How do Citi Bike Station Closures Affect Rides from Nearby Stations?

Bill Mabe

7/5/2019

Overview

This is just a basic test of a proposition that's so obviously true that it doesn't really need to be tested: Does the (temporary) closing of a Citi Bike station increase ridership at nearby stations?

The answer it turns out – don't even bother with the drumroll – is that yes, rides increase (by about 9 percent at the three nearest stations, after controlling for a few variables that likely affect ridership).

I've written this up mostly as a means to walk through how the data need to be cleaned, how the data need to be munged in order to answer the question, and some of the statistical tests that can be run to estimate the effect of station closure on ridership.

```
### Load Some Packages ----------------------------------------------------
require(readr)
require(dplyr)
require(ggplot2)
require(knitr)
require(R.utils)
require(lubridate)
require(raster)
require(broom)

setwd("~/Documents/")

###  Create a list of data frames, each of which is a month of Citi Bike data. ------
---------
filenames <- list.files(pattern = "csv")
citi_list <- vector(mode = "list", length(filenames))
for(i in seq_along(filenames)) {
  citi_list[[i]] <- read_csv(filenames[i])
  }

# Name the elements of the list.
names(citi_list) <- substr(filenames, 1, 6)

# Standardize the names of all columns across all data frames in the list.
citi_list <- lapply(citi_list, setNames, nm = c("tripduration", "starttime", "stoptim
e", "start_station_id", "start_station_name", "start_station_latitude", "start_statio
n_longitude", "end_station_id", "end_station_name", "end_station_latitude", "end_stat
ion_longitude", "bikeid", "usertype", "birth_year", "gender"))

### Create one large data from from the list ------------------------------------
---------

# Create one huge data frame (64 million or so rows) and add a variable indicating th
e month (which will become very useful below).
citi_df <- do.call("rbind", citi_list)
citi_df <- transform(citi_df, month = as.numeric(rep(names(citi_list), sapply(citi_li
st, nrow))))
rownames(citi_df) <- seq_len(nrow(citi_df))

# listwise delete all rows that don't have a station id (about 1600).
citi_df <- citi_df[!is.na(citi_df$start_station_id), ]
```

Cleaning Citi Bike Dates

From examining the data – i.e., in running code and having it bomb :(– it became clear that:

1. starting in October 2016 Citi Bike changed how it stored date data from a charcter string of m/d/yyyy hms to an integer; and

2. For the first 6 months of 2015, seconds were not recorded in the data variables and so the data was actually m/d/yyyy hm

The code below addresses both of these issues. To handle the first, I split the data into two data frames one inclduing all observations from September 2016 and earlier and the other consisting of all observations since then.

