RESEARCH TOOLS 2011

LECTURE 15

2011-Oct-20
Kurt Schwehr
http://schwehr.org

UNH CCOM/JHC
Python: Matplotlib part 1

http://creativecommons.org/licenses/by-nc-sa/3.0/
Research Tools Class 15

Python: plotting data with Matplotlib and spatial distances with proj

Kurt Schwehr
2011-Oct-20
http://schwehr.org
<table>
<thead>
<tr>
<th>No.</th>
<th>Date</th>
<th>Title/Notes</th>
<th>Audio</th>
<th>Present</th>
<th>Video</th>
<th>Blog</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>2011-10-20</td>
<td>Python: Matplotlib [org]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>2011-10-18</td>
<td>Python: parsing GPS data</td>
<td>pdf</td>
<td>key</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rd 1</td>
<td>2011-10-13</td>
<td>NOAA - Making Waves 3</td>
<td>txt</td>
<td></td>
<td></td>
<td>comment</td>
</tr>
<tr>
<td>Vid 13</td>
<td>2011-10-15</td>
<td>Python part 6 - parsing GPS data</td>
<td></td>
<td></td>
<td></td>
<td>comment</td>
</tr>
<tr>
<td>13</td>
<td>2011-10-13</td>
<td>Python: if, while, function</td>
<td>pdf</td>
<td>key</td>
<td></td>
<td>comment</td>
</tr>
<tr>
<td>Vid 12</td>
<td>2011-10-13</td>
<td>Python part 5 - while loops</td>
<td>pdf</td>
<td>key</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vid 11</td>
<td>2011-10-12</td>
<td>Python part 4 - if, modules, command line args [org]</td>
<td>pdf</td>
<td>key</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>No class - UNH Monday Schedule</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vid 10</td>
<td>2011-10-10</td>
<td>Emacs part 5 - modes [org]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vid 9</td>
<td>2011-10-09</td>
<td>Python part 3: parts [org]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vid 8</td>
<td>2011-10-09</td>
<td>Python part 2: ways to run python code [org]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>2011-10-06</td>
<td>Python: files, for loops [org]</td>
<td>mp3</td>
<td>m4a</td>
<td>ogg</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>2011-10-04</td>
<td>ipython and python data types [org]</td>
<td>mp3</td>
<td>m4a</td>
<td>ogg</td>
<td></td>
</tr>
<tr>
<td>Vid 7</td>
<td>2011-10-02</td>
<td>Python part 1: intro to python/ipython [org]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Friday, October 21, 11

/home/researchtools/class/15:
total used in directory 8 available 10

researchtools@ubuntu:~/class/15$ wget
wget: missing URL
Usage: wget [OPTION]... [URL]...

Try `wget --help' for more options.

Resolving vislab-ccom.unh.edu... 192.168.3.3
Connecting to vislab-ccom.unh.edu[192.168.3.3]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 7250 (7.1K) [text/plain]
Saving to: `15-matplotlib.org'

100%[=================================================================>] 7,250 --.-K/s in 0s
2011-10-20 11:04:11 (14.6 MB/s) - `15-matplotlib.org' saved [7250/7250]

researchtools@ubuntu:~/class/15$
**Class 15: matplotlib - graphing**

---

**Introduction**

This class is about starting to plot data. Matplotlib is a *super*
Class 15: matplotlib - graphing

Introduction

This class is about starting to plot data! Matplotlib is a *super*
# Introduction

This class is about starting to plot data! Matplotlib is a *super* powerful plotting system. I am definitely not an expert. Check out the range of examples:

http://matplotlib.sourceforge.net/

# Setup

I have pre-parsed the data for you.

```bash
#+BEGIN_SRC sh
mkdir -p ~/class/15
cd ~/class/15
#+END_SRC
```

