HPC Python Workshop: MatPlotLib

Dr. Glen R. Jenness

Catalysis Center for Energy Innovation
University of Delaware

October 22, 2015

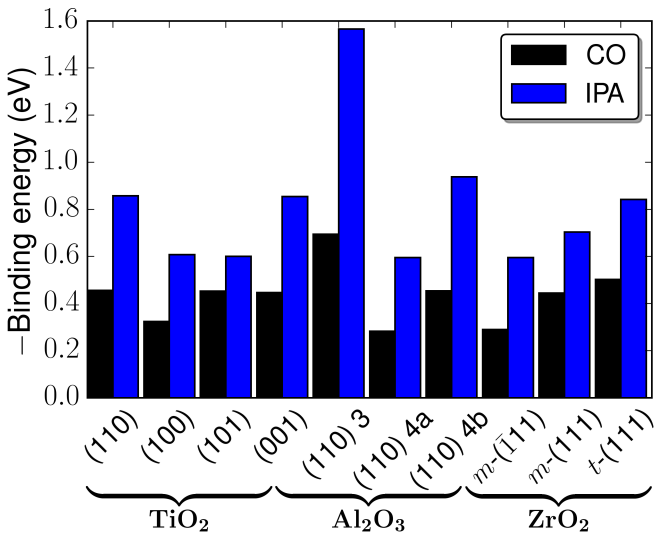
CCEI is an Energy Frontier Research Center funded by the U.S. Department of Energy, Office of Basic Energy Sciences

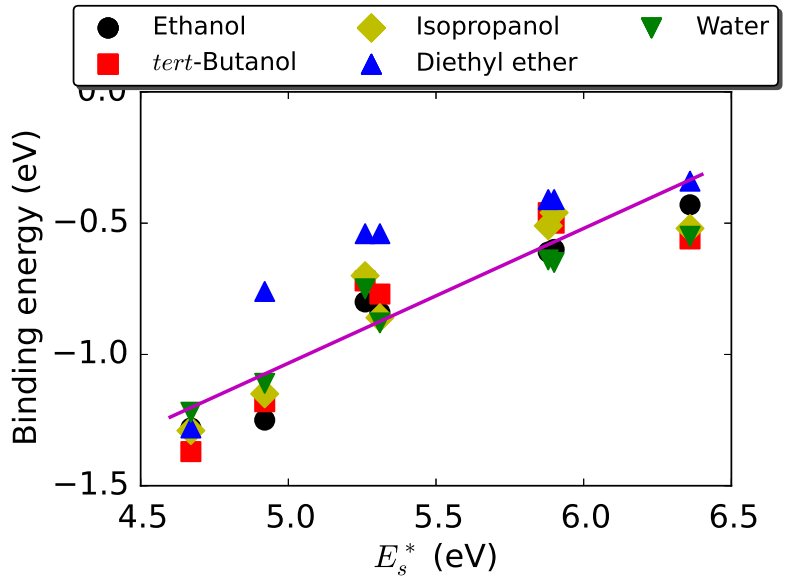
Introduction

What is MatPlotLib?

matplotlib

- Python plotting environment that integrates directly with NumPy and SciPy
- Designed to be cross platform, flexible, and capable of producing publication ready figures





Why MatPlotLib?

Advantages

- Cross platform
- Open Source (FREE!)
- High quality graphics production
- High reproducibility
- Interface with NumPy/SciPy allows for easy manipulation of data
- Variety of chart types supported (both 2D and 3D)

Disadvantages

- Intimidating interface
- Multiple ways to achieve a singular goal
- Not readily transparent

Why MatPlotLib?

Advantages

- Cross platform
- Open Source (FREE!)
- High quality graphics production
- High reproducibility
- Interface with NumPy/SciPy allows for easy manipulation of data
- Variety of chart types supported (both 2D and 3D)

Disadvantages

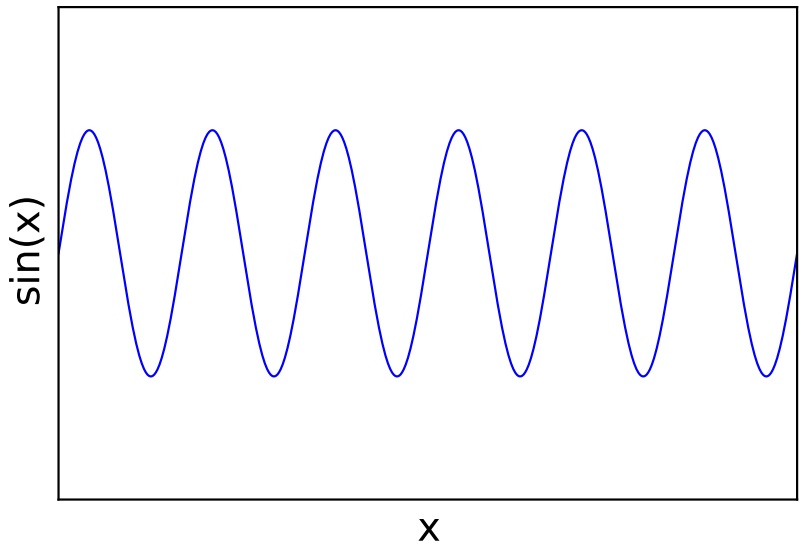
- Intimidating interface
- Multiple ways to achieve a singular goal
- Not readily transparent

If MatPlotLib is so flexible, what sort of things can it do?

EXAMPLES

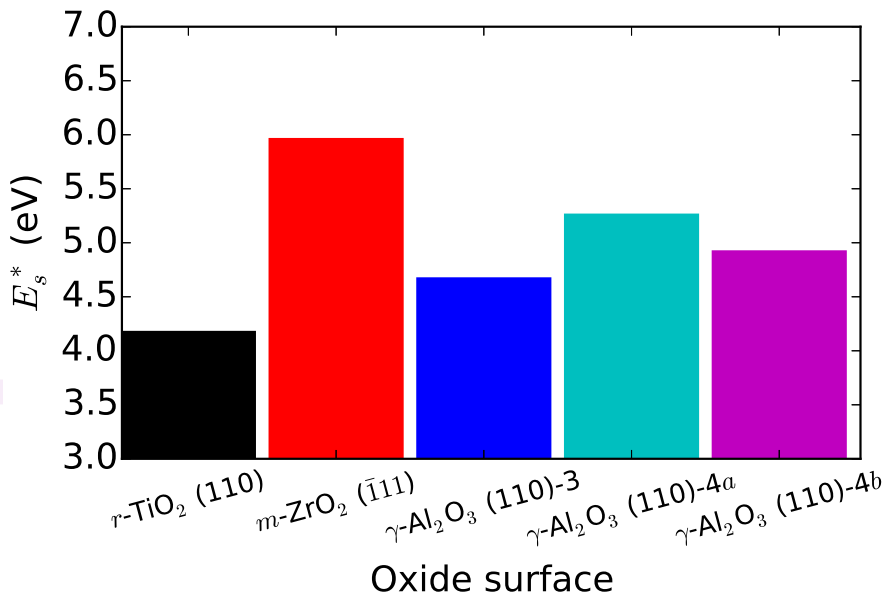
Example: Sine function

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

font = {'size': 18}
matplotlib.rc('font', **font)