The lack of seconds in the observations from the first half of 2015 affected how I read / cleaned the date. The incredibly helpful lubridate package has the very useful `mdy_hms()` function, but because there was no "s" in many of the records, I had to write my own kludgy workaround to make sure the dates were properly formatted.

```
# First, use the month variable created at the end of the prefious code block to spli
t the data into two different data frames, starting with the more recent data as it w
as easier to handle. Then create clean date and time variables.

### More recent data ------------------------------------------------------------
---------
citi_df_intdate <- citi_df %>%
                    filter(month > 201609) %>%
                    mutate(starttime_num = as.numeric(starttime),
                                           usable_date = as.POSIXct(starttime_num, t
z = "GMT", origin = "1970-01-01"),
                           Date_asdate = as.Date(usable_date),
                           hour = hour(usable_date)) %>%
                    dplyr::select(-starttime_num, -usable_date)

###  Older data -------------------------------------------------------------------
---------
citi_df_chardate <- citi_df %>% filter(month <= 201609)

# (i) Create a month variable
citi_df_chardate$mon <- substr(citi_df_chardate$starttime, 1, 2)
citi_df_chardate$mon <- sub("/", "", citi_df_chardate$mon)
citi_df_chardate$mon <- str_pad(string =  citi_df_chardate$mon, side = "left", width
= 2, pad = "0")

# (ii) Create a day variable
citi_df_chardate$small_date <- sub("^[^/]*", "", substr(citi_df_chardate$starttime, 1
, 10))
citi_df_chardate$day <- substr(citi_df_chardate$small_date, 2, 3)
citi_df_chardate$day <- sub("/", "", citi_df_chardate$day)
citi_df_chardate$day <- str_pad(string =  citi_df_chardate$day, side = "left", width
= 2, pad = "0")

# (iii) Create a year variable
citi_df_chardate$small_date_new <- substr(citi_df_chardate$small_date, 2, 8)
citi_df_chardate$year <- sub("^[^/]*", "", citi_df_chardate$small_date_new)
citi_df_chardate$year <- sub("/", "", citi_df_chardate$year, fixed = TRUE)
citi_df_chardate$year <- sub(" ", "", citi_df_chardate$year)
```

```
# (iv) Create a clean_date variable
citi_df_chardate$clean_date <- paste0(citi_df_chardate$mon, "/", citi_df_chardate$day
, "/", citi_df_chardate$year)

citi_df_chardate$Date_asdate <- mdy(citi_df_chardate$clean_date)

citi_df_chardate <- citi_df_chardate %>% dplyr::select(-mon, -small_date, -day, -smal
l_date_new, -year, -clean_date, -time)

table(is.na(citi_df_chardate$Date_asdate))

# create hour variable

citi_df_chardate <- citi_df_chardate %>% mutate(time = unlist(lapply(strsplit(citi_df
_chardate$starttime, " "), function(x) x[2])))
citi_df_chardate <- citi_df_chardate %>% mutate(hour = substr(time, 1, 2))

### Put humpty dumpty back together ------------------------------------------------
---------
citi_df_all <- rbind(citi_df_chardate, citi_df_intdate)

# Create month_year variable
month_year <- floor_date(citi_df_all$Date_asdate, unit = "month")
citi_df_all$month_year <- month_year + days_in_month(month_year) - 1
table(citi_df_all$month_year)

citi_df_all <- citi_df_all %>% dplyr::select(tripduration, starttime, stoptime, start
_station_id, start_station_name, start_station_latitude, start_station_longitude, end
_station_id, end_station_name, end_station_latitude, end_station_longitude, bikeid, u
sertype, birth_year, gender, Date_asdate, hour)

# Clean the hour variable from the older data
citi_df_all$hour <- ifelse(citi_df_all$hour %in% c("0", "0:"), "00",
                    ifelse(citi_df_all$hour %in% c("1", "1:"), "01",
                    ifelse(citi_df_all$hour %in% c("2", "2:"), "02",
                    ifelse(citi_df_all$hour %in% c("3", "3:"), "03",
                    ifelse(citi_df_all$hour %in% c("4", "4:"), "04",
                    ifelse(citi_df_all$hour %in% c("5", "5:"), "05",
                    ifelse(citi_df_all$hour %in% c("6", "6:"), "06",
                    ifelse(citi_df_all$hour %in% c("7", "7:"), "07",
                    ifelse(citi_df_all$hour %in% c("8", "8:"), "08",
                    ifelse(citi_df_all$hour %in% c("9", "9:"), "09", citi_df_all$hour
))))))))))
```

Creating Analysis Data Sets

Now that the data have been cleaned the real thinking work begins. How do the data need to be reshaped / rolled up to answer the question?

First thing was to summarize the data to get the number of trip originations per day per station. Of interest fromn this data frame are the stations that have missing values for certain days, indicating that no rides occurred on those days. Later, I will classify any station (i) that was missing from this data frame (citi_df_date) and (ii) that had more than zero rides on a date prior to the date for which ride data were missing as having zero rides (aka trip_originations in the data below) for that day.

```
citi_df_date <- citi_df_all %>% group_by(start_station_id, Date_asdate) %>%
                            summarize(number_trip_originations = n())
```

