Analysis of a public bike share network using Python

J. Michael Shockley

What makes a bike share network “tick”?

“Docked” bike share systems consist of a network of docking stations.

- Trip Start: a user takes a bike from a station
- Trip End: a user puts a bike into a station

The popularity of a bike share network at any given moment is affected by a multitude of factors, e.g.:

- Weather
- Time of day

https://en.wikipedia.org/wiki/Capital_Bikeshare#/media/File:Capital_Bikeshare_station_outside_Eastern_Market_Metro.jpg

What makes a bike share network "tick"?

Inevitably, some stations will be popular, leading to two negative situations:

- All the bikes are taken
- If all the docks are full: nowhere to park a bike
- Both of these scenarios must be avoided if possible.
- The solution: re-distribution!

Bike redistribution is expensive:

- Labor costs
- Transportation costs (trucks, gas, maintenance)
- Logistics!

To approach this logistics problem, the first step is to consider what factors influence the popularity of a given station:

Geographic location?
Time of day?
Day of the week?
Weather?

Let's get some data!

What sort of data is available?

Data from a San Francisco Bay Area public bike share
pilot project, 2014-09-01 to 2014-08-31

- station_data.csv
  - Station ID, latitude/longitude, city name
- trip_data.csv
  - Raw log of all trips during a 1 year time period
  - Start date/time and station ID
  - End date/time and station ID
- weather_data.csv
  - 22 columns of weather data (temperature, wind, precipitation, etc.)
  - 1 entry per day for the 5 zip codes covered by the bike share network

Connecting the data sets

Trip Data
(Start/End Station, Start/End Date/Time)

Station Data
(ID, city, lat/long)

Weather Data
(Date, Zip, Weather observations)

Station ID	Date/Hour	Net bikes per hour	Station Location	Weather Observations

Question relevant for logistics:
At a given station, in a given hour, at a given set of conditions:
What change in the number of bikes was observed?

Making bike share data useful for logistics purposes

1 Extract raw data

2 Create a single dataset of bikes entering and leaving each station, per hour

3 Incorporate station information and weather data into the hourly trip data set

4 Analyze relationships in the data

5 Summary and recommendations for future work

1 Extract raw data: Trip Data

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

#Trip data imported as dataframe TD
TD=pd.read_csv(
    "trip_data.csv",
    header=0,
    skip_blank_lines=False,
    parse_dates=['Start Date','End Date'],
    date_parser=lambda d: pd.to_datetime(
        d, format="%d/%m/%Y %H:%M", errors="coerce"
    )
)
TD.head()
```

Out[1]:

	Trip ID	Start Date	Start Station	End Date	End Station	Subscriber Type
0	913460	2015-08-31 23:26:00	50	2015-08-31 23:39:00	70	Subscriber
1	913459	2015-08-31 23:11:00	31	2015-08-31 23:28:00	27	Subscriber
2	913455	2015-08-31 23:13:00	47	2015-08-31 23:18:00	64	Subscriber
3	913454	2015-08-31 23:10:00	10	2015-08-31 23:17:00	8	Subscriber
4	913453	2015-08-31 23:09:00	51	2015-08-31 23:22:00	60	Customer

1 Extract raw data: weather, station data

Max TemperatureF
Mean TemperatureF
Min TemperatureF
Max Dew PointF
MeanDew PointF
Min DewpointF
Max Humidity
Mean Humidity
Min Humidity
Max Sea Level PressureIn
Mean Sea Level PressureIn
Min Sea Level PressureIn
Max VisibilityMiles
Mean VisibilityMiles
Min VisibilityMiles
Max Wind SpeedMPH
Mean Wind SpeedMPH
Max Gust SpeedMPH
PrecipitationIn
CloudCover
Events

Weather

	Date	Max TemperatureF	Mean TemperatureF	Min TemperatureF	Max Dew PointF	MeanDew PointF	Min DewpointF	Max Humidity	H
0	2014-09-01	83.0	70.0	57.0	58.0	56.0	52.0	86.0	
1	2014-09-02	72.0	66.0	60.0	58.0	57.0	55.0	84.0	
2	2014-09-03	76.0	69.0	61.0	57.0	56.0	55.0	84.0	
3	2014-09-04	74.0	68.0	61.0	57.0	57.0	56.0	84.0	
4	2014-09-05	72.0	66.0	60.0	57.0	56.0	54.0	84.0	

5 rows × 24 columns

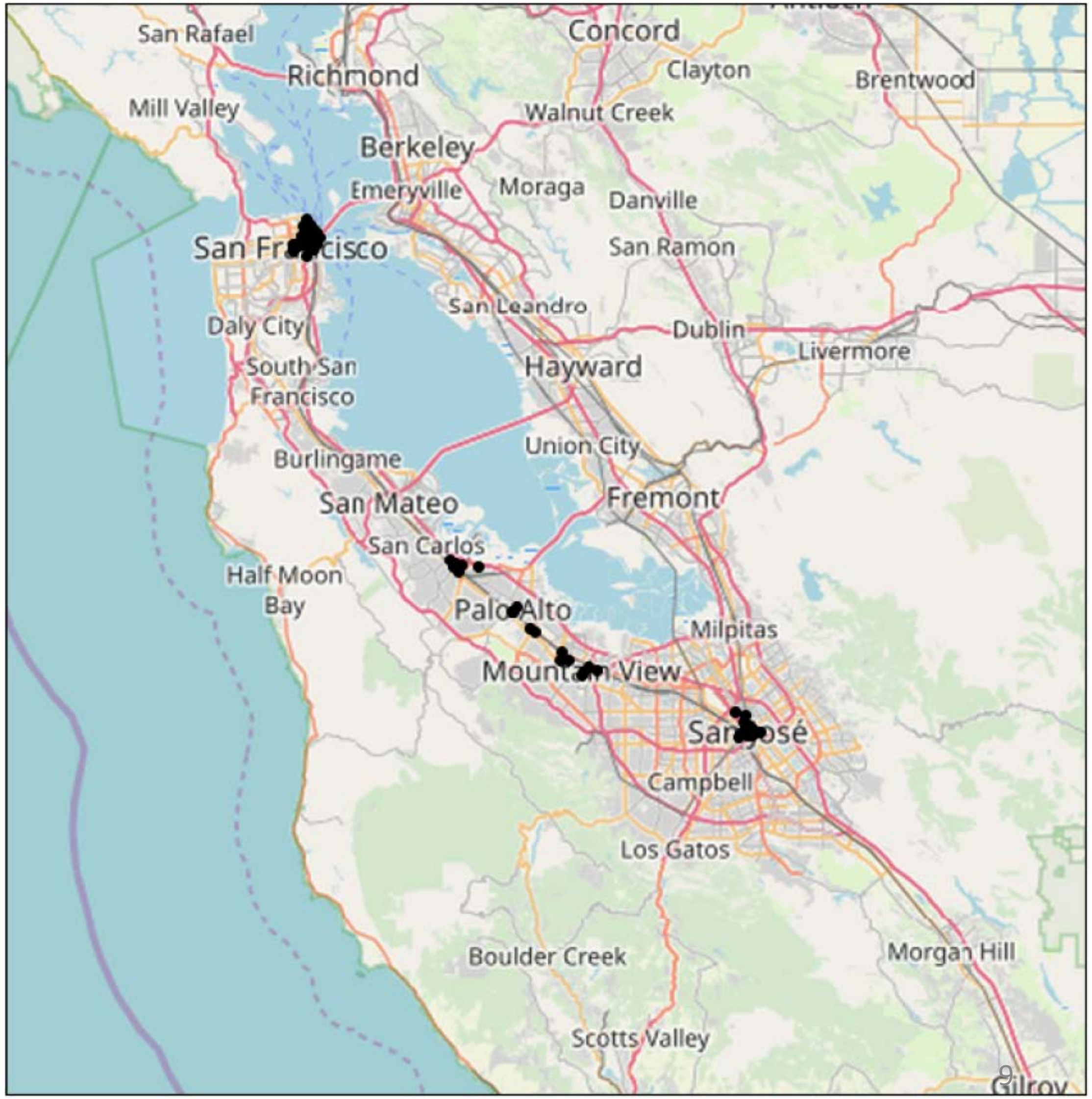
Stations

	Id	Name	Lat	Long	Dock Count	City
0	2	San Jose Diridon Caltrain Station	37.329732	-121.901782	27	San Jose
1	3	San Jose Civic Center	37.330698	-121.888979	15	San Jose
2	4	Santa Clara at Almaden	37.333988	-121.894902	11	San Jose
3	5	Adobe on Almaden	37.331415	-121.893200	19	San Jose
4	6	San Pedro Square	37.336721	-121.894074	15	San Jose

Visualizing Station Data

