Read and Plot data with Python

Student and Staff IT Introduction

ssiti2013@gmail.com*

27/06/2013

1 Read data from a .CSV file

There are different ways to open and read a CSV (Comma Separated Values) file in python. You can export the Excel spreadsheets into CSV. Here is the easiest way using only the standard libraries of Python. The CSV file in this case contains only one column with float numbers. Try the following code on file "csv1.csv":

```
import sys

# sys.argv is a list containing the command line parameters
# sys.argv[0] always contains the name of the python script
if len(sys.argv) != 2:
    print 'Error: give a file name on the command line'
    exit(1) # Exit and return the error code 1

csv_file_name = sys.argv[1]
csv = open(csv_file_name, 'r') #open the csv file
data = [] #initiation of the data list
for line in csv: #read the file
    line = line.rstrip() # remove the 'newline' character at the end of the line
    data.append(float(line))#append the current value in the data list
print data
csv.close() #close the csv file
```

It can be a bit harder to read files containing many columns and/or column names, but fortunately, there are python libraries allowing to read csv files. Here we will use a function call **genfromtxt** from the library **numpy**. Try to open the file "csv2.csv" using the following code. The file contains two columns representing values. The first row of the file corresponds to the names of the columns: "V1" and "V2". To load data from a CSV file using **genfromtxt**, you have to specify a delimiter (the comma here), the type of your data (not mandatory). If column names are present in the file, you should use the "skip_header" option.

```
import sys
import numpy as np

if len(sys.argv) != 2:
    print 'Error: give a file name on the command line'
    exit(1) # Exit and return the error code 1

data = np.genfromtxt(sys.argv[1], dtype=float, delimiter=',', skip_header=True)
```

*Document written by Pierre-Yves Dupont (p.y.dupont@massey.ac.nz), for IFS staff and students, Massey University.

```
print data # All data
print data[:,0] # column 1
print data[:,1] # column 2
```

You can find more information about **genfromtxt** options here: `http://docs.scipy.org/doc/numpy/user/basics.io.genfromtxt.html`. The **names** option might be interesting to work on columns only. You can find a tutorial about another library allowing to read and write CSV files here: `http://www.pythonforbeginners.com/systems-programming/using-the-csv-module-in-python/` Now you know how to read data from a simple text file, so let's see how to plot these data.

2 Scatter Plot

To plot data we will use the *pyplot* module from *matplotlib* library. We will modify the code we used to read the simple CSV file (csv1.csv) to plot this data.

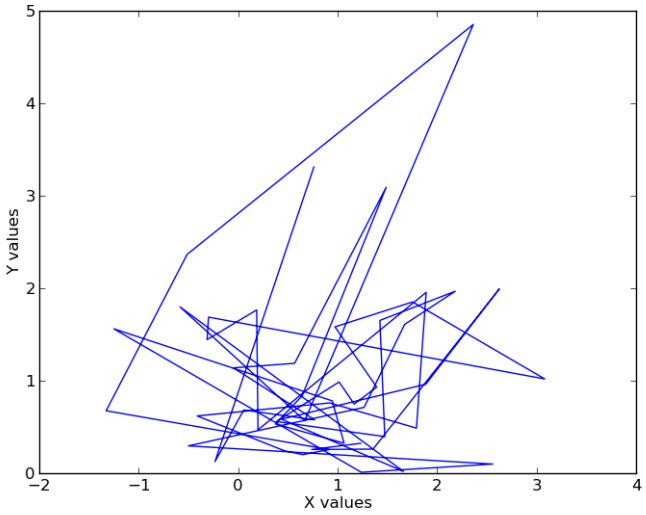
```
import matplotlib.pyplot as plt # import the pyplot module as plt
import sys

# sys.argv is a list containing the command line parameters
# sys.argv[0] always contains the name of the python script
if len(sys.argv) != 2:
    print 'Error: give a file name on the command line'
    exit(1) # Exit and return the error code 1

csv_file_name = sys.argv[1]
csv = open(csv_file_name, 'r') #open the csv file
data = [] #initiation of the data list
for line in csv: #read the file
    line = line.rstrip() # remove the 'newline' character at the end of the line
    data.append(float(line))#append the current value in the data list
print data
csv.close() #close the csv file

plt.plot(data) #Simple scatter plot of the data
plt.ylabel('My data') #title of the Y axis
plt.show() # show the plot
```