Finding “zero_ride” Stations

So, first I’m going to identify stations that had zero rides on certain days. This will be done by creating a variable that identifies breaks in a series of dates and then calculates the number of days between the start date and end date of the break. A station then gets classified as a zero ride station for a particular day if that day falls within this interval.

```
### Calculate breaks in the data ----------------------------------------------
---------
a <- citi_df_date %>%
         group_by(start_station_id) %>%
         mutate(date_lead = lead(Date_asdate),
                                                          unbroken = ifelse(Date_asdat
e == date_lead - 1, 1, 0))

broken <- a %>%
              filter(unbroken == 0) %>%
              mutate(days_between = interval(Date_asdate, date_lead)/days(1) - 1,
                                            start_plus1 = Date_asdate + 1,
                                            lead_min1 = date_lead - 1)

### Create a list, the elements of which are date vectors --------------------------
---------
date_list <- vector(mode = "list", length = length(broken))
for(i in 1:nrow(broken)) {
    date_list[[i]] <- broken[i, ]$start_plus1:broken[i, ]$lead_min1
}

zero_ride_days <- unlist(date_list)
zero_ride_stations <- rep(broken$start_station_id, broken$date_lead - broken$Date_asd
ate-1)

### Create the data frame of stations + dates when there were zero trip originations
---------
zero_rides <- tibble(zero_ride_days = zero_ride_days, zero_ride_stations = zero_ride_
stations)
```

Find Stations Closest to the Zero Ride Stations

Now I have to find the closest stations to these zero ride stations. This can be done using a Cartesian product of all stations that had zero rides on a given day with all stations that had rides on those days. The data_frame `station_list_daily` created below lists the station id number and the date of all stations that had GREATER THAN zero trip originations on any given day. The `zero_rides` data frame, created at the end of the last block of code, lists all stations the station id number and the date of all stations that ZERO trip originations on any given day.

These two can be merged together by date to associate all stations that had zero rides on a given day with all stations that had greater than zero rides on a given day. This data frame will be the basis for calculating distances between stations in order to identify nearby stations.

```
### Create a Data Frame of Open Stations Where Primary Key = Station + Date ---------
---------

station_list_daily <- citi_df_all %>% group_by(start_station_id, Date_asdate) %>% sli
ce(1:1)  %>% dplyr::select(start_station_id, Date_asdate, start_station_name, start_s
tation_latitude, start_station_longitude)

station_list_daily <- station_list_daily %>% mutate(date_numeric = as.numeric(Date_as
date))

### Create Cartesian Product of Closed Stations and Open Stations, by Date ----------
--------
ij <- inner_join(zero_rides, station_list_daily, by = c("zero_ride_days" = "date_nume
ric"))

# Pull relevant station info into a data frame whose primary key is start_station_id.
station_list_daily_unique <- station_list_daily %>% dplyr::select(start_station_id, s
tart_station_name, start_station_latitude, start_station_longitude) %>% group_by(star
t_station_id) %>% slice(1:1)

# Create Cartesian product
ij <- inner_join(zero_rides, station_list_daily, by = c("zero_ride_days" = "date_nume
ric"))

# Put the station specific info on each matching row
ij_pre <- inner_join(ij, station_list_daily_unique, by = c("zero_ride_stations" = "st
art_station_id"))

# Name the columns
names(ij_pre) <- c("zero_ride_days", "zero_ride_stations", "start_station_id", "Date_
asdate", "start_station_name", "start_station_latitude", "start_station_longitude", "
Date2", "zero_ride_station_name", "zero_ride_station_latitude", "zero_ride_station_lo
ngitude")

# Remove rows with  missing start_station_id and remove rows where stations ids may h
ave matched with themselves.
ij_pre <- ij_pre %>% filter(!is.na(start_station_id)) %>% filter(zero_ride_stations !
= start_station_id)
```

