Intervention in Bicycle Allocation - A Nested Group BH Approach to Online Hypothesis Testing

STAT 30850 Project Report

Bo Wei Ooi
ooibowei@uchicago.edu

Vu Phan
vuphan@uchicago.edu

Zhi Rong Tan
tanzhirong@uchicago.edu

April 10, 2019

1 Introduction

1.1 Background

In the field of Operations Research, successful resource allocation is a fundamental aspect of capability management for companies. Firms in the sharing economy face another challenge - demand and supply must be carefully handled to ensure that at all times, and at all locations, enough bicycles are provided to meet any high demand. Similarly, enough parking lots have to be available at stations with a high arrival of bicycles.

Hence, consider a scenario where the company wants to monitor the rate of bicycle arrival and departure at all stations, across all time. Fix a given time period, day, and weather condition e.g. 11am to 5pm, at station A, on a weekday with good weather. There are 3 possibilities:

1. The number of bicycles being returned exceeds the number of bicycles leaving the stations. If this trend is constant over time, this leads to the risk of future returning bicycles having no parking lots. In this case, cyclists have to search for a new station to return their bicycles. This is known as Oversupply.
2. Bicycles leaving the station exceeds the number of bicycles returning to the station. Over time, this leads to the risk of interested customers having no bicycles. This is known as Undersupply.
3. The number of bicycles leaving the station is equal to the number being returned. Here, the arms of Demand and Supply are balanced.

Possibilities 1 and 2 require intervention. In the former, bicycles have to be removed, and in the latter, new bicycles should be added to the station.

The decision for intervention is conservative. Consider the scenario - in the interval, 5 bicycles arrived and 1 left. However, if at the beginning of the time interval, there are 5 empty slots, this oversupply will not lead to concerns. Yet, we presume companies would want to prepare for the worst-case scenario, since if left unchecked, a persisting trend could lead to the problem described above.

Intervention requires labor costs, but the lack of intervention could generate customer frustration and lost revenue - an accurate decision rule for intervention is pertinent for maximizing profit.

There are 2 decisions a company has to make regarding intervention.

Firstly, there is a need to determine how long each time interval should be. The shorter the time interval, the more reactive the company will be to sudden oversupply or undersupply. However this will increase the manpower cost due to the frictional costs of deployment. Here, we will use 6 hours as our time interval.

Secondly, there is a need to determine how often the decision rule for intervention should be updated. As the company grows, it is likely that arrival and departure rates change. Hence, the company is required to update its decision rule periodically. 1 month is a pragmatic duration, since most manpower deployments and administrative planning are decided monthly. At the end of each month, the company updates its decision rule based on data from the previous month, and will determine the resources to be deployed for the following month based on the results of the tests, where rejection of the null hypothesis at a station under a given condition implies intervention is required. Subsequently, we will refer to each month as a period.

In our setting, we will use 6 hours as the length of each time interval and update the decision rule monthly. So we let each condition be a combination of 3 factors

1. Type of day: weekday, weekend
2. Weather: good, bad
3. Time: 0600-1159, 1200-1759, 1800-2359

We assume that the rate of bicycle arrival and the rate of bicycle departure are constant within each condition.

Therefore, in a given month, under a given condition and for a given station, the null hypothesis is that no intervention is required - the rates of incoming and outgoing bicycles are the same. This will form the company's decision rule for the next month.

Due to the large number of hypotheses, the company is also interested in controlling the false discovery rate within each period and across all the periods. Controlling the number of false positives helps minimise cost from intervention when it is not needed.

In the subsequent sections, we will proceed with the following situation:

1. Let us start from the end of the 1st month is 2012 - i.e. Jan 31 2012. At the end of the month, the company wants to make inference on the stability of stations in January, to inform operation decisions for February.
2. From the data and the test statistics, we will determine the p-value for each station, under (1) a given day type, (2) time interval, and (3) weather.
3. Apply a multiple testing correction to decide if we should reject or accept each null hypothesis. This gives us the intervention decisions for all of February.
4. At the end of February, we utilize the data from February to inform operation decisions in March.

Note that we are using the rates in January (for example) to make inference on the rates in the inference on the rates in February. This may not be completely accurate, but given the limited information, and desire to only update decision rules once every month, is the most optimal option we have.

1.2 Model

Let S be the set of all stations. Let C be the set defined as

$$\begin{aligned} C = \{&0600 - 1159, 1200 - 1759, 1800 - 2359\} \times \{\text{weekday}, \text{weekend}\} \\ &\times \{\text{good weather}, \text{bad weather}\} \end{aligned}$$

So each $c \in C$ corresponds to a time interval with specific characteristics, which accounts for the effect of weather and day of the week. We assume that weather remains constant during each time interval and across all stations. Subsequently, we will refer to $c \in C$ as a condition. This should be understood to mean a time interval under a specific weather and day combination.

Also, let $m_1, m_2, \ldots$ be a sequence of time periods of equal length. In our case, each m_k corresponds to a month and m_1 refers to the first month of 2012. This sequence is meant to be infinite, but with our limited data, we will only perform testing till the last month of 2012.

For each m_k, we are interested in testing the hypothesis that station $i \in S$ under $c \in C$ is stable. Specifically, we want to test if the expected number of arrivals into station i is equal to the expected number of departures from station i under condition c. We assume that this remains constant over m_k.

In the subsequent sections, we will omit the dependence on c and m_k in our descriptions except where necessary. We will also suppress the dependence of S on m_k (due to addition of new stations). Thus S should be taken to be the set of all stations in period m_k.

Let $D^{m_k}_{c,i,j}$ be the total number of departures from station i to station j under condition c and m_k. For example, if $c = (0600 - 1159, \text{weekday, good weather})$, then $D^{m_k}_{c,i,j}$ is the total number of departures from station i to j between 0600 to 1159 when it is a weekday with good weather, in m_k. Suppose that we can write

$$D^{m_k}_{c,i,j} = w_{c,i,j} N^{m_k}_{c,i}$$

where $N^{m_k}_{c,i} \sim \text{Pois}(\lambda^{m_k}_{c,i})$. In the above, $N^{m_k}_{c,i}$ is the number of departures from station i and $w_{c,i,j}$ represents the proportion of departures from i that arrive in j. Assume that the $N^{m_k}_{c,i}$'s are independent across stations i.

Note that $w_{c,i,j}$ does not depend on m_k. We assume that while the actual number of departures $N^{m_k}_{c,i}$ might change from one period to another, the proportion $w_{c,i,j}$ does not. This will factor into our estimation procedure in section 2.1.

2 Methodology

2.1 Estimation

We have data from Jan 2011 to Dec 2012. For a given m_k and c, let

1. $Q_c^{m_k}$ be the number of such intervals in m_k. For example, if we let

$$c = (0600 - 1159, \text{weekday, good weather})$$

   then $Q_c^{m_k}$ will be the total number of intervals in m_k that lie between $0600 - 1159$ on a weekday that has good weather.

2. $d_{c,i,j,q}^{m_k}$ be the observed number of departures from i to j in the interval q where $q = 1, \ldots, Q_c^{m_k}$

3. $d_{c,i,j}^{m_0}$ be the observed number of departures from i to j under condition c that occurred before period m_1.

Firstly, we estimate $w_{c,i,j}$. As described above, we assume that $w_{c,i,j}$ does not change over time. For each period m_k, we will compute $w_{c,i,j}$ using data up to m_k. So

$$\hat{w}_{c,i,j} = \frac{d_{c,i,j}^{m_0} + \sum_{l=1}^{k} \sum_{q=1}^{Q_c^{m_l}} d_{c,i,j,q}^{m_l}}{\sum_{j=1}^{|S|} (d_{c,i,j}^{m_0} + \sum_{l=1}^{k} \sum_{q=1}^{Q_c^{m_l}} d_{c,i,j,q}^{m_l})} \tag{2.1}$$

The numerator counts the total number of trips from i to j under condition c in all periods up to m_k, and the denominator counts the total number of departures from i under condition c in all periods up to m_k. Subsequently, we will drop the hat on $w_{c,i,j}$ and treat $w_{c,i,j}$ as a fixed quantity (whose estimate is updated with each period).

We believe that $N_{c,i}^{m_k}$ changes with period. So to estimate $\hat{N}_{c,i}^{m_k}$, we use

$$\hat{N}_{c,i}^{m_k} = \frac{\sum_{j=1}^{|S|} \sum_{q=1}^{Q_c^{m_k}} d_{c,i,j,q}^{m_k}}{Q_c^{m_k}} \tag{2.2}$$

The numerator counts the total number of departures from i over all the intervals satisfying condition c in m_k. The denominator counts the total number of such intervals in m_k.

2.2 Hypothesis Testing

We are interested in the null hypothesis that

$$H_{0,c,i}^{m_k} : \sum_{j \in S \setminus \{i\}} w_{c,i,j} \lambda_{c,i}^{m_k} = \sum_{j \in S \setminus \{i\}} w_{c,j,i} \lambda_{c,j}^{m_k}$$

The LHS represents the mean number of departures from station i and the RHS represents the mean number of arrivals at station i.

Define

$$T_{c,i}^{m_k} = \frac{\sum_{j \in S \setminus \{i\}} \left(w_{c,i,j} \hat{N}_{c,i}^{m_k} - w_{c,j,i} \hat{N}_{c,j}^{m_k} \right)}{\sqrt{\left(\sum_{j \in S \setminus \{i\}} w_{c,j,i} \lambda_{c,j}^{m_k} \right) \left(\sum_{j \in S \setminus \{i\}} w_{c,i,j} \right) + \sum_{j \in S \setminus \{i\}} w_{c,j,i}^2 \lambda_{c,j}^{m_k}} \Big/ \sqrt{Q_c^{m_k}}}$$

Proposition 2.1. *Suppose that $\lambda_{c,i}^{m_k}$ is large for all i and that the null hypothesis holds. Then $T_{c,i}^{m_k} \sim N(0,1)$ approximately.*

Proof. As $\lambda_{c,i}^{m_k}$ is large, $N_{c,i}^{m_k}$ is approximately $N(\lambda_{c,i}^{m_k}, \lambda_{c,i}^{m_k})$. So each $\sum_{j=1}^{|S|} d_{c,i,j,q}^{m_k}$ is drawn independently (across q's) from a $N(\lambda_{c,i}^{m_k}, \lambda_{c,i}^{m_k})$. Thus from our expression in (2.2), $\hat{N}_{c,i}^{m_k} \sim N(\lambda_{c,i}^{m_k}, \lambda_{c,i}^{m_k}/Q_c^{m_k})$. This implies that

$$\sum_{j\in S\setminus\{i\}} w_{c,i,j}\hat{N}_{c,i}^{m_k} \sim N\left(\sum_{j\in S\setminus\{i\}} w_{c,i,j}\lambda_{c,i}^{m_k}, \left(\sum_{j\in S\setminus\{i\}} w_{c,i,j}\right)^2 \frac{\lambda_{c,i}^{m_k}}{Q_c^{m_k}}\right)$$

and by independence,

$$\sum_{j\in S\setminus\{i\}} w_{c,j,i}\hat{N}_{c,j}^{m_k} \sim N\left(\sum_{j\in S\setminus\{i\}} w_{c,j,i}\lambda_{c,j}^{m_k}, \frac{1}{Q_c^{m_k}} \sum_{j\in S\setminus\{i\}} w_{c,j,i}^2\lambda_{c,j}^{m_k}\right)$$

Note that the null also implies that $\lambda_{c,i}^{m_k} = (\sum_{j\in S\setminus\{i\}} w_{c,j,i}\lambda_{c,j})/(\sum_{j\in S\setminus\{i\}} w_{c,i,j})$. By independence and imposing the null,

$$\sum_{j\in S\setminus\{i\}} \left(w_{c,i,j}\hat{N}_{c,i}^{m_k} - w_{c,j,i}\hat{N}_{c,j}^{m_k}\right) \sim N\left(0, V\right)$$

where

$$V = \frac{1}{Q_c^{m_k}}\left[\left(\sum_{j\in S\setminus\{i\}} w_{c,j,i}\lambda_{c,j}^{m_k}\right)\left(\sum_{j\in S\setminus\{i\}} w_{c,i,j}\right) + \sum_{j\in S\setminus\{i\}} w_{c,j,i}^2\lambda_{c,j}^{m_k}\right]$$

as required. □

To make $T_{c,i}^{m_k}$ feasible, we compute

$$\tilde{T}_{c,i}^{m_k} = \frac{\sum_{j\in S\setminus\{i\}}\left(w_{c,i,j}\hat{N}_{c,i}^{m_k} - w_{c,j,i}\hat{N}_{c,j}^{m_k}\right)}{\sqrt{\left(\sum_{j\in S\setminus\{i\}} w_{c,j,i}\hat{N}_{c,j}^{m_k}\right)\left(\sum_{j\in S\setminus\{i\}} w_{c,i,j}\right) + \sum_{j\in S\setminus\{i\}} w_{c,j,i}^2\hat{N}_{c,j}^{m_k}} \Big/ \sqrt{Q_c^{m_k}}} \tag{2.3}$$

which is approximately $N(0,1)$ under the null from that fact that $\hat{N}_{c,i}^{m_k}$ is a consistent estimator of $\lambda_{c,j}^{m_k}$ (as $Q_c^{m_k}$ increases).

