Data Bootcamp Final Project - Weather and Transportation in New York City ¶

Author: Aristo P. Joesoef

e-mail: aristopj@gmail.com

Data Report

This project will explore the effects of weather (and other parameters) on MTA subway ridership. I posit that weather (whether it is sunny, raining, snowing etc.) will affect people's willingness to take the subway instead of walking or using other means of transportation (e.g. CitiBike, and other available non-underground transport data).

The first step is to construct a cross-sectional data to compare the number of subway users in different weather conditions, and compare these to the number of CitiBike users (and other available non-underground transport). The end goal is to find the correlation, if any, between weather statistics (temperature, rain/snow, etc.) and MTA subway transport use.

The following data are used:

- Citibike data, specifically the daily usage (usages of Citibike in 24-hour intervals)
- MTA subway turnstile data, which gives us the daily subway ridership in New York City
- Weather data from the National Centers for Environmental Information, which is open to public and provides daily weather data including, but not limited to, maximum and minimum temperature, snowfall, and rainfall.

Packages

The most important packages are pandas and matplotlib, both of which are used to manipulate the data so that it is more easily understandable and to find connections or correlations between the available data sets. The numpy package assists the manipulation of the data by allowing for more complex numerical functions, while the datetime package allows us to manipulate date-time objects.

In [494]:
```
import pandas as pd #an important tool to create data frames and start t
he analysis

import matplotlib.pyplot as plt #a tool to visualize the available data

import numpy as np #this allows us to do more mathematical functions in
 our analysis

from datetime import date, datetime, timedelta
#this allows us to manipulate and create date-time objects
```

Cleaning the Data

In [66]:
```
mta_url = "http://web.mta.info/developers/data/nyct/turnstile/turnstile
_"
date = ["170107","170114","170121","170128",
        "170204","170211","170218","170225",
        "170304","170311","170318","170325",
        "170401","170408","170415","170422","170429",
        "170506","170513","170520","170527",
        "170603","170610","170617","170624",
        "170701","170708","170715","170722","170729",
        "170805","170812","170819","170826",
        "170902","170909","170916","170923","170930",
        "171007","171014","171021","171028",
        "171104","171111","171118","171125",
        "171202","171209","171216","171223","171230",
        "180106"]
#this is a list of the relevant dates from the MTA turnstile data
```

This list was generated manually. However, it is also possible to generate a set of dates through the datetime function with an interval of 7 days between each date, then format each date into a string with the correct format.

In [67]:
```
d = {} #d is an empty dictionary

for a in date:
    mta_data = mta_url + a +".txt"
    d[a] = pd.read_csv(mta_data)

#this for loop adds data frames from the MTA turnstile data into the emp
ty dictionary, d
```

In [69]:
```
mta_data = pd.concat(d.values())
#this merges all the dataframes in the dictionary d into one larger data
frame

mta_data.tail()
#this code is used to check the number of rows from all the data
```

Out[69]:

	C/A	UNIT	SCP	STATION	LINENAME	DIVISION	DATE	TIME	
200660	TRAM2	R469	00-03-01	RIT-ROOSEVELT	R	RIT	01/05/2018	04:00:00	RE
200661	TRAM2	R469	00-03-01	RIT-ROOSEVELT	R	RIT	01/05/2018	08:00:00	RE
200662	TRAM2	R469	00-03-01	RIT-ROOSEVELT	R	RIT	01/05/2018	12:00:00	RE
200663	TRAM2	R469	00-03-01	RIT-ROOSEVELT	R	RIT	01/05/2018	16:00:00	RE
200664	TRAM2	R469	00-03-01	RIT-ROOSEVELT	R	RIT	01/05/2018	20:00:00	RE

In [301]:
```
mta_data.LINENAME.value_counts()
```

Out[301]:

1	1238143
6	625908
7	472804
F	391871
25	348480
A	315781
BQ	273054
MR	235502
L	230813
23	202073
BD	180283
123	179931
FG	168718
ACE	168598
NRW	166336
R	147834
G	145089
4567S	144400
2345ACJZ	141143
45	132397
C	129498
D	127743
BC	125206
Q	123444
BDFMNQRW	121932
4	120692
JZ	115619
ACENQRS1237W	113555
NQRW	112525
3	93769
	...
JZ456	29042
NQR456W	26969
456LNQRW	26871
FM	22455
7BDFM	22382
2345	22341
BDE	22231
1237ACENQRS	20332
ABCD	20255
6DF	20234
ACENGRS1237W	20216
ACS	20116
2345S	18033
BQS	17952
FQ	17818
1AC	15793
ACJLZ	15686
GL	15662
245	15611
ND	14006
7NQW	13489
DNR	13467
R2345	11186
S	11176
LG	11173
2	11159

```
ACG                11147
7EFMR              11112
23ACE               9093
S2345               8941
Name: LINENAME, Length: 114, dtype: int64
```

The line of code above is used to see the values of the LINENAME column. LINENAME refers to the available subway lines in one station. For example, the 59th street station has 7 subway lines: N,Q,R,4,5,6, and W. Thus, the 59th street station has "NQR456W" in its LINENAME column.

In [287]:
```
mta = mta_data.loc[mta_data['LINENAME'] == 'NQR456W']
```

The 59th street station (LINENAME = "NQR456W"), the only station with the LINENAME "NQR456W", is arbitrarily chosen here as the main part of the MTA data. I am not using the whole of the MTA data due to a few key reasons:

- The MTA turnstile data is somewhat convoluted, with data being taken multiple times a day, at different devices within the same station.
- The ridership trend (increases and decreases) of the NYC subway is assumed to be similar from one station to another, thus the 59th street subway station can be used as a proxy to approximate the subway ridership trends of New York City's subway stations.
- 59th street is located in a fairly populated area, surrounded by large corporate offices. It should thus have large enough ridership numbers that will give a better picture of changes in ridership numbers.