Calculate Distance Between Stations and Keep Three Closest Stations

A fair question to ask is what qualifies a station as being “nearby” a closed Citi Bike station. There are certainly many ways this could be operationalized, but for the purpose of this quick study, I classified a station as being a closed station on a particular day if it (i) had greater than zero trip originations on that day and (ii) was one of the three stations closest to that other station.

```
### Use `pointDistance()` from Raster Package to Calculate Distance Between Stations
---------
ij1 <- ij_pre %>% rowwise() %>% mutate(pd = pointDistance(c(zero_ride_station_latitud
e, zero_ride_station_longitude), c(start_station_latitude, start_station_longitude),
lonlat = FALSE))

# Keep only the three stations closest to the zero ride station that had rides that d
ay.
ij2 <- ij1 %>%
       arrange(zero_ride_stations, zero_ride_days, pd) %>%
       group_by(zero_ride_stations, zero_ride_days) %>%
       slice(1:3) %>%
       ungroup()
```

```
### Pull in the daily citibike data to get the number of trips per day, so that I wil
l know how many trips were taken ad nearby stations on the days that the closed stati
ons were closed.
citi_df_date_small <- citi_df_date %>% dplyr::select(start_station_id, Date_asdate, n
umber_trip_originations)
```

Create the Analysis Data Set

I need to create a data set that allows me to know which rides were taken: 1. From closed stations when they were not closed 2. From stations located close to stations that were closed a. when the nearby stations were closed b. when the nearby stations were NOT closed

A key point in understanding the comments in the code below is that when I use the label “closed station,” I am referring to any station that was closed at some point during the sample period.

```
### Get unique list of all closed stations ----------------------------------------
---------
closed_stations <- ij2 %>%
                   dplyr::select(zero_ride_stations) %>%
                   group_by(zero_ride_stations) %>%
                   slice(1:1) %>%
                   ungroup()

### 1. Create a Data Frame of All Trips from Closed Stations (When They Were Not Clos
ed) --------------------
```

```
# The first line below pulls ALL rides taken from closed stations (obviously when tho
se stations were not closed).
# The second line removes closed stations that themselves had a closed station near t
hem when they were not closed.
# The third lines assigns a value of zero to the nearby_station_closed variable, beca
use after running the second line of code, no stations nearby were closed.

closed_station_trips <- inner_join(closed_stations, citi_df_date_small, by = c("zero_
ride_stations" = "start_station_id")) %>%
                        anti_join(ij2, by = c("zero_ride_stations" = "start_station_i
d", "Date_asdate" = "Date_asdate")) %>%
                        mutate(nearby_station_closed = 0)
names(closed_station_trips) <- c("start_station_id", "Date_asdate", "number_trip_orig
inations", "nearby_station_closed")

### 2a. Rides Taken from Nearby Stations When Closed Stations Were Closed ----------
-----

nearby_closed <- inner_join(ij2, citi_df_date_small, by = c("start_station_id" = "sta
rt_station_id", "Date_asdate" = "Date_asdate")) %>%
       mutate(nearby_station_closed = 1) %>%
       dplyr::select(start_station_id, Date_asdate, number_trip_originations, nearby_
station_closed) %>%
       group_by(start_station_id, Date_asdate) %>%
       slice(1:1) %>%
       ungroup()

### 2b. Rides Taken from Nearby Stations When Closed Stations Were Open ------------
---

nearby_open <- anti_join(citi_df_date_small, ij2, by = c("start_station_id" = "start_
station_id", "Date_asdate" = "Date_asdate")) %>%
       mutate(nearby_station_closed = 0) %>%
       dplyr::select(start_station_id, Date_asdate, number_trip_originations, nearby_
station_closed) %>%
       group_by(start_station_id, Date_asdate) %>%
       slice(1:1) %>%
       ungroup()

### rbind all these together to create the analysis data set ----------------------
-----
all_trips <- rbind(closed_station_trips, nearby_closed, nearby_open)

# Create some time variables
all_trips <- all_trips %>% mutate(year = year(Date_asdate),
                                  month = month(Date_asdate, label = TRUE),
                                  weekday = weekdays(Date_asdate))
```

Pull in the Weather Data for NYC (for Central Park) and Merge with Trip Data

```
w <- read_csv("weather1.csv")
weather <- w %>% filter(NAME == "NY CITY CENTRAL PARK, NY US")
all_trips_weather <- inner_join(all_trips, weather, by = c("Date_asdate" = "DATE"))
```