2.3 Multiple Hypothesis Testing

In this section, we will address the multiple hypothesis problem. Within each period m_k, we are testing $|S|\cdot|C|$ different hypotheses. Furthermore, this is nested within an online structure as at each period m_k, we do not observe the p-values in periods $m_{k+1}, m_{k+2}, \ldots$.

We are interested in obtaining FDR control over all the periods m_k where we test a vector of null hypotheses in each period.

To do this, we will use the group Benjamini-Hochberg procedure to control the FDR within each period. Then we will adapt the LOND method found in Javanmard and Montanari (2015) to control the FDR across periods.

2.3.1 LOND - Online FDR Control

Javanmard and Montanari (2015) developed LOND for FDR control in an online setting where in each period, we test a single hypothesis without knowledge of the future p-values. We will extend the method to allow for a vector of hypotheses in each period.

Suppose we have a sequence of periods $1, 2, \ldots$. In each period k, we test n_k different hypotheses. We start by choosing a sequence β_k such that $\sum_{k=1}^{\infty} \beta_k = \alpha$. We will then define a sequence α_k where α_k is the target FDR control level within period k.

Let f_k be the number of false discoveries in period k and r_k be the number of discoveries in period k. Also, let $R_k = \sum_{l=1}^{k} r_l$ be the total number of discoveries up to and including period k. In the following, we will use $x \vee y$ to denote $\max(x, y)$ and $\mathcal{F}_k = \sigma(r_1, \ldots, r_k)$.

Define

$$\alpha_k = \frac{\beta_k (R_{k-1} \vee 1)}{n_k} \tag{2.4}$$

Our following result adapts the proof found in Javanmard and Montanari (2015). It shows that if we have FDR control within each period at level α_k defined in (2.4), then $\mathrm{FDR}(K)$, the total FDR by period K, is bounded by α for any K.

Theorem 2.1. *Suppose that* $\mathbb{E}[f_k/(r_k \vee 1) \mid \mathcal{F}_{k-1}] \leq \alpha_k$ *where* α_k *is given in* (2.4)*. Then for any* K*,* $\mathrm{FDR}(K) \leq \alpha$*.*

Proof. Fix $k \leq K$. By the assumption of FDR control and that α_k is $\mathcal{F}_{k-1}$-measurable,

$$\mathbb{E}\left(\frac{f_k}{(r_k \vee 1)\alpha_k}\right) = \mathbb{E}\left[\frac{1}{\alpha_k}\mathbb{E}\left(\frac{f_k}{r_k \vee 1}\,\middle|\, \mathcal{F}_{k-1}\right)\right] \leq 1$$

Also,

$$\frac{(r_k \vee 1)\alpha_k}{R_K \vee 1} \leq \frac{(r_k \vee 1)\alpha_k}{R_k \vee 1} \leq \frac{n_k \alpha_k}{R_{k-1} \vee 1} = \beta_k$$

where the last inequality is tight for $n_k = 1$ and $r_k = 0$. Thus,

$$\mathrm{FDR}(K) = \mathbb{E}\left(\frac{\sum_{k=1}^{K} f_k}{R_K \vee 1}\right) = \sum_{k=1}^{K} \mathbb{E}\left[\frac{f_k}{(r_k \vee 1)\alpha_k} \cdot \frac{(r_k \vee 1)\alpha_k}{R_K \vee 1}\right] \leq \sum_{k=1}^{K} \beta_k \leq \alpha$$

□

2.3.2 Group BH - Within Period FDR Control

The adaptive group BH procedure was first introduced in Hu et al. (2010). Subsequently, Li and Barber (2018) introduced a general class of adaptive BH methods named SABHA that encompasses the adaptive group BH.

We will start by describing the procedure. Suppose that we have N different hypotheses and we can partition them into G different groups where the size of group g is n_g and the proportion of nulls is π_g. Then we compute $\hat{\pi}_g$ for each g. For a given α and τ in $[0, 1]$, we reject the hypotheses corresponding for $p_{(1)}, \ldots, p_{(k)}$ where k is given by

$$k = \max\left\{ i : p_{(i)} \leq \left(\frac{\alpha k}{N \hat{\pi}_{(i),g}} \wedge \tau\right)\right\}$$

and $\hat{\pi}_{(i),g}$ refers to the estimated proportion of nulls in the group belonging to hypothesis (i). If no such k exists, then we do not reject any hypotheses.

Li and Barber (2018) derived FDR bounds under different dependence assumptions. In particular, they showed that under sufficiently weak dependence, we can bound FDR by an expression that is close to α when $\epsilon \gg \sqrt{(G\log(N))/N}$.

We now turn to the problem of estimating $\hat{\pi}_g$. Fix ϵ and τ in $[0, 1]$. Following Li and Barber (2018), let $\pi = (\pi_1, \ldots, \pi_G)$ and define

$$\hat{\pi} = \text{argmax}\left[\sum_g \left(\sum_{i\,\in\,\text{group } g} \mathbb{1}_{p_i>\tau}\right)\log(\pi_g(1-\tau)) + \left(\sum_{i\,\in\,\text{group } g} \mathbb{1}_{p_i\leq\tau}\right)\log(1-\pi_g(1-\tau))\right]$$

subject to $\epsilon \leq \pi_g \leq 1$ for all g and $\sum_g \frac{\sum_{i\,\in\,\text{group } g} \mathbb{1}_{p_i>\tau}}{\pi_g(1-\tau)} \leq N$

2.3.3 Implementation

We will now describe our implementation of the two multiple testing procedures. We will set $\alpha = 0.01$.

For LOND, we specify our β sequence to be

$$\sum_{k=1}^{\infty}\beta_k = \sum_{k=1}^{\infty}\frac{C}{k\log^2(k \vee 2)} = \alpha \tag{2.5}$$

where C is chosen such that the sequence converges to α. Using numerical methods, we approximate C to be $\alpha/4$. Then we apply the group BH procedure to the hypotheses in each period, where the BH procedure for period m_k is applied at level α_k as defined in (2.4).

For the group BH procedure, consider period m_k and level α_k given by LOND. We will split the hypotheses into 24 different groups: we group the hypotheses under a specific condition together and within each condition, we further split the hypotheses into stations are in downtown and stations that are not. In our setting, it is natural to group the hypotheses by their condition. Also, we expect that the group of stations in downtown will have a different proportion of nulls compared to the group of stations not in downtown, for any given condition.

Specifically, we will declare any station that has a ZIP code in the set

$$\{20001, 20005, 20006, 20036\}$$

to be in downtown. All other stations will be labelled as not in downtown. Then we pick $\epsilon = 0.1$ and $\tau = 0.5$, and use the code provided by Li and Barber (2018) to estimate the proportion of nulls in each group.

3 Results

We compute the p-values and correct for multiple hypothesis testing, as described in the previous section, to determine which hypotheses should be rejected. A summary of the results for each condition in each period is presented in appendix A.

The following table shows the number of hypotheses and rejections in each period, as well as the values of α_k and β_k.

Month	α_k	β_k	# hypotheses	# rejections
1	3.136×10^{-6}	5.203×10^{-3}	1659	553
2	8.672×10^{-4}	2.602×10^{-3}	1659	709
3	5.252×10^{-4}	6.904×10^{-4}	1659	693
4	3.832×10^{-4}	3.252×10^{-4}	1659	675
5	3.060×10^{-4}	1.930×10^{-4}	1659	665
6	2.578×10^{-4}	1.298×10^{-4}	1659	660
7	2.249×10^{-4}	9.432×10^{-5}	1659	652
8	2.007×10^{-4}	7.227×10^{-5}	1659	647
9	1.822×10^{-4}	5.754×10^{-5}	1659	643
10	1.676×10^{-4}	4.715×10^{-5}	1659	642
11	1.558×10^{-4}	3.953×10^{-5}	1659	639
12	1.460×10^{-4}	3.374×10^{-5}	1659	638

Table 1: Number of hypotheses, rejections, α_k, and β_k for each period

In section 3.1, we will provide an example of how a company can analyse the results of the tests from the month of March and use them to inform their operations in April. Subsequently, we will compare this to the actual data from April. Nonetheless, our focus is on inference about the underlying parameters of the random variables and not on prediction.

In section 3.2, we will analyse our results over the entire year with the aim of identifying persistent trends under different conditions. For example, we are interested in which stations are consistently not stable (null hypothesis rejected) under different conditions. This might suggest that a structural change, such as installing new stations, is necessary.

3.1 Short Run Analysis

3.1.1 Analysis

Suppose the company has reached the end of March, and possesses data from start of 2011 till March 2012. With this information, we want to decide our manpower and resource allocation for the entire month of April 2012.

From the analysis, we determine:

1. The number of times a condition is predicted to appear in April, and how they are spaced throughout the month. For example, if there are bad weather continuously for a few days and intervention drops, human resources may be reallocated to other areas.
2. The number of stations requiring intervention for each condition, and subsequently the minimum manpower required for the condition. This may also enable the company to determine how much manpower is required for overtime ($1800 - 2359$), and adjust the salary accordingly.

3. From (1) and (2), the company may determine how the intervention number and pattern change throughout the hours, days, and in different predicted weather. This allows them to better allocate manpower to different locations. For example, suppose a worker is at station A in the time interval $0600 - 1159$. If station A ceases to require intervention in the next time interval, the worker can be reallocated to another nearby station requiring intervention.

Recall that we have 12 possible conditions with different kinds of weather, time interval, and day type. Assuming (perfect) weather predictions from AccuWeather, the company has the following scenarios for different days of April, listed in the first row of table 2. Weekends are labelled with (*).

Interval	01*	02	03	04	05	06	07*	08*	09	10	11	12	13	14*	15*
0600-1159	S	-	-	-	X	-	=	=	-	-	X	-	-	=	=
1200-1759	S	-	-	-	X	-	=	=	-	-	X	-	-	S	=
1800-2359	S	-	-	-	-	-	=	=	X	-	-	-	-	S	=
Interval	16	17	18	19	20	21*	22*	23	24	25	26	27	28*	29*	30
0600-1159	-	-	X	X	-	=	S	X	-	-	X	-	S	=	X
1200-1759	-	-	X	-	-	S	S	X	-	-	X	-	S	=	X
1800-2359	-	-	X	-	-	S	S	X	-	-	X	-	S	=	X

Table 2: Conditions in April

- - : Good Weather on Weekday
- X : Bad Weather on Weekday
- = : Good Weather on Weekend
- S : Bad Weather on Weekend

The count for each kind of condition in April is listed in table 3.

Interval	0600-1159	1200-1759	1800-2359
Good Weather, Weekday	14	15	16
Good Weather, Weekend	6	4	4
Bad Weather, Weekday	7	6	5
Bad Weather, Weekend	3	5	5

Table 3: Condition Count

Based on our multiple testing results for each condition, we will either reject the null hypothesis (indicating intervention is required to balance the demand/supply of bicycles), or accept the null hypothesis, where no intervention is required. The results are presented in figures 1 to 7.

