Introduction to the dataset

STATION DETAILS	
Name	PROVIDENCE, RI US
Network:ID	GHCND:USW00014765
Latitude/Longitude	41.7225°, -71.4325°
Elevation	16.8 m

PERIOD OF RECORD	
Start Date[1]	1942-08-01
End Date[1]	2018-04-21
Data Coverage[2]	93%
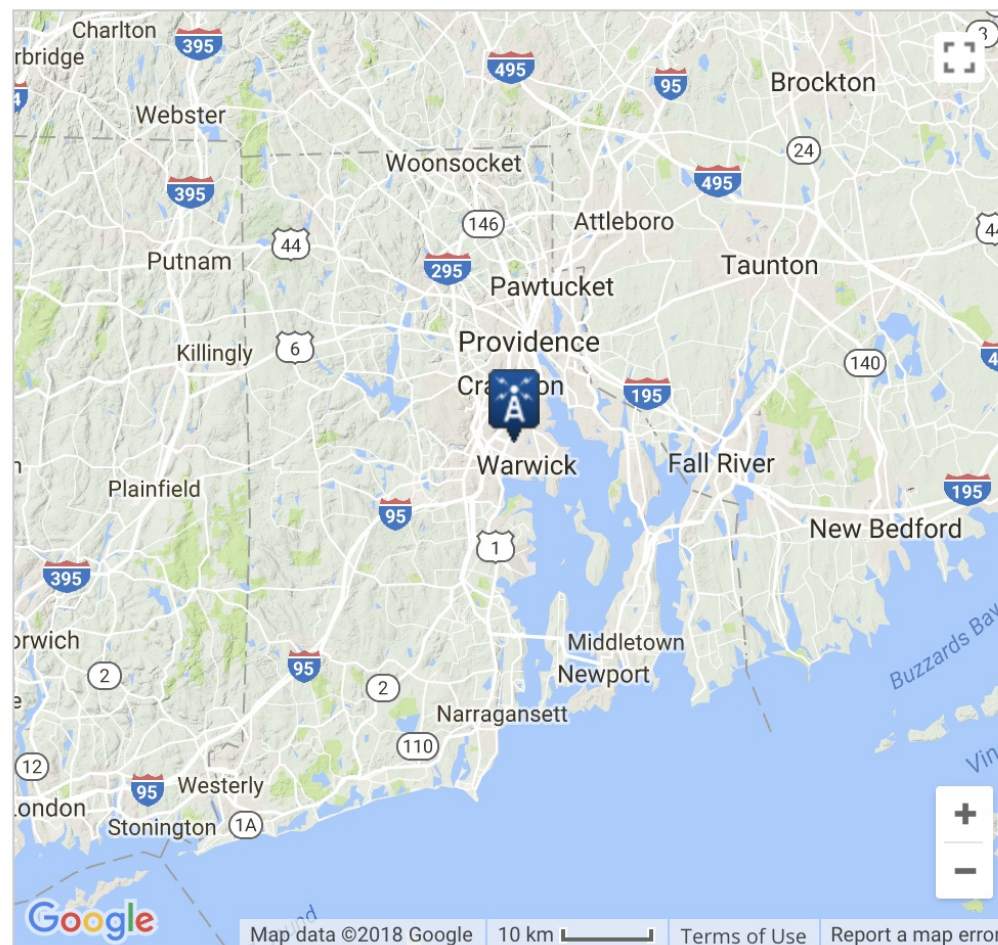
```
weather = pd.read_csv('weather.csv')
weather.head(3)
```

```
        STATION          DATE   TAVG   TMIN   TMAX   AWND   WSF2   WT01   WT02
0   USW00014765   2005-01-01   44.0     35     53   8.95   25.1    1.0    NaN
1   USW00014765   2005-01-02   36.0     28     44   9.40   14.1    NaN    NaN
2   USW00014765   2005-01-03   49.0     44     53   6.93   17.0    1.0    NaN


       ...   WT11   WT13   WT14   WT15   WT16   WT17   WT18   WT19   WT21   WT22
0      ...    NaN    1.0    NaN    NaN    NaN    NaN    NaN    NaN    NaN    NaN
1      ...    NaN    NaN    NaN    NaN    1.0    NaN    1.0    NaN    NaN    NaN
2      ...    NaN    1.0    NaN    NaN    1.0    NaN    NaN    NaN    NaN    NaN
```