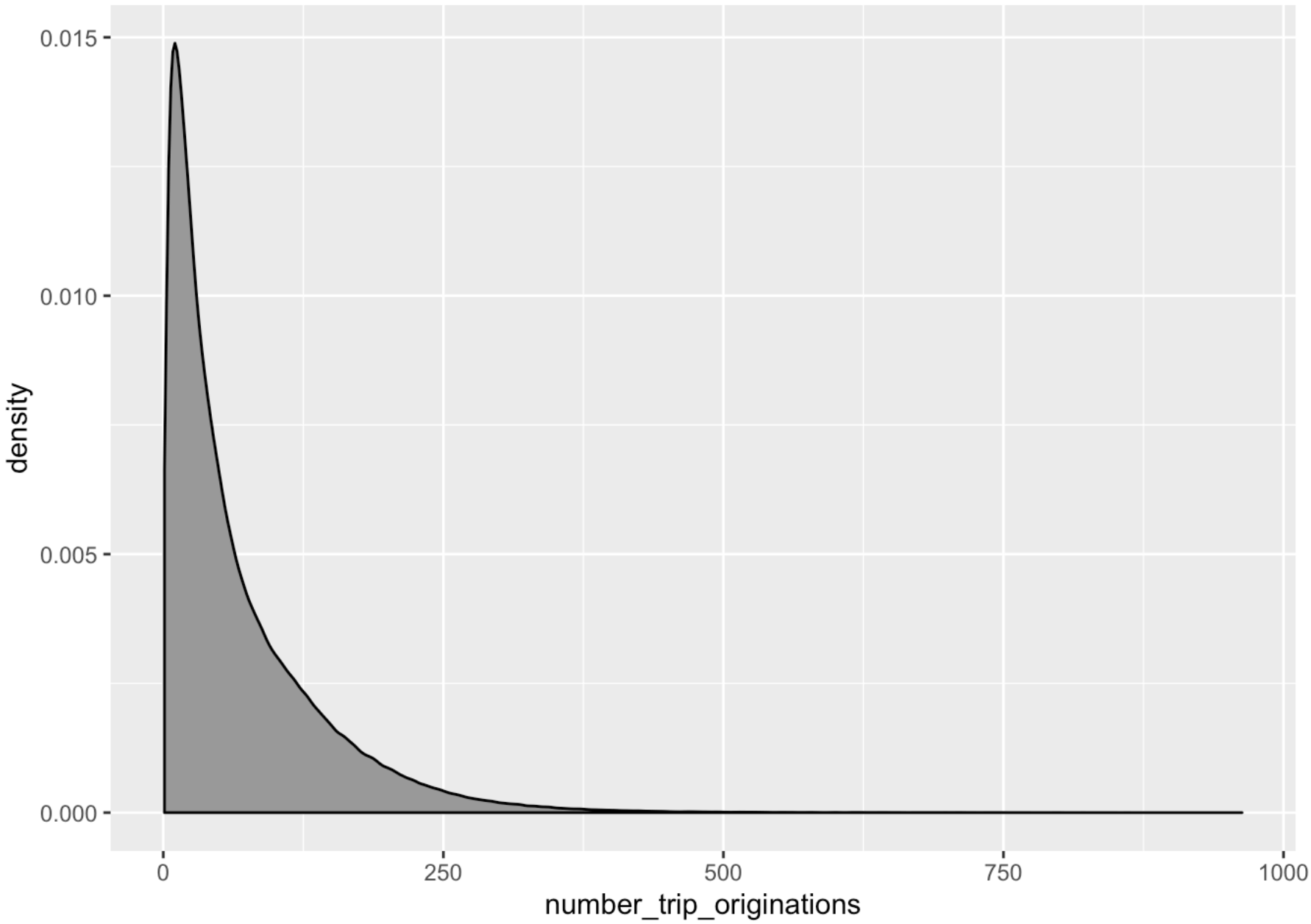
Conduct Data Analysis

```
## Parsed with column specification:
## cols(
##   start_station_id = col_double(),
##   Date_asdate = col_date(format = ""),
##   number_trip_originations = col_double(),
##   nearby_station_closed = col_double(),
##   year = col_double(),
##   month = col_character(),
##   weekday = col_character()
## )
```

```
## Warning: 154 parsing failures.
##    row              col expected actual                                  file
## 791459 start_station_id a double   NULL '~/Documents/Spatial Data/all_trips.csv'
## 791460 start_station_id a double   NULL '~/Documents/Spatial Data/all_trips.csv'
## 791461 start_station_id a double   NULL '~/Documents/Spatial Data/all_trips.csv'
## 791462 start_station_id a double   NULL '~/Documents/Spatial Data/all_trips.csv'
## 791463 start_station_id a double   NULL '~/Documents/Spatial Data/all_trips.csv'
## ...... ................ ........ ...... ......................................
## See problems(...) for more details.
```

```
## Parsed with column specification:
## cols(
##   STATION = col_character(),
##   NAME = col_character(),
##   DATE = col_date(format = ""),
##   PRCP = col_double(),
##   SNWD = col_double(),
##   TMAX = col_double()
## )
```

```
## # A tibble: 23 x 5
##    term                  estimate std.error statistic   p.value
##    <chr>                    <dbl>     <dbl>     <dbl>     <dbl>
##  1 (Intercept)           1674.      82.4         20.3 1.11e- 91
##  2 nearby_station_closed    6.41     0.228       28.1 1.43e-173
##  3 year                    -0.818    0.0408     -20.0 2.55e- 89
##  4 monthAug                 4.33     0.280       15.5 7.54e- 54
##  5 monthDec               -11.6      0.257      -45.0 0.
##  6 monthFeb               -11.3      0.260      -43.6 0.
##  7 monthJan               -12.9      0.266      -48.5 0.
##  8 monthJul                 2.94     0.285       10.3 6.35e- 25
##  9 monthJun                 8.23     0.276       29.8 1.12e-195
## 10 monthMar                -7.90     0.247      -32.0 6.11e-224
## # … with 13 more rows
```



```