x = np.arange(0, 12 * np.pi, 0.001)
y = np.sin(x)

fig = plt.figure()
ax = fig.add_subplot(111)

line = ax.plot(x, y)

plt.ylim([-2.0, 2.0])
plt.xlim([0, 12 * np.pi])

ax.set_xticks([])
ax.set_yticks([])

ax.set_xlabel(r'x')
ax.set_ylabel('sin(x)')

for ext in ['png', 'pdf', 'eps']:
    plt.savefig('sine_plot.%s' % ext, pad_inches=0.5,
                bbox_inches='tight', dpi=500)
```



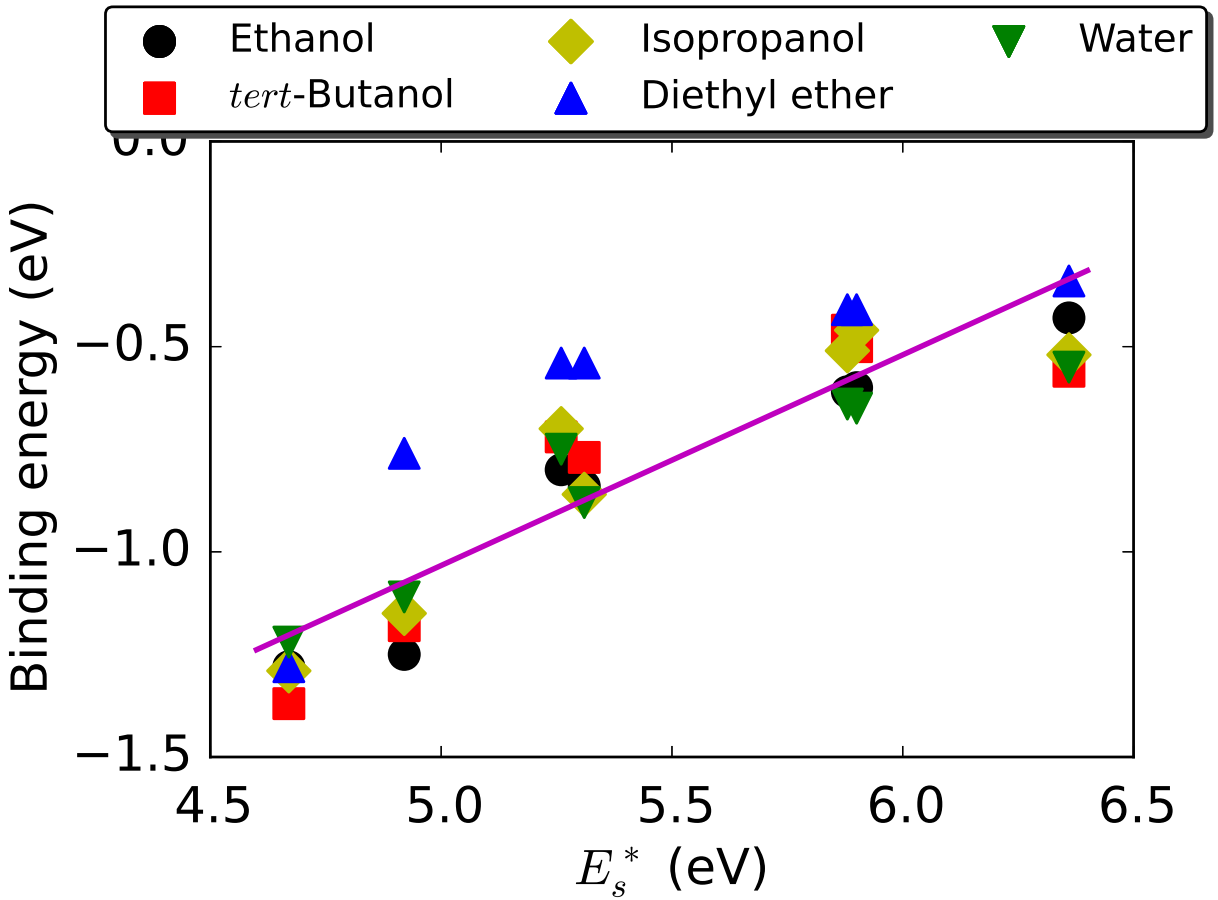
Example: Simple Bar Chart