- `TAVG` , `TMIN` , `TMAX` : Temperature

- `AWND` , `WSF2` : Wind speed

- `WT01` … `WT22` : Bad weather conditions

Examining the wind speed
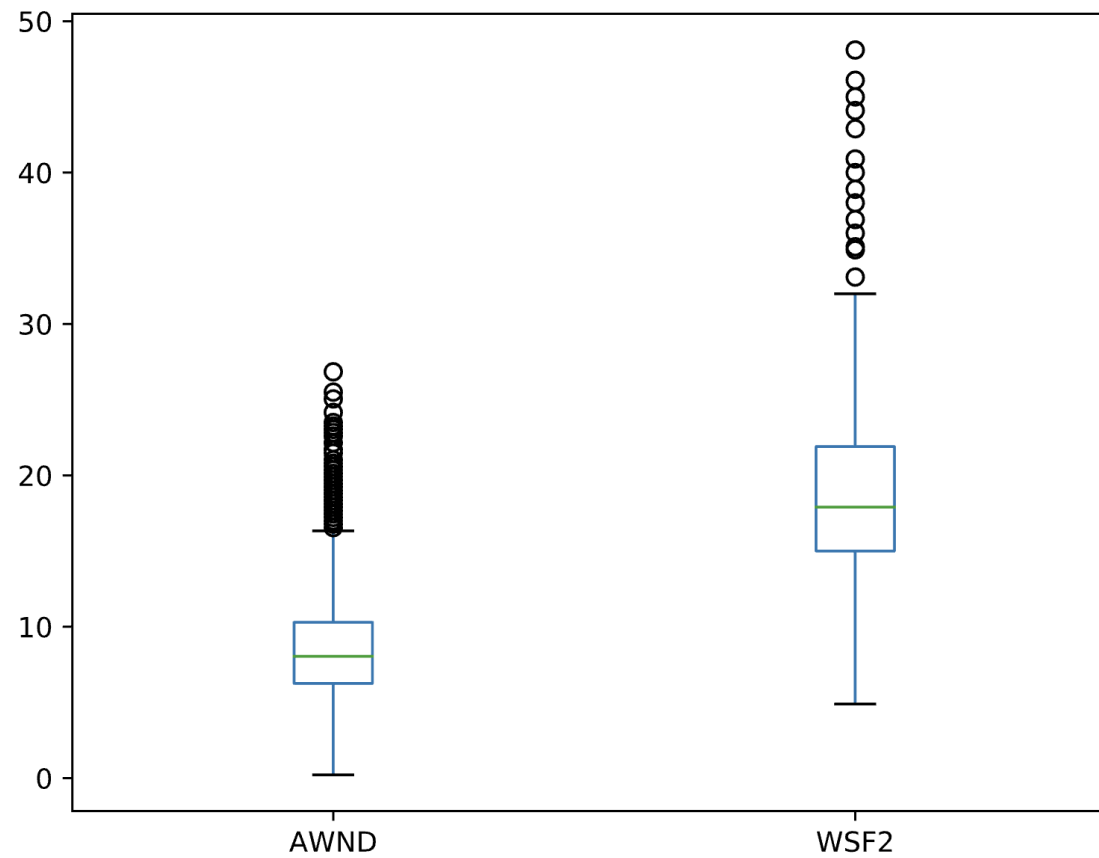
```
weather[['AWND', 'WSF2']].head()
```

```
weather[['AWND', 'WSF2']].describe()
```

```
     AWND  WSF2
0    8.95  25.1
1    9.40  14.1
2    6.93  17.0
