DATA SCIENCE EAST AFRICA

Importing Data In Python

Day 4/20

Data can be in any of the popular formats – CSV, TXT, XLS or XLSX for (Excel), sas7bdat (SAS), Stata, Rdata (R) etc. Loading data in python environment is the most initial step of analyzing data. We make use of python library Pandas in this process.

Install and Load pandas Package

pandas is a powerful data analysis package. It makes data exploration and manipulation easy. It has several functions to read data from various sources.

If you are using Anaconda, pandas must be already installed. You need to load the package by using the following command –

```
import pandas as pd
```

If pandas package is not installed, you can install it by running the following code in Ipython Console. If you are using Spyder, you can submit the following code in Ipython console within Spyder –

```
!pip install pandas
```

If you are using Anaconda and you doubt pandas is not installed, you can try the following line of code to install pandas –

```
!conda install pandas
```

1. **Import CSV files**

It is important to note that a singlebackslash does not work when specifying the file path. You need to either change it to forward slash or add one more backslash like below –

```
import pandas as pd
df= pd.read_csv("property data.csv")
```

If no header (title) in raw data file –

```
import pandas as pd
df= pd.read_csv("property data.csv", header = None)
```

You need to include header = None option to tell Python there is no column name (header) in data.

Note that we can include column names by using names= option.

```
import pandas as pd
df= pd.read_csv("property data.csv",
                header = None,
                names = ['ID',  'first_name'])
```

The variable names can also be added separately by using the following command.

```
df.columns  = ['ID',  'first_name']
```

2. **Import File from URL**

You don't need to perform additional steps to fetch data from URL. Simply put URL in read_csv() function (applicable only for CSV files stored in URL).

```
df=pd.read_csv("http://winterolympicsmedals.com/medals.csv")
df
```

3. **Read Text File**

We can use read_table() function to pull data from text file. We can also use read_csv() with sep= "\t" to read data from tab-separated file.

```
mydf = pd.read_table("example2.txt")
myddf = pd.read_csv("example2.txt", sep ="\t")
```

4. **Read Excel File**

The read_excel() function can be used to import excel data into Python.

```
df1=pd.read_excel("Mobile_Banking .xlsx", sheetname="Data 1",
skiprows=2)
```

If you do not specify name of sheet in sheetname = option, it would take by default first sheet.

5. **Read delimited file**

Suppose you need to import a file that is separated with white spaces.

```
df2 =
pd.read_table("http://www.ssc.wisc.edu/~bhansen/econometrics/i
nvest.dat",  sep="\s+", header = None)
```

To include variable names, use the names = option like below –

```
df3 =
pd.read_table("http://www.ssc.wisc.edu/~bhansen/econometrics/i
nvest.dat", sep="\s+", names=['a', 'b', 'c', 'd'])
```

6. **Read SAS File**

We can import SAS data file by using read_sas() function.

```
df4 = pd.read_sas('cars.sas7bdat')
```

If you have a large SAS File, you can try package named **pyreadstat** which is faster than pandas. It is equivalent to **haven** package in R which provides easy and fast way to read data from SAS, SPSS and Stata.

To install this package, you can use the command –

```
pip install pyreadstat
```

```
import pyreadstat
df, meta = pyreadstat.read_sas7bdat('cars.sas7bdat')
# done! let's see what we got
print(df.head())
print(meta.column_names)
print(meta.column_labels)
print(meta.number_rows)
print(meta.number_columns)
```

7. **Read Stata File**

We can load Stata data file via read_stata() function –

```
mydf1 = pd.read_stata('cars.dta')
```

Note: the pyreadstat package lets you to pull value labels from stata files –

```
import pyreadstat
df, meta = pyreadstat.read_dta("cars.dta")
```

And to get labels, set apply_value_formats as TRUE –

```
df, meta = pyreadstat.read_dta("cars.dta",
apply_value_formats=True)
```

8. **Import R Data File**

Using pyreadr package, you can load .Rdata and .Rds format files which in general contains R data frame. You can install this package using the command below –

```
pip install pyreadr
```

With the use of **read_r**( ) function, we can import R data format files.

```
import pyreadr
result = pyreadr.read_r('C:/Users/sampledata.RData')
print(result.keys()) # let's check what objects we got
df11 = result["df1"] # extract the pandas data frame for
object df11
```

Similarly, you can read .Rds formatted file.

9. **Import Data from SPSS File**

```
import pyreadstat
df, meta =
pyreadstat.read_sav("file.sav",apply_value_formats=True)
```

If you don't want value labels, make apply_value_formats as False.

10. **Read sample of rows and columns**

By specifying nrows= and usecols=, you can fetch specified number of rows and columns.

```
df7 =
pd.read_csv("http://winterolympicsmedals.com/medals.csv",
nrows=5, usecols=(1,5,7))
```

nrows = 5 implies you want to import only first 5 rows and usecols = refers to specified columns you want to import.

– **Skip rows while importing**

Suppose you want to skip first 5 rows and wants to read data from 6th row (6th row would be a header row)

```
df8 =
pd.read_csv("http://winterolympicsmedals.com/medals.csv",
skiprows=5)
```

– **Specify values as missing values**

By including na_values= option, you can specify values as missing values. In this case, we are telling python to consider dot (.) as missing cases.

```
df9 = pd.read_csv("workingfile.csv", na_values=['.'])
```

Read, practice and do exercise on:

1). How to read data from tables stored in SQL Server by building a connection.
2). How to install and use Teradata module and how to use it to integrate python with Teradata Database.

Additional materials: http://bit.ly/DSEAReadMore

All the best, Data Science East Africa.