```
 8 import numpy as np
 9 import matplotlib
10 import matplotlib.pyplot as plt
11
12 font = {'size': 16}
13 matplotlib.rc('font', **font)
14
15 sites = [r'$r$-TiO$_2$ (110)', r'$m$-ZrO$_2$ ($\bar{1}11$)',
16          r'$\gamma$-Al$_2$O$_3$ (110)-3', r'$\gamma$-Al$_2$O$_3$ (110)-4$a$',
17          r'$\gamma$-Al$_2$O$_3$ (110)-4$b$']
18
19 estars = [4.174374, 5.960088, 4.67, 5.26, 4.92]
20 colors = ['k', 'r', 'b', 'c', 'm', 'g', 'y']
21
22 index = np.arange(len(estars))
23 bar_width = 0.90
24
25 fig = plt.figure()
26 ax = fig.add_subplot(111)
27
28 bar1 = ax.bar(index, estars, width=bar_width)
29 for i in range(len(sites)):
30     bar1[i].set_color(colors[i])
31
32 ax.set_xticks(index + bar_width / 2)
33 ax.set_xticklabels(sites)
34 plt.xticks(rotation=15.)
35 for tick in ax.xaxis.get_major_ticks():
36     tick.label.set_fontsize(12)
37
38 plt.ylim([3, 7])
39 ax.set_xlabel('Oxide surface')
40 ax.set_ylabel('$E^*_s$ (eV)')
41
42 plt.subplots_adjust(bottom=0.22)
43 plt.savefig('oxide_band_energies.eps')
```



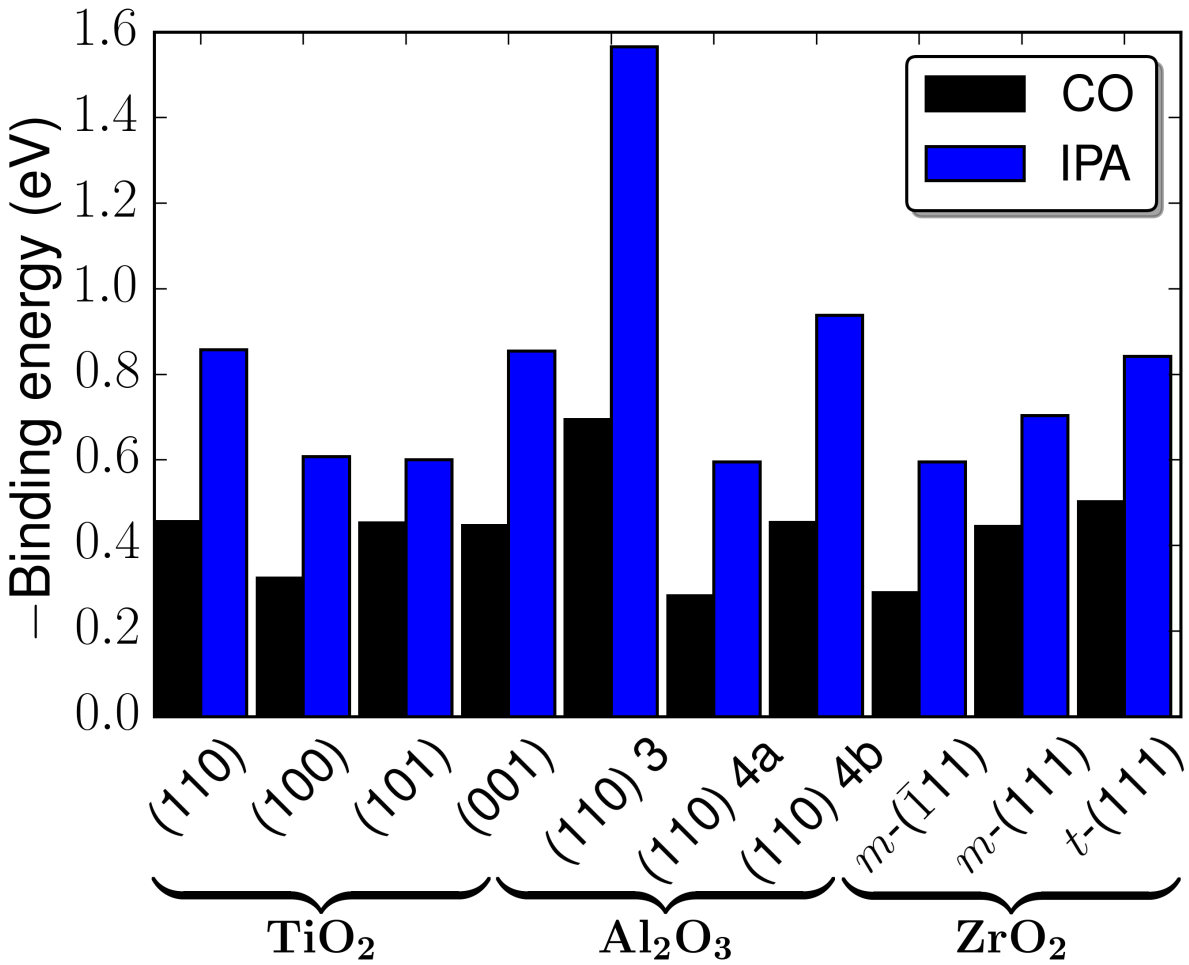
Example: Scatter with Best Fit

```
import numpy as np
from scipy.stats import linregress
import matplotlib
import matplotlib.pyplot as plt
import math

font = {'size': 18}
matplotlib.rc('font', **font)

sites = ['3', '4$a$', '4$b$', 'V$_a$', 'V$_b$', 'V$_c$', 'V$_d$']
etoh = np.array([-1.28, -0.80, -1.25, -0.84, -0.60, -0.43, -0.61])
tbutanol = np.array([-1.37, -0.72, -1.18, -0.77, -0.50, -0.56, -0.4
ipa = np.array([-1.29, -0.70, -1.15, -0.86, -0.46, -0.52, -0.51])
dee = np.array([-1.28, -0.54, -0.76, -0.54, -0.41, -0.34, -0.41])
h2o = np.array([-1.22, -0.75, -1.11, -0.88, -0.65, -0.55, -0.64])

index = np.array([4.67, 5.26, 4.92, 5.31, 5.90, 6.36, 5.88])

fig = plt.figure()
ax = fig.add_subplot(111)

size = 120
scatplot = ax.scatter(index, etoh, color='k', marker='o', s=size,
                      label='Ethanol')
scatplot = ax.scatter(index, tbutanol, color='r', marker='s', s=siz
                      label='$tert$-Butanol')
scatplot = ax.scatter(index, ipa, color='y', marker='D', s=size,
                      label='Isopropanol')
scatplot = ax.scatter(index, dee, color='b', marker='^', s=size,
                      label='Diethyl ether')
scatplot = ax.scatter(index, h2o, color='g', marker='v', s=size, lal

fita = [0.52, 0.52, 0.50]
fitb = [3.7, 3.6, 3.5]

a_ave = 0.0
b_ave = 0.0
for a in fita:
    a_ave += a

for b in fitb:
    b_ave += b

a_ave /= len(fita)
b_ave /= len(fitb)

print 'Ave. alpha = %f' % a_ave
print 'Ave. beta = %f' % b_ave

def linear(x):
    return a_ave * x - b_ave

xlist = list()
ylist = list()
for i in np.arange(4.6, 6.4, 0.1):
    xlist.append(i)
    ylist.append(linear(i))

linplot = ax.plot(xlist, ylist, color='m', linewidth=2)

ax.set_xlabel('$E^*_s$ (eV)')
ax.set_ylabel('Binding energy (eV)')

