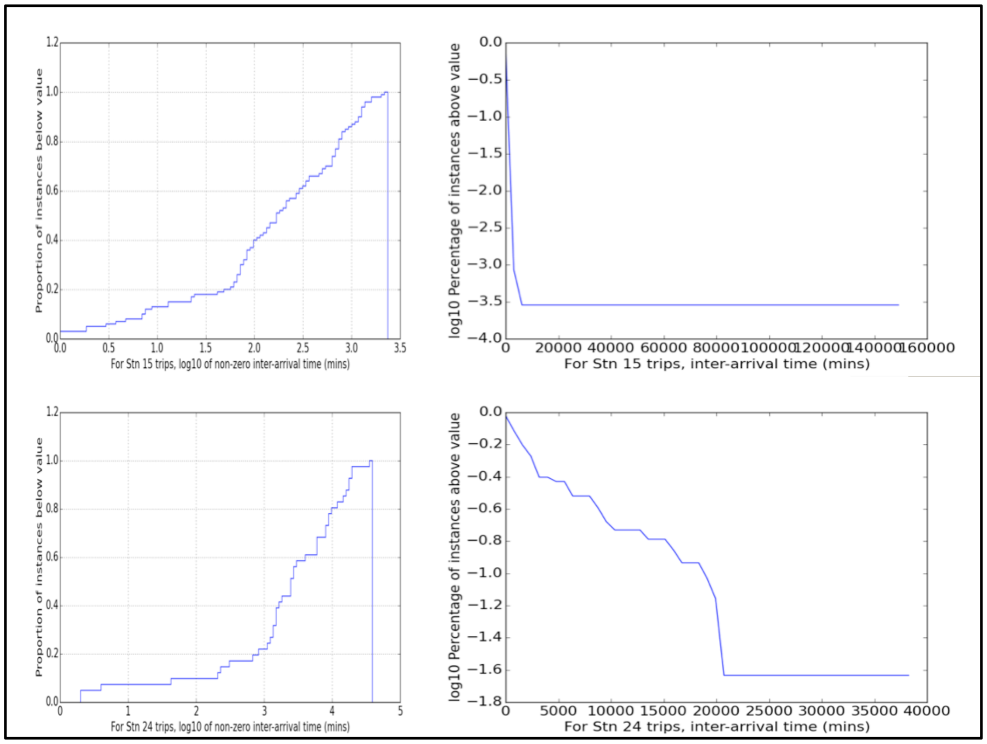
Taking logs helps reduce the effect of the outlier, but there are still no good matches here.



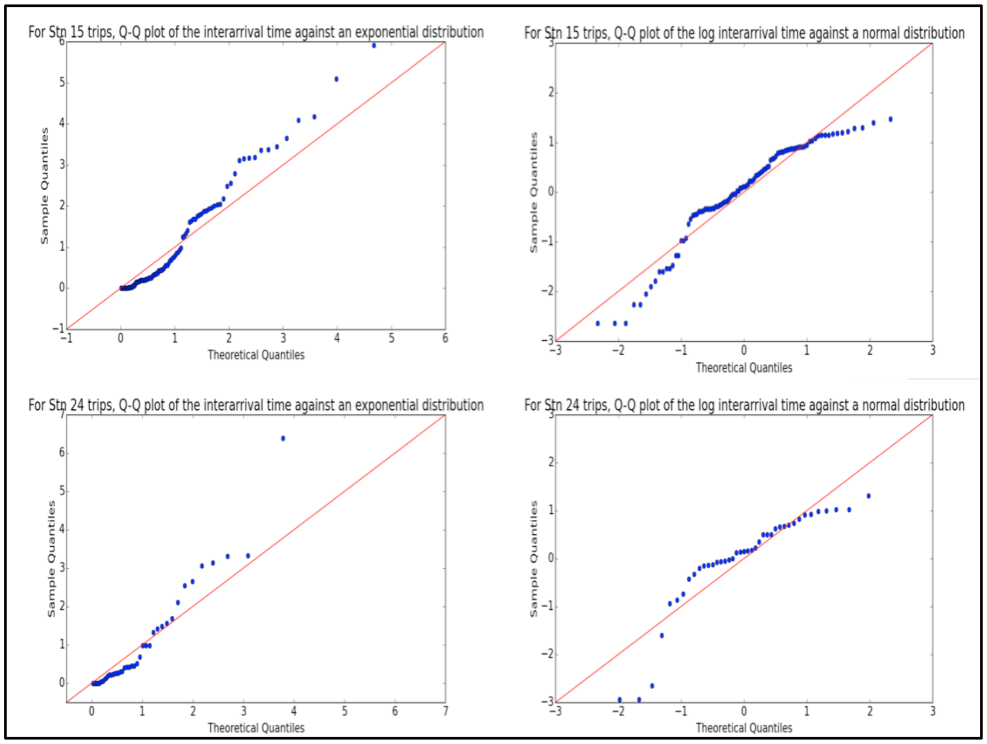
Since most models assume station specific poisson rates, let's look at station specific arrival rates rather than the arrival rate of the whole system.
Here, we look at the busiest stations in the entire system – station 22 for boston and station 70 for the bay area.



