Python for Science!

PyLadies Boston Meetup #3, December 18, 2013

Adrienne Bolger
(adrienne.bolger@gmail.com)

Why SciPy?

- Free (as in software and as in beer)
- Cross platform coding
- Real time integration
  - Live data measurement and plotting
  - Robotics
- Easy as Python!

SciPy Available Tool Stack

http://scipy.org/docs.html

- Numpy- numerical data structures and symbols for use in Python
- Scipy- Signal and image processing, linear algebra, Fourier transforms, statistics
- MatPlotLib- Graphs of all kinds, GUIs, and animations
- SymPy- Symbolic math (calculus, matrices, and geometric algebra)
- Pandas- Easy data file processing from many filetypes

Why all this stuff?

- In today's scientific environment, *everybody codes*
  - MATLAB
  - Excel
  - MySQL
  - Traditional computer languages
  - Code from the Web

Boring things in the way of Science:

- Parsing files from a scientific instrument or sensor
- String parsing data exported from a database

- Resampling data to process it

- Filling in all the blank spots with zeroes

- …and the list goes on.

Getting started: your tools

- Python 2.7 + Scipy + MatPlotLib all downloaded separately and installed on your machine

  OR…

- Canopy package manager and IDE installed on your machine
- Sample Data- housing price data from Ames, Iowa
  http://www.amstat.org/publications/jse/v19n3/decock/AmesHousing.txt
- Sample Python code (we'll create this as we go)

So you want to…

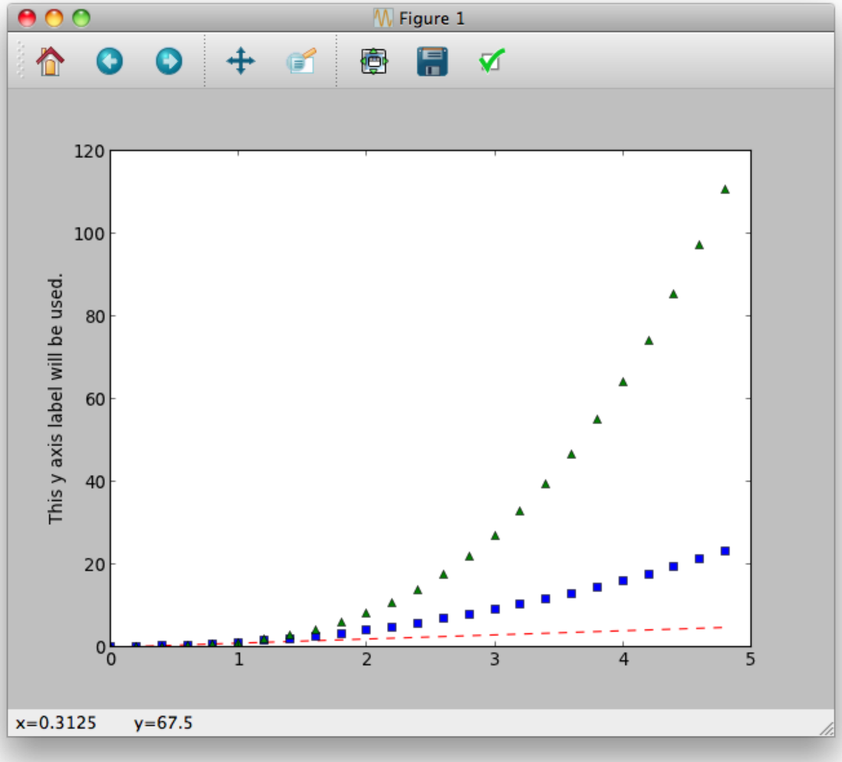
…plot data and arrays like MATLAB?

Enter Matplotlib

Hello Matplotlib!

http://matplotlib.org/users/pyplot_tutorial.html

```
import numpy

import matplotlib.pyplot as plt



#uses MatPlotLib the way you use Matlab-
as an interactive console

t = numpy.arange(0., 5., 0.2) # evenly
sampled time at 200ms intervals

plt.plot(t, t, 'r--', t, t**2, 'bs', t,
t**3, 'g^') # red dashes, blue squares
and green triangles

plt.ylabel('This y axis label will be
used.')

plt.show()



plt.ylabel('This label will NOT appear')
#isn't called until AFTER we "show" the
plot, so it's ignored
```



So you want to...
...have fast numerical computation?