## # A tibble: 23 x 5
##    term                  estimate std.error statistic  p.value
##    <chr>                    <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)           26.6     0.151        177.   0.
##  2 nearby_station_closed  0.0876  0.000402     218.   0.
##  3 year                  -0.0114  0.0000746   -153.   0.
##  4 monthAug               0.00314 0.000476       6.60 4.05e-11
##  5 monthDec              -0.235   0.000518    -454.   0.
##  6 monthFeb              -0.236   0.000531    -444.   0.
##  7 monthJan              -0.305   0.000560    -545.   0.
##  8 monthJul              -0.00746 0.000486     -15.3  4.10e-53
##  9 monthJun               0.0610  0.000463     132.   0.
## 10 monthMar              -0.142   0.000484    -294.   0.
## # … with 13 more rows
```

```
##                    term exponentiated_coefficient
## 2  nearby_station_closed                 1.0915374
## 3                  year                 0.9886301
## 4              monthAug                 1.0031466
## 5              monthDec                 0.7904381
## 6              monthFeb                 0.7897428
## 7              monthJan                 0.7371104
## 8              monthJul                 0.9925699
## 9              monthJun                 1.0629431
## 10             monthMar                 0.8674264
## 11             monthMay                 1.0074672
## 12             monthNov                 0.9771918
## 13             monthOct                 1.1155629
## 14             monthSep                 1.0655735
## 15       weekdayMonday                  0.9635457
## 16     weekdaySaturday                  0.8350816
## 17       weekdaySunday                  0.7787968
## 18     weekdayThursday                  1.0396190
## 19      weekdayTuesday                  1.0274262
## 20    weekdayWednesday                  1.0391208
## 21                PRCP                 0.6407795
## 22                SNWD                 0.9403296
## 23                TMAX                 1.0130047
```

Results

As expected, when a station is closed nearby stations see increases in ridership (of about 9 percent), controlling for year, month, day of the week, and weather.