There is no need to uncompress the data!

```bash
#+BEGIN_SRC sh
bzcat 2011-10-11.gga.dat.bz2 | head
# x y z quality satellites hdop
-70.9395833333 43.1354166667 35.7 2 9 1.1
-70.9395766667 43.135415 36.1 2 9 1.1
-70.9395766667 43.135415 36.1 2 9 1.1
-70.9395766667 43.1354133333 36.5 2 9 1.1
-70.9395666667 43.1354133333 37.0 2 9 1.1
-70.9395633333 43.1354133333 37.4 2 9 1.1
```

Friday, October 21, 11

Do NOT uncompress the 2011-10-11.gga.dat.bz2
x is longitude
y is latitude
x y z is a right handed coordinate frame
make sure to start ipython with "--pylab" or you will be missing a lot of the functions we will be using today.
# Introduction

This class is about starting to use a powerful plotting system. I am going to use the range of examples:

http://matplotlib.sourceforge.net

* Setup

I have pre-parsed the data for you:

```bash
#BEGIN_SRC sh
mkdir -p ~/class/15
cd ~/class/15
#END_SRC
```

There is no need to uncompress the data!

```bash
#BEGIN_SRC sh
bzcat 2011-10-11.gga.dat.bz2 | head
# x y z quality satellites hdo
d0.93595633333 43.154133333 37.0 2 9 1.1
#-70.9395666667 43.135413333 37.4 2 9 1.1
#-70.939563333 43.154133333 37.8 2 9 1.1
#-70.939563333 43.135413333 38.3 2 9 1.1
#-70.9395666667 43.154133333 38.7 2 9 1.1
#-70.939563333 43.135413333 39.1 2 9 1.1
```

Welcome to pylab, a matplotlib-based Python environment.

For more information, type `help(pylab)`.

Friday, October 21, 11
Asking for help with `loadtxt`
Look at the “File:” path and you will see that loadtext is a part of “numpy”
loadtxt\( (\text{fname}, \text{dtype}=\text{type 'float'}, \text{comments}='#', \text{delimiter}=None, \text{converters}=None, \text{skiprows}=0, \text{usecols}=None, \text{unpack}=False) \)

**Docstring:**

Load data from a text file.

Each row in the text file must have the same number of values.

**Parameters**

\begin{itemize}
\item \texttt{fname} : file or str
  File or filename to read. If the filename extension is `.`gz` or `.`bz2`, the file is first decompressed.
\item \texttt{dtype} : data-type, optional
  Data-type of the resulting array; default: float. If this is a record data-type, the resulting array will be 1-dimensional, and each row will be interpreted as an element of the array. In this case, the number of columns used must match the number of fields in the data-type.
\item \texttt{comments} : str, optional
  The character used to indicate the start of a comment; default: `#`.
\item \texttt{delimiter} : str, optional
\end{itemize}

```python
# The results
array([[ 70.9395633333, 43.1354133333, 35.7 , 2. ,
        9. , 1.1 ],
       [ 70.9395633333, 43.1354133333, 35.7 , 2. ,
        9. , 1.1 ]])
```
The string used to separate values. By default, this is any whitespace.

collectors : dict, optional
A dictionary mapping column number to a function that will convert
that column to a float. E.g., if column 0 is a date string:
``collectors = {0: datestr2num}``. Converters can also be used to
provide a default value for missing data:
``collectors = {3: lambda s: float(s or 0)}``. Default: None.

skiprows : int, optional
Skip the first `skiprows` lines; default: 0.

usecols : sequence, optional
Which columns to read, with 0 being the first. For example,
``usecols = (1,4,5)`` will extract the 2nd, 5th and 6th columns.
The default, None, results in all columns being read.

unpack : bool, optional
If True, the returned array is transposed, so that arguments may be
unpacked using ```x, y, z = loadtxt(...)```. The default is False.

