BikeShare analysis

Chris Murray, Adam Spitzig, Tom Yedwab

April 2, 2016

```r
library(dplyr)
library(ggmap)
library(magrittr)
```

Load all the data from the CSVs

```r
stations_201402 = read.csv("../data/bikeshare_raw/201402_station_data.csv")
stations_201408 = read.csv("../data/bikeshare_raw/201408_station_data.csv")
stations_201508 = read.csv("../data/bikeshare_raw/201508_station_data.csv")

trips_201402 = read.csv("../data/bikeshare_raw/201402_trip_data.csv")
trips_201408 = read.csv("../data/bikeshare_raw/201408_trip_data.csv")
trips_201508 = read.csv("../data/bikeshare_raw/201508_trip_data.csv")

weather_201402 = read.csv("../data/bikeshare_raw/201402_weather_data.csv")
weather_201408 = read.csv("../data/bikeshare_raw/201408_weather_data.csv")
weather_201508 = read.csv("../data/bikeshare_raw/201508_weather_data.csv")

status_201402 = read.csv("../data/bikeshare_raw/201402_status_data.csv")
status_201408 = read.csv("../data/bikeshare_raw/201408_status_data.csv")
status_201508 = read.csv("../data/bikeshare_raw/201508_status_data.csv")
```

Concatenate the partitions of the trips dataset & merge to make one big dataset

```r
# Rename some columns so they're consistent
colnames(trips_201402)[10] = "Subscriber.Type"

# Concatenate all three time ranges
trips = bind_rows(trips_201402, trips_201408, trips_201508)

# Rename some columns so they're consistent
colnames(weather_201402) = colnames(weather_201408)

# Concatenate all three time ranges
weather = bind_rows(weather_201402, weather_201408, weather_201508)
colnames(weather)[1] = "Date"
weather$Date = as.Date(weather$Date, format="%m/%d/%Y")

# Clean up zip code
colnames(trips)[11] = "Zip"
trips$Zip = as.numeric(trips$Zip)

# Parse out just the date part of the start date column
```

```
trips$Date = as.Date(trips$Start.Date, format="%m/%d/%Y")

# Join on zip code
trips_and_weather = left_join(trips, weather, by=c("Zip", "Date"))
head(trips_and_weather[c("Date", "Start.Station","Zip","Mean.TemperatureF")], 40)
```

```
## Source: local data frame [40 x 4]
##
##          Date            Start.Station   Zip Mean.TemperatureF
## 1  2013-08-29 South Van Ness at Market 94127                NA
## 2  2013-08-29       San Jose City Hall 95138                NA
## 3  2013-08-29  Mountain View City Hall 97214                NA
## 4  2013-08-29       San Jose City Hall 95060                NA
## 5  2013-08-29 South Van Ness at Market 94103                NA
## 6  2013-08-29      Golden Gate at Polk 94109                NA
## 7  2013-08-29   Santa Clara at Almaden 95112                NA
## 8  2013-08-29      San Salvador at 1st 95112                NA
## 9  2013-08-29 South Van Ness at Market 94103                NA
## 10 2013-08-29       San Jose City Hall 95060                NA
## ..        ...                      ...   ...               ...
```

Unfortunately, we discover here that while the `trips` dataset contains thousands of unique zip codes (many of them clearly invalid) we only have weather data for a handful of zip codes. So joining on zip code is not going to work, and since we do not have zip codes for the stations we cannot join there either.

This leads us to prefer joining with an external dataset from forecast.io that we can query for the specific locations and dates we need. For that we'll need a set of coordinates to look up weather data for.

Figure out where all the stations are located

We use the "landmark" column of the stations to group them into 5 distinct areas, since we cannot fetch weather data for each station individually.