ax.legend(loc='top center', prop={'size': 14}, scatterpoints=1,
          frameon=True, shadow=True,
          fancybox=True, ncol=3, bbox_to_anchor=(1.10, 1.25))
```



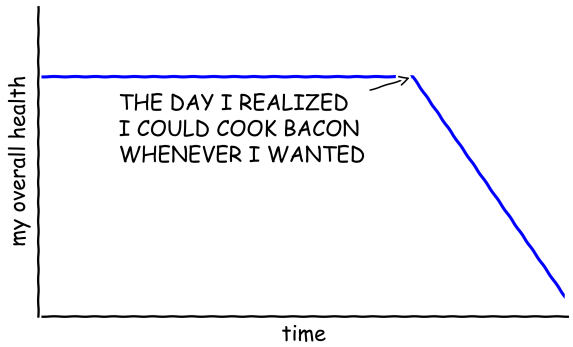
Example: LaTeX+ MatPlotLib

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

font = {'size': 18}
matplotlib.rc('font', **font)
matplotlib.rc('text', usetex=True)

latex = r'\usepackage[dvips]{graphicx}\usepackage{xfrac}'
matplotlib.rcParams["text.latex.preamble"].append(latex)

sites   = ['(110)', '(100)', '(101)', '(001)',
           '(110) 3', '(110) 4a', '(110) 4b',
           r'$m$-($\bar{1}$11)', r'$m$-(111)', r'$t$-(111)'
           ]

co_be = [-0.456298, -0.323406, -0.453106, -0.446507,
         -0.694236, -0.282624, -0.453253,
         -0.289229, -0.444831, -0.502208
         ]

ipa_be = [-0.857123, -0.607922, -0.600221, -0.854681,
          -1.565287, -0.595605, -0.937462,
          -0.595479, -0.703180, -0.841475]

co_be = np.array(co_be)
ipa_be = np.array(ipa_be)

index = np.arange(len(sites))
bar_width = 0.90

fig = plt.figure()
ax = fig.add_subplot(111)

barplot1 = ax.bar(index, -co_be, width=bar_width / 2, color='k', la
barplot2 = ax.bar(index+bar_width/2, -ipa_be, width=bar_width/2, co
                  label='IPA')

ax.set_xticks(index + bar_width / 2)
ax.set_xticklabels(sites, fontsize=16, rotation=45)

plt.ylim([0.0, 1.6])

brace = '~~~~~~~~~~~~~~~~~~'
down = -0.15
font = 22

text = r'$\underbrace{%s}_{\rm \mathbf{%s}}$' % (brace, 'TiO_2')
plt.figtext(0.25, down, text, fontsize=font, ha='center')

text = r'$\underbrace{%s}_{\rm \mathbf{%s}}$' % (brace, 'Al_2O_3')
plt.figtext(0.51, down, text, fontsize=font, ha='center')

text = r'$\underbrace{%s}_{\rm \mathbf{%s}}$' % (brace, 'ZrO_2')
plt.figtext(0.771, down, text, fontsize=font, ha='center')

ax.set_ylabel(r'$-$Binding energy (eV)')

location = 'upper right'
ax.legend(loc=location, prop={'size': 16},
          frameon=True, shadow=True,
          fancybox=True, bbox_to_anchor=(1.0, 1.0), ncol=1)

for ext in ['png', 'pdf']:
    plt.savefig('results/oxides_co_ipa_ebind.%s' % ext, pad_inches=
                bbox_inches='tight', dpi=500)
```

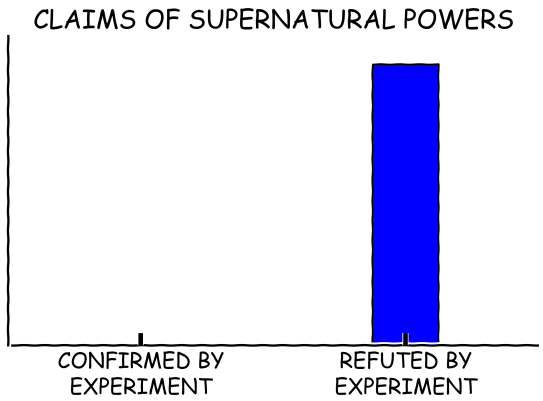


Example: XKCD

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

with plt.xkcd():
    # Based on "Stove Ownership" from XKCD by Randall Monroe
    # http://xkcd.com/418/

    fig = plt.figure()
    ax = fig.add_axes((0.1, 0.2, 0.8, 0.7))
    ax.spines['right'].set_color('none')
    ax.spines['top'].set_color('none')
    plt.xticks([])
    plt.yticks([])
    ax.set_ylim([-30, 10])

    data = np.ones(100)
    data[70:] -= np.arange(30)

    plt.annotate(
        'THE DAY I REALIZED\nI COULD COOK BACON\nWHENEVER I WANTED',
        xy=(70, 1), arrowprops=dict(arrowstyle='->'), xytext=(15, -10))

    plt.plot(data)

    plt.xlabel('time')
    plt.ylabel('my overall health')
    fig.text(
        0.5, 0.05,
        '"Stove Ownership" from xkcd by Randall Monroe',
        ha='center')

    for ext in ['png', 'pdf']:
        plt.savefig('xkcd418.%s' % ext, pad_inches=0.5,
                    bbox_inches='tight', dpi=500)
    # Based on "The Data So Far" from XKCD by Randall Monroe
    # http://xkcd.com/373/

    fig = plt.figure()
    ax = fig.add_axes((0.1, 0.2, 0.8, 0.7))
    ax.bar([-0.125, 1.0-0.125], [0, 100], 0.25)
    ax.spines['right'].set_color('none')
    ax.spines['top'].set_color('none')
    ax.xaxis.set_ticks_position('bottom')
    ax.set_xticks([0, 1])
    ax.set_xlim([-0.5, 1.5])
    ax.set_ylim([0, 110])
    ax.set_xticklabels(['CONFIRMED BY\nEXPERIMENT', 'REFUTED BY\nEXPERIMENT'])
    plt.yticks([])

    plt.title("CLAIMS OF SUPERNATURAL POWERS")

    fig.text(
        0.5, -0.05,
        '"The Data So Far" from xkcd by Randall Monroe',
        ha='center')

    for ext in ['png', 'pdf']:
        plt.savefig('xkcd373.%s' % ext, pad_inches=0.5,
                    bbox_inches='tight', dpi=500)

```



"Stove Ownership" from xkcd by Randall Monroe



"The Data So Far" from xkcd by Randall Monroe

How to get MatPlotLib

Those graphs look great! Where do I get MatPlotLib?

- Source code: http://matplotlib.org
- Linux Package installer (look for the SciPy stack!)
- Enthought Canopy (https://enthought.com/)
- Anaconda Python (https://www.continuum.io/downloads)
- If on Farber, load the following (`vpkg_require`):
  - python-numpy python-scipy python-matplotlib

USING MATPLOTLIB

Plotting a Sine function

```
1 import numpy as np
```

```
1 Import numpy
```

Plotting a Sine function

```
1 import numpy as np
2 import matplotlib
3 import matplotlib.pyplot as plt
```

1 Import `numpy`

2-3 Import `MatPlotLib`

- pylab vs. pyplot
- pylab: used for interactive plotting, MatLab like
- pyplot: used more for scripting

Plotting a Sine function

```
1 import numpy as np
2 import matplotlib
3 import matplotlib.pyplot as plt
4 font = {'size':  16}
5 matplotlib.rc('font', **font)
```

1 Import `numpy`

2-3 Import `MatPlotLib`

4-5 Change the graph font size to 16

- matplotlib.rc is used to change many of the behaviors of MatPlotLib!
- Can even use rc parameters to utilize LaTeX

Plotting a Sine function

```
1 import numpy as np
2 import matplotlib
3 import matplotlib.pyplot as plt
4 font = {'size':  16}
5 matplotlib.rc('font', **font)
6 x = np.arange(0, 12 * np.pi, 0.001)
7 y = np.sin(x)
```

1 Import `numpy`

2-3 Import `MatPlotLib`

4-5 Change the graph font size to 16

6-7 Generate some (x, y) data

Plotting a Sine function

