Importing Data

Again, getting data into the programming environment for analysis is a fundamental and first step for any data analytics or machine learning task. In practice, data usually comes in a comma-separated value, **csv,** format.

```
my_DF = pd.read_csv('link_to_file/csv_file', sep=',', header = None)
```

To export a DataFrame back to **csv**

```
my_DF.to_csv('file_name.csv')
```

For the next example, the dataset 'states.csv' is found in the chapter folder of the code repository of this book.

```
my_DF = pd.read_csv('states.csv', sep=',', header = 0)

# read the top 5 rows
my_DF.head()

# save DataFrame to csv
my_DF.to_csv('save_states.csv')
```

Timeseries with Pandas

One of the core strengths of Pandas is its powerful set of functions for manipulating timeseries datasets. A couple of these functions are covered in this material.

Importing a Dataset with a DateTime Column

When importing a dataset that has a column containing datetime entries, Pandas has an attribute in the **read_csv** method called **parse_dates** that converts the datetime column from strings into Pandas **date** datatype. The attribute **index_col** uses the column of datetimes as an index to the DataFrame.

The method **head()** prints out the first five rows of the DataFrame, while the method **tail()** prints out the last five rows of the DataFrame. This function is very useful for taking a peek at a large DataFrame without having to bear the computational cost of printing it out entirely.

```
# load the data
data = pd.read_csv('crypto-markets.csv', parse_dates=['date'], index_
       col='date')
data.head()
'Output':
  slug date  symbol name  ranknow     open    high     low   close
volume   market    close_ratio  spread
2013-04-28  bitcoin BTC Bitcoin 1   135.30  135.98  132.10  134.21
     0   1500520000     0.5438    3.88
2013-04-29  bitcoin BTC Bitcoin 1   134.44  147.49  134.00  144.54
     0   1491160000     0.7813   13.49
2013-04-30  bitcoin BTC Bitcoin 1   144.00  146.93  134.05  139.00
     0   1597780000     0.3843   12.88
2013-05-01  bitcoin BTC Bitcoin 1   139.00  139.89  107.72  116.99
     0   1542820000     0.2882   32.17
2013-05-02  bitcoin BTC Bitcoin 1   116.38  125.60  92.28   105.21
     0   1292190000     0.3881   33.32
```

Let's examine the index of the imported data. Notice that they are the datetime entries.

```
# get the row indices
data.index
'Output':
DatetimeIndex(['2013-04-28', '2013-04-29', '2013-04-30', '2013-05-01',
               '2013-05-02', '2013-05-03', '2013-05-04', '2013-05-05',
               '2013-05-06', '2013-05-07',
               ...
               '2018-01-01', '2018-01-02', '2018-01-03', '2018-01-04',
               '2018-01-05', '2018-01-06', '2018-01-07', '2018-01-08',
               '2018-01-09', '2018-01-10'],
              dtype='datetime64[ns]', name='date', length=659373,
              freq=None)
```

Selection Using DatetimeIndex

The **DatetimeIndex** can be used to select the observations of the dataset in various interesting ways. For example, we can select the observation of an exact day or the observations belonging to a particular month or year. The selected observation can be subsetted by columns and grouped to give more insight in understanding the dataset.

Let's see some examples.

Select a Particular Date

Let's select a particular date from a DataFrame.

```
# select a particular date
data['2018-01-05'].head()
'Output':
                  slug  symbol         name  ranknow      open   high  \
date
2018-01-05       bitcoin    BTC       Bitcoin        1  15477.20  17705.20
2018-01-05      ethereum    ETH      Ethereum        2    975.75   1075.39
2018-01-05        ripple    XRP        Ripple        3      3.30      3.56
2018-01-05  bitcoin-cash    BCH  Bitcoin Cash        4   2400.74   2648.32
2018-01-05       cardano    ADA       Cardano        5      1.17      1.25

                   low         close       volume     market  \
date
2018-01-05  15202.800000  17429.500000  23840900000  259748000000
2018-01-05    956.330000    997.720000   6683150000   94423900000
2018-01-05      2.830000      3.050000   6288500000  127870000000
2018-01-05   2370.590000   2584.480000   2115710000   40557600000
2018-01-05      0.903503      0.999559    508100000   30364400000

            close_ratio   spread
date
2018-01-05       0.8898  2502.40
2018-01-05       0.3476   119.06
2018-01-05       0.3014     0.73
2018-01-05       0.7701   277.73
2018-01-05       0.2772     0.35
```