```
import cartopy.crs as ccrs
from cartopy.io.img_tiles import OSM

# Initialize figure and specify Open Street Map as map tile source
plt.figure(figsize=(12, 12))
imagery = OSM()
ax = plt.axes(projection=imagery.crs)
# Define axes in terms of longitude and latitude and add map tile imager
y
ax.set_extent([-122.75, -121.5,37, 38])
ax.add_image(imagery, 9,interpolation='spline36')
# Add scatter plot of station locations based longitude and latitude
plt.scatter(SD['Long'],SD['Lat'],transform=ccrs.Geodetic(),color='k')
plt.show()
```



Stations are in multiple cities around the San Francisco Bay area

2 Create a single dataset of bikes entering and leaving each station, per hour

In [5]:
```
#Use .dt.floor method to truncate trip data to the hour.
TD['Start Date Truncated']=TD['Start Date'].dt.floor('h')
TD['End Date Truncated']=TD['End Date'].dt.floor('h')
TD.head()
```

Out[5]:

	Trip ID	Start Date	Start Station	End Date	End Station	Subscriber Type	Start Date Truncated	End Date Truncated
0	913460	2015-08-31 23:26:00	50	2015-08-31 23:39:00	70	Subscriber	2015-08-31 23:00:00	2015-08-31 23:00:00
1	913459	2015-08-31 23:11:00	31	2015-08-31 23:28:00	27	Subscriber	2015-08-31 23:00:00	2015-08-31 23:00:00
2	913455	2015-08-31 23:13:00	47	2015-08-31 23:18:00	64	Subscriber	2015-08-31 23:00:00	2015-08-31 23:00:00
3	913454	2015-08-31 23:10:00	10	2015-08-31 23:17:00	8	Subscriber	2015-08-31 23:00:00	2015-08-31 23:00:00
4	913453	2015-08-31 23:09:00	51	2015-08-31 23:22:00	60	Customer	2015-08-31 23:00:00	2015-08-31 23:00:00

2 Create a single dataset of bikes entering and leaving each station, per hour

```
TD_bystation_byenddate = (pd.DataFrame(TD.groupby(['End Station','End Da
te Truncated'])
                            ['End Date Truncated']
                            .count()
                            .fillna(0)))
TD_bystation_byenddate.columns=['End Count']
TD_bystation_byenddate.reset_index(inplace=True)
TD_bystation_byenddate.head()
```

	End Station	End Date Truncated	End Count
0	2	2014-09-01 14:00:00	1
1	2	2014-09-02 06:00:00	3
2	2	2014-09-02 07:00:00	6
3	2	2014-09-02 08:00:00	1
4	2	2014-09-02 09:00:00	2

Pandas methods:
.groupby() and .count()

2 Create a single dataset of bikes entering and leaving each station, per hour

- DFstation: list of each station ID
- DFtime: list of Date/Hour combinations between 2014-09-01 and 2015-08-31

Combine by Cartesian Product: every station at every Date/Hour combination

	Station	Date/Hour
0	2	2014-09-01 00:00:00
1	2	2014-09-01 01:00:00
2	2	2014-09-01 02:00:00
3	2	2014-09-01 03:00:00
4	2	2014-09-01 04:00:00

```
DFMerge=pd.MultiIndex.from_product([DFstation,DFtime],names=['Station',
'Date/Hour'])
#Convert the multi-level indices to column values
DFMerge=pd.DataFrame(index=DFMerge).reset_index()
DFMerge.head()
```

2 Create a single dataset of bikes entering and leaving each station, per hour

```
DFall=pd.merge(left=DFMerge, right=TD_bystation_bystartdate, how='left',
on=None, left_on=['Station','Date/Hour'], right_on=['Start Station','Sta
rt Date Truncated'])
```

	Station	Date/Hour	Start Count	End Count
0	2	2014-09-01 00:00:00	0.0	0.0
1	2	2014-09-01 01:00:00	0.0	0.0
2	2	2014-09-01 02:00:00	0.0	0.0
3	2	2014-09-01 03:00:00	0.0	0.0
4	2	2014-09-01 04:00:00	0.0	0.0

2 Create a single dataset of bikes entering and leaving each station, per hour

```
DFall['Net Bikes Per Hour']=DFall['End Count']-DFall['Start Count']
```

	Station	Date/Hour	Start Count	End Count	Net Bikes Per Hour
0	2	2014-09-01 00:00:00	0.0	0.0	0
1	2	2014-09-01 01:00:00	0.0	0.0	0
2	2	2014-09-01 02:00:00	0.0	0.0	0
3	2	2014-09-01 03:00:00	0.0	0.0	0
4	2	2014-09-01 04:00:00	0.0	0.0	0

3 Incorporate station information and weather data into the hourly trip data set

Merge hourly trip data and Station data using Station Id as a join key