Returns
-------
out : ndarray
# Loading the data

Start ipython with pylab loaded

```
# BEGIN_SRC sh
ipython --pylab
# END_SRC
```

We can load the data using numpy's handle gzip and bzip2 compressed files.

```
# BEGIN_SRC python
loadtxt?
# q to quit out of the pager

data = loadtxt('2011-10-11.gga.dat.bz2')
```

```
In [5]:
```

```
data

array([[ -70.93958333,  43.13541667,  35.74025295,  2. ,  9. ,
        1.1 ,  1.1 ],
       [-70.93957667,  43.13541547,  36.14481789,  2. ,  9. ,
        1.1 ,  1.1 ],
       [-70.93957,  43.13541333,  36.5094802 ,  2. ,  9. ,
        1.1 ,  1.1 ],
       ...
       [-70.93955167,  43.13545667,  43.22315989,  2. ,  8. ,
        1.2 ,  1.2 ],
       [-70.93955167,  43.13545667,  43.32772469,  2. ,  8. ,
        1.2 ,  1.2 ],
       [-70.93955167,  43.13545833,  43.56963994,  2. ,  8. ,
        1.2 ,  1.2 ]])
```
That's not really what we want. We would like an array for each variable. There is a "unpack" option for loadtxt that will let us assign each of the 6 columns separately.

```python
x, y, z, quality, satellites, hdop = np.loadtxt('2011-10-11.dat', unpack=True)
```

```python
average(x)
```

#-70.9395601490675187
That's not really what we want. We would like an array for each variable. There is a "unpack" option for loadtxt that will let us assign each of the 6 columns separately.

```python
x, y, z, quality, satellites, hdop = np.loadtxt('2011-10-11.dat', unpack=True)
```

# Average of x
```
average(x)
```

# Minimum of x
```
min(x)
```
That's not really what we want. We would like an array for each variable. There is a "unpack" option for loadtxt that will let us assign each of the 6 columns separately.
That's not really what we want. We would like an array for each variable. There is a "unpack" option for loadtxt that will let us assign each of the 6 columns separately.

```python
x, y, z, quality, satellites, hdop = np.loadtxt('2011-10-11.dat', unpack=True)
```

average(x)
---
- 70.939601490675187
Figure 1

-7.09394e-1
-0.00005
-0.00010
-0.00015
-0.00020
-0.00025
-0.00030
-0.00035

6 columns separately.

```python
# BEGIN_SRC python
x, y, z, quality, satellites, hdop = np.loadtxt('2011-10-11.dat', unpack=True)
average(x)
```

15-matplotlib.org  26% L63 (Org)
min(iterable[, key=func]) -> value
min(a, b, c, ...[, key=func]) -> value

With a single iterable argument, return its smallest item.
With two or more arguments, return the smallest argument.

In [14]: who
Out[14]: <function who at 0x899764c>

In [15]: %who
data hdop quality satellites x y z

In [16]: x
Out[16]: array([-70.93958333, -70.93957667, -70.93957 , ..., -70.93955167,
            -70.93955167, -70.93955167])

In [17]: plot(x)
Out[17]: [<matplotlib.lines.Line2D object at 0x9e1056c>]

In [18]: plot(y)

6 columns separately.

# BEGIN_SRC python
x,y,z,quality,satellites,hdop = np.loadtxt('2011-10-11.dat', unpack=True)
average(x)
#
-70.939601490675187
6 columns separately.

```
# BEGIN_SRC python
x,y,z,quality,satellites,hdop = np.loadtxt('2011-10-11.dat', unpack=True)
average(x)
```

---

15-matplotlib.org 26% L63 (Org)
In [20]: cla()
    
    Docstring:
    
    Clear the current axes

In [21]: plot(y)
Out[21]: [<matplotlib.lines.Line2D object at 0x91c5fcc>]

In [22]: figure(2)
Out[22]: <matplotlib.figure.Figure object at 0x91be22c>

In [23]: plot(x)
Out[23]: [<matplotlib.lines.Line2D object at 0x93a4e6c>]