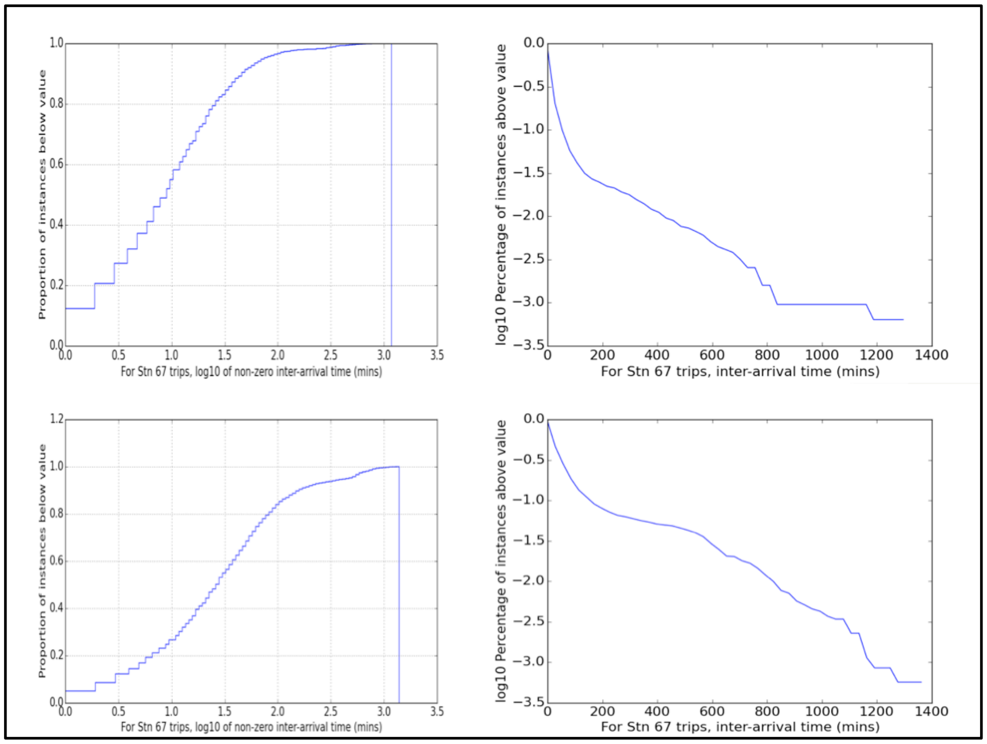
There is no match for the poisson (top qq-plot), but the middle part of the pareto looks pretty good. The pareto doesn't match at either of the tails, though. Note that the lower tail is actually more pronounced because we had to strip out the zeros for the log plot.