```
1 import numpy as np
2 import matplotlib
3 import matplotlib.pyplot as plt
4 font = {'size':  16}
5 matplotlib.rc('font', **font)
6 x = np.arange(0, 12 * np.pi, 0.001)
7 y = np.sin(x)
8 fig = plt.figure()
9 ax = fig.add_subplot(111)
```

1 Import `numpy`

2-3 Import `MatPlotLib`

4-5 Change the graph font size to 16

6-7 Generate some (x, y) data

8-9 Generate the plotting environment

- `figure` is the top level graphics object
- `add_subplot` adds a plot to the `figure` object
- `111` does mean something, but ignore it for now

Plotting a Sine function

```
1  import numpy as np
2  import matplotlib
3  import matplotlib.pyplot as plt
4  font = {'size':  16}
5  matplotlib.rc('font', **font)
6  x = np.arange(0, 12 * np.pi, 0.001)
7  y = np.sin(x)
8  fig = plt.figure()
9  ax = fig.add_subplot(111)
10 line = ax.plot(x, y)
```

1 Import `numpy`

2-3 Import `MatPlotLib`

4-5 Change the graph font size to 16

6-7 Generate some (x, y) data

8-9 Generate the plotting environment

10 Add the (x, y) data to the plot!

Plotting a Sine function

```
1  import numpy as np
2  import matplotlib
3  import matplotlib.pyplot as plt
4  font = {'size':  16}
5  matplotlib.rc('font', **font)
6  x = np.arange(0, 12 * np.pi, 0.001)
7  y = np.sin(x)
8  fig = plt.figure()
9  ax = fig.add_subplot(111)
10 line = ax.plot(x, y)
11 plt.ylim([-2.0, 2.0])
12 plt.xlim([0, 12 * np.pi])
13 ax.set_xticks([])
14 ax.set_yticks([])
```

1 Import `numpy`

2-3 Import `MatPlotLib`

4-5 Change the graph font size to 16

6-7 Generate some (x, y) data

8-9 Generate the plotting environment

10 Add the (x, y) data to the plot!

11-14 Change some behavior of the axes

Plotting a Sine function

```
1  import numpy as np
2  import matplotlib
3  import matplotlib.pyplot as plt
4  font = {'size':  16}
5  matplotlib.rc('font', **font)
6  x = np.arange(0, 12 * np.pi, 0.001)
7  y = np.sin(x)
8  fig = plt.figure()
9  ax = fig.add_subplot(111)
10 line = ax.plot(x, y)
11 plt.ylim([-2.0, 2.0])
12 plt.xlim([0, 12 * np.pi])
13 ax.set_xticks([])
14 ax.set_yticks([])
15 ax.set_xlabel('x')
16 ax.set_ylabel('sin(x)')
```

- 1 Import `numpy`
- 2-3 Import `MatPlotLib`
- 4-5 Change the graph font size to 16
- 6-7 Generate some (x, y) data
- 8-9 Generate the plotting environment
- 10 Add the (x, y) data to the plot!
- 11-14 Change some behavior of the axes
- 15-16 Label the axes

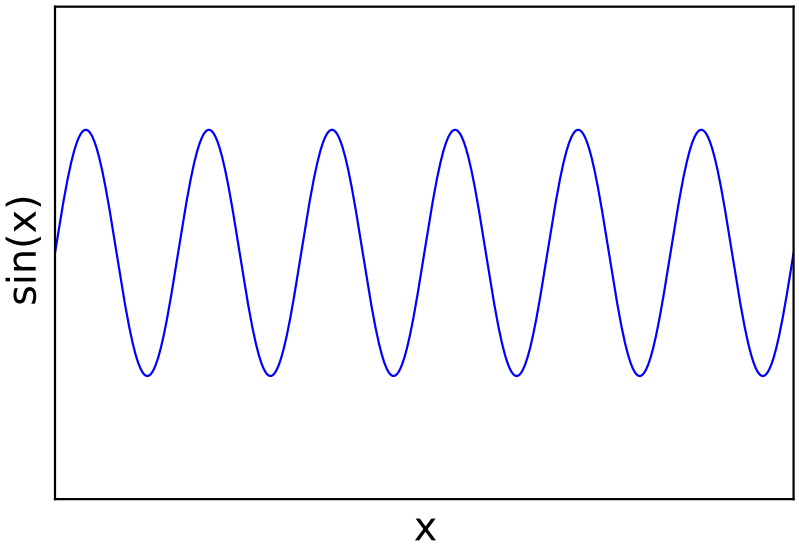
Plotting a Sine function

```
 6 x = np.arange(0, 12 * np.pi, 0.001)
 7 y = np.sin(x)
 8 fig = plt.figure()
 9 ax = fig.add_subplot(111)
10 line = ax.plot(x, y)
11 plt.ylim([-2.0, 2.0])
12 plt.xlim([0, 12 * np.pi])
13 ax.set_xticks([])
14 ax.set_yticks([])
15 ax.set_xlabel('x')
16 ax.set_ylabel('sin(x)')
17 plt.savefig('sine_plot.png')
   or
17 plt.show()
```

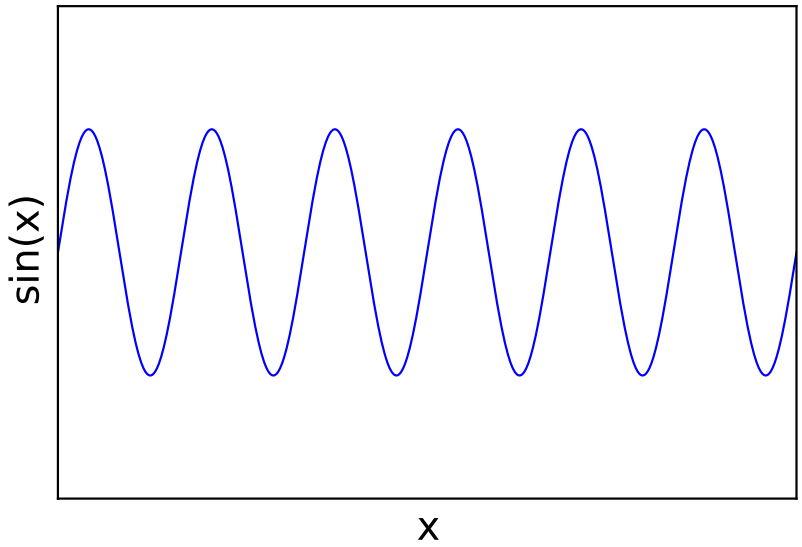
- 1 Import `numpy`
- 2-3 Import `MatPlotLib`
- 4-5 Change the graph font size to 16
- 6-7 Generate some (x, y) data
- 8-9 Generate the plotting environment
- 10 Add the (x, y) data to the plot!
- 11-14 Change some behavior of the axes
- 15-16 Label the axes
- 17 Visualize and/or save the plot

Plotting a Sine function