In [24]: cla()
In [21]: plot(y)
Out[21]: [<matplotlib.lines.Line2D object at 0x91c5fccc>]

In [22]: figure(2)
Out[22]: <matplotlib.figure.Figure object at 0x91be22c>

In [23]: plot(x)
Out[23]: [<matplotlib.lines.Line2D object at 0x93a4e6c>]

In [24]: cla()

In [25]: plot(x,y)
Out[25]: [<matplotlib.lines.Line2D object at 0x9e1ca8c>]

In [26]: title('GPS wander for 1 day')

---

#+BEGIN_SRC python
x,y,z,quality,sate
average(x)
# -0.939601490675
---

average(y)
---
In [22]: figure(2)
Out[22]: <matplotlib.figure.Figure object at 0x91be22c>

In [23]: plot(x)
Out[23]: [<matplotlib.lines.Line2D object at 0x93a4e6c>]

In [24]: cla()

In [25]: plot(x,y)
Out[25]: [<matplotlib.lines.Line2D object at 0x9e1ca8c>]

In [26]: title('GPS wander for 1 day')
Out[26]: <matplotlib.text.Text object at 0x91be1ec>

In [27]: xlabel('Longitude')
Out[27]: <matplotlib.text.Text object at 0xe0320c>

In [28]: ylabel('Latitude')
Out[28]: <matplotlib.text.Text object at 0xe03eec>

In [29]:
Try something nicer - the lines with the average of x and y marked:

```python
# BEGIN_SRC python
cla()
plot(x, y)
# END_SRC
```
In [23]: plot(x)
Out[23]: [<matplotlib.lines.Line2D object at 0x93a4e6c>]

In [24]: cla()

In [25]: plot(x,y)
Out[25]: [<matplotlib.lines.Line2D object at 0x9e1ca8c>]

In [26]: title('GPS wander for 1 day')
Out[26]: <matplotlib.text.Text object at 0x91belec>

In [27]: xlabel('Longitude')
Out[27]: <matplotlib.text.Text object at 0x9e0320c>

In [28]: ylabel('Latitude')
Out[28]: <matplotlib.text.Text object at 0x9e03eeb>

In [29]: plot(average(x), average(y), 'ro')
Out[29]: [<matplotlib.lines.Line2D object at 0x9e1bd8c>]

In [30]:

Try something nicer - the lines with the average of x and y marked:

#+BEGIN_SRC python
cla()
plot(x,y)

---

Try something nicer - the lines with the average of x and y marked:
Try something nicer - the lines with the average of x and y marked:

```python
#BEGIN_SRC python
cla()
plot(x,y)
```
Try something nicer - the lines with the average of x and y marked:

```python
# BEGIN_SRC python
cla()
plot(x, y)
```

---

`15-matplotlib.org` 46% L146 (Org)
```
In [34]: geod = pyproj.Geod

In [35]: geod = pyproj.Geod
```

```
performs forward and inverse geodetic, or Great Circle, computations. The forward computation (using the 'fwd' method) involves determining latitude, longitude and back azimuth of a terminus point given the latitude and longitude of an initial point, plus azimuth and distance. The inverse computation (using the 'inv' method) involves determining the forward and back azimuths and distance given the latitudes and longitudes of an initial and terminus point.
```

```
x=67131.5   y=-70.9396
```

Friday, October 21, 11
GPS wander for 1 day

...
#!/usr/bin/env python

import pyproj

def wander_list():
    geod = pyproj.Geod(ellps='WGS84')

    x, y, z, quality, satellites, hdop = loadtxt('2011-10-11.gga.dat.bz2', unpack=True)
    x
    y
    average(x)
    average(y)
    min(x)
    #?min
    who
    _ip.magic("who ")
    x
    plot(x)
    plot(y)
    cla()
    #?cla
    plot(y)
    figure(2)
    plot(x)
    cla()
    plot(x,y)
    title('GPS wander for 1 day')
    xlabel('Longitude')
    ylabel('Latitude')
    plot(average(x), average(y), 'ro')
#!/usr/bin/env python

import pyproj # as some_really_annoying_name
import numpy as np

def wander_list():
    geod = pyproj.Geod(ellps='WGS84')
    x,y,z,quality,satellites,hdop = loadtxt('2011-10-11.gga.dat.bz2', unpack=True)

    x_ave = average(x)
    y_ave = average(y)