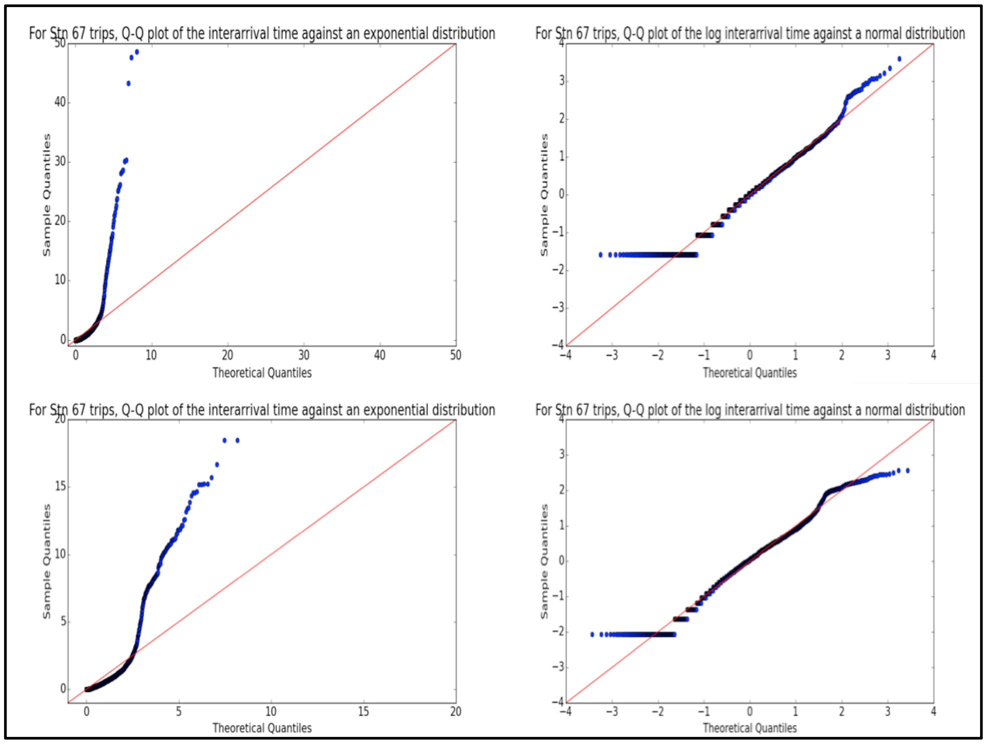
These are the plots for station 15, a station with one of the lowest loads. We can see a very differently shaped cdf here, much less steep around zero inter arrival time.



For this station, the poisson is not a bad fit, although the pareto is still a bit better.



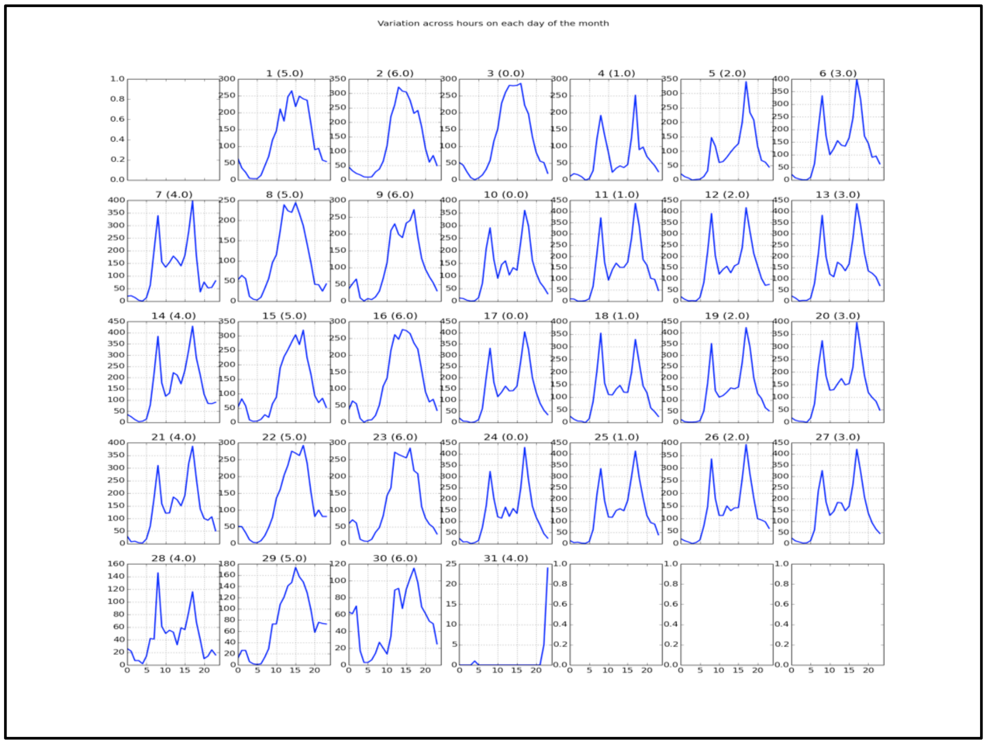
These are the curves for station 67, which I picked randomly using the randrange function. It is a lot closer to the busy station than the idle station in terms of inter arrivals.



