Problem set #6

Release 0.1

Brian Granger, John Hunter and Fernando Pérez.

February 17, 2010

Contents

1 Descriptive statistics

The first step in any statistical analysis should be to describe, charaterize and importantly, visualize your data. The normal distribution (aka Gaussian or bell curve) lies at the heart of much of formal statistical analysis, and normal distributions have the tidy property that they are completely characterized by their mean and variance. As you may have observed in your interactions with family and friends, most of the world is not normal, and many statistical analyses are flawed by summarizing data with just the mean and standard deviation (square root of variance) and associated signficance tests (eg the T-Test) as if it were normally distributed data.

In the exercise below, we write a class to provide descriptive statistics of a data set passed into the constructor, with class methods to pretty print the results and to create a battery of standard plots which may show structure missing in a casual analysis. Many new programmers, or even experienced programmers used to a proceedural environment, are uncomfortable with the idea of classes, having heard their geekier programmer friends talk about them but not really sure what to do with them. There are many interesting things one can do with classes (aka object oriented programming) but at their heart they are a way of bundling data with methods that operate on that data. The `self` variable is special in python and is how the class refers to its own data and methods. Here is a toy example:

```
In [2]: class MyData:
   ...:     def __init__(self, x):
   ...:         self.x = x
   ...:     def sumsquare(self):
   ...:         return (self.x**2).sum()
   ...:
   ...:

In [7]: mydata = MyData(np.random.rand(100))

In [8]: mydata.sumsquare()
Out[8]: 37.557501139463412
```

For this exercise, you should write a class that can be initialized with a dataset (a 1-d sequence) and will compute and store the following statistics from the data: length, median, min, max, mean, std, var, skew, kurtosis and range (max-min).

Printing an instance of this class should print a nicely formatted text summary with the above quantities, such as:

```
Name     = ../bookdata/hsales.dat
Npts     = 275
Mean     = 52.2873
Median   = 53.0000
Min      = 24.0000
Max      = 89.0000
Range    = 65.0000
Std      = 11.9170
Skew     = 0.1801
Kurtosis = 0.0748
```

The object should also have a **plots()** method, which when called, produces a figure with a few subplots: the data itself, its histogram, an autocorrelation plot, a power spectral density plot and a spectrogram. See the figure for an example of such a plot.

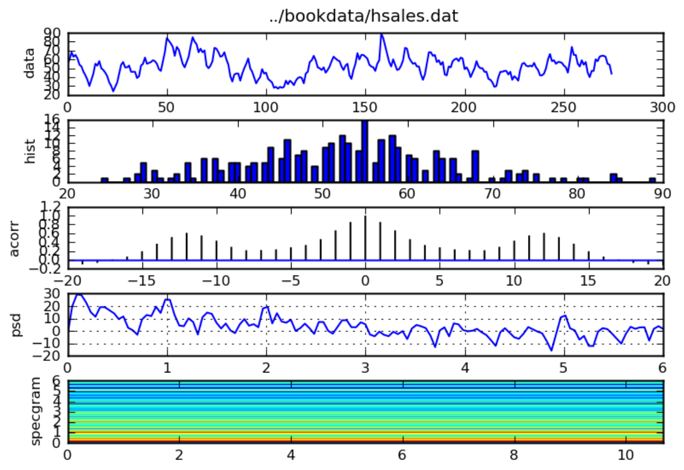

Figure 1: Simple descriptive statistics from a dataset of home sale volumes.

Hints

- You can load the data from the example files provided.

- In addition to the methods of numpy arrays, you may find the `scipy.stats` module useful for some of the required statistics.
- When you call in python `print(x)`, Python will print the output of the special `__str__()` method.
- For the necessary plots, see the `acorr()`, `psd()` and `specgram()` methods of Axis objects.