Landmark locations are calculated as the arithmetic mean of the station positions associated with the landmark.

```
stations_201402$name = as.character(stations_201402$name)
stations_201408$name = as.character(stations_201408$name)
stations_201508$name = as.character(stations_201508$name)

stations_201402$installation = as.Date(stations_201402$installation, format="%m/%d/%Y")
stations_201408$installation = as.Date(stations_201408$installation, format="%m/%d/%Y")
stations_201508$installation = as.Date(stations_201508$installation, format="%m/%d/%Y")

stations = union(union(stations_201402, stations_201408), stations_201508)

landmark_centers = stations[c('lat','long','landmark')] %>%
  group_by(landmark) %>%
  summarize(lat=mean(lat), long=mean(long))

landmark_centers
```

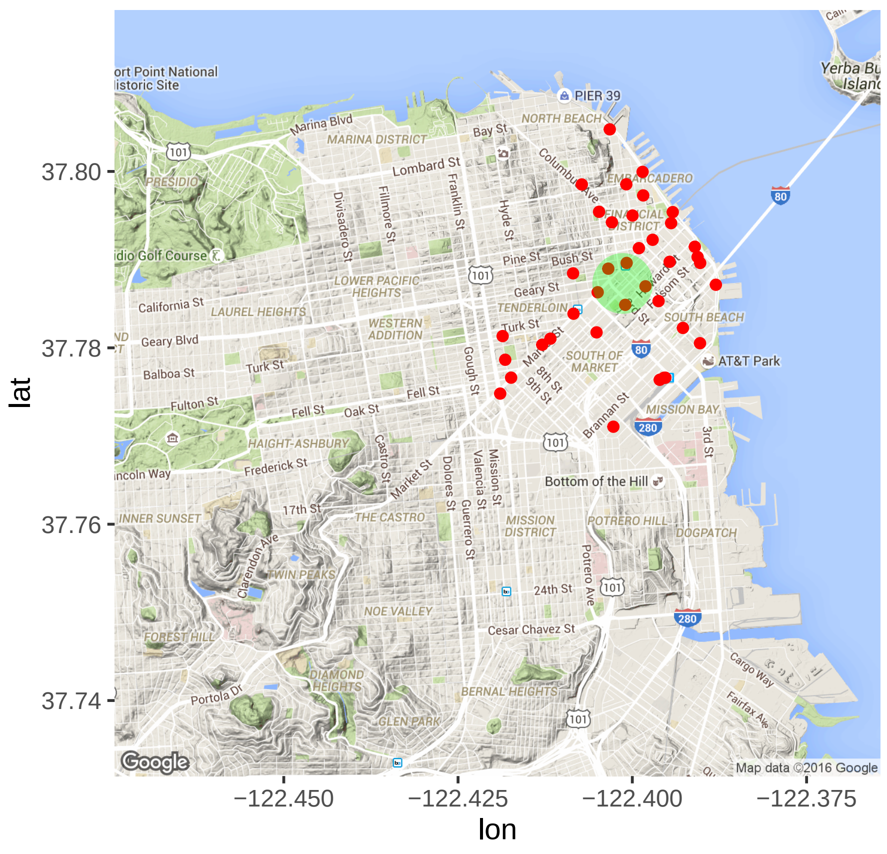
```
## Source: local data frame [5 x 3]
##
##         landmark      lat      long
## 1 Mountain View 37.39535 -122.0890
## 2     Palo Alto 37.43837 -122.1536
## 3  Redwood City 37.48658 -122.2262
## 4 San Francisco 37.78730 -122.4014
## 5      San Jose 37.33737 -121.8924
```

Plot the landmark locations on a few maps

```r
map1 = get_map(location="San Francisco", zoom=13)
```

```
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=San+Francisco&zoom=13&size=640x64
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=San%20Francisco&sens
```