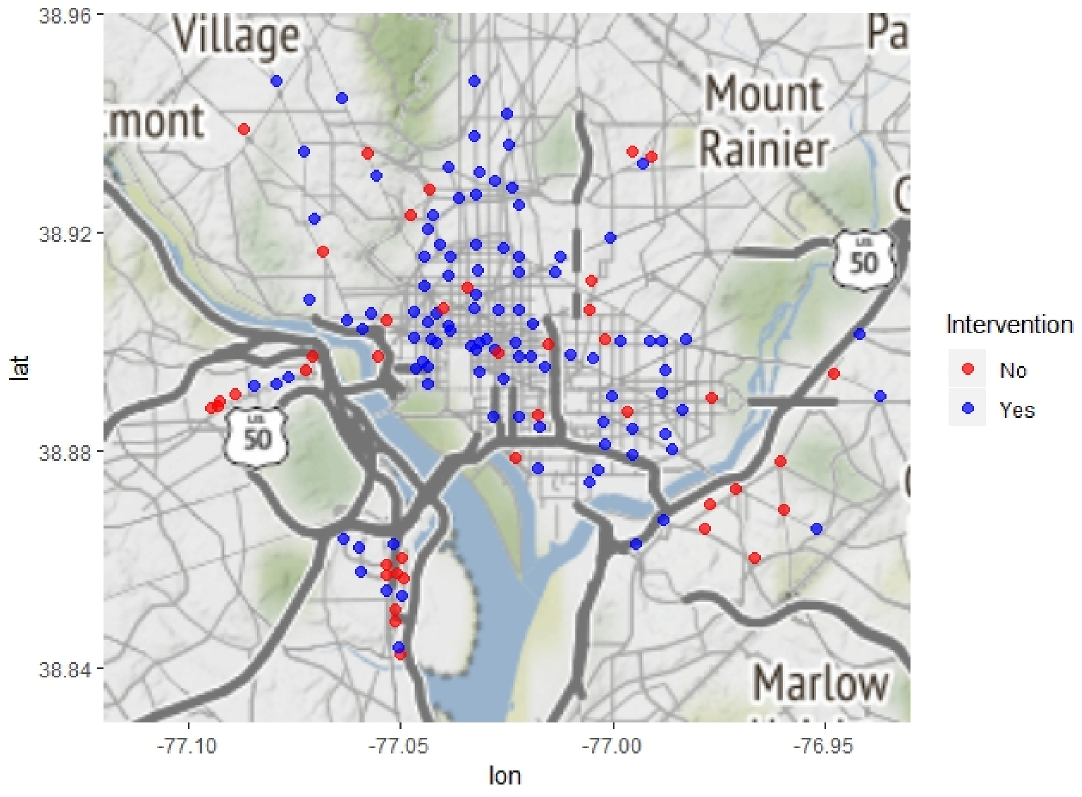


Figure 1: Good Weather, 0600 – 1159, Weekday

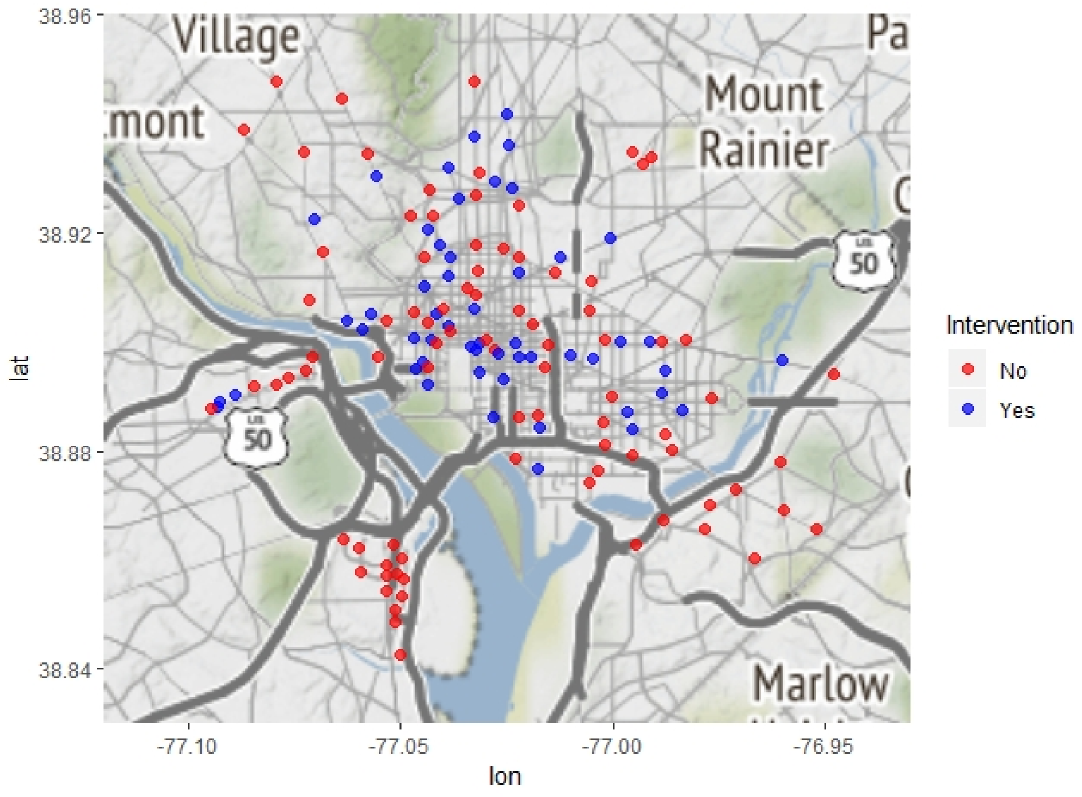
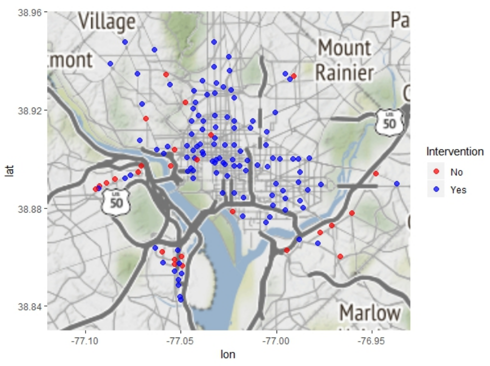
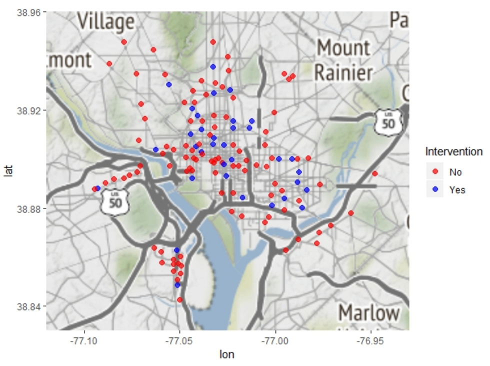


Figure 2: Good Weather, 0600 – 1159, Weekend

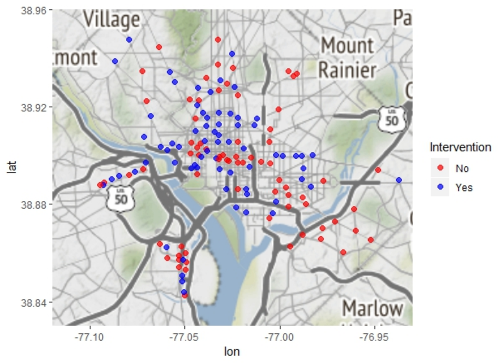


(a) Weekday

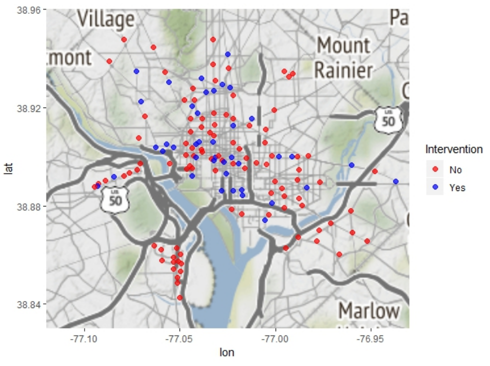


(b) Weekend

Figure 3: Bad Weather, 0600 – 1159

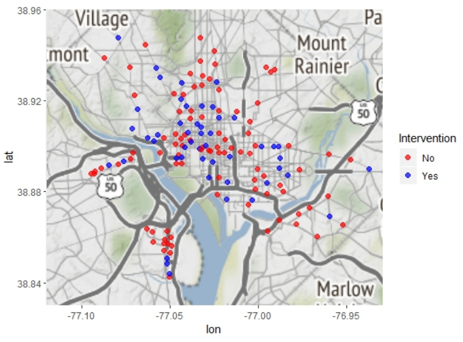


(a) Weekday

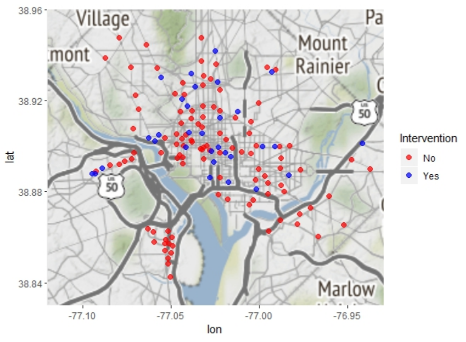


(b) Weekend

Figure 4: Good Weather, 1200 – 1759

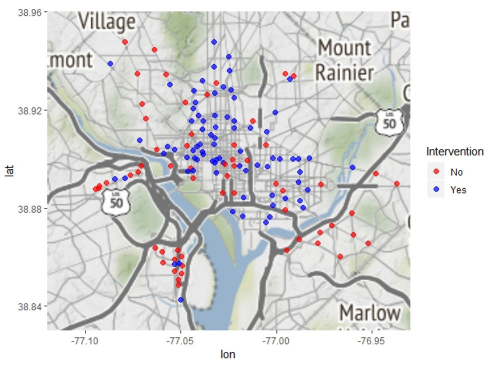


(a) Weekday

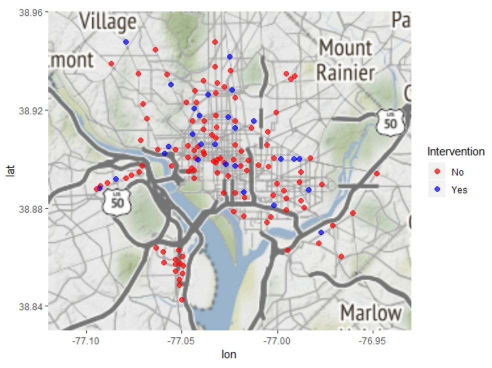


(b) Weekend

Figure 5: Bad Weather, 1200 – 1759

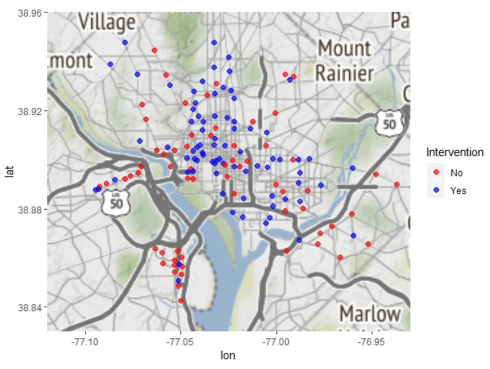


(a) Weekday

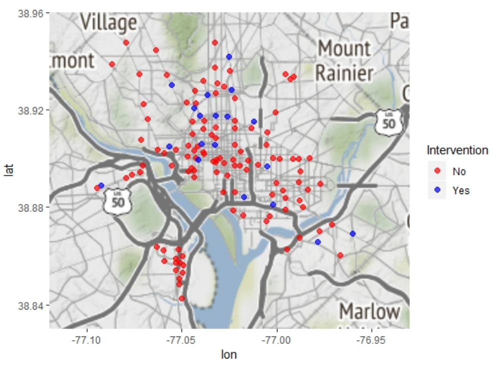


(b) Weekend

Figure 6: Good Weather, 1800 – 2359



(a) Weekday



(b) Weekend

Figure 7: Bad Weather, 1800 – 2359

From the plots, we perform a visual and qualitative comparison of results across the conditions.