3    6.93  16.1
4    7.83  17.0
```

```
                AWND          WSF2
count    4017.000000   4017.000000
mean        8.593707     19.274782
std         3.364601      5.623866
min         0.220000      4.900000
25%         6.260000     15.000000
50%         8.050000     17.900000
75%        10.290000     21.900000
max        26.840000     48.100000
```
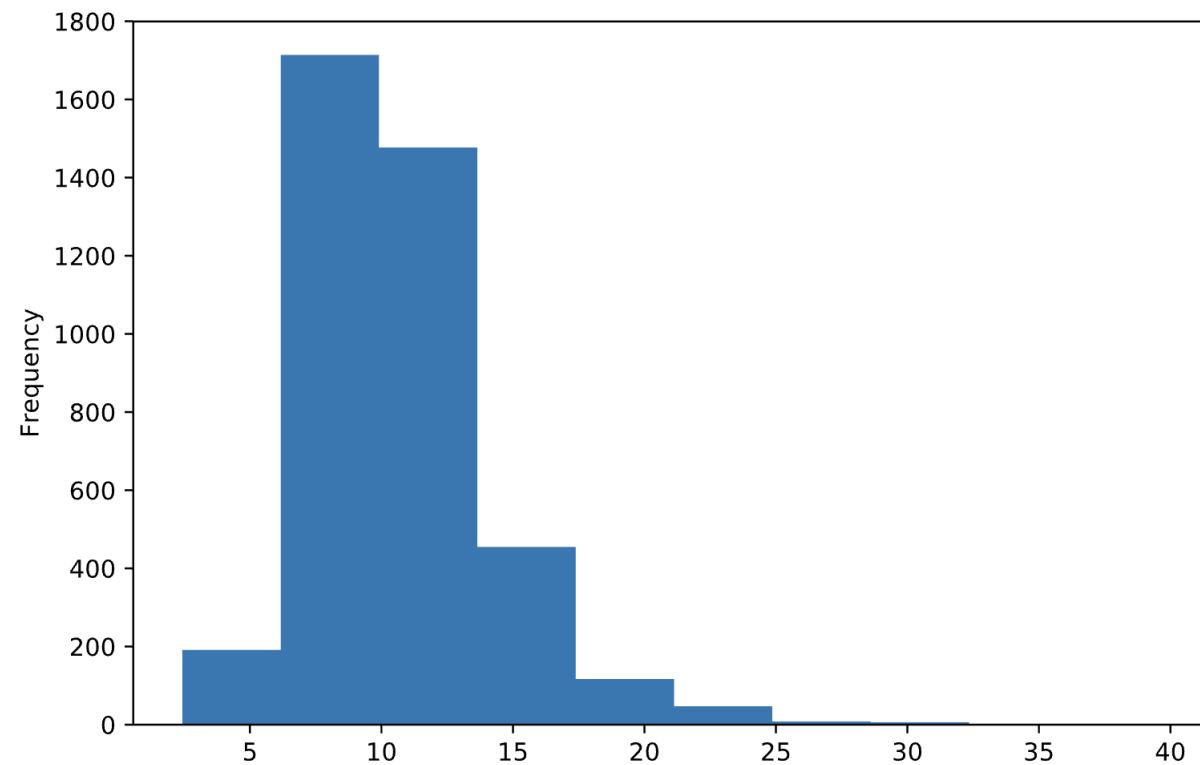
Creating a box plot

```
weather[['AWND', 'WSF2']].plot(kind='box')
plt.show()
```
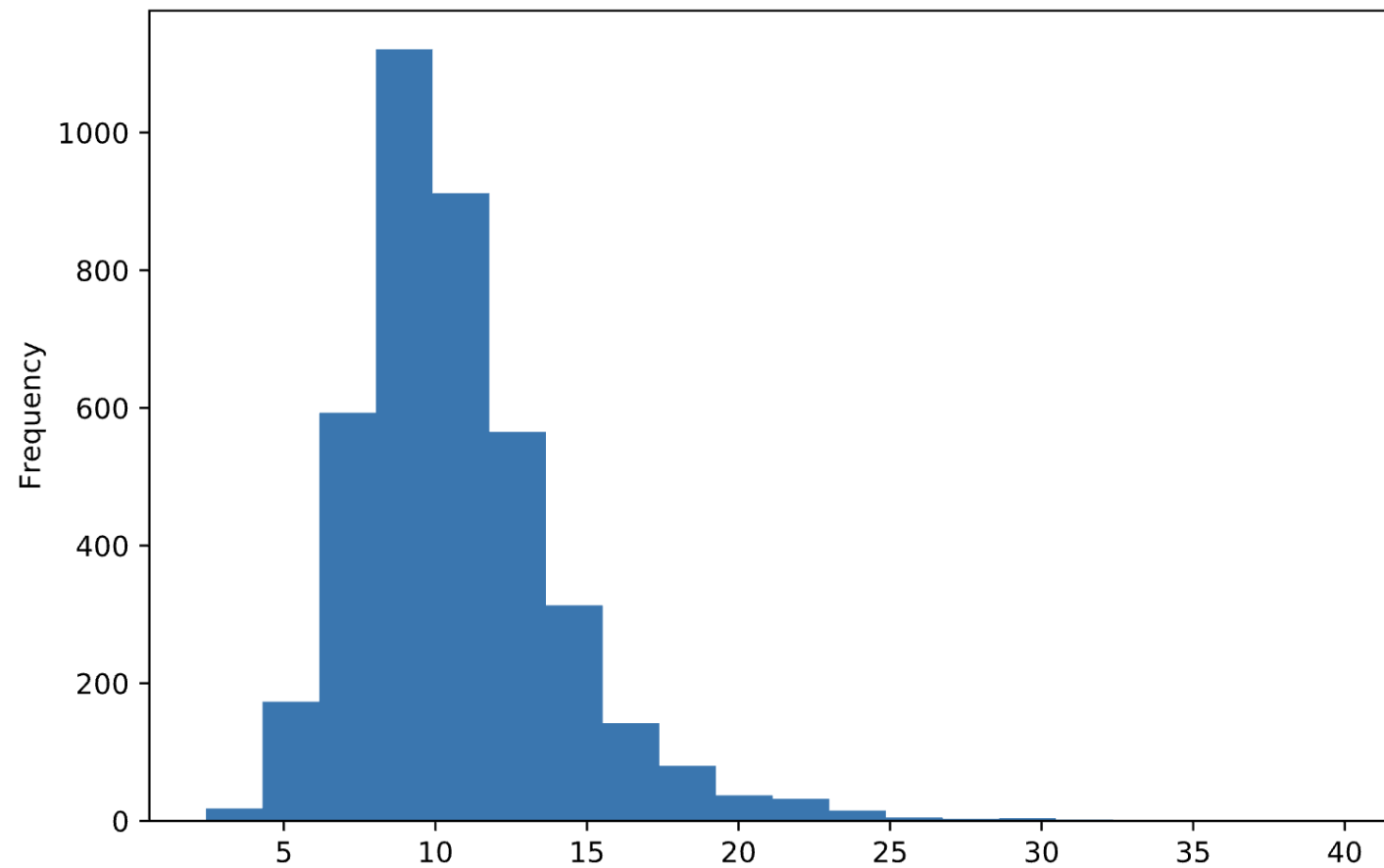
Creating a histogram (1)

```python
weather['WDIFF'] = weather.WSF2 - weather.AWND

weather.WDIFF.plot(kind='hist')
plt.show()
```