```
DFall=pd.merge(left=DFall, right=SD, how='left', left_on=['Station'], ri
ght_on=['Id'])
#Remove extraneous columns
DFall=DFall.drop(columns=['Id','Dock Count'])
DFall.head()
```

	Station	Date/Hour	Start Count	End Count	Net Bikes Per Hour	Name	Lat	Long	City	Zip
0	2	2014-09-01 00:00:00	0.0	0.0	0	San Jose Diridon Caltrain Station	37.329732	-121.901782	San Jose	95113
1	2	2014-09-01 01:00:00	0.0	0.0	0	San Jose Diridon Caltrain Station	37.329732	-121.901782	San Jose	95113
2	2	2014-09-01 02:00:00	0.0	0.0	0	San Jose Diridon Caltrain Station	37.329732	-121.901782	San Jose	95113
3	2	2014-09-01 03:00:00	0.0	0.0	0	San Jose Diridon Caltrain Station	37.329732	-121.901782	San Jose	95113
4	2	2014-09-01 04:00:00	0.0	0.0	0	San Jose Diridon Caltrain Station	37.329732	-121.901782	San Jose	95113

3 Incorporate station information and weather data into the hourly trip data set

```
DFfinal=pd.merge(left=DFall, right=WD, how='left', on=['Zip','Date'])
```

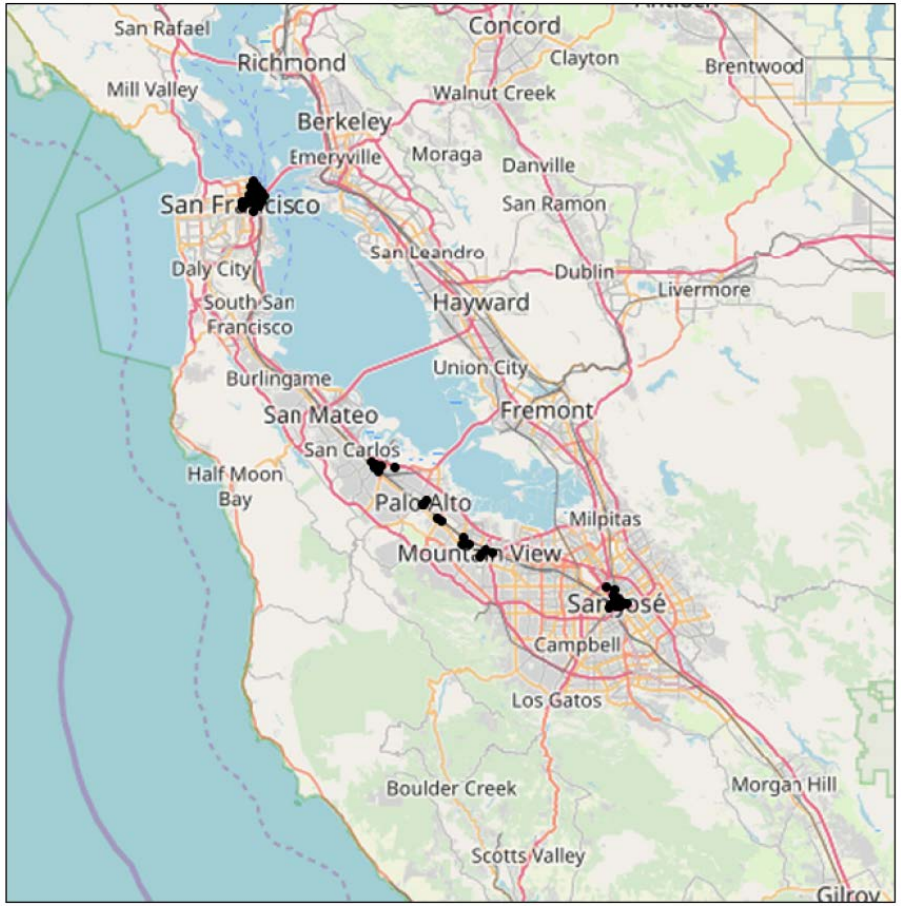
Merge daily weather using date and zip as join keys

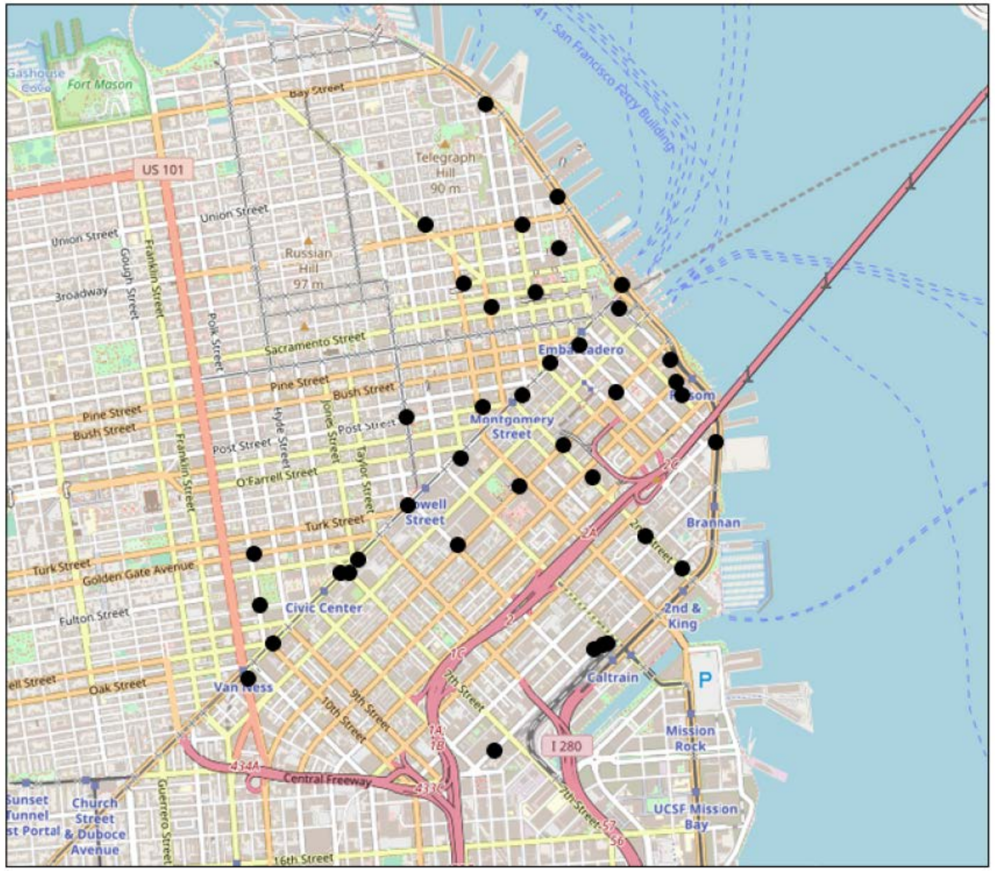
	Station	Date/Hour	Start Count	End Count	Net Bikes Per Hour	Name	Lat	Long	City	Zip	...	V
0	2	2014-09-01 00:00:00	0.0	0.0	0	San Jose Diridon Caltrain Station	37.329732	-121.901782	San Jose	95113	...	
1	2	2014-09-01 01:00:00	0.0	0.0	0	San Jose Diridon Caltrain Station	37.329732	-121.901782	San Jose	95113	...	
2	2	2014-09-01 02:00:00	0.0	0.0	0	San Jose Diridon Caltrain Station	37.329732	-121.901782	San Jose	95113	...	
3	2	2014-09-01 03:00:00	0.0	0.0	0	San Jose Diridon Caltrain Station	37.329732	-121.901782	San Jose	95113	...	
4	2	2014-09-01 04:00:00	0.0	0.0	0	San Jose Diridon Caltrain Station	37.329732	-121.901782	San Jose	95113	...	

5 rows × 34 columns

Max TemperatureF
Mean TemperatureF