First, let us compare between weekdays and weekends. In general, on weekdays, there are more stations in downtown and the outlying districts which require intervention than on weekends. This is understandable as most people commute into downtown from their suburban homes during the working days. Hence, in the morning, commuters will leave their homes to reach their workplaces, causing stations near residential areas to have a lack of supply, and the downtown stations to require additional parking lots.

Nonetheless, notice that the difference between weekdays and weekends is small in the interval 1200 – 1759 when there is bad weather. This is understandable since during this time period, commuters may be reluctant to travel, and have an option not to travel during both weekdays and weekends, the former by staying in their workplace, and the latter by staying at home.

On average, the time interval 0600 – 1159 has the most stations requiring intervention, followed by 1800 – 2359, then 1200 – 1759. This can be explained by how the morning commute is entirely captured within 0600 – 1159. The evening commute tends to range from 1700 to 1900, which is evenly split between the other 2 time intervals.

Finally, observe that we see a significant impact of bad weather during Weekday, 0600 – 1159 - if the weather is bad, more stations require intervention. This trend reverses in the interval 1200 – 1759: if the weather is bad, less stations require intervention. We do not observe any effect of bad weather for other time period and day type.

3.1.2 Comparison with Observed Data

Despite our focus on inference about the parameters instead of prediction, comparing the results of our tests based on the March data, to the actual April statistics, is a useful indicator of performance. Specifically, we will focus on the data from 4 randomly selected stations in the month of April under the condition of weekdays, $0600 - 1159$, with good weather. These stations are

1. ID 31100: Station is located in downtown. Null hypothesis rejected
2. ID 31619: Station is located outside of downtown. Null hypothesis rejected
3. ID 31201: Station is located in downtown, Null hypothesis not rejected
4. ID 31106: Station is located outside of downtown. Null hypothesis not rejected

The null hypotheses above refer to the hypotheses made at the end of March. In the following tables, we will present the total number of departures and arrivals at these 4 stations under the specified condition. Table 4 will list the arrival and departure counts for the two stations where intervention is required (null hypothesis rejected). Table 5 will list the arrival and departure counts for the two stations where intervention is not required (null hypothesis not rejected). For instance, the second column of table 4 refers to the departures from station ID 31100 at each date, under the condition of weekdays, $0600 - 1159$, with good weather.

	Downtown		Not downtown	
Date	Departures	Arrivals	Departures	Arrivals
2	7	18	36	2
3	5	25	41	8
5	5	27	41	6
6	3	19	48	11
9	6	21	28	5
10	3	24	37	5
12	5	26	39	3
13	7	16	36	4
16	6	20	37	6
17	16	32	40	11
20	10	32	43	7
24	9	24	37	3
25	3	26	56	6
27	9	25	29	4

Table 4: Departure and arrival counts for the two stations with intervention

	Downtown		Not downtown	
Date	Departures	Arrivals	Departures	Arrivals
2	30	15	31	21
3	32	19	31	22
5	35	19	27	22
6	33	20	20	21
9	31	25	30	15
10	28	20	22	20
12	32	17	15	11
13	28	24	21	20
16	34	21	27	15
17	30	27	29	15
20	22	18	28	16
24	27	24	17	16
25	30	25	30	21
27	30	27	29	20

Table 5: Departure and arrival counts for the two stations without intervention

We observe that both stations in table 4 require intervention as the number of arrivals is vastly different from the number of departures. Our test statistic is also able to correctly identify the type of intervention required. Station 31100 has a test statistic value of -10.52 which indicates that there are more arrivals than departures, while station 31619 has a test statistic value of 58.66 which indicates that there are more departures than arrivals.

On the other hand, the stations in table 5 appear to be stable as the number of arriving bikes is relatively similar to the number of departing bikes.

From table 4, we can also see the impact of a station being in downtown. As the condition is $0600 - 1159$ on a weekday, it is expected that there will be a large number of trips being made from stations outside of downtown to downtown. This is reflected in the data as station 31100 had significantly more arrivals than departures, and vice versa for station 31619.

To obtain an overview of all the stations under this particular condition, we will proceed as follows:

1. Declare a station to have required intervention on a particular date if

$$2\min(\#\text{ arrivals}, \#\text{departures}) < \max(\#\text{ arrivals}, \#\text{ departures})$$

   where the number of arrivals and departures are the counts for that date. Essentially, a station with fewer arrivals than departures required intervention on that date if the ratio of arrivals to departures is less than half, and vice versa for stations with more arrivals than departures. We will not distinguish between the type of intervention required and

2. For each station, we will compute the number of dates where intervention was required. If the station required intervention on more than half of the dates, we will say that the station required intervention in the month of April.

3. For each station, we will compare the decision to intervene (this is from the hypothesis test performed at the end of March) to whether the station required intervention as defined in the previous step. In figure 8 below, we will mark the stations as follows

   - Purple: decision to reject the hypothesis matches the requirement for intervention

- Red: Hypothesis not rejected but intervention required (labelled "FALSE 1")
- Blue: Hypothesis rejected but intervention not required (labelled "FALSE 2")

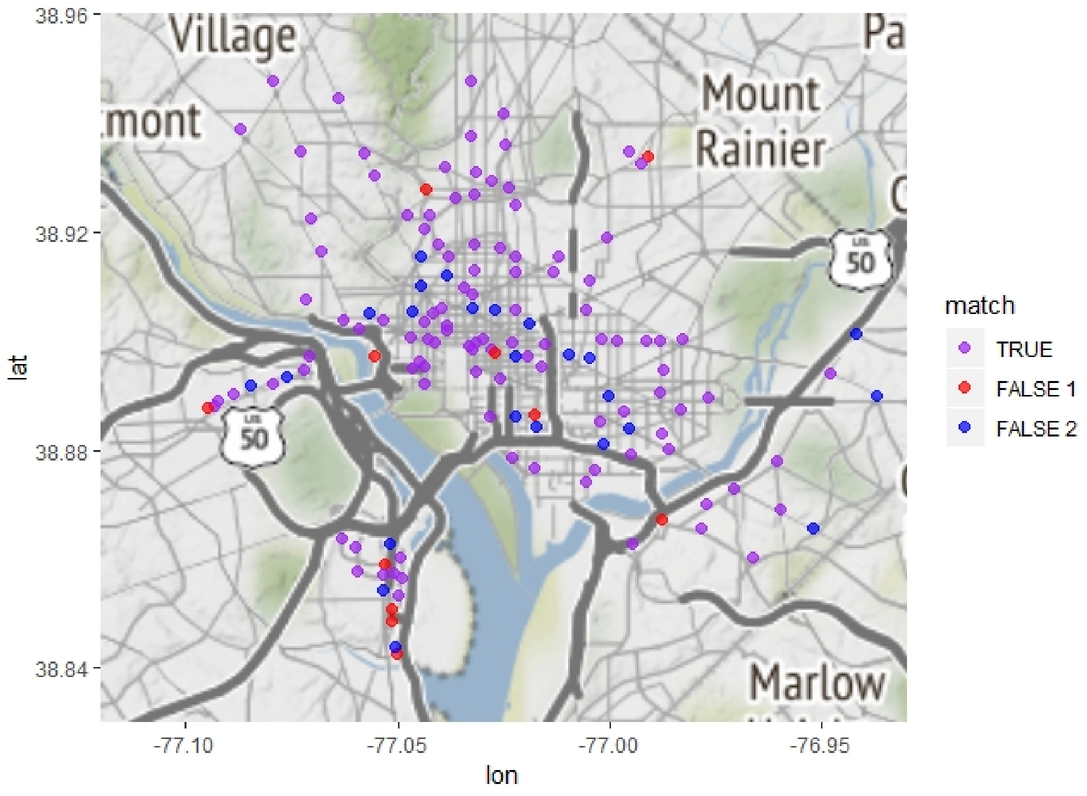


Figure 8: Hypothesis rejected vs intervention required

We observe that 107 out of 142 stations (approximately 75%) registered a match between the rejection of the hypothesis and whether an intervention was required. 11 stations were classified as "FALSE 1" and 24 were classified as "FALSE 2". We consider "FALSE 1" to be the costlier error as this represents a station where intervention was required but not deployed.

Our results in this subsection suggests that our method performs well and can be used to make intervention decisions.

3.2 Long Run Analysis

Suppose the company has reached the end of 2012 and would like to identify any long run trends in station stability (more rejections corresponds to lower stability) under different conditions throughout the year. This could be useful in making more structural adjustments as opposed to merely allocating resources and personnel to fixing oversupply and undersupply at stations.

We will start by comparing rejections aggregated over all periods across different conditions in sections 3.2.1—3.2.3. We are interested in any patterns that emerge as well as identifying which stations are consistently rejected. In the next three sections, we will classify the stability of stations into 3 categories:

1. Stable (blue): stations that do not have any rejections after aggregating over all periods
2. Unstable (purple): stations that have at least one rejection

3. Highly unstable (red): stations that are rejected more than 75% of the time. For example, a station is highly unstable during weekdays if, after aggregating over all time intervals, weather conditions and periods, more than 75% of the hypotheses (there are 72 in this case) are rejected.

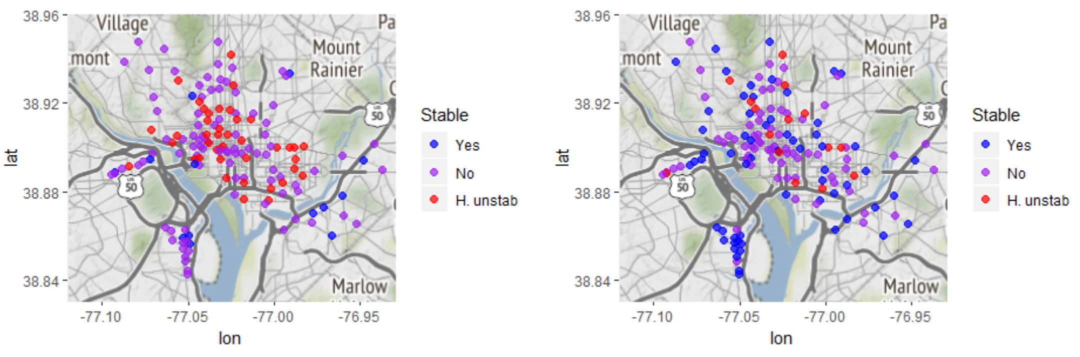
In section 3.2.4, we will examine the stations that are revealed to be consistently highly unstable, regardless of condition.

Lastly, in section 3.2.5, we will pick a specific condition and track the rejections over different periods, with the aim of identifying time trends in the stations being rejected.

3.2.1 Weekday vs Weekend

	Weekday	Weekend
Stable (%)	9.028	41.667
Unstable (%)	63.194	47.917
Highly unstable (%)	27.778	10.417

Table 6: Percentage of stations in each stability category – Weekday vs Weekend



(a) Weekday (b) Weekend

Figure 9: Aggregated over all intervals, weather conditions and months

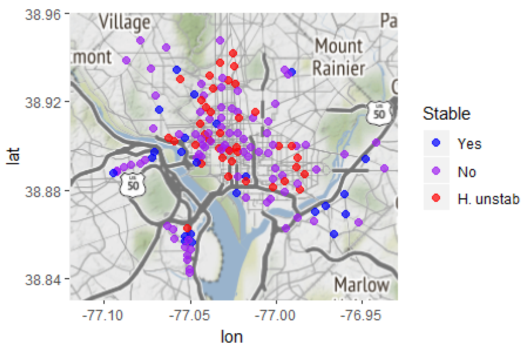
On weekends, 41.67% of all stations are stable whereas on weekdays, only 9.03% of all stations are stable. This supports our intuition that people's commuting activity and consequently demand for bike sharing service is higher on weekdays compared to weekends. From the map, we can see that it is largely the stations that are outside of downtown that change from stable to unstable when going from weekdays to weekends. The highly unstable stations during weekdays tend to be clustered at or near downtown. There are also fewer highly unstable station on weekends compared to weekdays.