Creating a histogram (2)

```python
weather.WDIFF.plot(kind='hist', bins=20)
plt.show()
```

Let's practice!

ANALYZING POLICE ACTIVITY WITH PANDAS

Categorizing the weather

ANALYZING POLICE ACTIVITY WITH PANDAS

Kevin Markham
Founder, Data School

DataCamp

Selecting a DataFrame slice (1)

```
weather.shape
```

```
(4017, 28)
```

```
weather.columns
```

```
Index(['STATION', 'DATE', 'TAVG', 'TMIN', 'TMAX', 'AWND',
       'WSF2', 'WT01', 'WT02', 'WT03', 'WT04', 'WT05',
       'WT06', 'WT07', 'WT08', 'WT09', 'WT10', 'WT11',
       'WT13', 'WT14', 'WT15', 'WT16', 'WT17', 'WT18',
       'WT19', 'WT21', 'WT22', 'TDIFF'], dtype='object')
```

Selecting a DataFrame slice (2)

```
temp = weather.loc[:, 'TAVG':'TMAX']
```

```
temp.shape
```

```
(4017, 3)
```

```
temp.columns
```

```
Index(['TAVG', 'TMIN', 'TMAX'], dtype='object')
```

DataFrame operations

`temp.head()`

```
     TAVG   TMIN   TMAX
0    44.0     35     53
1    36.0     28     44
2    49.0     44     53
3    42.0     39     45
4    36.0     28     43
```

`temp.sum(axis='columns').head()`

```
0     132.0
1     108.0
2     146.0
3     126.0
4     107.0
```

`temp.sum()`

```
TAVG      63884.0
TMIN     174677.0
TMAX     246116.0
```

ANALYZING POLICE ACTIVITY WITH PANDAS

Mapping one set of values to another

```
ri.stop_duration.unique()
```

```
array(['0-15 Min', '16-30 Min', '30+ Min'], dtype=object)
```

```python
mapping = {'0-15 Min':'short',
           '16-30 Min':'medium',
           '30+ Min':'long'}

ri['stop_length'] = ri.stop_duration.map(mapping)

ri.stop_length.dtype
```

```
dtype('O')
```

Changing data type from object to category (1)

```
ri.stop_length.unique()
```

```
array(['short', 'medium', 'long'], dtype=object)
```

- Category type stores the data more efficiently
- Allows you to specify a logical order for the categories

```
ri.stop_length.memory_usage(deep=True)
```

```
8689481
```

Changing data type from object to category (2)

```python
cats = ['short', 'medium', 'long']
```

```python
ri['stop_length'] = ri.stop_length.astype('category',
                                           ordered=True,
                                           categories=cats)
```

```python
ri.stop_length.memory_usage(deep=True)
```

```
3400602
```

Using ordered categories (1)

```
ri.stop_length.head()
```

```
stop_datetime
2005-01-04 12:55:00      short
2005-01-23 23:15:00      short
2005-02-17 04:15:00      short
2005-02-20 17:15:00     medium
2005-02-24 01:20:00      short
Name: stop_length, dtype: category
Categories (3, object): [short < medium < long]
```

Using ordered categories (2)

```python
ri[ri.stop_length > 'short'].shape
```

```
(16959, 16)
```

```python
ri.groupby('stop_length').is_arrested.mean()
```

```
stop_length
short      0.013654
medium     0.093595
long       0.261572
Name: is_arrested, dtype: float64
```

Let's practice!

ANALYZING POLICE ACTIVITY WITH PANDAS

Merging datasets

ANALYZING POLICE ACTIVITY WITH PANDAS

Kevin Markham
Founder, Data School

```
apple
```

```
                         date    time    price
date_and_time
2018-02-14 09:30:00  2/14/18    9:30   163.04
2018-02-14 16:00:00  2/14/18   16:00   167.37
2018-02-15 09:30:00  2/15/18    9:30   169.79
2018-02-15 16:00:00  2/15/18   16:00   172.99
```

```
apple.reset_index(inplace=True)

apple
```

```
        date_and_time     date    time    price
0 2018-02-14 09:30:00  2/14/18    9:30   163.04
1 2018-02-14 16:00:00  2/14/18   16:00   167.37
2 2018-02-15 09:30:00  2/15/18    9:30   169.79
3 2018-02-15 16:00:00  2/15/18   16:00   172.99
```

Preparing the second DataFrame

```
high_low
```

```
      DATE     HIGH      LOW
0  2/14/18   167.54   162.88
1  2/15/18   173.09   169.00
2  2/16/18   174.82   171.77
```

```
high = high_low[['DATE', 'HIGH']]
high
```

```
      DATE     HIGH
0  2/14/18   167.54
1  2/15/18   173.09
2  2/16/18   174.82
```

Merging the DataFrames

```
apple_high = pd.merge(left=apple, right=high,
                      left_on='date', right_on='DATE',
                      how='left')
```