In [288]:
```
mta_proper = mta.reset_index()
#we reset the index after choosing the data we want to use, so the index
 starts from 0 again.
```

In [289]:
```
mta_proper.head(20)
#double check whether the data is properly sliced from the larger data f
rame
```

Out[289]:

	index	C/A	UNIT	SCP	STATION	LINENAME	DIVISION	DATE	TIME	DE
0	0	A002	R051	02-00-00	59 ST	NQR456W	BMT	12/31/2016	03:00:00	REGUI
1	1	A002	R051	02-00-00	59 ST	NQR456W	BMT	12/31/2016	07:00:00	REGUI
2	2	A002	R051	02-00-00	59 ST	NQR456W	BMT	12/31/2016	11:00:00	REGUI
3	3	A002	R051	02-00-00	59 ST	NQR456W	BMT	12/31/2016	15:00:00	REGUI
4	4	A002	R051	02-00-00	59 ST	NQR456W	BMT	12/31/2016	19:00:00	REGUI
5	5	A002	R051	02-00-00	59 ST	NQR456W	BMT	12/31/2016	23:00:00	REGUI
6	6	A002	R051	02-00-00	59 ST	NQR456W	BMT	01/01/2017	03:00:00	REGUI
7	7	A002	R051	02-00-00	59 ST	NQR456W	BMT	01/01/2017	07:00:00	REGUI
8	8	A002	R051	02-00-00	59 ST	NQR456W	BMT	01/01/2017	11:00:00	REGUI
9	9	A002	R051	02-00-00	59 ST	NQR456W	BMT	01/01/2017	15:00:00	REGUI
10	10	A002	R051	02-00-00	59 ST	NQR456W	BMT	01/01/2017	19:00:00	REGUI
11	11	A002	R051	02-00-00	59 ST	NQR456W	BMT	01/01/2017	23:00:00	REGUI
12	12	A002	R051	02-00-00	59 ST	NQR456W	BMT	01/02/2017	03:00:00	REGUI

	index	C/A	UNIT	SCP	STATION	LINENAME	DIVISION	DATE	TIME	DE
13	13	A002	R051	02-00-00	59 ST	NQR456W	BMT	01/02/2017	07:00:00	REGUI
14	14	A002	R051	02-00-00	59 ST	NQR456W	BMT	01/02/2017	11:00:00	REGUI
15	15	A002	R051	02-00-00	59 ST	NQR456W	BMT	01/02/2017	15:00:00	REGUI
16	16	A002	R051	02-00-00	59 ST	NQR456W	BMT	01/02/2017	19:00:00	REGUI
17	17	A002	R051	02-00-00	59 ST	NQR456W	BMT	01/02/2017	23:00:00	REGUI
18	18	A002	R051	02-00-00	59 ST	NQR456W	BMT	01/03/2017	03:00:00	REGUI
19	19	A002	R051	02-00-00	59 ST	NQR456W	BMT	01/03/2017	07:00:00	REGUI

In [290]:
```
mta_proper.loc[mta_proper['DATE'] == '01/07/2017']
#this step was done to illustrate the large number of data assigned to o
ne date, even
#though the data had been narrowed down to one station.
```

Out[290]:

	index	C/A	UNIT	SCP	STATION	LINENAME	DIVISION	DATE	TIME	D
499	0	A002	R051	02-00-00	59 ST	NQR456W	BMT	01/07/2017	03:00:00	REGU
500	1	A002	R051	02-00-00	59 ST	NQR456W	BMT	01/07/2017	07:00:00	REGU
501	2	A002	R051	02-00-00	59 ST	NQR456W	BMT	01/07/2017	11:00:00	REGU
502	3	A002	R051	02-00-00	59 ST	NQR456W	BMT	01/07/2017	15:00:00	REGU
503	4	A002	R051	02-00-00	59 ST	NQR456W	BMT	01/07/2017	19:00:00	REGU
504	5	A002	R051	02-00-00	59 ST	NQR456W	BMT	01/07/2017	23:00:00	REGU
541	42	A002	R051	02-00-01	59 ST	NQR456W	BMT	01/07/2017	03:00:00	REGU
542	43	A002	R051	02-00-01	59 ST	NQR456W	BMT	01/07/2017	07:00:00	REGU
543	44	A002	R051	02-00-01	59 ST	NQR456W	BMT	01/07/2017	11:00:00	REGU
544	45	A002	R051	02-00-01	59 ST	NQR456W	BMT	01/07/2017	15:00:00	REGU
545	46	A002	R051	02-00-01	59 ST	NQR456W	BMT	01/07/2017	19:00:00	REGU
546	47	A002	R051	02-00-01	59 ST	NQR456W	BMT	01/07/2017	23:00:00	REGU
583	84	A002	R051	02-03-00	59 ST	NQR456W	BMT	01/07/2017	03:00:00	REGU

	index	C/A	UNIT	SCP	STATION	LINENAME	DIVISION	DATE	TIME	D
584	85	A002	R051	02-03-00	59 ST	NQR456W	BMT	01/07/2017	07:00:00	REGU
585	86	A002	R051	02-03-00	59 ST	NQR456W	BMT	01/07/2017	11:00:00	REGU
586	87	A002	R051	02-03-00	59 ST	NQR456W	BMT	01/07/2017	15:00:00	REGU
587	88	A002	R051	02-03-00	59 ST	NQR456W	BMT	01/07/2017	19:00:00	REGU
588	89	A002	R051	02-03-00	59 ST	NQR456W	BMT	01/07/2017	23:00:00	REGU
625	126	A002	R051	02-03-01	59 ST	NQR456W	BMT	01/07/2017	03:00:00	REGU
626	127	A002	R051	02-03-01	59 ST	NQR456W	BMT	01/07/2017	07:00:00	REGU
627	128	A002	R051	02-03-01	59 ST	NQR456W	BMT	01/07/2017	11:00:00	REGU
628	129	A002	R051	02-03-01	59 ST	NQR456W	BMT	01/07/2017	15:00:00	REGU
629	130	A002	R051	02-03-01	59 ST	NQR456W	BMT	01/07/2017	19:00:00	REGU
630	131	A002	R051	02-03-01	59 ST	NQR456W	BMT	01/07/2017	23:00:00	REGU
667	168	A002	R051	02-03-02	59 ST	NQR456W	BMT	01/07/2017	03:00:00	REGU
668	169	A002	R051	02-03-02	59 ST	NQR456W	BMT	01/07/2017	07:00:00	REGU

