Getting Started with Python

For students unfamiliar with Python or any other data analysis tools, this documents gives a step by step instructions on how to get started.

The tools

We are going to use the following tools for data analysis

- Python: easy and versatile programming language
- Numpy: python library for fast array manipulation and linear algebra
- Matplotlib: python library for plotting a multitude of graphs

Installing

Because installing all the packages you may need in a data analysis project, dealing with their dependencies and setting up a development environment can be a lot of work, there are several distributions that pack all of this together aka SciPy stack. We are going to use the free Anaconda distribution.

1. Download and install Anaconda: http://continuum.io/downloads , Python 2.7 is fine.
2. From your start menu/screen, open the IPython Notebook and do **not** close the console window that opens.
3. Your browser should open automatically to url: http://localhost:8888/
4. Click “New Notebook”
5. Copy the matplotlib histogram demo from http://matplotlib.org/examples/statistics/histogram_demo_features.html into the Notebook.
6. Click “Run” and a new window with the plot should open.

You can now do work in this environment, save your notebooks and plot them.

Using your own editor

If you wish to use your own editor (like notepad++), you can open the “Anaconda Command Prompt” and type in “python C:\...\path_to_your_file". Note that if you are using relative paths like “wine.data” in your code, you need to launch the command from the project folder and use the full anaconda installation path or set your system PATH variable.

The basic skeleton

Pyplot is a procedural interface to matplotlib, which you can use to plot arrays the easy way.

```
import numpy as np
from matplotlib import pyplot
# Create your numpy array
x = np.random.random(10)
# Plot the array
pyplot.plot(X)
# Show the figure
pyplot.show()
```

You need to find the appropriate plotting command to give pyplot, and possibly do some array manipulation in Python to get the data to a state you want to plot.

Tips for the exercises

1. There is `numpy.loadtxt` function that will import the wine data as it is. You also need to split, aka array slice, the data into two: the labels in the first column, and features in the other columns. If you want all of the plots in the same figure, you can use `pyplot.subplot`.
2. It may be a good idea to use transparency and different colors: `pyplot.hist(…, color = 'r', alpha=0.6)` etc.
3. Paraller plot can be done with just the usual `pyplot.plot` command. Just plot each sample separately and use the label as a color.
4. The coefficient matrix can be saved with `numpy.savetxt` as so: `np.savetxt('corcoeff.txt', corcoeff, fmt='%.2f', delimiter="\t")`
5. If you found many correlations, use `pyplot.subplot` to keep them in the same figure.
6. You can find the functions to calculate e.vals and e.vecs from either scipy.linalg or sklearn.decomposition.
7. PCA visualization is just a `pyplot.scatter` plot of the data transformed into 2d.

So where to go from here?

Matplotlib has good documentation. You can start with the Pyplot Tutorial, which should take just 20 minutes. It is part of the larger Beginner's Guide. Often one can just steal needed plots from the Gallery.

If during your plotting quest you find you need to do some array manipulation, reading chapter "2. The Basics" from Numpy Tutorial, again around 20 mins, will answer most of your questions. The rest of the tutorial will answer all of them.

If you need to do some linear algebra, the references for Numpy and Scipy will be helpful. In particular take a look at these modules: numpy.linalg and scipy.linalg. Most of the time it is anything from a bad idea to an absolutely horrible idea to write your linear algebra functions.

For data analysis related decompositions, tools and models, see the package Scikit-learn. In particular you may find sklearn.decomposition useful.