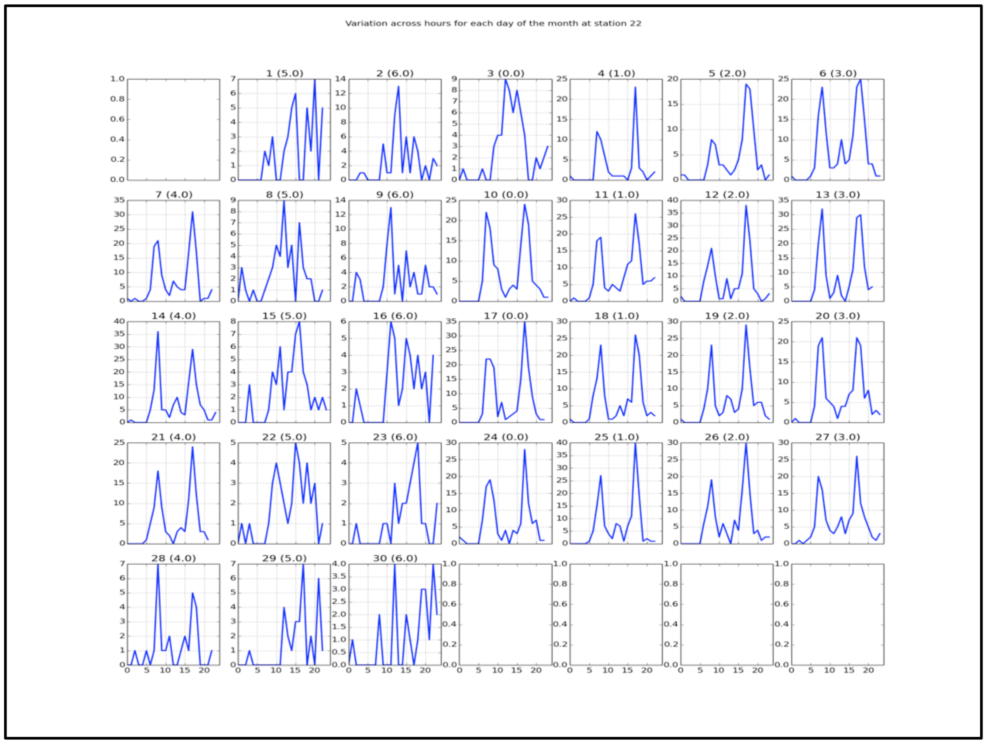
And this is reflected in the Q-Q plots as well.