	index	C/A	UNIT	SCP	STATION	LINENAME	DIVISION	DATE	TIME	D
669	170	A002	R051	02-03-02	59 ST	NQR456W	BMT	01/07/2017	11:00:00	REGU
670	171	A002	R051	02-03-02	59 ST	NQR456W	BMT	01/07/2017	15:00:00	REGU
671	172	A002	R051	02-03-02	59 ST	NQR456W	BMT	01/07/2017	19:00:00	REGU
672	173	A002	R051	02-03-02	59 ST	NQR456W	BMT	01/07/2017	23:00:00	REGU
...	...	...	...	...	...	...	...	...	...	...
793	294	A002	R051	02-03-05	59 ST	NQR456W	BMT	01/07/2017	03:00:00	REGU
794	295	A002	R051	02-03-05	59 ST	NQR456W	BMT	01/07/2017	07:00:00	REGU
795	296	A002	R051	02-03-05	59 ST	NQR456W	BMT	01/07/2017	11:00:00	REGU
796	297	A002	R051	02-03-05	59 ST	NQR456W	BMT	01/07/2017	15:00:00	REGU
797	298	A002	R051	02-03-05	59 ST	NQR456W	BMT	01/07/2017	19:00:00	REGU
798	299	A002	R051	02-03-05	59 ST	NQR456W	BMT	01/07/2017	23:00:00	REGU
835	336	A002	R051	02-03-06	59 ST	NQR456W	BMT	01/07/2017	03:00:00	REGU
836	337	A002	R051	02-03-06	59 ST	NQR456W	BMT	01/07/2017	07:00:00	REGU
837	338	A002	R051	02-03-06	59 ST	NQR456W	BMT	01/07/2017	11:00:00	REGU

	index	C/A	UNIT	SCP	STATION	LINENAME	DIVISION	DATE	TIME	D
838	339	A002	R051	02-03-06	59 ST	NQR456W	BMT	01/07/2017	15:00:00	REGU
839	340	A002	R051	02-03-06	59 ST	NQR456W	BMT	01/07/2017	19:00:00	REGU
840	341	A002	R051	02-03-06	59 ST	NQR456W	BMT	01/07/2017	23:00:00	REGU
877	378	A002	R051	02-05-00	59 ST	NQR456W	BMT	01/07/2017	03:00:00	REGU
878	379	A002	R051	02-05-00	59 ST	NQR456W	BMT	01/07/2017	07:00:00	REGU
879	380	A002	R051	02-05-00	59 ST	NQR456W	BMT	01/07/2017	11:00:00	REGU
880	381	A002	R051	02-05-00	59 ST	NQR456W	BMT	01/07/2017	15:00:00	REGU
881	382	A002	R051	02-05-00	59 ST	NQR456W	BMT	01/07/2017	19:00:00	REGU
882	383	A002	R051	02-05-00	59 ST	NQR456W	BMT	01/07/2017	23:00:00	REGU
919	420	A002	R051	02-05-01	59 ST	NQR456W	BMT	01/07/2017	03:00:00	REGU
920	421	A002	R051	02-05-01	59 ST	NQR456W	BMT	01/07/2017	07:00:00	REGU
921	422	A002	R051	02-05-01	59 ST	NQR456W	BMT	01/07/2017	11:00:00	REGU
922	423	A002	R051	02-05-01	59 ST	NQR456W	BMT	01/07/2017	15:00:00	REGU

	index	C/A	UNIT	SCP	STATION	LINENAME	DIVISION	DATE	TIME	D
923	424	A002	R051	02-05-01	59 ST	NQR456W	BMT	01/07/2017	19:00:00	REGU
924	425	A002	R051	02-05-01	59 ST	NQR456W	BMT	01/07/2017	23:00:00	REGU
961	462	A002	R051	02-06-00	59 ST	NQR456W	BMT	01/07/2017	03:00:00	REGU
962	463	A002	R051	02-06-00	59 ST	NQR456W	BMT	01/07/2017	07:00:00	REGU
963	464	A002	R051	02-06-00	59 ST	NQR456W	BMT	01/07/2017	11:00:00	REGU
964	465	A002	R051	02-06-00	59 ST	NQR456W	BMT	01/07/2017	15:00:00	REGU
965	466	A002	R051	02-06-00	59 ST	NQR456W	BMT	01/07/2017	19:00:00	REGU
966	467	A002	R051	02-06-00	59 ST	NQR456W	BMT	01/07/2017	23:00:00	REGU

72 rows × 12 columns

In [291]:
```
mta_proper.SCP.value_counts()
```

Out[291]:
```
02-03-00    2250
02-03-02    2250
02-00-00    2249
02-03-03    2249
02-03-05    2249
02-05-00    2249
02-05-01    2249
02-03-04    2249
02-03-06    2248
02-06-00    2247
02-00-01    2245
02-03-01    2235
Name: SCP, dtype: int64
```

Here, we check the value of the SCP column, which is a column for the Subunit Channel Position, which is information on the specific address of a "device" (as the language is somewhat ambiguous, I will assume that "device" refers to the individual turnstile).