3.2.2 Interval 1 vs Interval 2 vs Interval 3

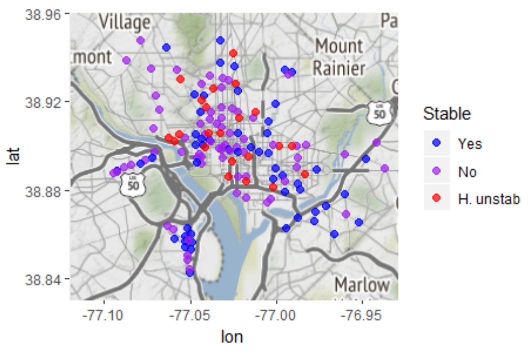
We will use time interval 1 to refer to 0600 – 1159, time interval 2 to refer to 1200 – 1759, and time interval 3 to refer to 1800 – 2359. We proceed as before.

	Interval 1	Interval 2	Interval 3
Stable (%)	15.972	36.806	30.556
Unstable (%)	57.639	47.917	58.333
Highly unstable (%)	26.389	15.278	11.111

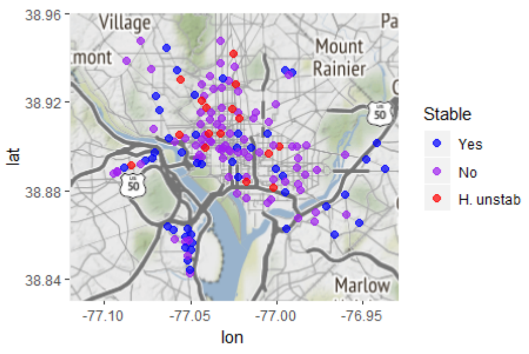
Table 7: Percentage of stations in each stability category – Intervals 1, 2, and 3



(a) Interval 1



(b) Interval 2



(c) Interval 3

Figure 10: Aggregated over all days of week, weather conditions and months

The trend is interesting. Interval 1 seems to get the most activity (lowest stable percentage), interval 3 seems to get the second most activity while interval 2 has the least activity (most % stable). This can potentially be explained by the work day pattern: interval 1 corresponds to people getting to work, interval 2 middle of the workday (thus lowest activity) and interval 3 people leaving work to get home. The difference between interval 1 and 3 can be explained by the fact that there is generally more variance to when work starts compared to when work ends. For example, most work starts at 8 or 9 but can end at 5, 6, 7 or even later depending on the workload of that day.

The highly unstable stations that occur in interval 2 and 3 tend to be similar. These stations also tend to be highly unstable in interval 1. This suggests that the high instability of these 10% or so cannot be explained by the different time intervals.

At this point, we need to recall that the percentages calculated above aggregate over both weekdays and weekends but our work day hypothesis only applies to weekdays. We can refine our analysis by computing the percentages for weekends and weekdays separately.

	Interval 1	Interval 2	Interval 3
Stable (%)	18.056	49.306	36.111
Unstable (%)	17.361	22.917	18.750
Highly unstable (%)	64.583	27.778	45.139

Table 8: Stability percentage – Intervals 1, 2, 3 – Weekday only

	Interval 1	Interval 2	Interval 3
Stable (%)	56.338	66.197	74.648
Unstable (%)	29.577	15.493	17.606
Highly unstable (%)	14.0845	18.3099	7.7465

Table 9: Stability percentage – Intervals 1, 2, 3 – Weekend only

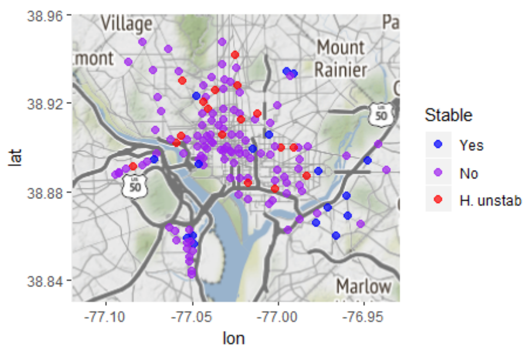
Indeed, when limiting the analysis to weekdays, the observed work day pattern is still there: interval 1 has the lowest percentage of stable stations and interval 2 has the highest. Also, we can see the effect of the work commute from the large proportion of highly unstable stations in interval 1 and 3, and the relatively smaller number in interval 2.

On the other hand, when limiting the analysis to weekends, the work day pattern disappears: the percentage of stable stations does not change significantly across intervals. This seems to indicate that the work day explanation is sensible.

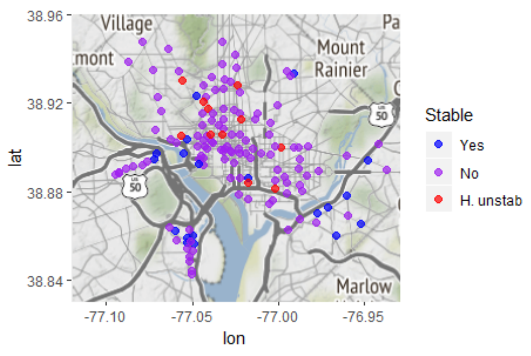
3.2.3 Good weather vs bad weather

	Good weather 1	Bad weather
Stable (%)	11.806	13.194
Unstable (%)	76.389	79.167
Highly unstable (%)	11.8056	7.6389

Table 10: Stability percentage – Good Weather vs Bad Weather



(a) Good weather



(b) Bad weather

Figure 11: Aggregated over all intervals, days of week, and months

We can see that demand is slightly higher (stability is lower) when weather is good, which is intuitive as under bad weather, people may be more inclined to stay inside or take other modes of transportation instead of cycling. However, the percentages are quite similar, which seems to indicate that unlike day of week and interval of day, weather is not a significant determining factor in the stability of stations.

3.2.4 Consistently Highly Unstable Stations

From our earlier analysis, we observe that some stations appear to be consistently classified as highly unstable, across all conditions and periods. With this knowledge, the company can explore other methods to reducing the amount of intervention needed at these stations. We will plot these stations.

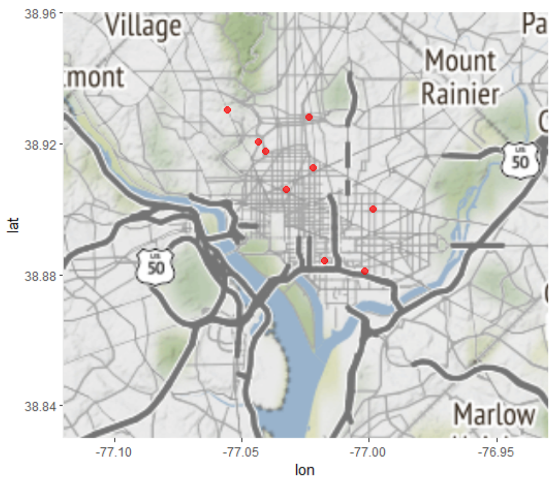


Figure 12: Consistently Highly Unstable Stations

3.2.5 Stability over Time

We will analyse the stability over time for a specific condition: weekdays, $0600 - 1159$, good weather. In particular, we are interested in examining if there are any trends in the stations that are rejected as we move from one period to another. The total number of rejections in each period is presented in the table below. We observe that the number of rejections does not

Period	1	2	3	4	5	6	7	8	9	10	11	12
# rejections	97	101	101	99	98	98	98	98	98	98	98	98

Table 11: # rejections by period

change significantly from period to period. In fact, the 97 stations rejected in period 1 are also rejected in each of the other 11 periods. Thus we conclude that the stability of stations, under this condition, did not change over time.

4 Extensions

In this section, we will discuss several improvements and extensions to our method.

4.1 Prediction

An alternative approach is to frame this problem as one of prediction instead of inference. At each point in time and for each station, we can use past data with similar characteristics to predict the number of incoming and outgoing bikes at the next time point. For example, on the 1st of March 2012 5pm, we want to predict the number of incoming and outgoing bikes by the 1st of March 2012 6pm.

To do this, we can implement a similar estimation procedure. Based on the current conditions (for example: evening, weekday, bad weather), we can estimate the expected number of bikes leaving each station and the proportion of them heading to each of the other stations. This prediction can be further refined by taking into consideration the number of bikes that are currently active at 5pm but have not arrived at a destination.

At the next time point 6pm, we can then check the actual number of arrivals and departures from each station and use this to compute a prediction error for our method. This gives us an empirical distribution of the residuals under different conditions which can be used to compute a prediction interval.

4.2 Other Parameters

In our analysis, we fixed certain parameters such as the length of each period (1 month) and the types of conditions. With more data, this can be tuned for better inference.

For each condition, we use one rate parameter to capture the number of departures from each station. Thus by considering conditions at a more granular level, with sufficient data, we can capture the local trend more accurately. For example, we can use shorter time intervals to account for the fact that there are likely more departures at 7am compared to 11am on weekdays. Also, we can partition the type of weather into finer groups such as thunderstorm, drizzle and no rain.

Regarding the length of each period: we are, essentially, using inference on the parameters of the previous period to guide decisions for the upcoming period. Thus it is desirable for the length of each period to be short to minimise the effect of trends that vary over time.

4.3 Tracking Demand and Supply

Recall that as part of our method to estimate the demand and available supply of bicycles at each station, we relate demand with the number of bicycles leaving the station, and supply as the number of bicycles arriving at the station.

However, this is based on the assumption that the company has been constantly intervening. Consider the possibility that returning cyclists are forced to look for a new station because their target station has no parking lots, or that interested passengers cannot start a new trip since there is no bikes left at the station. Since our data only captures trips, we are unable to acknowledge the extent of this missing data. Hence, we can resolve this through a few methods:

- Capture more data that highlight this information. For example, the company may request riders to indicate on their app if they fail to retrieve any bicycles or find an empty slot at their intended stations. For non-members, the company may also analyze the trip path to determine any anomaly which indicate the same issue (e.g. a rider visiting 1 or more stations before returning the bicycle).
- Introduce 2 hyperparameters, γ_D and γ_S, which scale the trip-reliant estimate by a proportion greater than 1 to reflect a more realistic demand and supply. These hyperparameters may be fixed, depending on the company's priorities (larger γ is, the more concerned they are about lost revenue), or to be tuned based on another downstream task.

5 Conclusion

In this project, we introduced a parametric model for the number departures and arrivals at each station. Under this parametric model, we estimated the parameters and conducted inference on the stability of stations in different periods and under different conditions. In order to correct for multiple hypothesis testing, we implemented a group BH procedure for each period as well as a LOND-type procedure to account for the online testing structure.

In the results section, we analysed the results of these hypotheses tests. Firstly, we demonstrated how a company can use the results of these tests to inform its operations for the coming month, by determining which stations require intervention and under which conditions. Next, to gauge the accuracy of our method, we compared the results of our tests to observed data to check if intervention was required. Lastly, we used the information from the tests over the entire year of 2012 to perform a long run analysis, identifying trends and patterns over periods and conditions.

References

Hu, James X., et al. 2010. "False Discovery Rate Control With Groups." *Journal of the American Statistical Association* 105 (491): 1215–1227. doi:10.1198/jasa.2010.tm09329.

Javanmard, Adel, and Andrea Montanari. 2015. "On Online Control of False Discovery Rate." *arXiv e-prints.*

Li, Ang, and Rina Foygel Barber. 2018. "Multiple testing with the structure-adaptive Benjamini-Hochberg algorithm." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 81 (1): 45–74. doi:10.1111/rssb.12298.