MORE COMPLEXITY AT SMALLER SCALES

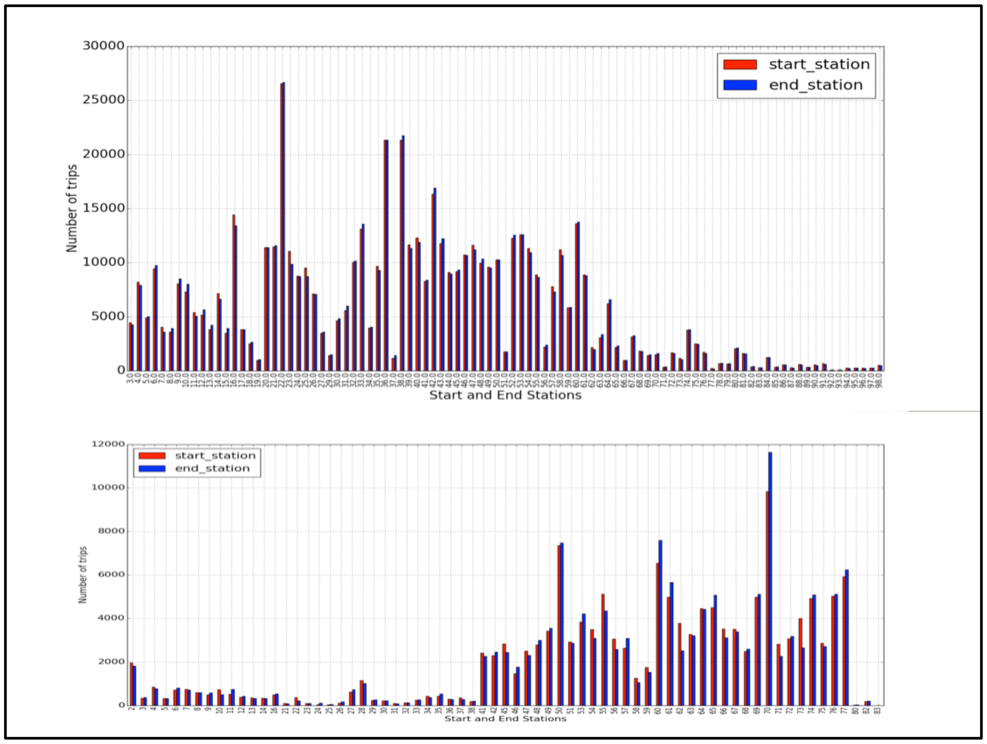
So far, we have been looking at all the data, across large time scales.
However, building algorithms based on smoothed data across large time scales can run into issues with local perturbations that don't match the smoothed data.
So I look a look at smaller scale data as well and saw how well the curves matched.



Here's the variation across hours on each day of Sep 2012 (the busiest month) of the Boston data. We can see that although the two humped pattern is prety consistent, the magnitude of the peaks varies – sometimes, the first one is much smaller than the second one and sometimes it is not. Even the shape can be different sometimes – check out day 3 or day 5 versus the same day in other weeks.



Going to an even smaller scale, here's the variation over hours per day for a single station (22) in Boston in Sept 2012. As we can see, the curves seem a lot less consistent here. Not just the curves, but also the magnitudes are very different. Consider day of week = 4. The peak appears to range between 7 and 40. Planning for the mean will fail miserably since the pool of bikes and slots is not infinite – an unexpected surge will lead to exhaustion.



A final note on static versus dynamic rebalancing. If you recall the material from slide 23 – in static rebalancing, the rebalancing trucks run overnight, and the system is assumed to be in stasis, with no load, during the run. In dynamic rebalancing, there is ongoing load in the system. There is an interesting mix of arrival demand assumptions and algorithms. The dynamic algorithms either assume homogenous poisson, or (I think) a uniform distribution. The static algorithms assume either homogenous or non-homogenous poission.

What I wanted to show in this slide was that in the San Francisco case, static rebalancing looks like it will help. There are certain sections that are clearly trip sources and certain sections that are trip sinks, and so you would want to rebalance between them. But in the Boston case, we don't see such a pattern. Incoming and outgoing trips seem pretty evenly balanced at all stations. So in this case, the unavailability is primarily caused by an insufficiently large buffer, and dynamic rebalancing is needed to handle availability. I think that there is the opportunity to come up with an algorithm for this case that doesn't rely on homogenous poisson.

Questions that I had

- Some of the papers assume non-homogenous processes. How do you check if your distribution fits a non-homogenous process?
- Do you look at all the data? Or look at subsets?
  - If at subsets, how to select them?
- It is not enough to look at arrival rates of users, we also need to look at arrival rates of bikes. If a particular station has a user arrival rate (bike departure rate) of n, but a bike arrival and return rate also of n, then it doesn't need any rebalancing. So we really need to look at flow. But flow implies a time period for the flow. How do we deal with that?