In [292]:
```
scptest = mta_proper.loc[mta_proper['SCP'] == '02-03-00']
#Here, we take data that is associated with a specific turnstile/device
```

In [293]:
```
scptest.DATE.value_counts()
```

```
Out[293]: 07/05/2017    9
          07/26/2017    9
          02/08/2017    8
          02/10/2017    8
          12/21/2017    7
          10/16/2017    7
          09/05/2017    7
          11/17/2017    7
          12/20/2017    7
          04/24/2017    7
          11/09/2017    7
          07/07/2017    7
          02/09/2017    7
          03/18/2017    7
          04/21/2017    7
          12/13/2017    7
          01/04/2018    7
          11/05/2017    7
          10/27/2017    7
          02/15/2017    7
          08/04/2017    6
          08/23/2017    6
          12/06/2017    6
          04/10/2017    6
          01/30/2017    6
          11/13/2017    6
          02/12/2017    6
          02/19/2017    6
          03/08/2017    6
          08/17/2017    6
                        ..
          03/29/2017    6
          07/30/2017    6
          10/01/2017    6
          09/08/2017    6
          11/20/2017    6
          11/19/2017    6
          03/11/2017    6
          05/28/2017    6
          05/21/2017    6
          03/14/2017    6
          09/28/2017    6
          08/21/2017    6
          06/09/2017    6
          01/22/2017    6
          05/10/2017    6
          11/07/2017    6
          10/29/2017    6
          07/22/2017    6
          03/03/2017    6
          10/23/2017    6
          06/01/2017    6
          10/04/2017    6
          09/21/2017    6
          12/31/2016    6
          09/01/2017    6
          09/17/2017    6
```

```
06/25/2017    6
11/18/2017    6
03/12/2017    5
03/09/2017    5
Name: DATE, Length: 371, dtype: int64
```

Here, I am checking the dates to see if there are any anomalies or strange outliers. The difference in the number of data points in each date is cause by the inconsistent data gathering by the MTA - from data gathering activities that are done outside of MTA's schedule, to missing data points where there should be one. For example, the MTA often gathers data everyday up to 11.00pm or 23.00. However, there are many data points for the 23.00 time slot that is missing.

In [482]:
```
scptest["date"] = pd.to_datetime(scptest["DATE"])
#Here, we change the format of values in the "DATE" column from strings
 to a date time format.
```

```
/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:1: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-d
ocs/stable/indexing.html#indexing-view-versus-copy
  """Entry point for launching an IPython kernel.
```

In [483]:
```
scptest["date"]
#Again, this is another double check that the data has been manipulated
 properly.
```

Out[483]:

83	2016-12-31
84	2016-12-31
85	2016-12-31
86	2016-12-31
87	2016-12-31
88	2016-12-31
89	2017-01-01
90	2017-01-01
91	2017-01-01
92	2017-01-01
93	2017-01-01
94	2017-01-01
95	2017-01-02
96	2017-01-02
97	2017-01-02
98	2017-01-02
99	2017-01-02
100	2017-01-02
101	2017-01-03
102	2017-01-03
103	2017-01-03
104	2017-01-03
105	2017-01-03
106	2017-01-03
107	2017-01-04
108	2017-01-04
109	2017-01-04
110	2017-01-04
111	2017-01-04
112	2017-01-04
...	...
26552	2018-01-01
26553	2018-01-01
26554	2018-01-01
26555	2018-01-01
26556	2018-01-01
26557	2018-01-02
26558	2018-01-02
26559	2018-01-02
26560	2018-01-02
26561	2018-01-02
26562	2018-01-02
26563	2018-01-03
26564	2018-01-03
26565	2018-01-03
26566	2018-01-03
26567	2018-01-03
26568	2018-01-03
26569	2018-01-04
26570	2018-01-04
26571	2018-01-04
26572	2018-01-04
26573	2018-01-04
26574	2018-01-04
26575	2018-01-04
26576	2018-01-05
26577	2018-01-05

```
26578   2018-01-05
26579   2018-01-05
26580   2018-01-05
26581   2018-01-05
Name: date, Length: 2250, dtype: datetime64[ns]
```

In [484]:
```
group = scptest.groupby("date")
#the groupby function allows for more straightforward operations on the
 data frame
```

In [485]:
```
daily_ave = group.mean()
```

The daily average of each day is taken instead of the difference between maximum and minimum values for the following reasons:

- The inconsistent data gathering activities of he MTA causes missing data points, which would mean that taking the difference between the largest and lowest value associated with a certain date will not always be comparable.
- A daily average is assumed to better represent the daily changes in ridership.

It is important to note that the numerical values under the "ENTRIES" and "EXITS" columns are accumulated values instead of daily values.

In [486]:
```
daily_ave
```

Out[486]:

	index	ENTRIES	EXITS
date			
2016-12-31	85.5	8.420265e+05	3.132030e+06
2017-01-01	91.5	8.424167e+05	3.132957e+06
2017-01-02	97.5	8.427612e+05	3.133955e+06
2017-01-03	103.5	8.432092e+05	3.135692e+06
2017-01-04	109.5	8.437845e+05	3.137988e+06
2017-01-05	115.5	8.443613e+05	3.140394e+06
2017-01-06	121.5	8.449892e+05	3.142870e+06
2017-01-07	86.5	8.455353e+05	3.144729e+06
2017-01-08	92.5	8.458835e+05	3.145984e+06
2017-01-09	98.5	8.463568e+05	3.147778e+06
2017-01-10	104.5	8.471307e+05	3.150136e+06
2017-01-11	110.5	8.479560e+05	3.152424e+06
2017-01-12	116.5	8.486662e+05	3.154724e+06
2017-01-13	122.5	8.493482e+05	3.157069e+06
2017-01-14	86.5	8.498993e+05	3.158912e+06
2017-01-15	92.5	8.502403e+05	3.160154e+06
2017-01-16	98.5	8.505988e+05	3.161602e+06
2017-01-17	104.5	8.510810e+05	3.163571e+06
2017-01-18	110.5	8.516375e+05	3.165864e+06
2017-01-19	116.5	8.522588e+05	3.168215e+06
2017-01-20	122.5	8.529182e+05	3.170465e+06
2017-01-21	86.5	8.534868e+05	3.172466e+06
2017-01-22	92.5	8.539057e+05	3.174051e+06
2017-01-23	98.5	8.542548e+05	3.175682e+06
2017-01-24	104.5	8.547620e+05	3.177818e+06
2017-01-25	110.5	8.552973e+05	3.179921e+06
2017-01-26	116.5	8.558157e+05	3.182163e+06
2017-01-27	122.5	8.563692e+05	3.184379e+06
2017-01-28	86.5	8.568762e+05	3.186136e+06
2017-01-29	92.5	8.572247e+05	3.187403e+06
...	...	...	...

	index	ENTRIES	EXITS
date			
2017-12-07	116.5	1.010577e+06	3.742424e+06
2017-12-08	122.5	1.011183e+06	3.744704e+06
2017-12-09	88.5	1.011680e+06	3.746498e+06
2017-12-10	94.5	1.011972e+06	3.747662e+06
2017-12-11	100.5	1.012299e+06	3.749249e+06
2017-12-12	106.5	1.012789e+06	3.751358e+06
2017-12-13	113.0	1.013333e+06	3.753641e+06
2017-12-14	119.5	1.013891e+06	3.755932e+06
2017-12-15	125.5	1.014381e+06	3.758080e+06
2017-12-16	90.5	1.014869e+06	3.759676e+06
2017-12-17	96.5	1.015232e+06	3.760697e+06
2017-12-18	102.5	1.015620e+06	3.762314e+06
2017-12-19	108.5	1.016172e+06	3.764617e+06
2017-12-20	115.0	1.016697e+06	3.766895e+06
2017-12-21	122.0	1.017275e+06	3.769115e+06
2017-12-22	128.5	1.017932e+06	3.771429e+06
2017-12-23	86.5	1.018388e+06	3.772897e+06
2017-12-24	92.5	1.018647e+06	3.773784e+06
2017-12-25	98.5	1.018839e+06	3.774346e+06
2017-12-26	104.5	1.019110e+06	3.775456e+06
2017-12-27	110.5	1.019672e+06	3.777356e+06
2017-12-28	116.5	1.020321e+06	3.779525e+06
2017-12-29	122.5	1.020965e+06	3.781627e+06
2017-12-30	88.5	1.021526e+06	3.783157e+06
2017-12-31	94.5	1.021857e+06	3.784095e+06
2018-01-01	100.5	1.022104e+06	3.784811e+06
2018-01-02	106.5	1.022438e+06	3.786084e+06
2018-01-03	112.5	1.022967e+06	3.788188e+06
2018-01-04	119.0	1.023411e+06	3.789989e+06
2018-01-05	125.5	1.023783e+06	3.791632e+06

371 rows × 3 columns

In [488]:
```
daily_ave_diff = daily_ave.diff()
#Here, we take the difference between the accumulated value up until day
 t and day t-1
#this gives us the number of people who enter and/or exit in day t

daily_ave_diff.reset_index(inplace = True)
#As with the previous index reset, this is done for convenience so that
 the index
#starts at 0 again

daily_ave_diff["date"] = daily_ave_diff["date"].dt.strftime('%m/%d/%Y')
#Here, the format of the date is changed to match the format in the next
 data sets

daily_ave_diff.set_index("date", inplace = True)
#The date is now set as the new index

daily_ave_diff.drop(["12/31/2016", "01/01/2018","01/02/2018",
                     "01/03/2018","01/04/2018","01/05/2018"], inplace = T
rue)
#As the dates are the index, it allows for easier removal of irrelevant
 dates

daily_ave_diff.reset_index(inplace = True)
#the index is reset after the irrelevant rows are removed

daily_ave_diff
```

Out[488]:

	date	index	ENTRIES	EXITS
0	01/01/2017	6.0	390.166667	927.000000
1	01/02/2017	6.0	344.500000	998.333333
2	01/03/2017	6.0	448.000000	1736.666667
3	01/04/2017	6.0	575.333333	2296.333333
4	01/05/2017	6.0	576.833333	2406.333333
5	01/06/2017	6.0	627.833333	2475.500000
6	01/07/2017	-35.0	546.166667	1858.833333
7	01/08/2017	6.0	348.166667	1255.333333
8	01/09/2017	6.0	473.333333	1794.166667
9	01/10/2017	6.0	773.833333	2358.166667
10	01/11/2017	6.0	825.333333	2288.000000
11	01/12/2017	6.0	710.166667	2299.333333
12	01/13/2017	6.0	682.000000	2345.500000
13	01/14/2017	-36.0	551.166667	1842.500000
14	01/15/2017	6.0	341.000000	1242.166667
15	01/16/2017	6.0	358.500000	1448.666667
16	01/17/2017	6.0	482.166667	1968.666667
17	01/18/2017	6.0	556.500000	2293.166667
18	01/19/2017	6.0	621.333333	2350.500000
19	01/20/2017	6.0	659.333333	2250.500000
20	01/21/2017	-36.0	568.666667	2000.166667
21	01/22/2017	6.0	418.833333	1585.500000
22	01/23/2017	6.0	349.166667	1631.166667
23	01/24/2017	6.0	507.166667	2135.500000
24	01/25/2017	6.0	535.333333	2103.666667
25	01/26/2017	6.0	518.333333	2241.333333
26	01/27/2017	6.0	553.500000	2216.000000
27	01/28/2017	-36.0	507.000000	1757.500000
28	01/29/2017	6.0	348.500000	1266.500000
29	01/30/2017	6.0	377.166667	1699.666667
...	...	...	...	...
335	12/02/2017	-36.0	496.500000	1665.000000