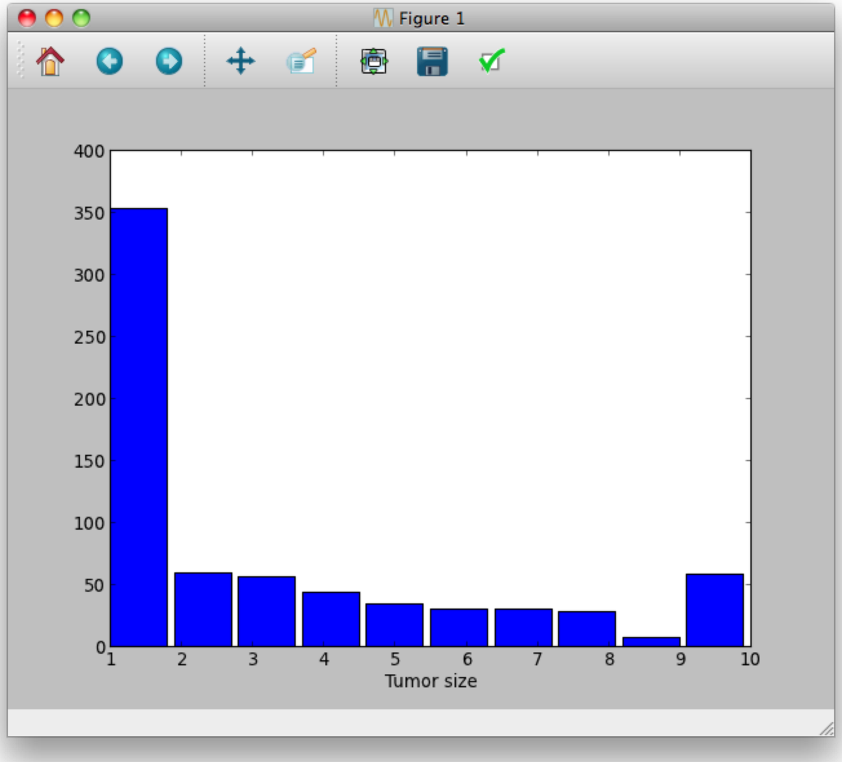
You've noticed that...
... Python lists don't replace matrices very well?

Enter Numpy.

Hello Numpy!

http://docs.scipy.org/doc/numpy/reference/generated/numpy.histogram.html

```
import numpy as np

import matplotlib.pyplot as plt

import os, sys

#Numpy has a BUILT IN function for data
loading... if all your data is numeric

present_directory =
os.path.dirname(sys.argv[0])

data = np.loadtxt(present_directory +
os.sep + 'breast-cancer-wisconsin.data',
delimiter=",", skiprows=0)

tumor_cell_size = data[:,3]

#Numpy has built in statistics functions
as well:

histogram_data, bin_edges =
np.histogram(tumor_cell_size);

plt.bar(bin_edges[:-1], histogram_data)
#Bar graph the generated histogram data

plt.xlabel("Tumor size")

plt.xlim(min(bin_edges), max(bin_edges))

plt.show()
```



So you want to …
… avoid spending all your time parsing data files?