Appendix A Results

Month	Weather	Time	Weekday	# stations	# rejections
1	Good	0600 - 1159	TRUE	142	97
1	Good	0600 - 1159	FALSE	139	38
1	Good	1200 - 1759	TRUE	141	52
1	Good	1200 - 1759	FALSE	140	31
1	Good	1800 - 2359	TRUE	139	63
1	Good	1800 - 2359	FALSE	135	22
1	Bad	0600 - 1159	TRUE	138	101
1	Bad	0600 - 1159	FALSE	134	19
1	Bad	1200 - 1759	TRUE	142	32
1	Bad	1200 - 1759	FALSE	138	26
1	Bad	1800 - 2359	TRUE	141	58
1	Bad	1800 - 2359	FALSE	130	14
2	Good	0600 - 1159	TRUE	142	101
2	Good	0600 - 1159	FALSE	139	53
2	Good	1200 - 1759	TRUE	141	70
2	Good	1200 - 1759	FALSE	140	43
2	Good	1800 - 2359	TRUE	139	82
2	Good	1800 - 2359	FALSE	135	29
2	Bad	0600 - 1159	TRUE	138	113
2	Bad	0600 - 1159	FALSE	134	36
2	Bad	1200 - 1759	TRUE	142	50
2	Bad	1200 - 1759	FALSE	138	32
2	Bad	1800 - 2359	TRUE	141	79
2	Bad	1800 - 2359	FALSE	130	21
3	Good	0600 - 1159	TRUE	142	101
3	Good	0600 - 1159	FALSE	139	53
3	Good	1200 - 1759	TRUE	141	69
3	Good	1200 - 1759	FALSE	140	41
3	Good	1800 - 2359	TRUE	139	79
3	Good	1800 - 2359	FALSE	135	27
3	Bad	0600 - 1159	TRUE	138	112
3	Bad	0600 - 1159	FALSE	134	34
3	Bad	1200 - 1759	TRUE	142	49
3	Bad	1200 - 1759	FALSE	138	31
3	Bad	1800 - 2359	TRUE	141	78
3	Bad	1800 - 2359	FALSE	130	19
4	Good	0600 - 1159	TRUE	142	99
4	Good	0600 - 1159	FALSE	139	51
4	Good	1200 - 1759	TRUE	141	66
4	Good	1200 - 1759	FALSE	140	41
4	Good	1800 - 2359	TRUE	139	79
4	Good	1800 - 2359	FALSE	135	25
4	Bad	0600 - 1159	TRUE	138	112
4	Bad	0600 - 1159	FALSE	134	29
4	Bad	1200 - 1759	TRUE	142	47
4	Bad	1200 - 1759	FALSE	138	31

4	Bad	1800 - 2359	TRUE	141	77
4	Bad	1800 - 2359	FALSE	130	18
5	Good	0600 - 1159	TRUE	142	98
5	Good	0600 - 1159	FALSE	139	51
5	Good	1200 - 1759	TRUE	141	64
5	Good	1200 - 1759	FALSE	140	40
5	Good	1800 - 2359	TRUE	139	77
5	Good	1800 - 2359	FALSE	135	25
5	Bad	0600 - 1159	TRUE	138	112
5	Bad	0600 - 1159	FALSE	134	28
5	Bad	1200 - 1759	TRUE	142	45
5	Bad	1200 - 1759	FALSE	138	31
5	Bad	1800 - 2359	TRUE	141	77
5	Bad	1800 - 2359	FALSE	130	17
6	Good	0600 - 1159	TRUE	142	98
6	Good	0600 - 1159	FALSE	139	50
6	Good	1200 - 1759	TRUE	141	64
6	Good	1200 - 1759	FALSE	140	40
6	Good	1800 - 2359	TRUE	139	76
6	Good	1800 - 2359	FALSE	135	24
6	Bad	0600 - 1159	TRUE	138	112
6	Bad	0600 - 1159	FALSE	134	28
6	Bad	1200 - 1759	TRUE	142	44
6	Bad	1200 - 1759	FALSE	138	31
6	Bad	1800 - 2359	TRUE	141	76
6	Bad	1800 - 2359	FALSE	130	17
7	Good	0600 - 1159	TRUE	142	98
7	Good	0600 - 1159	FALSE	139	49
7	Good	1200 - 1759	TRUE	141	61
7	Good	1200 - 1759	FALSE	140	40
7	Good	1800 - 2359	TRUE	139	75
7	Good	1800 - 2359	FALSE	135	23
7	Bad	0600 - 1159	TRUE	138	112
7	Bad	0600 - 1159	FALSE	134	28
7	Bad	1200 - 1759	TRUE	142	43
7	Bad	1200 - 1759	FALSE	138	31
7	Bad	1800 - 2359	TRUE	141	76
7	Bad	1800 - 2359	FALSE	130	16
8	Good	0600 - 1159	TRUE	142	98
8	Good	0600 - 1159	FALSE	139	47
8	Good	1200 - 1759	TRUE	141	61
8	Good	1200 - 1759	FALSE	140	40
8	Good	1800 - 2359	TRUE	139	74
8	Good	1800 - 2359	FALSE	135	23
8	Bad	0600 - 1159	TRUE	138	111
8	Bad	0600 - 1159	FALSE	134	28
8	Bad	1200 - 1759	TRUE	142	43
8	Bad	1200 - 1759	FALSE	138	31
8	Bad	1800 - 2359	TRUE	141	75
8	Bad	1800 - 2359	FALSE	130	16

9	Good	0600 - 1159	TRUE	142	98
9	Good	0600 - 1159	FALSE	139	47
9	Good	1200 - 1759	TRUE	141	61
9	Good	1200 - 1759	FALSE	140	40
9	Good	1800 - 2359	TRUE	139	74
9	Good	1800 - 2359	FALSE	135	23
9	Bad	0600 - 1159	TRUE	138	110
9	Bad	0600 - 1159	FALSE	134	28
9	Bad	1200 - 1759	TRUE	142	43
9	Bad	1200 - 1759	FALSE	138	31
9	Bad	1800 - 2359	TRUE	141	73
9	Bad	1800 - 2359	FALSE	130	15
10	Good	0600 - 1159	TRUE	142	98
10	Good	0600 - 1159	FALSE	139	47
10	Good	1200 - 1759	TRUE	141	61
10	Good	1200 - 1759	FALSE	140	40
10	Good	1800 - 2359	TRUE	139	73
10	Good	1800 - 2359	FALSE	135	23
10	Bad	0600 - 1159	TRUE	138	110
10	Bad	0600 - 1159	FALSE	134	28
10	Bad	1200 - 1759	TRUE	142	43
10	Bad	1200 - 1759	FALSE	138	31
10	Bad	1800 - 2359	TRUE	141	73
10	Bad	1800 - 2359	FALSE	130	15
11	Good	0600 - 1159	TRUE	142	98
11	Good	0600 - 1159	FALSE	139	46
11	Good	1200 - 1759	TRUE	141	61
11	Good	1200 - 1759	FALSE	140	40
11	Good	1800 - 2359	TRUE	139	72
11	Good	1800 - 2359	FALSE	135	23
11	Bad	0600 - 1159	TRUE	138	110
11	Bad	0600 - 1159	FALSE	134	28
11	Bad	1200 - 1759	TRUE	142	43
11	Bad	1200 - 1759	FALSE	138	31
11	Bad	1800 - 2359	TRUE	141	72
11	Bad	1800 - 2359	FALSE	130	15
12	Good	0600 - 1159	TRUE	142	98
12	Good	0600 - 1159	FALSE	139	46
12	Good	1200 - 1759	TRUE	141	61
12	Good	1200 - 1759	FALSE	140	40
12	Good	1800 - 2359	TRUE	139	72
12	Good	1800 - 2359	FALSE	135	23
12	Bad	0600 - 1159	TRUE	138	110
12	Bad	0600 - 1159	FALSE	134	27
12	Bad	1200 - 1759	TRUE	142	43
12	Bad	1200 - 1759	FALSE	138	31
12	Bad	1800 - 2359	TRUE	141	72
12	Bad	1800 - 2359	FALSE	130	15

Appendix B   Code - Data Processing and Test Statistics

```r
library(tidyverse)
library(lubridate)

# load in data, bike_sharing_hourly.csv has to be in
# same folder with this script
# stations.csv in same folder

stations=read_csv("stations.csv")

data = NULL

for(i in 2011:2012){ # the data goes up to 2017, but the files are extremely large from 2011 onwards - you can decide to just use a subset
   file = paste0('https://s3.amazonaws.com/capitalbikeshare-data/',i,'-capitalbikeshare-tripdata.zip')
   download.file(file,destfile='bikedata.zip')
   unzip('bikedata.zip')
  if (i == 2011) {
    data = rbind(data,read_csv(paste0(i,'-capitalbikeshare-tripdata.csv')))
  }
  if (i == 2012) {
    for (j in 1:4) {
      data = rbind(data,read_csv(
        paste0(i, 'Q', j, '-capitalbikeshare-tripdata.csv')))
    }
  }
}
colnames(data) = make.names(colnames(data))
data = data %>%
  filter(hour(Start.date) >= 6 & hour(Start.date) <= 23 &
           hour(End.date) >= 6 & hour(End.date) <= 23) %>%
  mutate(interval = cut(hour(Start.date), breaks = c(6, 11, 17, 23),
                        include.lowest = T, labels = 1:3),
         dteday = date(Start.date))

weather = read_csv("bike_sharing_hourly.csv")
weather2 = weather %>%
  filter(hr >= 6 & hr <= 23) %>%
  mutate(interval = cut(hr, breaks = c(6, 11, 17, 23),
                        include.lowest = T, labels = 1:3),
         is.good = weathersit == 1) %>%
  group_by(dteday, interval) %>%
  summarise(good.pct = mean(is.good)) %>%
  mutate(good.weather = good.pct > 2/3)

datafull = left_join(data, weather2,
                     by=c('dteday'='dteday', 'interval'='interval'))

# Function to filter dataframe
# Input: Conditions
# Output: filtered dataframe containing only the relevant observations

get_df = function(cutoff1 = ymd("2011-01-01"),
                  cutoff2 = ymd("2012-12-31"),
                  t1 = 0, t2 = 23,
                  which.wkdays = 1:7,
                  which.weather = c(T, F),
                  current_data = datafull) {

  df = current_data %>%
    # get trips between the cutoffs
    filter(date(Start.date) >= cutoff1 &
             date(End.date) <= cutoff2) %>%

    # choose between weekdays (2:6) or weekends (c(1, 7))
    filter(wday(Start.date) %in% which.wkdays) %>%

    # choose kind of weather
    filter(good.weather %in% which.weather) %>%

    # departure time must be inside time window
    filter(t1 <= hour(Start.date) &
             hour(Start.date) <= t2) %>%

    # arrival time must be inside time window
    filter(t1 <= hour(End.date) &
             hour(End.date) <= t2)

  df
}

# Weights Function
# Input: Conditions
# Output: filter dataframe using get_df then calculate weights


get_weights = function(cutoff1 = ymd("2011-01-01"),
                       cutoff2 = ymd("2012-12-31"),
                       t1 = 0, t2= 23,
                       which.wkdays = 1:7,
                       which.weather = 1:4,
                       current_data = datafull) {

  df = get_df(cutoff1, cutoff2, t1, t2,
              which.wkdays, which.weather, current_data)
  df %>%
    group_by(Start.station.number, End.station.number) %>%
    summarise(n = n()) %>%
    mutate(w = n / sum(n))
}



# p value function

timing = matrix(c(6,11,12,17,18,23), 2,3)
rownames(timing) = c("start", "end")


T_t_m = function(month, which.weather, interval.no,
                 which.wkdays) {
  cutoff1 = ymd(paste0("2012-", toString(month), "-01"))
  cutoff2 = ceiling_date(cutoff1, unit = 'month') - 1
  t1 = timing[1,interval.no]
  t2 = timing[2,interval.no]

  n_days = weather2 %>%
    filter(cutoff1 <= dteday & dteday <= cutoff2
           & interval == interval.no
           & good.weather %in% which.weather) %>% nrow()

  df = get_df(cutoff1, cutoff2, t1, t2,
              which.wkdays, which.weather) %>%
    group_by(Start.station.number, End.station.number) %>%
```

```r
    summarise(n = n())

  weights.df = get_weights(ymd("2011-01-01"), cutoff2,
                             t1, t2, which.wkdays, which.weather)

  curr.mth = df %>%
    group_by(Start.station.number) %>%
    summarise(Nstart = sum(n) / n_days)

  weights.full = left_join(weights.df, curr.mth,
                             by="Start.station.number") %>%
    mutate(wN = w * replace_na(Nstart, 0))

  depart = weights.full %>%
    filter(Start.station.number != End.station.number) %>%
    group_by(Start.station.number) %>%
    summarise(dep = sum(wN),
              sum_w = sum(w))

  arrive = weights.full %>%
    filter(Start.station.number != End.station.number) %>%
    group_by(End.station.number) %>%
    summarise(arr = sum(wN),
              var_3rdpart = sum(wN * w))

  dep.n.arr = full_join(depart, arrive,
                          by=c("Start.station.number"=
                                 "End.station.number"))
  colnames(dep.n.arr)[1]="id"
  rv = dep.n.arr %>%
    filter(arr!=0) %>%
    mutate(dep = replace_na(dep,0),
           sum_w = replace_na(sum_w, 0),
           Tstat = (dep - arr) /
             ( sqrt(arr * sum_w + var_3rdpart) / sqrt(n_days) ),
           pval = 2 * pnorm( -abs(Tstat) ) ,
           weather = which.weather,
           interval = interval.no,
           wkday = rep(identical(which.wkdays,2:6),nrow(dep.n.arr %>% filter(arr!
  =0)))
           )
  rv2=merge(rv,select(stations,id,zip))
  rv2$downtown=ifelse(rv2$zip %in% c(20001,20005,20006,20036),1,0)
  return(rv2)
}


#each month has its own list
#each condition in each month is its own df
#name of each df in a particular month is coded as weather/interval/wkday

#weather: T if good weather. F if bad
#interval: 1 is 0600-1159, 2 is 1200-1759, 3 is 1800-2350
#wkday: T if weekday. F if weekend

output=list()
wthr=c(T,F)
interval=1:3
wkday=list(2:6,c(1,7))

for (m in 1:12){
  output[[paste0("m",m)]]=list()
  for (w in wthr){
    for (i in interval){
      for (k in wkday){
        k2=identical(k,2:6)
        output[[paste0("m",m)]][[paste(w,i,k2)]]=T_t_m(month=1,which.weather=w,
    interval.no=i,which.wkdays=k)
      }
    }
  }
}

save(output,file="output.RData")



#subset dataframes for comparison with observed data
df1=subset(get_df("2012-04-01","2012-04-30",6,11,2:6,T,datafull),
           Start.station.number==31100 | End.station.number==31100)
df2=subset(get_df("2012-04-01","2012-04-30",6,11,2:6,T,datafull),
           Start.station.number==31619 | End.station.number==31619)
df3=subset(get_df("2012-04-01","2012-04-30",6,11,2:6,T,datafull),
           Start.station.number==31201 | End.station.number==31201)
df4=subset(get_df("2012-04-01","2012-04-30",6,11,2:6,T,datafull),
           Start.station.number==31106 | End.station.number==31106)

list.bw=list()
for (i in 1:length(output[[4]][[1]]$id)){
  id=output[[4]][[1]]$id[i]
  list.bw[[paste(id)]]=subset(get_df("2012-04-01","2012-04-30",6,11,2:6,T,
    datafull),
                (Start.station.number==id | End.station.number==id))
}
```