Min TemperatureF
Max Dew PointF
MeanDew PointF
Min DewpointF
Max Humidity
Mean Humidity
Min Humidity
Max Sea Level PressureIn
Mean Sea Level PressureIn
Min Sea Level PressureIn
Max VisibilityMiles
Mean VisibilityMiles
Min VisibilityMiles
Max Wind SpeedMPH
Mean Wind SpeedMPH
Max Gust SpeedMPH
PrecipitationIn
CloudCover
Events

4 Analyze relationships in the data





4 Analyze relationships in the data

Focus on 2 stations:

Station 70: close proximity to Caltrain commuter rail station

Station 65:several blocks removed from public transport

4 Analyze relationships in the data

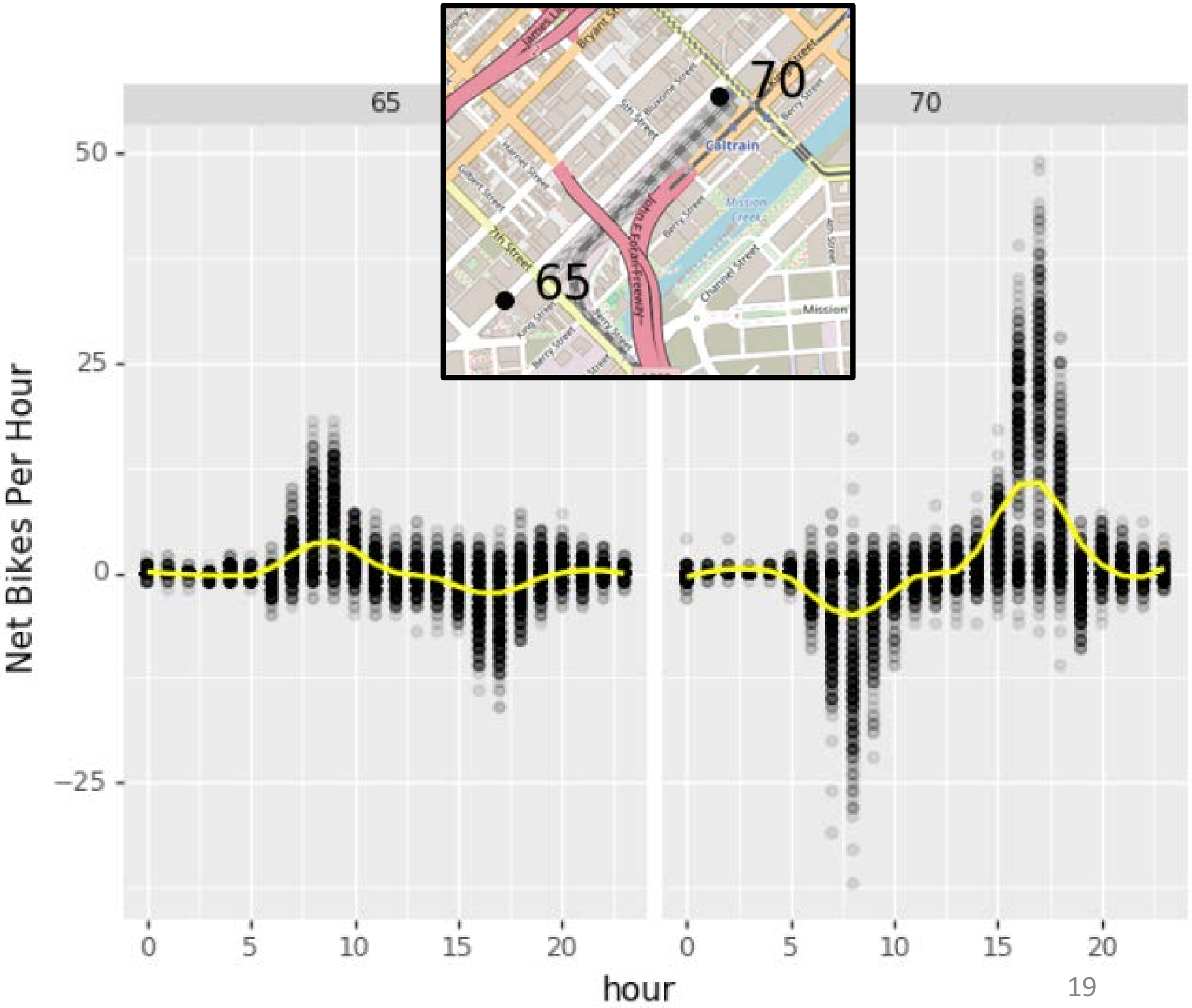
Scatter plot: Net bikes per hour vs. hour of the day

Station 70:

- Drop in bikes in the morning: commuters!
- Surplus in bikes the afternoon: commuters!

Station 65:

- Opposite behavior by time of day: commuters!

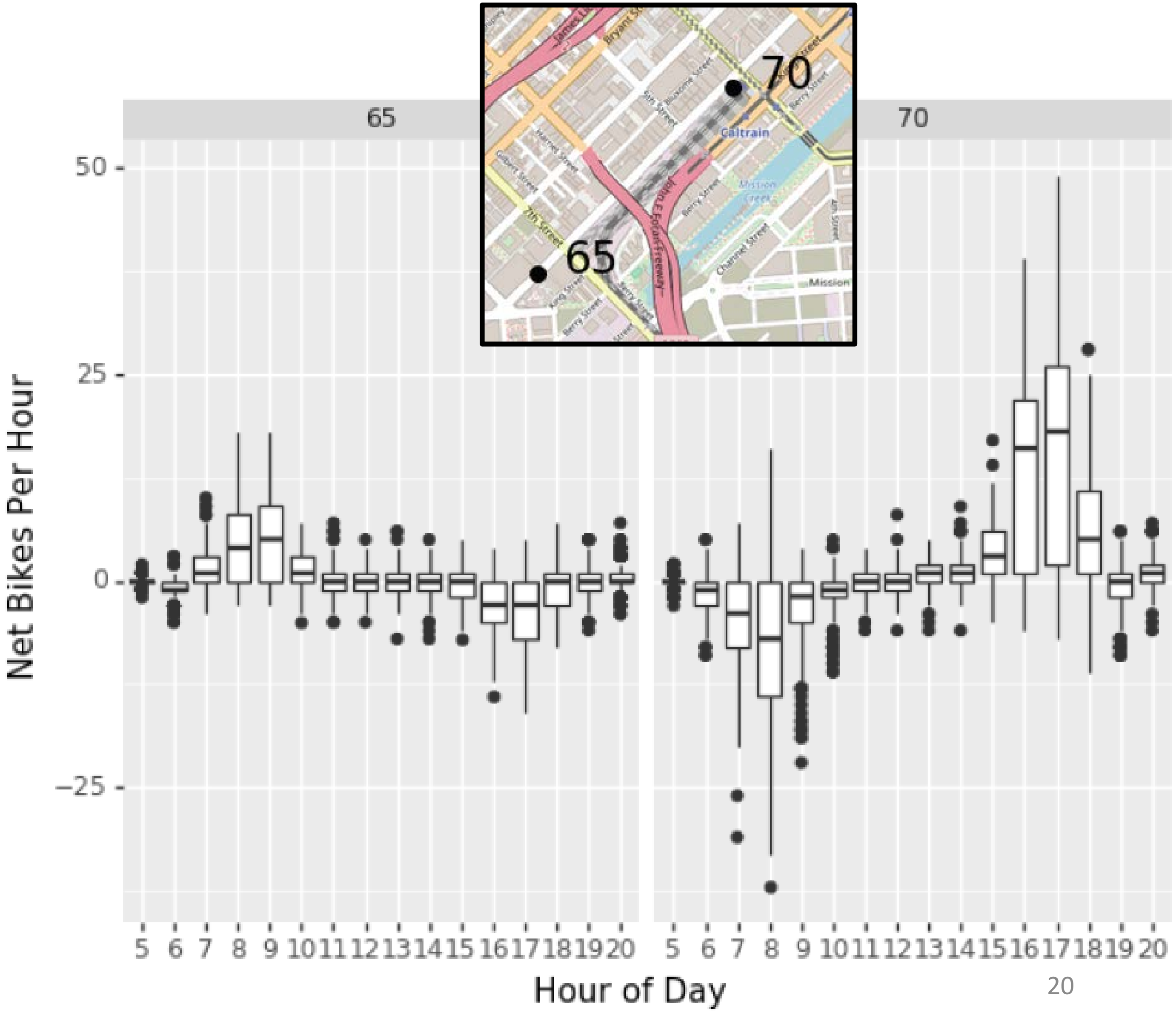