```
6  x = np.arange(0, 12 * np.pi, 0.001)
7  y = np.sin(x)
8  fig = plt.figure()
9  ax = fig.add_subplot(111)
10 line = ax.plot(x, y)
11 plt.ylim([-2.0, 2.0])
12 plt.xlim([0, 12 * np.pi])
13 ax.set_xticks([])
14 ax.set_yticks([])
15 ax.set_xlabel('x')
16 ax.set_ylabel('sin(x)')
17 plt.savefig('sine_plot.png')
   or
17 plt.show()
```

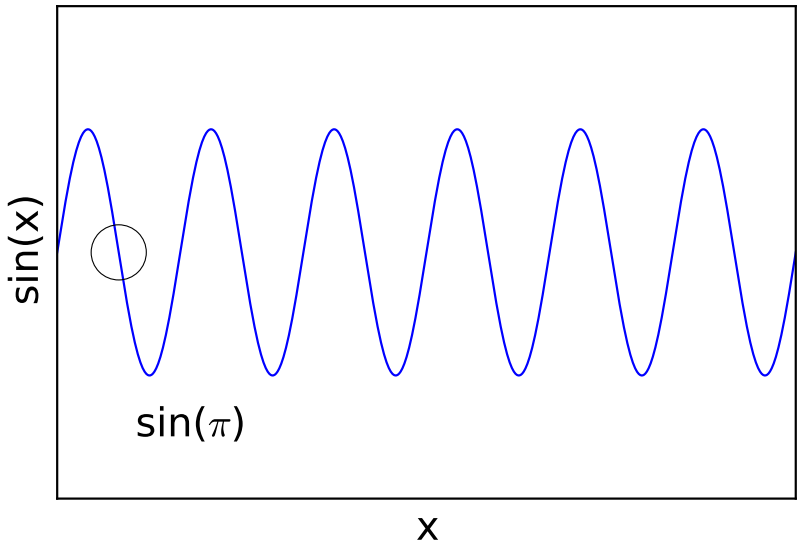


Plotting a Sine function — Modifications



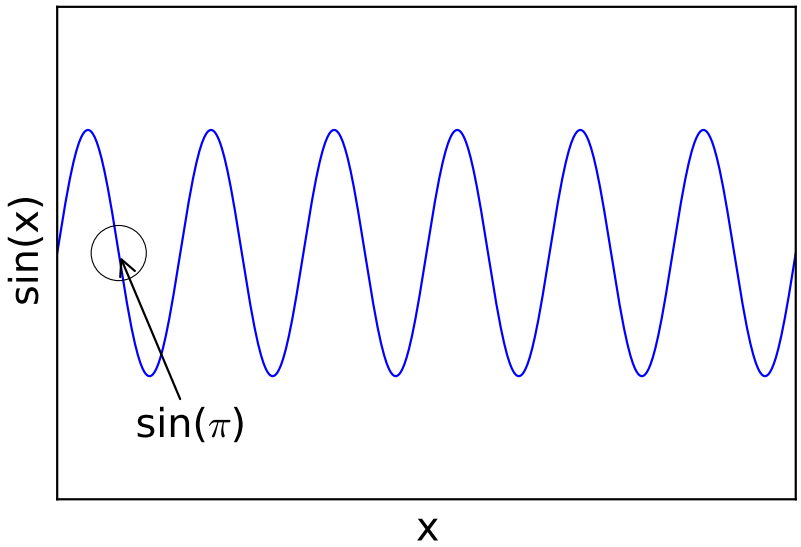
Plotting a Sine function — Modifications

- `circle = ax.plot(np.pi, 0.0, 'bo', fillstyle='none', markersize=25, color='k')`



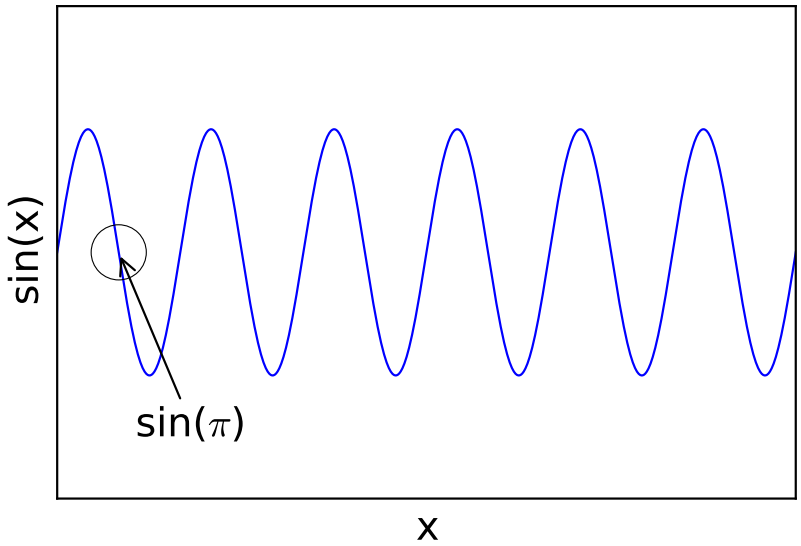
Plotting a Sine function — Modifications

- `circle = ax.plot(np.pi, 0.0, 'bo', fillstyle='none', markersize=25, color='k')`
- `ax.annotate(r'sin($\pi$)', xy=(np.pi, 0.0), xycoords='data', xytext=(4.0, -1.5), textcoords='data')`



Plotting a Sine function — Modifications

- ```
  circle = ax.plot(np.pi, 0.0, 'bo', fillstyle='none', markersize=25,
                   color='k')
  ```
- ```
  arrow = {'arrowstyle':  '->'}
  ax.annotate(r'sin($\pi$)', xy=(np.pi, 0.0), xycoords='data',
              xytext=(4.0, -1.5), textcoords='data',
              arrowprops=arrow)
  ```



Reading from external file

First, write the (x, y) data to a file to have something to work with:

```
with open('data.txt', 'w') as dfile:
  for i in range(len(x)):
     dfile.write('%16.6f%16.6f\n' % (x[i], y[i]))
```

Now read this data in using numpy:

```
x, y = np.loadtxt('data.txt', unpack=True)
```

Fitting data

```
1 import numpy as np
2 import matplotlib
3 import matplotlib.pyplot as plt
4 from scipy.optimize import curve_fit
```

1-4 Load the requisite namespaces

Fitting data

```
4 def func(x, a, b, c):
5     return a * np.exp(-b * x) + c
6 xdata = np.linspace(0, 4, 100)
7 y = func(xdata, 2.5, 1.3, 0.5)
8 ydata = y + 0.2 *
  np.random.normal(size=len(xdata))
```

4-5 Define a function to create data

6-8 Generate some data. The y data is generated via a normal Gaussian distribution for some extra randomness

Fitting data