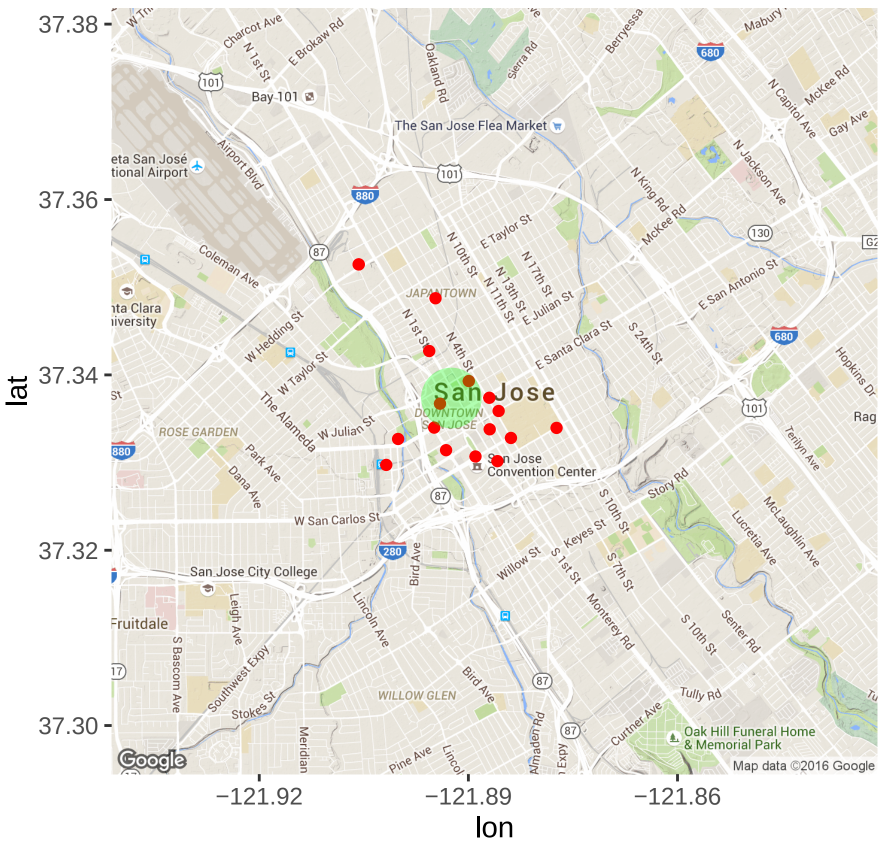
```r
ggmap(map1) +
  geom_point(aes(x=long, y=lat), data=stations, color='red', alpha=1.0) +
  geom_point(aes(x=long, y=lat), data=landmark_centers, color='green', alpha=0.3, size=10)
```



```r
map2 = get_map(location="San Jose", zoom=13)
```

```
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=San+Jose&zoom=13&size=640x640&sca
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=San%20Jose&sensor=fal
```

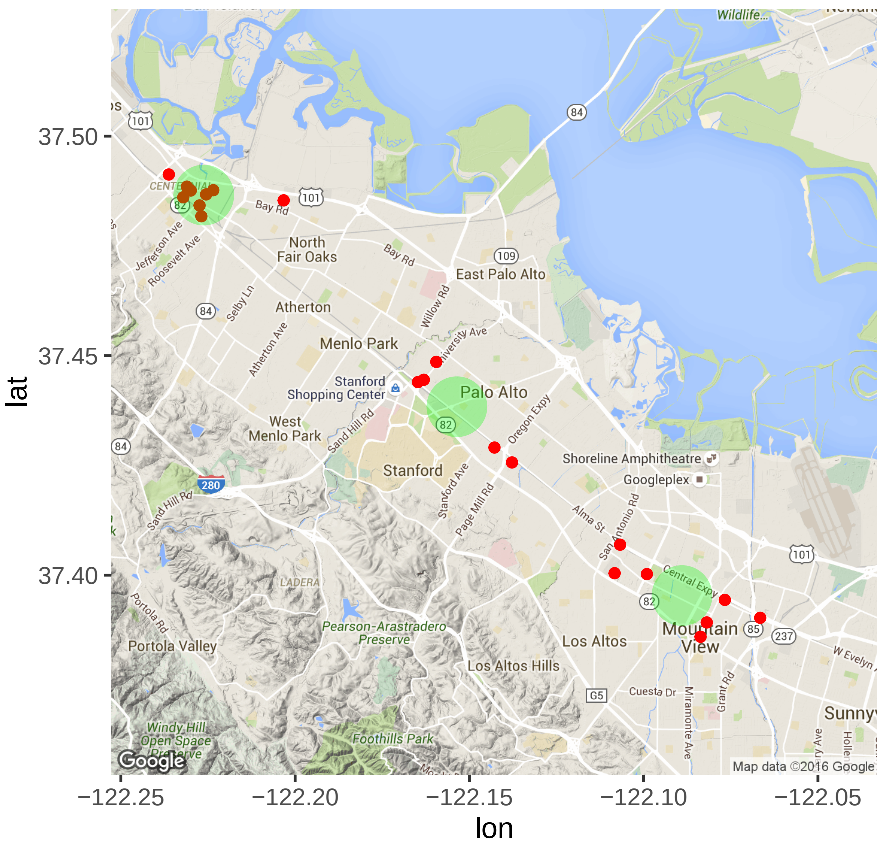
```
ggmap(map2) +
  geom_point(aes(x=long, y=lat), data=stations, color='red', alpha=1.0) +
  geom_point(aes(x=long, y=lat), data=landmark_centers, color='green', alpha=0.3, size=10)
```



```
map3 = get_map(location="Palo Alto", zoom=12)
```

```
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=Palo+Alto&zoom=12&size=640x640&s
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Palo%20Alto&sensor=f
```

```
ggmap(map3) +
  geom_point(aes(x=long, y=lat), data=stations, color='red', alpha=1.0) +
  geom_point(aes(x=long, y=lat), data=landmark_centers, color='green', alpha=0.3, size=10)
```