4 Analyze relationships in the data

Box plot: Net bikes per hour vs. hour of the day

More quantitative information than a scatter plot

The same relationship exists!

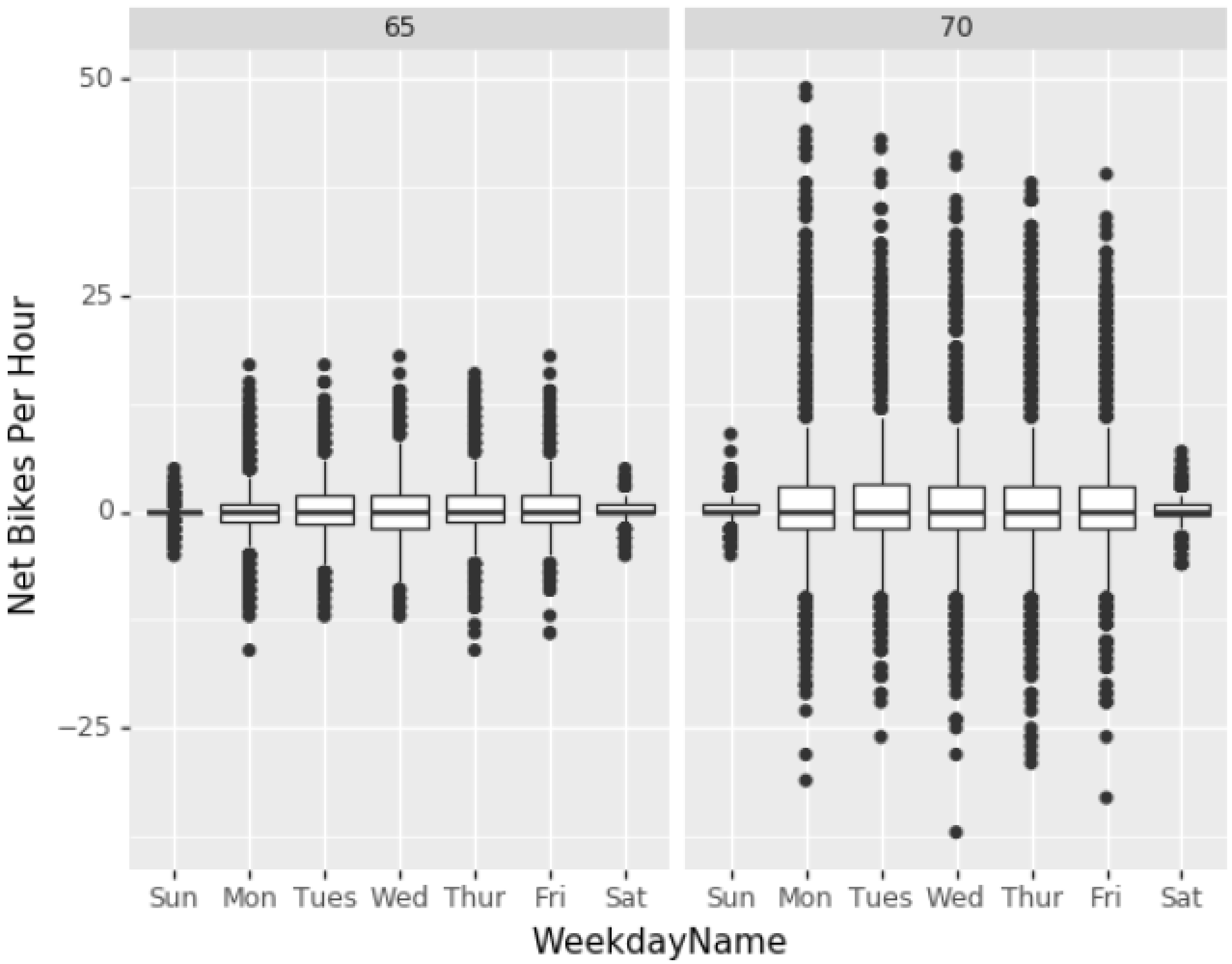


4 Analyze relationships in the data

Box plot: Net bikes per hour vs. day of the week

The system is more popular on weekdays than on weekends:

Commuters!



4 Analyze relationships in the data

Weather data:

- Certain relationships in daily weather data may correlate with one another

- To simplify the analysis, let's first examine which columns may introduce redundancy

	Date	Max TemperatureF	Mean TemperatureF	Min TemperatureF	Max Dew PointF	MeanDew PointF	Min DewpointF	Max Humidity	H
0	2014-09-01	83.0	70.0	57.0	58.0	56.0	52.0	86.0	
1	2014-09-02	72.0	66.0	60.0	58.0	57.0	55.0	84.0	
2	2014-09-03	76.0	69.0	61.0	57.0	56.0	55.0	84.0	
3	2014-09-04	74.0	68.0	61.0	57.0	57.0	56.0	84.0	