	date	index	ENTRIES	EXITS
336	12/03/2017	6.0	340.000000	1058.666667
337	12/04/2017	6.0	342.166667	1586.666667
338	12/05/2017	6.0	538.166667	2209.166667
339	12/06/2017	6.0	566.333333	2346.833333
340	12/07/2017	6.0	612.500000	2276.500000
341	12/08/2017	6.0	606.000000	2280.000000
342	12/09/2017	-34.0	497.166667	1793.666667
343	12/10/2017	6.0	292.666667	1163.833333
344	12/11/2017	6.0	326.500000	1587.000000
345	12/12/2017	6.0	490.333333	2108.666667
346	12/13/2017	6.5	543.809524	2282.904762
347	12/14/2017	6.5	557.857143	2291.595238
348	12/15/2017	6.0	490.000000	2148.000000
349	12/16/2017	-35.0	488.000000	1595.666667
350	12/17/2017	6.0	362.666667	1021.500000
351	12/18/2017	6.0	388.333333	1616.500000
352	12/19/2017	6.0	552.333333	2303.500000
353	12/20/2017	6.5	524.666667	2277.952381
354	12/21/2017	7.0	577.714286	2219.428571
355	12/22/2017	6.5	657.619048	2313.952381
356	12/23/2017	-42.0	455.833333	1468.666667
357	12/24/2017	6.0	258.833333	886.833333
358	12/25/2017	6.0	191.666667	562.333333
359	12/26/2017	6.0	271.833333	1109.500000
360	12/27/2017	6.0	561.333333	1899.833333
361	12/28/2017	6.0	649.166667	2168.833333
362	12/29/2017	6.0	643.666667	2102.000000
363	12/30/2017	-34.0	561.333333	1530.666667
364	12/31/2017	6.0	331.333333	938.000000

365 rows × 4 columns

In [489]:
```
github = "https://raw.githubusercontent.com/AristoJoesoef/my_first_repos
itory/master/"
citi = "CitiBike_Ridership_2017.csv"
citibike_2017 = github+citi
```

The Citibike data consists of multiple csv files, which have been merged and uploaded onto my github site as a show of an alternative method of combining csv files (doing so manually before importing it into the iPython notebook), as compared to the method used to merge the csv files from the MTA turnstile data.

In [406]:
```
citibike = pd.read_csv(citibike_2017)
```

In [407]:
```
citibike.head(20)
```

Out[407]:

	Date	Trips over the past 24-hours (midnight to 11:59pm)	Miles traveled today (midnight to 11:59 pm)	Total Annual Members (All Time)	24-Hour Passes Purchased (midnight to 11:59 pm)	3-Day Passes Purchased (midnight to 11:59 pm)
0	1/1/17	16009	50761	206624	1237	33
1	1/2/17	8921	21545	206675	142	15
2	1/3/17	14198	25879	206728	42	15
3	1/4/17	34039	69966	206778	475	29
4	1/5/17	28393	52446	206845	214	17
5	1/6/17	24177	48964	206879	157	17
6	1/7/17	4425	23556	206892	21	4
7	1/8/17	6417	22756	206907	27	2
8	1/9/17	15856	35209	206933	30	2
9	1/10/17	23232	46216	206955	67	12
10	1/11/17	32419	63099	206981	208	37
11	1/12/17	39783	84763	207021	479	30
12	1/13/17	33611	75847	207059	373	43
13	1/14/17	13819	43090	207087	252	33
14	1/15/17	17384	50738	207120	476	19
15	1/16/17	24110	51664	207163	448	30
16	1/17/17	18646	40697	207198	63	15
17	1/18/17	30857	57978	207224	146	17
18	1/19/17	36561	72077	207262	391	27
19	1/20/17	26752	56479	207288	184	19

In [334]:
```
nyc_weather_2017 = github + "NYC_Weather_2017.csv"
```

The weather data is taken from the National Center for Environmental Information, from who I had to request the weather data in a specific locatin (in this case, Central Park, NYC) and a specific date range (in this case, from the 1st of January of 2017 to the 31st of December of 2017). Thus, I have a csv file for my weather data, which I have uploaded to my github account.

In [338]:
```
nyc_weather = pd.read_csv(nyc_weather_2017, usecols = ['DATE','STATION',
'NAME','PRCP',
                                                       'SNOW','TMAX','TM
IN'])
```

The relevant columns taken from the weatehr data are:

- Date of the data
- The weather station code where the data was recorded, in this case the code for Central Park in New York City
- The name of the weather station, in this case Central Park in New York City
- Precipitation, which is a measure of rainfall
- Snow, which is a measure of snowfall
- Maximum temperature at any given date
- Minimum temperature at any given date

In [339]:
```
nyc_weather.head(20)
```

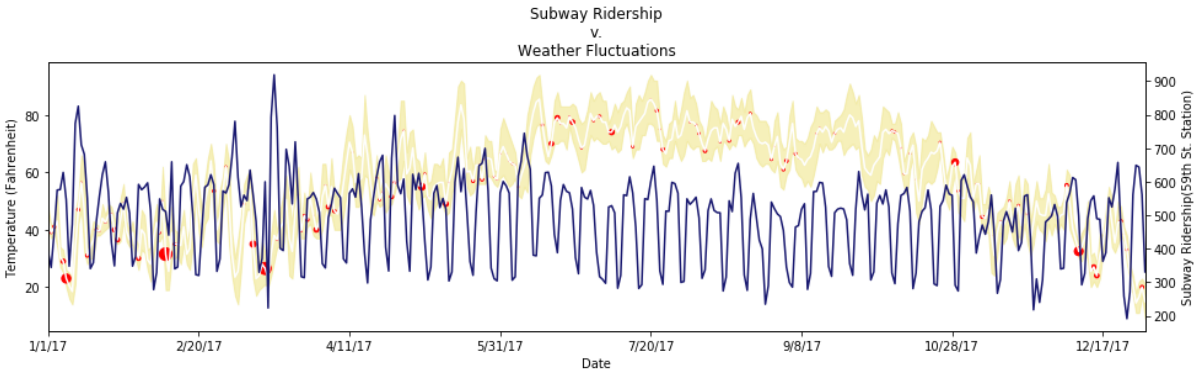
Out[339]:

	STATION	NAME	DATE	PRCP	SNOW	TMAX	TMIN
0	USW00094728	NY CITY CENTRAL PARK, NY US	1/1/17	0.00	0.0	48	40
1	USW00094728	NY CITY CENTRAL PARK, NY US	1/2/17	0.21	0.0	41	37
2	USW00094728	NY CITY CENTRAL PARK, NY US	1/3/17	0.58	0.0	43	39
3	USW00094728	NY CITY CENTRAL PARK, NY US	1/4/17	0.00	0.0	52	34
4	USW00094728	NY CITY CENTRAL PARK, NY US	1/5/17	0.00	0.0	34	27
5	USW00094728	NY CITY CENTRAL PARK, NY US	1/6/17	0.05	1.2	33	25
6	USW00094728	NY CITY CENTRAL PARK, NY US	1/7/17	0.32	5.1	26	20
7	USW00094728	NY CITY CENTRAL PARK, NY US	1/8/17	0.00	0.0	25	16
8	USW00094728	NY CITY CENTRAL PARK, NY US	1/9/17	0.00	0.0	23	14
9	USW00094728	NY CITY CENTRAL PARK, NY US	1/10/17	0.00	0.0	46	21
10	USW00094728	NY CITY CENTRAL PARK, NY US	1/11/17	0.52	0.0	52	42
11	USW00094728	NY CITY CENTRAL PARK, NY US	1/12/17	0.05	0.0	66	47
12	USW00094728	NY CITY CENTRAL PARK, NY US	1/13/17	0.00	0.0	62	32
13	USW00094728	NY CITY CENTRAL PARK, NY US	1/14/17	0.12	0.6	34	28
14	USW00094728	NY CITY CENTRAL PARK, NY US	1/15/17	0.00	0.0	38	30
15	USW00094728	NY CITY CENTRAL PARK, NY US	1/16/17	0.00	0.0	43	30
16	USW00094728	NY CITY CENTRAL PARK, NY US	1/17/17	0.35	0.0	42	38
17	USW00094728	NY CITY CENTRAL PARK, NY US	1/18/17	0.06	0.0	41	38
18	USW00094728	NY CITY CENTRAL PARK, NY US	1/19/17	0.00	0.0	49	39

	STATION	NAME	DATE	PRCP	SNOW	TMAX	TMIN
19	USW00094728	NY CITY CENTRAL PARK, NY US	1/20/17	0.10	0.0	45	40

In [340]:
```
nyc_weather["TAVE"] = (nyc_weather["TMAX"]+nyc_weather["TMIN"])/2
#The average daily temperature is then calculated (and named "TAVE")
```

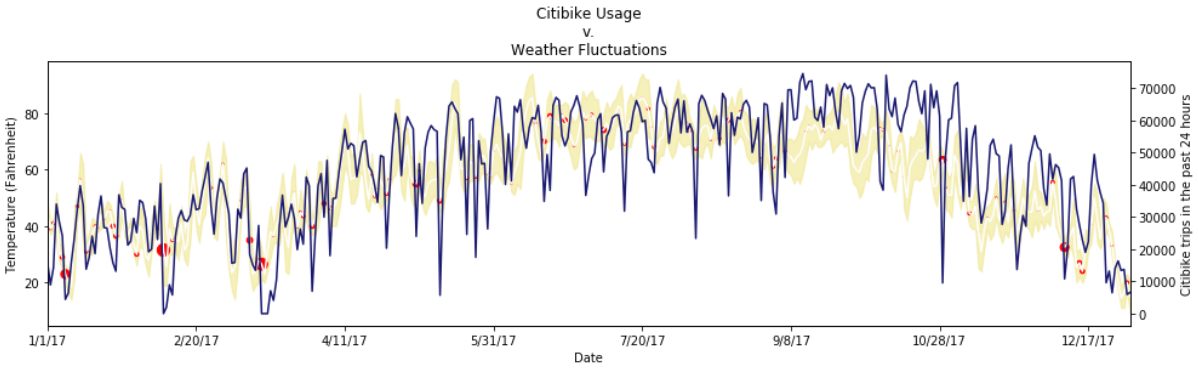
In [492]:
```
fig, ax1  = plt.subplots(figsize = (16,4))
#for this large dataset, a longer figure is preferable

nyc_weather["TAVE"].plot(ax = ax1, color = "white")
plt.fill_between(nyc_weather.index, nyc_weather["TMIN"],
                 nyc_weather["TMAX"], color = "#F0E68C", alpha = 0.6)
#Here, we create a shaded region between maximum and minimum daily tempe
rature,
#and highlight the average daily temperature

ax1.scatter(nyc_weather.index, nyc_weather["TAVE"],
            s=10*(nyc_weather["PRCP"]+nyc_weather["SNOW"]),
            alpha= 1.0, color = "red")
#This creates a scatterplot of both rainfall and snowfall (represented b
y the size of the dot)

ax1.set_xlim(0,364)
#Set the plotted graph to only go as far as the available data (365th da
ta point)

ax1.set_xlabel("Date")

ax1.set_title("Subway Ridership\n"+"v.\n"+"Weather Fluctuations")

ax1.set_ylabel("Temperature (Fahrenheit)")

#Set labels and titles

labels = [item.get_text() for item in ax1.get_xticklabels()]
#Make a list of the x tick labels

labels[0] = "1/1/17"
labels[1] = "2/20/17"
labels[2] = "4/11/17"
labels[3] = "5/31/17"
labels[4] = "7/20/17"
labels[5] = "9/8/17"
labels[6] = "10/28/17"
labels[7] = "12/17/17"
#Register the planned changes to the relevant x tick label

ax1.set_xticklabels(labels)
#Rename the x tick labels

ax2 = ax1.twinx()
#Set up a second y-axis on the other end of the graph

daily_ave_diff["ENTRIES"].plot(ax = ax2, color = "#191970")
#Plot the daily subway ridership

ax2.set_ylabel("Subway Ridership(59th St. Station)")
#Set y-label for the second y-axis

plt.show()
```