Concatenate the partitions of the status dataset & merge to make one big dataset

```r
cols = c("station_id","name","dockcount","landmark")

status_201402$Date = as.Date(status_201402$time, format="%Y/%m/%d")
status_joined_201402 = left_join(
  status_201402, stations_201402[cols], by="station_id")

status_201408$Date = as.Date(status_201408$time, format="%Y-%m-%d")
status_joined_201408 = left_join(
  status_201408, stations_201408[cols], by="station_id")

status_201508$Date = as.Date(status_201508$time, format="%Y-%m-%d")
status_joined_201508 = left_join(
  status_201508, stations_201508[cols], by="station_id")

# This thing is pretty big
status_joined = rbind(status_joined_201402, status_joined_201408,
                      status_joined_201508)
```

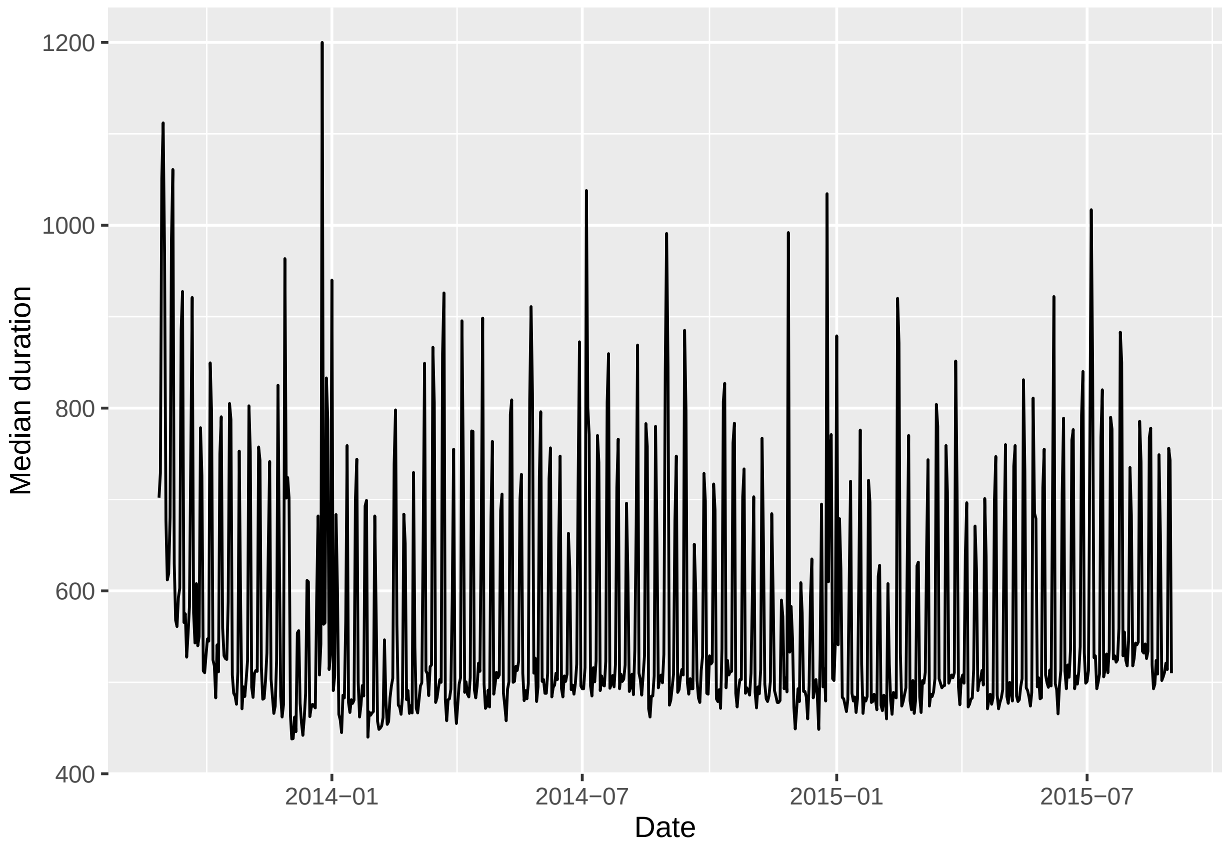
Look at trends in the datasets over time

```r
# Aggregate median duration and count by day
duration_by_day = trips %>%
  group_by(Date) %>%
  summarize(Duration=median(as.numeric(Duration)), Count=n())

# Aggregate bike and dock availability statistics by day
availability_by_day = status_joined %>%
  group_by(Date, name) %>%
  summarize(Bikes=median(bikes_available),
            Docks=median(docks_available),
            NoBikes=mean(bikes_available == 0),
            NoDocks=mean(docks_available == 0),
            Samples=n())
```