4	2014-09-05	72.0	66.0	60.0	57.0	56.0	54.0	84.0	

5 rows × 24 columns

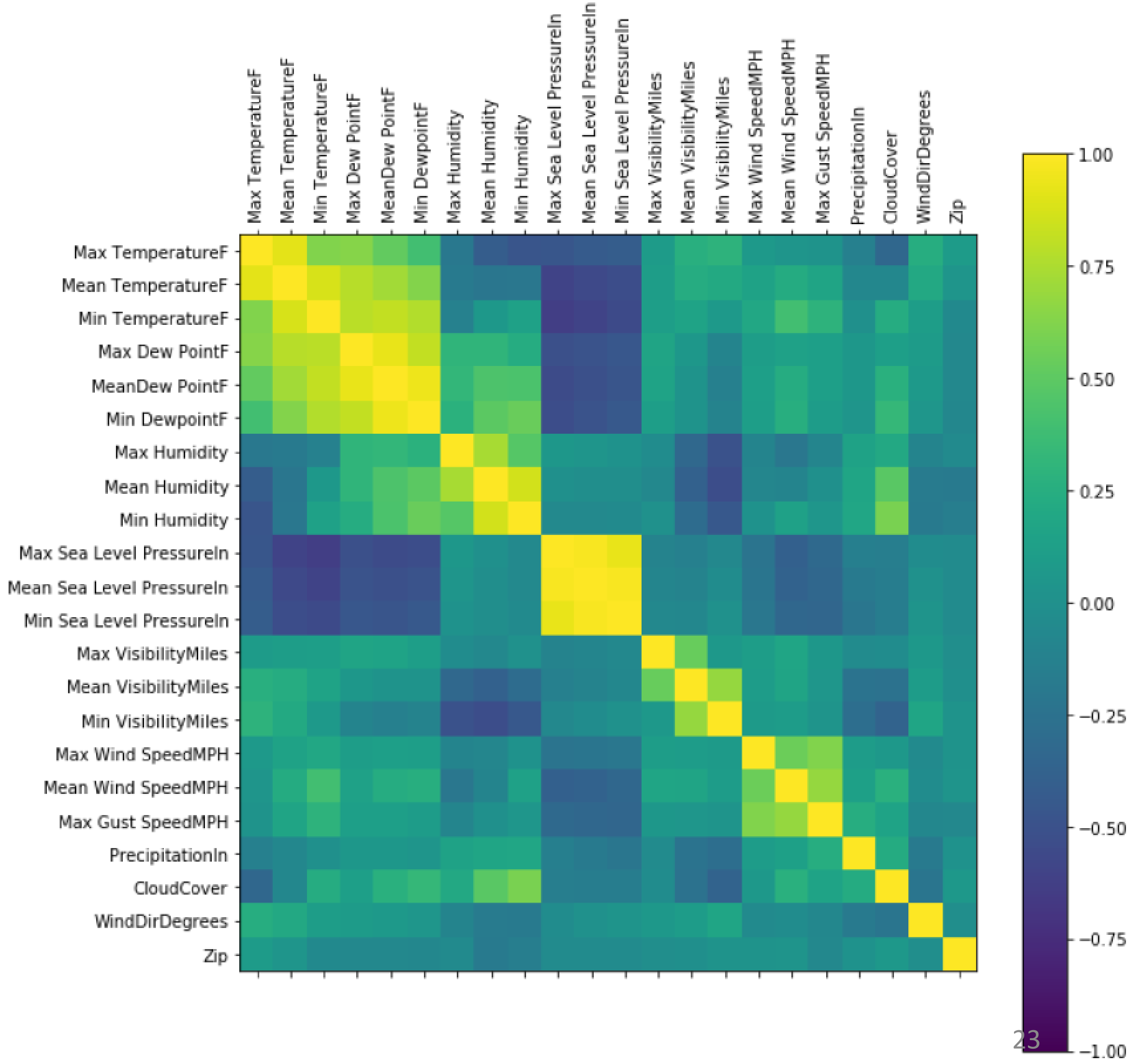
4 Analyze relationships in the data

Weather data:
correlation matrix

Several clusters in the correlation matrix exist.

Max/mean/min sets tend to correlate:

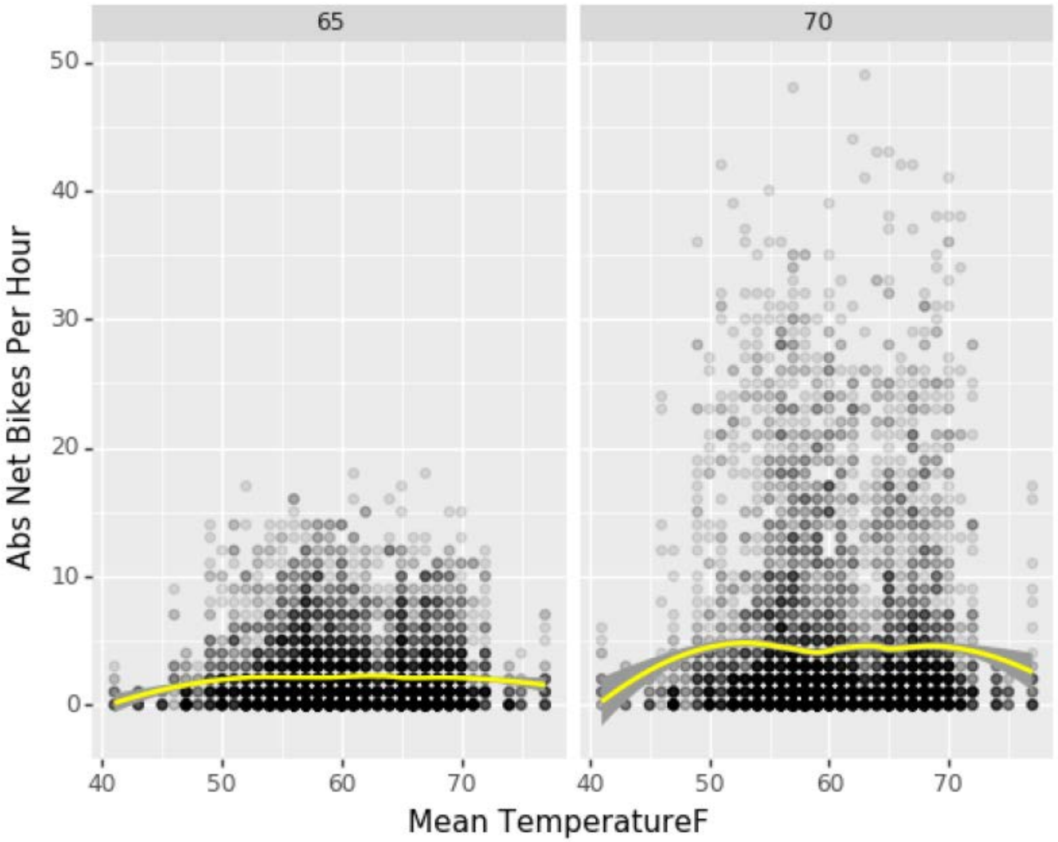
- Temperature
- dew point
- humidity
- pressure
- wind speed



4 Analyze relationships in the data

Most rides take place at temperatures between 50 and 70°F

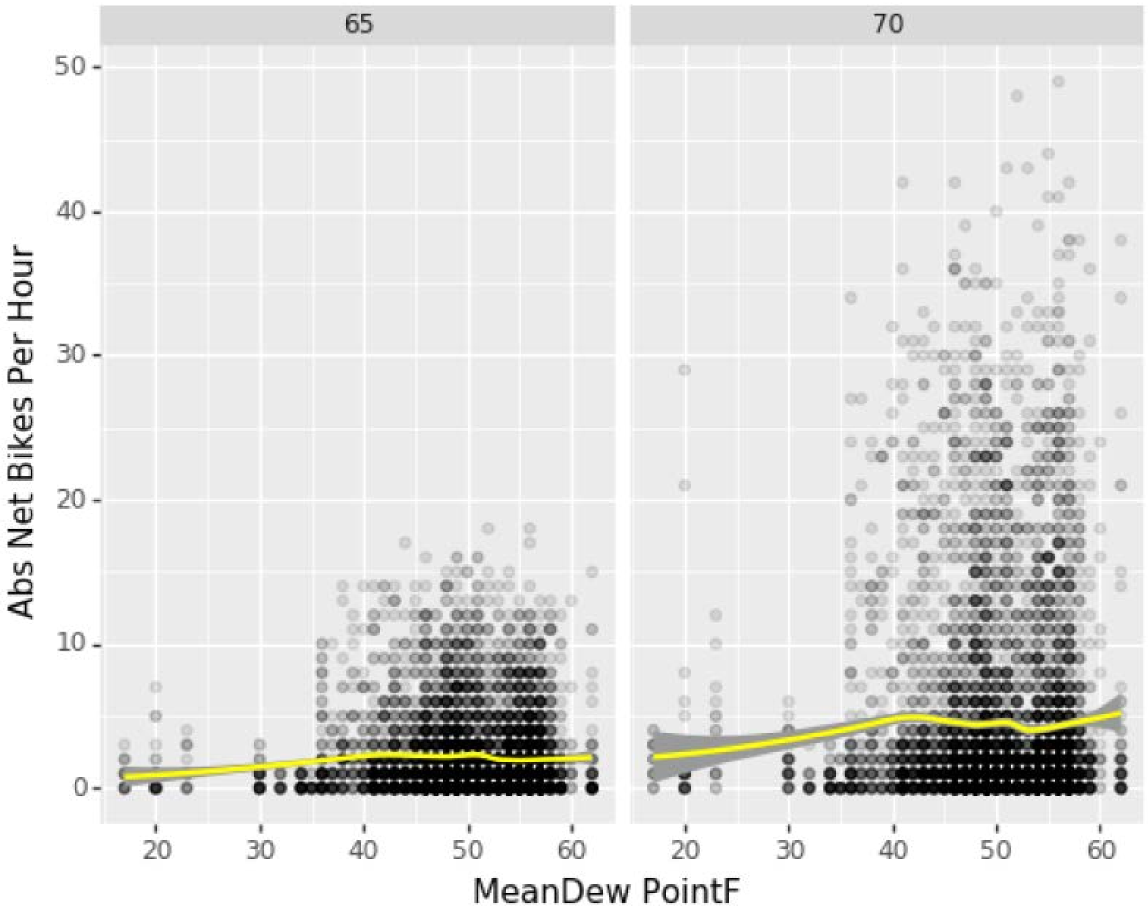
This may be due to comfort or due to stable temperature of San Francisco, which has few data points outside this range



4 Analyze relationships in the data

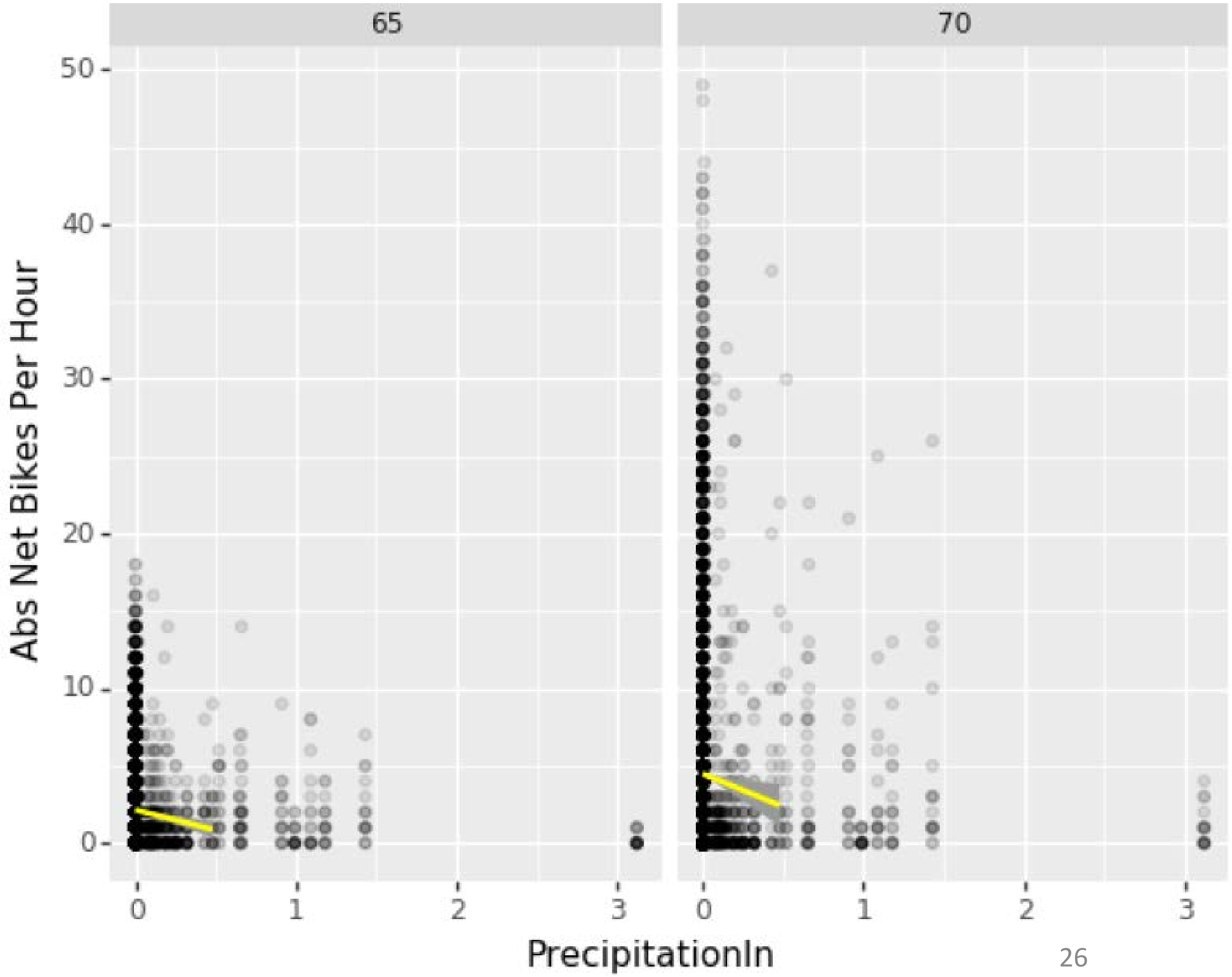
Dew point also does not predict ride numbers very well, though more outliers exist at higher dew points for Station 70

Clearly the data would look very different in Washington, DC!