Appendix C   Code - Multiple Testing

```
################################
#### Code by Li and Barber ####
################################

Solve_q_block = function(Pvals, tau, eps, blocks, ADMM_params){
  # blocks[i] gives the index of the block to which Pvals[i] belongs
  block_proj = create_block_function(blocks)
  q_init = block_proj((Pvals>tau)/(1-tau))
  if(min(q_init)>=eps & max(q_init)<=1){
    q = q_init
  }else{
    M = diag(length(Pvals))
    q = Solve_q_ADMM(Pvals, tau, eps, M, block_proj, ADMM_params)
  }
  q
}

create_block_function = function(blocks){
  function(y){
    # solving: min{1/2 ||x-y||^2_2 : x is constant over the blocks}
    x = y
    block_inds = sort(unique(blocks))
    for(i in block_inds){
      x[which(blocks==block_inds[i])] = mean(x[which(blocks==block_inds[i])])
    }
    x
  }
}

Solve_q_ADMM = function(Pvals, tau, eps, M, projection, ADMM_params){
  # min -sum_i (B[i]*log((1-tau) q[i]) + (1-B[i])*log(1-(1-tau) q[i]))
  # subject to (1) q \in Qset (characterized by M*q \in Mset)
  # and (2) sum_i B[i]/q[i] <= gamma and (3) eps<=q<=1
  # introduce auxiliary variables x, y under the constraint Mq = x, q = y
  # ADMM optimization:
  # minimize -sum_i (B_i*log((1-tau) q_i)+(1-B_i)*log(1-(1-tau) q_i)) + <u, Mq-x>
    + <v, q-y> + alpha/2 ||Mq-x||^2 + beta/2 ||q-y||^2 + alpha/2 (q-qt)'(eta I -
   M'M)(q-qt)
  # where qt is the previous iteration's q value

  # ADMM_params are: alpha, beta, eta, max_iters, converge_thr
  alpha_ADMM = ADMM_params[1]
  beta = ADMM_params[2]
  eta = ADMM_params[3]
  max_iters = ADMM_params[4]
  converge_thr = ADMM_params[5]

  n = length(Pvals)
  B = (Pvals > tau)
  gamma = n*(1-tau) # bound on sum_i (Pvals[i]>tau) / q[i]*(1-tau)
  q = y = rep(1,n)
  v = rep(0,n)
  u = x = rep(0,dim(M)[1])

  converge_check = function(q,x,y,u,v,q_old,x_old,y_old,u_old,v_old){
    max(c(sqrt(sum((q-q_old)^2))/sqrt(1+sum(q_old^2)),
          sqrt(sum((x-x_old)^2))/sqrt(1+sum(x_old^2)),
          sqrt(sum((y-y_old)^2))/sqrt(1+sum(y_old^2)),
          sqrt(sum((u-u_old)^2))/sqrt(1+sum(u_old^2)),
          sqrt(sum((v-v_old)^2))/sqrt(1+sum(v_old^2))))
  }
```

```r

  stop = FALSE
  iter = 0
  while(!stop){
    iter = iter+1
    q_old = q; x_old = x; y_old = y; u_old = u; v_old = v
    q = q_update(B, M, tau,eps,q,x,y,u,v,alpha_ADMM,gamma,beta, eta)
    x = x_update(B, M, tau,eps,q,x,y,u,v,alpha_ADMM,gamma,beta, eta, projection)
    y = y_update(B, M, tau,eps,q,x,y,u,v,alpha_ADMM,gamma,beta, eta)
    u = u_update(B, M, tau,eps,q,x,y,u,v,alpha_ADMM,gamma,beta, eta)
    v = v_update(B, M, tau,eps,q,x,y,u,v,alpha_ADMM,gamma,beta, eta)
    if(converge_check(q,x,y,u,v,q_old,x_old,y_old,u_old,v_old)<=converge_thr){
   stop=TRUE}
    if(iter>=max_iters){stop=TRUE}
  }

  return(q)

}


# inverse_sum_prox solves: min{1/2 ||x-y||^2 : x_i>0, sum_i 1/x_i <= bound}
# Used in y-update step of ADMM
inverse_sum_prox = function(y,bound){

  y = pmax(0,y) # the solution will have all positive x_i's now
  # and we can now ignore the constraint x_i>0

  if(sum(1/y)<= bound){
    x=y
  }else{ # use Lagrange multipliers

    # we should have - lambda * d/dx_j (sum_i 1/x_i) = d/dx_j (1/2 ||x-y||^2)
    # for all j, for some single lambda>0
    # in other words, lambda / x^2 = x-y (this holds elementwise)
    # rearranging, lambda = x^3 - x^2*y
    # let c = log(lambda) so that it's real-valued
    # we need to solve x^3 - x^2*y - exp(c) = 0 (elementwise)

    cuberoot = function(c){ # this solves the cubic equation x^3-x^2*y-exp(c)=0
      temp1 = ((y/3)^3 + exp(c)/2 + (exp(c)*(y/3)^3 + exp(c)^2/4)^0.5)
      temp2 = ((y/3)^3 + exp(c)/2 - (exp(c)*(y/3)^3 + exp(c)^2/4)^0.5)
      x = sign(temp1)*abs(temp1)^(1/3) + sign(temp2)*abs(temp2)^(1/3) + (y/3)
      x
    }

    # now we need to choose c, i.e. choose the lagrange multiplier lambda=exp(c)
    # the right value of c is the one that produces an x satisfying sum_i 1/x_i =
     bound

    c = uniroot(function(c){sum(1/cuberoot(c))-bound},c(-100,100))$root
    x = cuberoot(c)
  }
  x
}

q_update = function(B, M, tau,eps,q,x,y,u,v,alpha,gamma,beta, eta){
  # minimize -sum_i (B_i*log((1-tau) q_i)+(1-B_i)*log(1-(1-tau) q_i)) + <u, Mq-x>
     + <v, q-y> + alpha/2 ||Mq-x||^2 + beta/2 ||q-y||^2 + alpha/2 (q-qt)'(eta I -
   M'M)(q-qt)
  # where qt is the previous iteration's q value
  # equivalently, -sum_i (B_i*log((1-tau) q_i)+(1-B_i)*log(1-(1-tau) q_i)) + (
    alpha eta + beta)/2 * ||q-w||_2^2
  # where w = - (M'(ut + alpha (M qt - xt)) + (vt - beta yt - alpha eta qt))/(
    alpha eta + beta)

  w = - ( t(M)%*%(u + alpha*(M%*%q - x)) + (v - beta*y - alpha*eta*q) )/(alpha*
    eta + beta)

  q[B==1] = (w[which(B==1)]+sqrt(w[which(B==1)]^2+4/(alpha*eta + beta)))/2
  q[B==0] = ((w[which(B==0)]+1/(1-tau))-sqrt((w[which(B==0)]-1/(1-tau))^2+4/(
    alpha*eta+beta)))/2
  q[q<eps] = eps
  q[q>1] = 1
  q
}

x_update = function(B, M, tau,eps,q,x,y,u,v,alpha,gamma,beta, eta, projection){
  # Proj_Mset (M q + u/alpha)
  x = projection(M%*%q + u/alpha)
}

y_update = function(B, M, tau,eps,q,x,y,u,v,alpha,gamma,beta, eta){
  # Prof_B (q + v/beta)
  # where B = {sum_i B[i]/y[i]<= gamma}
  y = q + v/beta
  y[which(B==1)] = inverse_sum_prox((q+v/beta)[which(B==1)], gamma)
  y
}

u_update = function(B, M, tau,eps,q,x,y,u,v,alpha,gamma,beta, eta){
  u = u + alpha * (M%*%q -x)
  u
}

v_update = function(B, M, tau,eps,q,x,y,u,v,alpha,gamma,beta, eta){
  v = v + beta * (q-y)
  v
}

#################################
###### Compute rejections ######
#################################

#output.RData in same folder

#each month has its own list
#each condition in each month is its own df
#name of each df in a particular month is coded as weather/interval/wkday

#weather: T if good weather. F if bad
#interval: 1 is 0600-1159, 2 is 1200-1759, 3 is 1800-2350
#wkday: T if weekday. F if weekend
#downtown: 1 if stn in downtown, 0 otherwise


#set group id. 1-24 for each month. 2 for each condition.
#odd numbers are non-downtown, even numbers are downtown.

library(dplyr)
load("output.RData")

for (i in 1:12){
  g=1
  for (j in 1:12){
    output[[i]][[j]]$group=ifelse(output[[i]][[j]]$downtown==0,g,g+1)
    g=g+2
  }
}


#overall alpha
a=0.01

#beta(k)
b=function(k){
  return(a/(4*k*(log(max(k,2)))^2))
}

#compute rejections
rej.count=c()
a.seq=c()
b.seq=c()
for (k in 1:12){
  prd=select(bind_rows(output[[k]]),id,pval,group)

  #compute alpha(k)
  ak=(b(k)*max(1,sum(rej.count)))/nrow(prd)
  a.seq=c(a.seq,ak)
  b.seq=c(b.seq,b(k))
  params=c(1,1,2,5000,0.0001)
  q=Solve_q_block(prd$pval,0.5,0.1,prd$group,params)
  k.hat=max(c(0,which(sort(q*prd$pval)<=ak*(1:length(prd$pval))/length(prd$pval))
    ))
  rej.ind=(which(q*prd$pval<=ak*k.hat/length(prd$pval)))
  rej.count[k]=length(rej.ind)
  for (i in 1:12){
    len=nrow(output[[k]][[i]])
    output[[k]][[i]]$rej=ifelse(row(output[[k]][[i]])[,1] %in% rej.ind,T,F)
    rej.ind=rej.ind-len
  }
}

output.final=output
#rej: true if rej. false otherwise
save(output.final,file="output.final.RData")

#create results table
d=data.frame(matrix(nrow=0,ncol=6))
colnames(d)=c("Month","Weather","Time","Weekday","# stations","# rejections")
for (i in 1:12){
  for (j in 1:12){
    Month=i
    Weather=ifelse(output.final[[i]][[j]]$weather[1]=="TRUE","Good","Bad")
    Time=ifelse(output.final[[i]][[j]]$interval[1]==1,"0600 - 1159",
                  ifelse(output.final[[i]][[j]]$interval[1]==2,"1200 - 1759","1800
    - 2359"))
    Weekday=output.final[[i]][[j]]$wkday[1]
    `No. of stations`=nrow(output.final[[i]][[j]])
    `No. of rejections`=sum(output.final[[i]][[j]]$rej)
    d[nrow(d)+1,]=c(Month,Weather,Time,Weekday,`No. of stations`,`No. of
    rejections`)
  }
}
d %>%
  kable(format="latex",booktabs=TRUE,linesep="")

d2=d %>%
  group_by(Month=as.numeric(Month)) %>%
  summarise(`# hypotheses`=sum(as.numeric((`# stations`))),`# rejections`=sum(as.
    numeric((`# rejections`))))
d2$alpha.k=a.seq
d2$beta.k=b.seq
d2[,c(1,4,5,2,3)] %>%
  kable(format="latex",booktabs=TRUE,linesep="")
```