The default values for the X axis are a simple **range(N)**, but it is possible to set these values using the same **plot** function. Here is a simple example:

```
x = [1, 2, 3, 4, 5, 6]
y = [1, 2, 4, 3, 6, 5]
plt.plot( x, y )
plt.xlabel( "X values" )
plt.ylabel( "Y values" )
plt.show()
```

Exercise: Try to build a script to plot the data of the file "csv2.csv". You should obtain a graphic like:

These data points should not be linked by lines in the plot. Here is an example showing how to configure the plot display:




```
import numpy as np
import matplotlib.pyplot as plt

# evenly sampled time at 200ms intervals
t = np.arange(0., 5., 0.2) #Simple data data. Same as range() function

# red dashes, blue squares and green triangles
plt.plot(t, t, 'r--')     #Red (r) dashes (--)
plt.plot(t, t**2, 'bs')   #Blue (b) squares (s)
plt.plot(t, t**3, 'g^')   #Green (g) triangles (^)
plt.plot(t, t**4, 'b+-') #Blue (b) crosses (+) linked by a line (-)
plt.show()
```

Using this example, change your script to display the data using independent blue dots. The different markers (symbols) are:

```
[ 7   |  4   |   5     |   6   | 'o' | 'D' |
  '_' |  ''  | 'None'  | None  | ' ' | '8' |
  'p' |  ','  |  '+'   |  '.'  | 's' | '*' |
  3   |  0   |  1      |  2    | '1' | '3' |
  'v' | '<'  |  '>'    |  '^'  | '|' | 'x' |
  'h' | 'H'  |  'd'    |  '4'  | '2' ]
```

The different line styles are:

```
[ '-' | '--' | '-.' | ':' | 'None' | ' ' | '' ]
```

Now you know how to build line plots and scatter plots, lets see how to build box plots.

3 Box plot

Here is an example of a code that builds a box plot from random data. Try it, then display the values of the file "cvs3.csv" using a box plot.

```
import matplotlib.pyplot as plt
import numpy as np

#Fake data generation (random position on a normal distribution)
e1 = np.random.normal(0, 1, size=(500,))
e2 = np.random.normal(0, 1, size=(500,))
e3 = np.random.normal(0, 2, size=(500,))
e4 = np.random.normal(0, 2, size=(500,))

data=[e1,e2,e3,e4]
plt.boxplot(data)
plt.show()
```

The **plt.boxplot** function makes a box and whisker plot for each column of the given data. The box extends from the lower to upper quartile values of the data, with a line indicating the median. The whiskers extend from the box to show the range of the data. The length of the whiskers is defined as a function of the inner quartile range (by default 1.5 times longer). Flier points are those past the end of the whiskers.

Remarks: PyPlot allows you to specify the whiskers length and plot another value (like the mean) instead of the median.

4 More resources

You can find many resources online for python, numpy and matplotlib

- There is a good introdution to NumPy and Matplotlib for python beginners here: `http://youtu.be/3Fp1zn5ao2M`
- A NumPy tutorial on SciPy website: `http://wiki.scipy.org/Tentative_NumPy_Tutorial`
- A Matplotlib tutorial on SciPy website: `http://wiki.scipy.org/Cookbook/Matplotlib/`
- Matplotlib documentation is really good, it contains also a lot of examples and tutorials: `http://matplotlib.org`
- A tutorial with a lot of styling options: `http://cs.smith.edu/dftwiki/index.php/MatPlotLib_Tutorial_1`
- Another really good introduction to most of the plot types available in Matplotlib: `http://www.loria.fr/~rougier/teaching/matplotlib/`