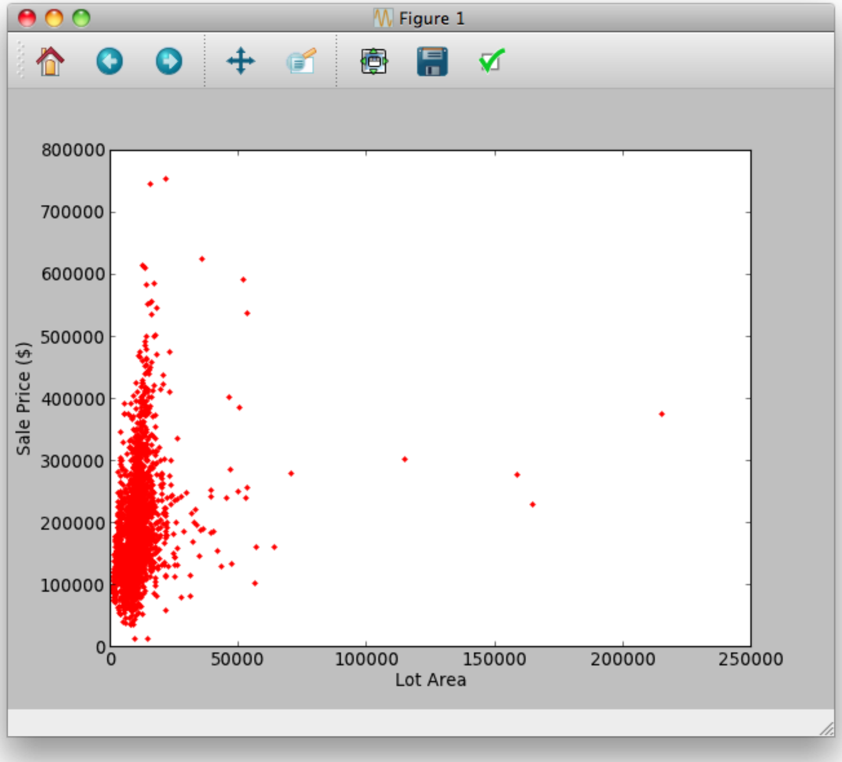
You want…
… **one line of code** to import your files into labeled rows and columns like Excel?

Enter Pandas.

Hello Pandas!

http://pandas.pydata.org/pandas-docs/dev/cookbook.html

```
import matplotlib.pyplot as plt

import pandas

import os, sys

present_directory = os.path.dirname(sys.argv[0])

Filename = present_directory + os.sep +
'ames_iowa_housing_dataset.txt'

data = pandas.read_csv( filename,  header=0,
delimiter="\t")



print "Listing available columns:", data.columns

plt.plot(data["Lot Area"], data["SalePrice"], 'r.')
#Uses MATLAB style data formatting: red dot for plotted
points

plt.xlabel("Lot Area")

plt.ylabel("Sale Price ($)")

plt.show()
```