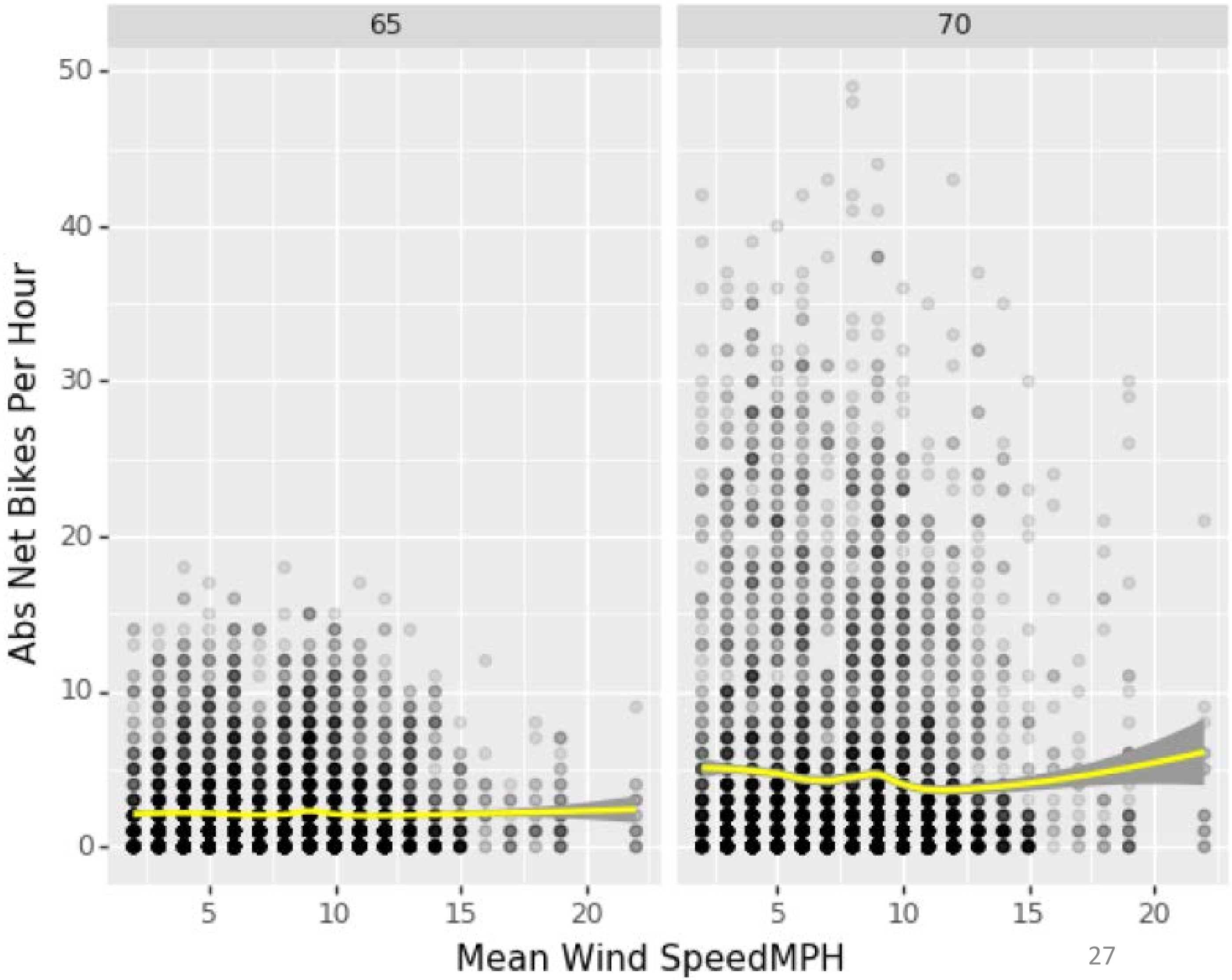
4 Analyze relationships in the data

Precipitation leads to decreased bike share popularity at both stations



4 Analyze relationships in the data

Wind speed does not appear to strongly affect bike share popularity



5 Summary and recommendations for future work

The present project has revealed relationships within a San Francisco bay area bike share network during the period of 09/2014 to 08/2015.

The geographic location of stations, time of day, day of the week, and weather parameters all affect the ridership numbers to greater or lesser extents.

To fully leverage this data for the logistics problem stated in the introduction, a regression model may be used to fit and predict the Net Bikes Per Hour data to the many features that have been shown to influence it. This is left as a consideration for future work.