```
4 def func(x, a, b, c):
5     return a * np.exp(-b * x) + c
6 xdata = np.linspace(0, 4, 100)
7 y = func(xdata, 2.5, 1.3, 0.5)
8 ydata = y + 0.2 *
  np.random.normal(size=len(xdata))
9 popt, pcov = curve_fit(func, xdata, ydata)
```

9 Use `scipy`'s `curve_fit` to fit the data. The parameters are in `popt`, and error information is in `pconv`

Fitting data

```
4  def func(x, a, b, c):
5      return a * np.exp(-b * x) + c
6  xdata = np.linspace(0, 4, 100)
7  y = func(xdata, 2.5, 1.3, 0.5)
8  ydata = y + 0.2 *
   np.random.normal(size=len(xdata))
9  popt, pcov = curve_fit(func, xdata, ydata)
10 ydata2 = func(xdata, popt[0], popt[1], popt[2])
```

9 Use `scipy`'s `curve_fit` to fit the data. The parameters are in `popt`, and error information is in `pconv`

10 `popt` is then used to generate a 2^{nd} set of y-data

Fitting data

```
4  def func(x, a, b, c):
5      return a * np.exp(-b * x) + c
6  xdata = np.linspace(0, 4, 100)
7  y = func(xdata, 2.5, 1.3, 0.5)
8  ydata = y + 0.2 *
   np.random.normal(size=len(xdata))
9  popt, pcov = curve_fit(func, xdata, ydata)
10 ydata2 = func(xdata, popt[0], popt[1], popt[2])

11 fig = plt.figure()
12 ax = fig.add_subplot(111)
```

9 Use `scipy`'s `curve_fit` to fit the data. The parameters are in `popt`, and error information is in `pconv`

10 `popt` is then used to generate a 2^{nd} set of y-data

11-12 Create the plotting environment

Fitting data

```
4  def func(x, a, b, c):
5      return a * np.exp(-b * x) + c
6  xdata = np.linspace(0, 4, 100)
7  y = func(xdata, 2.5, 1.3, 0.5)
8  ydata = y + 0.2 *
   np.random.normal(size=len(xdata))
9  popt, pcov = curve_fit(func, xdata, ydata)
10 ydata2 = func(xdata, popt[0], popt[1], popt[2])

11 fig = plt.figure()
12 ax = fig.add_subplot(111)
13 ax.scatter(xdata, ydata,)
14 ax.plot(xdata, ydata2, linewidth=2, color='r')
```

9 Use `scipy`'s `curve_fit` to fit the data. The parameters are in `popt`, and error information is in `pconv`

10 `popt` is then used to generate a 2^{nd} set of y-data

11-12 Create the plotting environment

13-14 Plot the data. The random data is plotted as a scatter plot.

Fitting data

```
4  def func(x, a, b, c):
5      return a * np.exp(-b * x) + c
6  xdata = np.linspace(0, 4, 100)
7  y = func(xdata, 2.5, 1.3, 0.5)
8  ydata = y + 0.2 *
   np.random.normal(size=len(xdata))
9  popt, pcov = curve_fit(func, xdata, ydata)
10 ydata2 = func(xdata, popt[0], popt[1], popt[2])

11 fig = plt.figure()
12 ax = fig.add_subplot(111)
13 ax.scatter(xdata, ydata,)
14 ax.plot(xdata, ydata2, linewidth=2, color='r')
15 plt.xlim([0, 4])
16 plt.ylim([0, 3.0])
17 plt.savefig('fit_test.eps')
```

9 Use `scipy`'s `curve_fit` to fit the data. The parameters are in `popt`, and error information is in `pconv`

10 `popt` is then used to generate a 2^{nd} set of y-data

11-12 Create the plotting environment

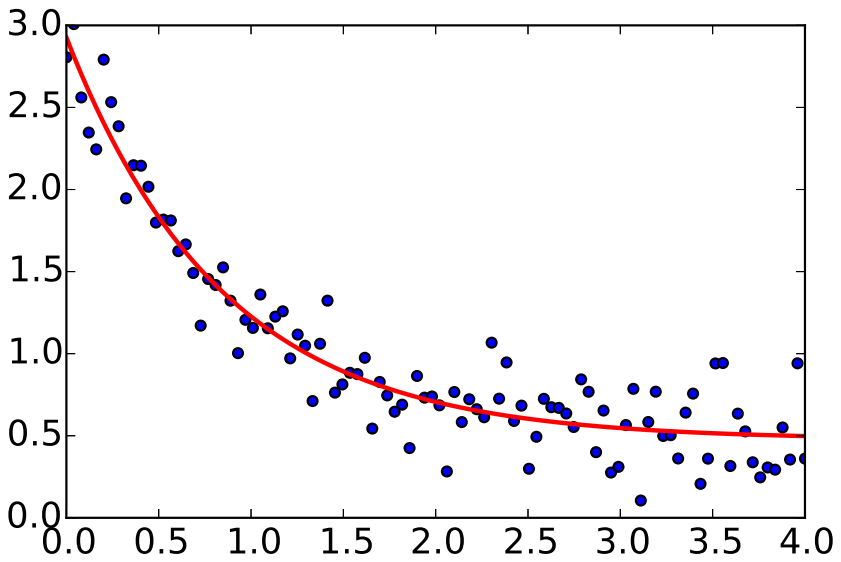
13-14 Plot the data. The random data is plotted as a scatter plot.

15-16 Set axes limits

17 Save the plot

Fitting data

```
4  def func(x, a, b, c):
5      return a * np.exp(-b * x) + c
6  xdata = np.linspace(0, 4, 100)
7  y = func(xdata, 2.5, 1.3, 0.5)
8  ydata = y + 0.2 *
   np.random.normal(size=len(xdata))
9  popt, pcov = curve_fit(func, xdata, ydata)
10 ydata2 = func(xdata, popt[0], popt[1], popt[2])

11 fig = plt.figure()
12 ax = fig.add_subplot(111)
13 ax.scatter(xdata, ydata,)
14 ax.plot(xdata, ydata2, linewidth=2, color='r')
15 plt.xlim([0, 4])
16 plt.ylim([0, 3.0])
17 plt.savefig('fit_test.eps')
```



Bar Plot

```
1 import numpy as np
2 import matplotlib
3 import matplotlib.pyplot as plt
4 font = {'size': 16}
5 matplotlib.rc('font', **font)
6 colors = ['k', 'r', 'b', 'c', 'm']
```

1-6 Load up the requisite namespace, set font, and pick some colors

Bar Plot

```
 7 sites = [r'$r$-TiO$_2$ (110)',
 8          r'$m$-ZrO$_2$ ($\bar111$)',
 9          r'$\gamma$-Al$_2$O$_3$ (110)-3',
10          r'$\gamma$-Al$_2$O$_3$
   (110)-4$a$',
11          r'$\gamma$-Al$_2$O$_3$
   (110)-4$b$']
10 index = np.arange(len(estars))
11 estars = [4.174374, 5.960088, 4.67, 5.26, 4.92]
12 bar_width = 0.90
```

7-12 Define some data. Here, we define two datasets for the x-coordinate. Also define a "width" of each bar.

Bar Plot