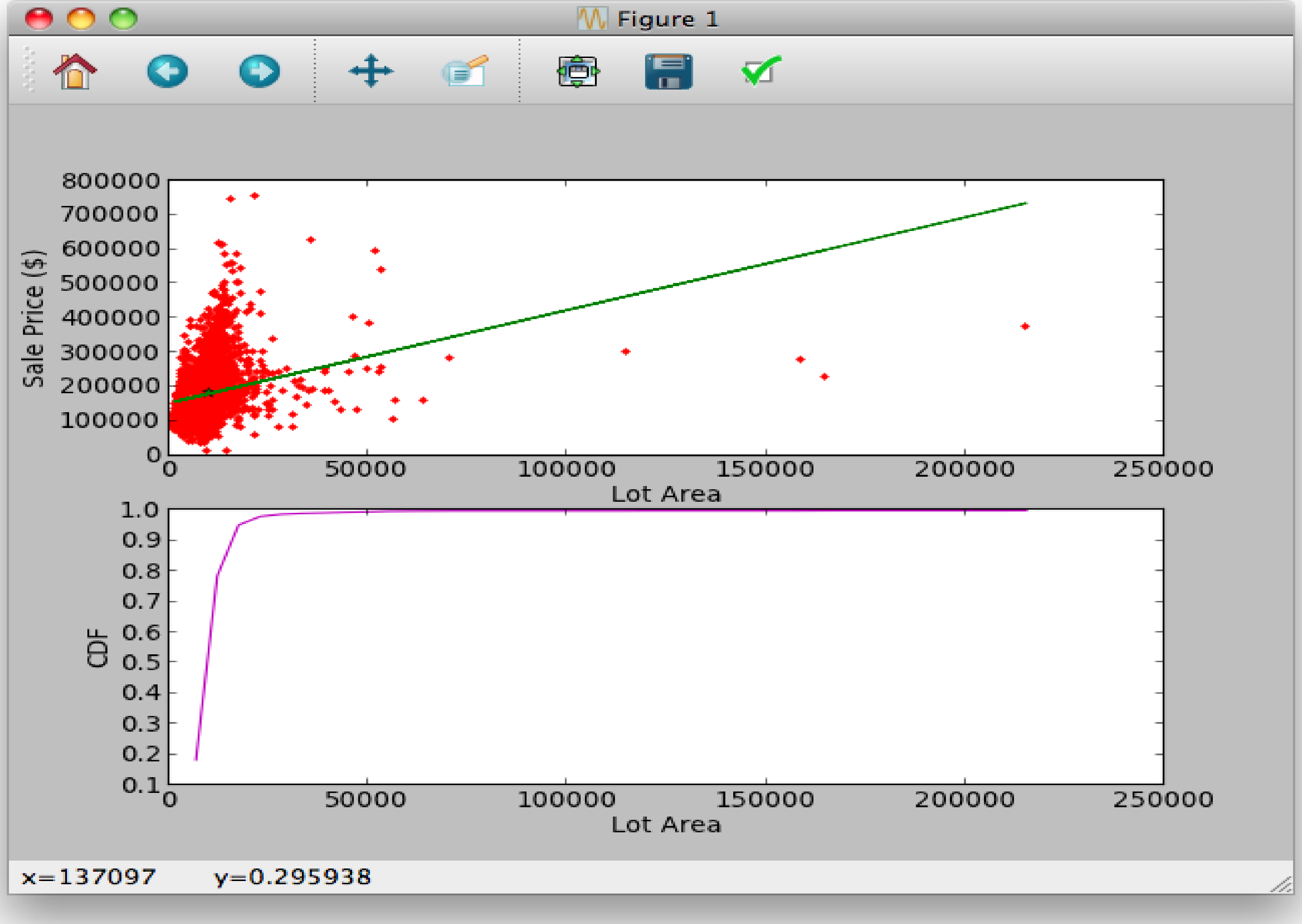
But I was promised a statistics library!

... Ok fine.

Hello statistics!

```
#Calculate some statistics about these houses:

mean_sale_price = np.mean(data["SalePrice"])

mean_lot_area = np.mean(data["Lot Area"])


#Calculate and plot a correlation line using the numpy linear regression linalg
library:

A = np.array([data["Lot Area"], np.ones(len(data["Lot Area"]))])

w = np.linalg.lstsq(A.T,data["SalePrice"])[0]

line = w[0]*data["Lot Area"] +w[1] # regression line


#CDF for the Lot Area:

n_counts, bin_edges = np.histogram(data["Lot Area"],bins=40,normed=True)

cdf = np.cumsum(n_counts)  # cdf not normalized, despite above

scale = 1.0/cdf[-1]

ncdf = scale * cdf
```

Hello statistics!