"suck it up"
#!/usr/bin/env python
import pyproj
import numpy as np

def wander_list():
    geod = pyproj.Geod(ellps='WGS84')
    x, y, z, quality, satellites, hdop = loadtxt('2011

    x_ave = np.average(x)
    y_ave = np.average(y)
```python
#!/usr/bin/env python

import pyproj
import numpy as np

def wander_list():
    geod = pyproj.Geod(ellps='WGS84')
    x,y,z,quality,satellites,hdop = loadtxt('2011
    x_ave = np.average(x)
    y_ave = np.average(y)
```

In [42]: np.average?

In [43]: 

"suck it up"

Wrote /home/researchtools/class/15/wander.py
#!/usr/bin/env python

import pyproj
import numpy as np

def wander_list():
    geod = pyproj.Geod(ellps='WGS84')
    x, y, z, quality, satellites, hdop = loadtxt('2011-10-11.gga.dat.b22')
    x_ave = np.average(x)
    y_ave = np.average(y)

    print x_ave, y_ave

In [46]: wander.wander_list()
```python
#!/usr/bin/env python

import pyproj
import numpy as np

def wander_list():
    geod = pyproj.Geod(ellps='WGS84')
    x, y, z, quality, satellites, hdop = np.loadtxt('2011-10-11.gga.dat.b22', unpack=True)

    x_ave = np.average(x)
    y_ave = np.average(y)

    print(x_ave, y_ave)

/home/researchtools/class/15/wander.py in wander_list()
     6 def wander_list():
     7     geod = pyproj.Geod(ellps='WGS84')
 ---> 8     x, y, z, quality, satellites, hdop = np.loadtxt('2011-10-11.gga.dat.b22', unpack=True)
     9
----> 10     x_ave = np.average(x)

NameError: global name 'loadtxt' is not defined

In [49]: reload wander
Out[49]: <module 'wander' from 'wander.py'>

In [50]: wander.wander_list()
   -70.9396014907 43.135434976

In [51]:  "suck it up"
```

Friday, October 21, 11
#!/usr/bin/env python

import pyproj
import numpy as np

def wander_list():
    geod = pyproj.Geod(ellps='WGS84')
    x, y, z, quality, satellites, hdop = np.loadtxt('2011-10-11.gga.dat.bz2', unpack=True)

    x_ave = np.average(x)
    y_ave = np.average(y)

    print x_ave, y_ave

if __name__ == '__main__':
    wander_list()
```python
#!/usr/bin/env python

import pyproj
import numpy as np

def wander_list():
    geod = pyproj.Geod(ellps='WGS84')
    x, y, z, quality, satellites, hdop = np.loadtxt('2011-10-11.gga.dat.bz2', unpack=True)

    x_ave = np.average(x)
    y_ave = np.average(y)

    print(x_ave, y_ave)

    print('__name__:', __name__)
    if __name__ == '__main__':
        wander_list()
```

Friday, October 21, 11
There was a question outside of class about “glob” for finding files. I will try to do a video that talks more about working with files and directories.
#!/usr/bin/env python

import pyproj
import numpy as np

def wander_list():
    geod = pyproj.Geod(ellps='WGS84')
    x,y,z,quality,satellites,hdop = np.loadtxt('2011-10-11.gga.dat.bz2', unpack=True)

    x_ave = np.average(x)
    y_ave = np.average(y)

    print x_ave,y_ave

print '__name__:',__name__
if __name__ == '__main__':
    wander_list()