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.