The data gives us some interesting observations. Most notably, the subway ridership numbers are cyclical, most likely caused by weekends or holidays. After all, the chosen location at 59th Street is located near many large corporate buildings. This means that the workforce in the area likely has a strong impact on the subway ridership. Moreover, the cyclical nature of the subway ridership seems to be largely unaffected by the changes in temperature. Again, this can be attributed to the nature of the people who often use this station (corporate office workers). These people need to go to work, at times regardless of weather conditions. Travel through this station is thus more of a necessity than a choice, which means that weather should not affect the subway ridership much or at all. There is also a lack of evidence that precipitation and/or snow has a strong impact on subway ridership. Rainfall and snowfall is represented by the red dots, where larger dots mean heavier rainfall and/or snowfall. Similarly to temperature changes, subway ridership is largely indifferent to changes in rainfall or snowfall. Thus, subway ridership in the 59th Street Station is largely cyclical in nature, unaffected by most weather fluctuations.

In [493]:
```
fig, ax1  = plt.subplots(figsize = (16,4))
#for this large dataset, a longer figure is preferable

nyc_weather["TAVE"].plot(ax = ax1, color = "white")
#Here, we create a shaded region between maximum and minimum daily tempe
rature,
#and highlight the average daily temperature

plt.fill_between(nyc_weather.index, nyc_weather["TMIN"],
                 nyc_weather["TMAX"], color = "#F0E68C", alpha = 0.6)
#This creates a scatterplot of both rainfall and snowfall (represented b
y the size of the dot)

ax1.scatter(nyc_weather.index, nyc_weather["TAVE"],
            s=10*(nyc_weather["PRCP"]+nyc_weather["SNOW"]),
            alpha= 1.0, color = "red")

ax1.set_xlim(0,364)
#Set the plotted graph to only go as far as the available data (365th da
ta point)

ax1.set_xlabel("Date")

ax1.set_title("Citibike Usage\n"+"v.\n"+"Weather Fluctuations")

ax1.set_ylabel("Temperature (Fahrenheit)")

#Set labels and titles

labels = [item.get_text() for item in ax1.get_xticklabels()]
#Make a list of the x tick labels

labels[0] = "1/1/17"
labels[1] = "2/20/17"
labels[2] = "4/11/17"
labels[3] = "5/31/17"
labels[4] = "7/20/17"
labels[5] = "9/8/17"
labels[6] = "10/28/17"
labels[7] = "12/17/17"
#Register the planned changes to the relevant x tick label

ax1.set_xticklabels(labels)
#Rename the x tick labels

ax2 = ax1.twinx()
#Set up a second y-axis on the other end of the graph

citibike["Trips over the past 24-hours (midnight to 11:59pm)"].plot(ax =
 ax2,
                                                                    color
 = "#191970")
#Plot the 24-hour usage data of Citibike

ax2.set_ylabel("Citibike trips in the past 24 hours")
#Set y-label for the second y-axis
```

```
plt.show()
```



Unlike the case with subway ridership, Citibike usage can loosely be said to be affected by weather fluctuations. As the average temperature of the day increases, there is a higher likelihood for people to use Citibike. The red dots that signify rainfall and/or snowfall can be seen to be closer, in the most part, to the throughs/valleys of the Citibike usage graph instead of the crests/peaks. However, without further rigorous statistical analysis, it is not possible to make a conclusion on the effects of rainfall and/or snowfall on Citibike usage. An interesting thing to note is the large fluctuations in the daily Citibike usage, which can be caused by various reasons such as lack of promotion or renewals of Citibike membership, shifting preferences for other forms of transportation etc.

Conclusion

Interestingly, increases in Citibike usage does not have a strong impact on subway ridership. This implies that the consumers that Citibike and the MTA are offering their services to are from a different market segment, eitehr due to price constraints or due to differences in preferences. For example, the corporate workers theorized to use the 59th Street Station can be less inclined to use Citibike for transportation because they want to avoid sweating, because they prefer being in the subway where they can relax while waiting for the train to arrive or to reach their stop, etc.

Though the subway ridership is largely unaffected, Citibike usage does seem to fluctuate with increases and decreases in temperature. From the fluctuations of Citibike usage, there is some positive correlation between Citibike usage and warmth (higher temperatures). Holding the assumption that users of Citibike are at least partially motivated to use Citibike so that they can enjoy the weather, we can concluded that higher temperatures are often viewed to be "good" weather, while colder temperatures disincentivize people from spending too much time outside. Of course, there are various factors taht are unaccounted for in this analysis, stemming from a priori knowledge of what is generally said to be "good" weather. For example, the absence of strong winds, tbe presence of warm sunlight, the lack of piled up snow etc. are often considered to be conditions that contribute to "good" weather.

Unfortunately, the data set, specifically the data set on the weather in New York City, lacks many important parameters. It does not tell us, for example, of the strength of sunlight in each day.