- `left=apple` : Left DataFrame

- `right=high` : Right DataFrame

- `left_on='date'` : Key column in left DataFrame

- `right_on='DATE'` : Key column in right DataFrame

- `how='left'` : Type of join

apple_high

```
     date_and_time       date    time   price     DATE    HIGH
0 2018-02-14 09:30:00  2/14/18    9:30  163.04  2/14/18  167.54
1 2018-02-14 16:00:00  2/14/18   16:00  167.37  2/14/18  167.54
2 2018-02-15 09:30:00  2/15/18    9:30  169.79  2/15/18  173.09
3 2018-02-15 16:00:00  2/15/18   16:00  172.99  2/15/18  173.09
```

apple

```
     date_and_time       date    time   price
0 2018-02-14 09:30:00  2/14/18    9:30  163.04
1 2018-02-14 16:00:00  2/14/18   16:00  167.37
2 2018-02-15 09:30:00  2/15/18    9:30  169.79
3 2018-02-15 16:00:00  2/15/18   16:00  172.99
```

high

```
     DATE    HIGH
0  2/14/18  167.54
1  2/15/18  173.09
2  2/16/18  174.82
```

Setting the index

```
apple_high.set_index('date_and_time', inplace=True)

apple_high
```

```
                           date     time     price        DATE       HIG
date_and_time
2018-02-14 09:30:00    2/14/18     9:30    163.04     2/14/18     167.5
2018-02-14 16:00:00    2/14/18    16:00    167.37     2/14/18     167.5
2018-02-15 09:30:00    2/15/18     9:30    169.79     2/15/18     173.0
2018-02-15 16:00:00    2/15/18    16:00    172.99     2/15/18     173.0
```

Let's practice!

DataCamp

Driver gender and vehicle searches

```
ri.search_conducted.mean()
```

```
0.0382153092354627
```

```
ri.groupby('driver_gender').search_conducted.mean()
```

```
driver_gender
F    0.019181
M    0.045426
```

```
ri.groupby(['violation', 'driver_gender']).search_conducted.mean()
```

```
violation            driver_gender
Equipment            F                0.039984
                     M                0.071496
Moving violation     F                0.039257
                     M                0.061524
Other                F                0.041018
                     M                0.046191
Registration/plates  F                0.054924
                     M                0.108802
Seat belt            F                0.017301
                     M                0.035119
Speeding             F                0.008309
                     M                0.027885
```

```
search_rate = ri.groupby(['violation',
                           'driver_gender']).search_conducted.mean()

search_rate
```

```
violation          driver_gender
Equipment          F                  0.039984
                   M                  0.071496
Moving violation   F                  0.039257
                   M                  0.061524
...                ...                     ...
```

```
type(search_rate)

type(search_rate.index)
```

```
pandas.core.series.Series
pandas.core.indexes.multi.MultiIndex
```

```
violation        driver_gender
Equipment        F                0.039984
                 M                0.071496
Moving violation F                0.039257
                 M                0.061524
...              ...                   ...
```

```
search_rate.loc['Equipment']
```

```
driver_gender
F    0.039984
M    0.071496
```

```
search_rate.loc['Equipment', 'M']
```

```
0.07149643705463182
```

Converting a multi-indexed Series to a DataFrame

```
search_rate.unstack()
```

```
driver_gender              F          M
violation
Equipment           0.039984   0.071496
Moving violation    0.039257   0.061524
Other               0.041018   0.046191
...                      ...        ...
```

```
type(search_rate.unstack())
```

```
pandas.core.frame.DataFrame
```

Converting a multi-indexed Series to a DataFrame

```python
ri.pivot_table(index='violation',
               columns='driver_gender',
               values='search_conducted')
```

```
driver_gender        F         M
violation
Equipment      0.039984  0.071496
Moving violation  0.039257  0.061524
Other          0.041018  0.046191
...                 ...       ...
```

Let's practice!

ANALYZING POLICE ACTIVITY WITH PANDAS

Conclusion

ANALYZING POLICE ACTIVITY WITH PANDAS

Kevin Markham
Founder, Data School

Stanford Open Policing Project

- Download data: **https://openpolicing.stanford.edu/**

Thank you!

ANALYZING POLICE ACTIVITY WITH PANDAS