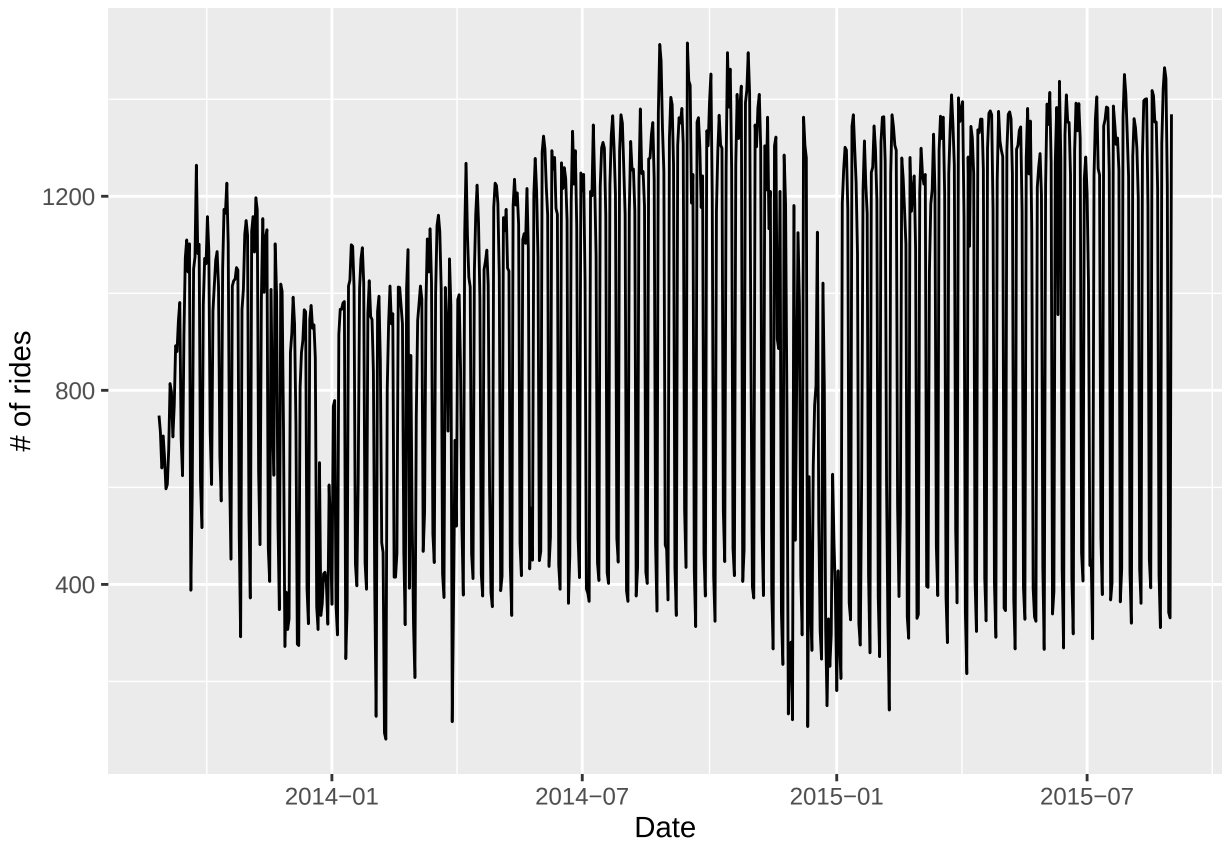
We can see a fairly stable trendline for median duration, with upward spikes we would guess correspond to weekends:

```r
ggplot(duration_by_day, aes(x=Date, y=Duration)) +
  labs(x="Date", y="Median duration") +
  geom_line()
```