Appendix D Code - Results (Short Run Analysis)

```r
library(tidyverse)
library(lubridate)
library(ggmap)

#good weather, interval, Weekday = T,1,T
load.Rdata("output.final.RData", "output.final")

#First get the map
dc <- c(left = -77.12, bottom = 38.83,
            right = -76.93, top = 38.96)
dc_map <- get_stamenmap(bbox = dc, zoom = 11, maptype ='terrain')
p <- ggmap(dc_map, color = 'color')
stations=read_csv("stations.csv")


##March, Good/Bad Weather, interval, weekday
Intervention = ifelse(output.final$m3$`FALSE 3 TRUE`$rej == TRUE, "Yes", "No")
color_name = ifelse(output.final$m3$`FALSE 3 TRUE`$rej == TRUE, "blue", "red")

present_stations = output.final$m3$`FALSE 3 TRUE`$id
stations_filtered = stations[which(stations$id %in% present_stations), ]
p + geom_point(aes(x = long, y = lat, colour = Intervention), data=stations_filtered,
               size = 2, alpha=0.7) + scale_colour_manual(breaks = Intervention,
                                                         values = unique(as.character(color_name)))


##March, Good/Bad Weather, interval, Weekend
Intervention = ifelse(output.final$m3$`FALSE 3 FALSE`$rej == TRUE, "Yes", "No")
color_name = ifelse(output.final$m3$`FALSE 3 FALSE`$rej == TRUE, "blue", "red")

present_stations = output.final$m3$`FALSE 3 FALSE`$id
stations_filtered = stations[which(stations$id %in% present_stations), ]
p + geom_point(aes(x = long, y = lat, colour = Intervention), data=stations_filtered,
               size = 2, alpha=0.7) + scale_colour_manual(breaks = Intervention,
                                                                  values = unique(as.character(color_name)))
```

Appendix E Code - Results (Short Run Comparison)

```r
library(tidyverse)
library(lubridate)
library(ggmap)

dates=unique(day(df1$Start.date))

dep.arr=function(df,dates,station){
  data=data.frame()
  for (d in dates){
    data.t=df %>%
      filter(day(Start.date)==d) %>%
      summarise(dep=sum(Start.station.number==station),arr=sum(End.station.number
   ==station)) %>%
      mutate(date=d)
    data=rbind(data,data.t)
  }
  data=data[,c(3,1,2)]
  data
}

#part 1 of comparison
load("df1.RData")
load("df2.RData")
load("df3.RData")
load("df4.RData")
stn1=dep.arr(df1,dates,31100)
stn2=dep.arr(df2,dates,31619)
stn3=dep.arr(df3,dates,31201)
stn4=dep.arr(df4,dates,31106)
merge(stn1,stn2,by="date") %>% kable(format="latex",booktabs=TRUE,linesep="")
merge(stn3,stn4,by="date") %>% kable(format="latex",booktabs=TRUE,linesep="")

#part 2 of comparison
stations=read_csv("stations.csv")
load("list.bw.RData")
load("output.final.RData")
df.rej=output.final[[4]][[1]]

list.bw2=list()
for (i in 1:142){
  list.bw2[[names(list.bw)[i]]]=cbind(data.frame(id=rep(as.numeric(names(list.bw)
    [i]),length(dates))),
                                              dep.arr(list.bw[[i]],dates,as.numeric(names
    (list.bw)[i])))
}


df.bw=Reduce(function(...) merge(..., all=T),list.bw2) %>%
  group_by(id) %>%
  transmute(date=date,
            intervention=ifelse((2*pmin(dep,arr)<pmax(dep,arr)),1,0)) %>%
  summarise(intervention=(sum(intervention)>0.5*length(date)))
df.bw2=left_join(left_join(select(df.rej,id,rej),df.bw,by="id"),select(stations,
    id,lat,long),by="id") %>%
    mutate(match=factor(ifelse(rej==intervention,TRUE,ifelse(rej==TRUE,"FALSE 2",
    "FALSE 1")),
                         levels=c("TRUE","FALSE 1","FALSE 2")))

dc=c(left=-77.12,bottom=38.83,right=-76.93,top=38.96)
dc_map=get_stamenmap(bbox = dc, zoom = 11, maptype ='terrain')
p=ggmap(dc_map,color='color')
```

```
p+geom_point(aes(x=long,y=lat,colour=match),data=df.bw2,size=2,alpha=0.7)+
  scale_colour_manual(breaks=c("TRUE","FALSE 1","FALSE 2"),values=c("purple","red
    ","blue"))

sum(df.bw2$match=="TRUE")
sum(df.bw2$match=="FALSE 1")
sum(df.bw2$match=="FALSE 2")
```

Appendix F Code - Results (Long Run Analysis)

```r
library(tidyverse)
library(ggmap)

# loading results
load("output.final.RData")

mth.names = 1:12

df = data.frame()

for (i in 1:12) {
  for (j in 1:12) {
    dfij = output.final[[i]][[j]] %>%
      select(id, weather, interval, wkday, rej) %>%
      mutate(mth = mth.names[i])
    df = rbind(df, dfij)
  }
}

# getting the map for DC
dc <- c(left = -77.12, bottom = 38.83,
            right = -76.93, top = 38.96)
dc_map <- get_stamenmap(bbox = dc, zoom = 11, maptype ='terrain')
p <- ggmap(dc_map, color = 'color')

# loading coordinates of stations
stations=read_csv("stations.csv")

# wrap map drawing code into a function for repeatable use
draw_map = function() {
  p +
    geom_point(aes(x = long, y = lat, colour = Stable),
              data = stations_filtered,
              size = 2, alpha=0.7) +
    scale_colour_manual(breaks = Stable,
                        values = unique(color_name))
}

# LONG RUN TREND IN WEEKDAYS VS WEEKENDS
overview1 = df %>%
  group_by(id, wkday) %>%
  summarise(rej = sum(rej)) %>%
  spread(wkday, rej, fill=0) %>%
  rename(!!'Weekends':=`FALSE`, !!'Weekdays':=`TRUE`)

stations_filtered = stations[stations$id %in% overview1$id, ]

# generate percentage table
colSums(overview1==0)[-1] * 100 / nrow(overview1)
colSums(overview1>0 & overview1<54)[-1] * 100 / nrow(overview1)
colSums(overview1>=54)[-1] * 100 / nrow(overview1)

# stable stations on weekdays
Stable = ifelse(overview1$Weekdays == 0,
                     "Yes",ifelse(overview1$Weekdays>=54,"H. unstab","No"))
color_name = c("Red","Purple","Blue")

draw_map()

# stable stations on weekends
Stable = ifelse(overview1$Weekends == 0,
                "Yes",ifelse(overview1$Weekends>=54,"H. unstab","No"))
color_name = c("Red","Purple","Blue")
draw_map()

##################################

# LONG RUN TREND IN INTERVALS 1, 2, 3
overview2 = df %>%
  group_by(id, interval) %>%
  summarise(rej = sum(rej)) %>%
  spread(interval, rej, fill=0)

stations_filtered = stations[stations$id %in% overview2$id, ]

# generate percentage table
colSums(overview2==0)[-1] * 100 / nrow(overview2)
colSums(overview2>0 & overview2<36)[-1] * 100 / nrow(overview2)
colSums(overview2>=36)[-1] * 100 / nrow(overview2)

# stable stations in interval 1
Stable = ifelse(overview2$`1` == 0,
                "Yes",ifelse(overview2$`1`>=36,"H. unstab","No"))
color_name = c("Red","Purple","Blue")
draw_map()

# stable stations in interval 2
Stable = ifelse(overview2$`2` == 0,
                "Yes",ifelse(overview2$`2`>=36,"H. unstab","No"))
color_name = c("Red","Purple","Blue")
draw_map()

# stable stations in interval 3
Stable = ifelse(overview2$`3` == 0,
                "Yes",ifelse(overview2$`3`>=36,"H. unstab","No"))
color_name = c("Red","Purple","Blue")
draw_map()

# generate percentage tables for refined analysis
overview2.wkday = df %>%
  filter(wkday) %>%
  group_by(id, interval) %>%
  summarise(rej = sum(rej)) %>%
  spread(interval, rej, fill=0)

overview2.wkend = df %>%
  filter(!wkday) %>%
  group_by(id, interval) %>%
  summarise(rej = sum(rej)) %>%
  spread(interval, rej, fill=0)

colSums(overview2.wkday==0)[-1] * 100 / nrow(overview2.wkday)
colSums(overview2.wkday>0 & overview2.wkday<18)[-1] * 100 / nrow(overview2.wkday)
colSums(overview2.wkday>=18)[-1] * 100 / nrow(overview2.wkday)

colSums(overview2.wkend==0)[-1] * 100 / nrow(overview2.wkend)
colSums(overview2.wkend>0 & overview2.wkend<18)[-1] * 100 / nrow(overview2.wkend)
colSums(overview2.wkend>=18)[-1] * 100 / nrow(overview2.wkend)

##################################

# LONG RUN TREND IN GOOD WEATHER VS BAD WEATHER

overview3 = df %>%
```

```r
  group_by(id, weather) %>%
  summarise(rej = sum(rej)) %>%
  spread(weather, rej, fill=0) %>%
  rename(!!'Bad.weather':=`FALSE`, !!'Good.weather':=`TRUE`)

stations_filtered = stations[stations$id %in% overview3$id, ]

# stable stations under good weather

Stable = ifelse(overview3$Good.weather == 0,
                "Yes",ifelse(overview3$Good.weather>=54,"H. unstab","No"))
color_name = c("Red","Purple","Blue")
draw_map()

# stable stations under bad weather
Stable = ifelse(overview3$Bad.weather == 0,
                "Yes",ifelse(overview3$Bad.weather>=54,"H. unstab","No"))
color_name = c("Red","Purple","Blue")
draw_map()

colSums(overview3==0)[-1] * 100 / nrow(overview3)
colSums(overview3>0 & overview3<54)[-1] * 100 / nrow(overview3)
colSums(overview3>=54)[-1] * 100 / nrow(overview3)

################################

# CONSISTENTLY HIGHLY UNSTABLE
s1=which(overview1$Weekends>=54 & overview1$Weekdays>=54)
s2=which(overview3$Good.weather>=54 & overview3$Bad.weather>=54)
s3=which(overview2$`1`>=36 & overview2$`2`>=36 & overview2$`3`>=36)
stations_filtered=stations[stations$id %in% overview1[intersect(intersect(s1,s2),
    s3),]$id,]

p + geom_point(aes(x = long, y = lat), colour = "red",
               data = stations_filtered,
               size = 2, alpha=0.7) + theme(legend.position="none")


################################

# OVER TIME

df2=data.frame()
for (i in 1:12) {
    df2i = output.final[[i]][[1]] %>%
      select(id, rej) %>%
      mutate(mth = mth.names[i])
    df2 = rbind(df2, df2i)
}

for (i in 1:12){
  print(nrow(subset(df2,mth==i & rej==TRUE)))
}

stn.id=output.final[[1]][[1]]$id
for (i in 1:12){
  stn.id=intersect(stn.id,subset(df2,mth==i & rej==TRUE)$id)
}
```