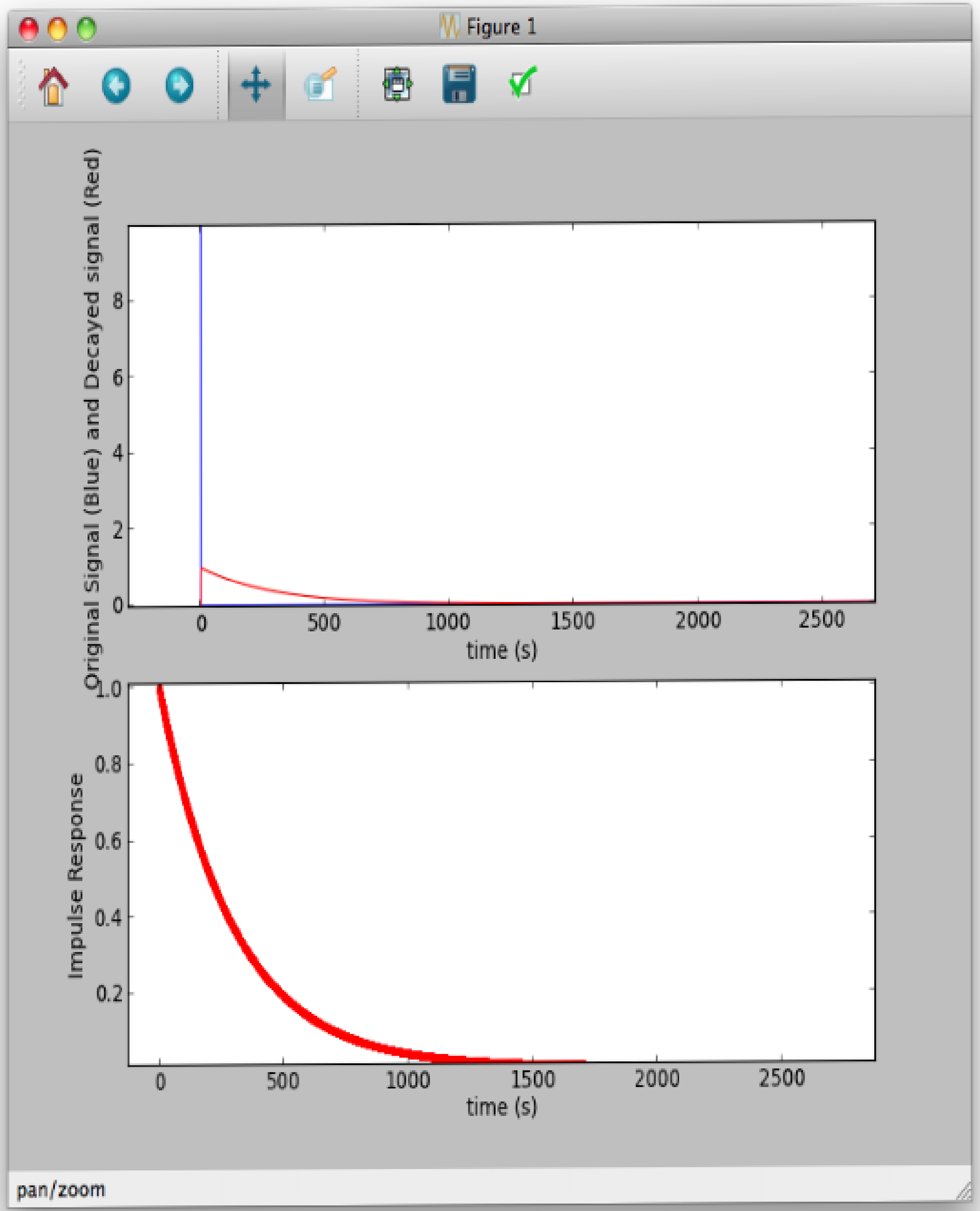


And I can't afford MATLAB's Signal Processing Toolbox!

Hello signal processing!

http://docs.scipy.org/doc/scipy/reference/signal.html

```
#...Code continued from previous slide


plt.subplot(2, 1, 1)

plt.plot(h_times, impulse_function, 'b-')

plt.plot(system_output_t, system_output, 'r-')

plt.xlabel('time (s)')

plt.ylabel('Original Signal (Blue) and Decayed
signal (Red)')


plt.subplot(2, 1, 2)

plt.plot(h_times,
impulse_response/impulse_response.max(), 'r.-')

plt.xlabel('time (s)')

plt.ylabel('Impulse Response')

#

plt.show()
```



Hello signal processing!

http://docs.scipy.org/doc/scipy/reference/signal.html

```
import numpy as np

from matplotlib import pyplot as plt

from scipy import signal



tau = 5.0*60   # 5 minutes

h_times = np.arange(0.0, 10*tau, 0.1)

impulse_function = len(h_times)*[0]

impulse_function[1] = 10

#Simple first order system model. Example =  X = Xo * e^(-t/T)

#Physical examples: Voltage in a capacitor in an RC circuit, temperature of a cooling off pot of water

sys = signal.lti(1,[1, 1/tau]) #Signal with a 1/t decay, like a swinging door

#2 ways to simulate the same basic dynamic

#1. we can  simulate an input and plot the output (more general cases)

system_output_t, system_output, system_output_state_vector = signal.lsim(sys, impulse_function, h_times)

#2. Or, we can use the LTI class to just ask for an impulse response

impulse_response = sys.impulse(T=h_times)[1]
```