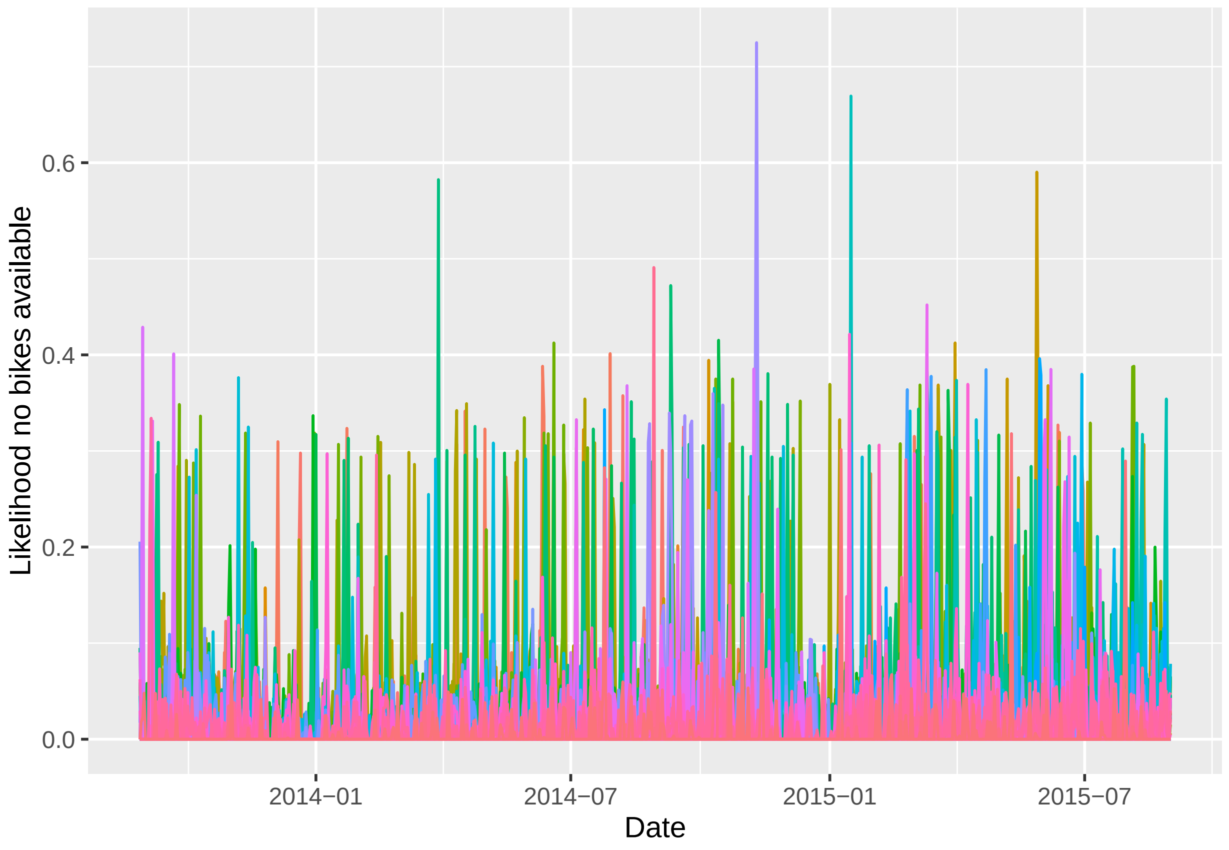
The number of counts might show a slight upward trend over the years, but it also be stabilizing. This graph appears to show downward spikes on weekends:

```r
ggplot(duration_by_day, aes(x=Date, y=Count)) +
  labs(x="Date", y="# of rides") +
  geom_line()
```



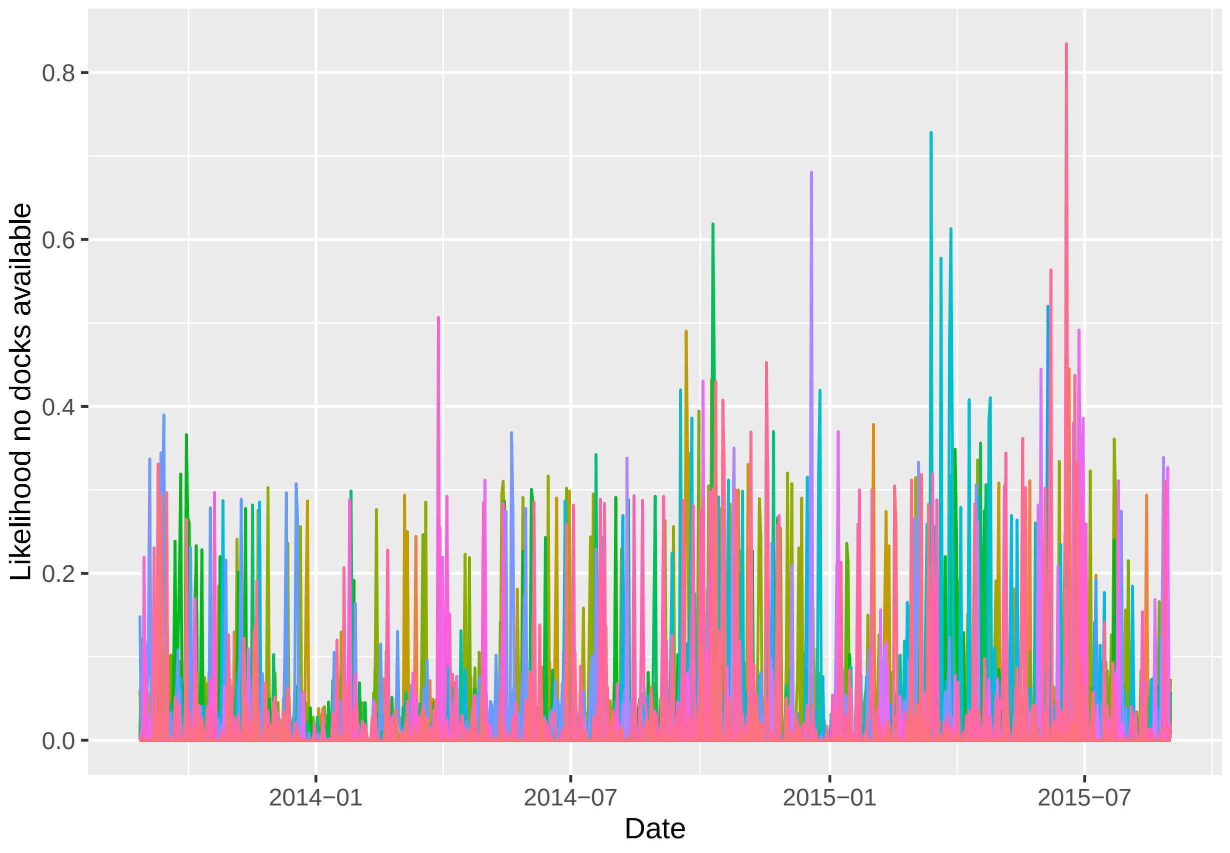
The likelihood of getting to a station and finding no bikes available is very spiky, which is promising since we want to see a wide variety of values for this if we're trying to predict it:

```r
ggplot(availability_by_day, aes(x=Date, y=NoBikes, color=name)) +
  labs(x="Date", y="Likelihood no bikes available") +
  guides(color=FALSE) +
  geom_line()
```



The same is true for the opposite situation where no docks are available:

```r
ggplot(availability_by_day, aes(x=Date, y=NoDocks, color=name)) +
  labs(x="Date", y="Likelihood no docks available") +
  guides(color=FALSE) +
  geom_line()
```


Likelihood no docks available
0.8
0.6
0.4
0.2
0.0
2014−01
2014−07
2015−01
2015−07
Date