```
13 fig = plt.figure()
14 ax = fig.add_subplot(111)
```

13-14 Define a `figure` and `axes` objects

Bar Plot

```
13 fig = plt.figure()
14 ax = fig.add_subplot(111)
15 bar1 = ax.bar(index, estars, width=bar_width)
16 for i in range(len(sites)):
17       bar1[i].set_color(colors[i])
```

15-17 Add bars to the plot. Want each bar to be a different color, so iterate through them.

Bar Plot

```
13 fig = plt.figure()
14 ax = fig.add_subplot(111)
15 bar1 = ax.bar(index, estars, width=bar_width)
16 for i in range(len(sites)):
17     bar1[i].set_color(colors[i])
18 ax.set_xticks(index + bar_width / 2)
19 ax.set_xticklabels(sites)
20 plt.xticks(rotation=15.)
21 for tick in ax.xaxis.get_major_ticks():
22     tick.label.set_fontsize(12)
23 plt.ylim([3, 7])
24 ax.set_xlabel('Oxide surface')
25 ax.set_ylabel('$E^*_s$ (eV)')
```

18-19 Default is that each bar is labeled via a number. Want a text label, and these lines enable this.

20-22 Rotate the labels by 15°, and change the font size on the labels.

23-25 Set a limit to y axis, and label both axes

Bar Plot

```
13 fig = plt.figure()
14 ax = fig.add_subplot(111)
15 bar1 = ax.bar(index, estars, width=bar_width)
16 for i in range(len(sites)):
17     bar1[i].set_color(colors[i])
18 ax.set_xticks(index + bar_width / 2)
19 ax.set_xticklabels(sites)
20 plt.xticks(rotation=15.)
21 for tick in ax.xaxis.get_major_ticks():
22     tick.label.set_fontsize(12)
23 plt.ylim([3, 7])
24 ax.set_xlabel('Oxide surface')
25 ax.set_ylabel('$E^*_s$ (eV)')
26 plt.subplots_adjust(bottom=0.22)
27 plt.savefig('oxide_band_energies.eps')
```

18-19 Default is that each bar is labeled via a number. Want a text label, and these lines enable this.

20-22 Rotate the labels by 15°, and change the font size on the labels.

23-25 Set a limit to y axis, and label both axes

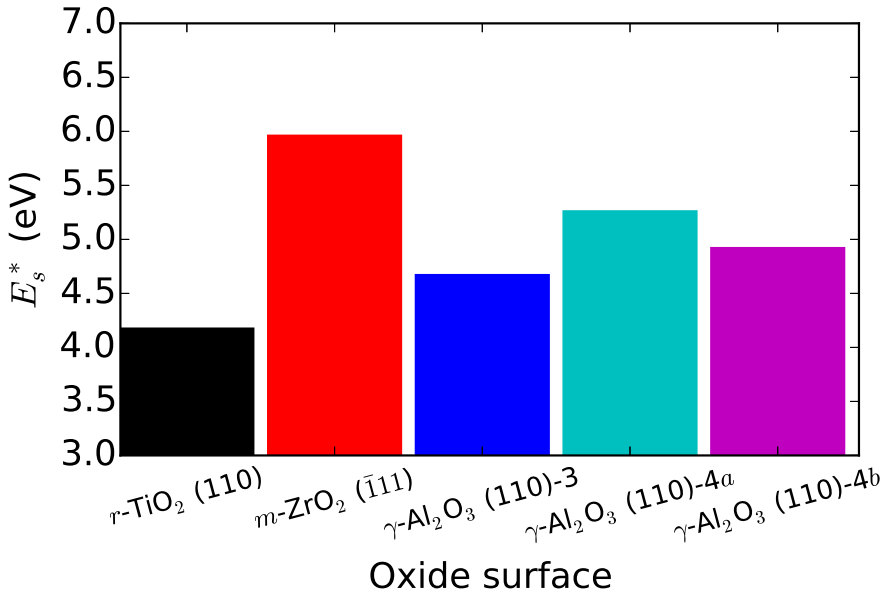
26 Pad out the bottom

27 Save the figure

- The following can be added to the `savefig` function to avoid the padding: pad_inches=0.5, bbox_inches='tight'

Bar Plot

```
13 fig = plt.figure()
14 ax = fig.add_subplot(111)
15 bar1 = ax.bar(index, estars, width=bar_width)
16 for i in range(len(sites)):
17     bar1[i].set_color(colors[i])
18 ax.set_xticks(index + bar_width / 2)
19 ax.set_xticklabels(sites)
20 plt.xticks(rotation=15.)
21 for tick in ax.xaxis.get_major_ticks():
22     tick.label.set_fontsize(12)
23 plt.ylim([3, 7])
24 ax.set_xlabel('Oxide surface')
25 ax.set_ylabel('$E^*_s$ (eV)')
26 plt.subplots_adjust(bottom=0.22)
27 plt.savefig('oxide_band_energies.eps')
```



Subplots

11-12 Create the plotting environment

```
11 fig = plt.figure()
12 ax = fig.add_subplot(221)
```

Subplots

```
11 fig = plt.figure()
12 ax = fig.add_subplot(221)
13 ax.scatter(xdata, ydata,)
14 ax.plot(xdata, ydata2, linewidth=2, color='r')
```

11-12 Create the plotting environment

13-14 Plot the data. The random data is plotted as a scatter plot.

- subplot now takes something besides (111)
- (ncols, nrows, number)

Subplots

```
11 fig = plt.figure()
12 ax = fig.add_subplot(221)
13 ax.scatter(xdata, ydata,)
14 ax.plot(xdata, ydata2, linewidth=2, color='r')
15 plt.xlim([0, 4])
16 plt.ylim([0, 3.0])
17 ax.set_xticks([])
18 ax.set_yticks(np.arange(0, 4, 1))
```

11-12 Create the plotting environment

13-14 Plot the data. The random data is plotted as a scatter plot.

15-16 Set axes limits

17 Clear the x, set the y

Subplots

```
19 ax2 = fig.add_subplot(222)
20 ax2.scatter(xdata, ydata,)
21 ax2.plot(xdata, ydata2, linewidth=2, color='k')
22 ax2.tick_params(labelcolor='w', top='on',
   bottom='on', left='off', right='off')
```

15-16 Set axes limits

17 Clear the x, set the y

19 Second plot

20-21 Add the same data (being lazy!)

22 Clear axes labels

Subplots

```
19 ax2 = fig.add_subplot(222)
20 ax2.scatter(xdata, ydata,)
21 ax2.plot(xdata, ydata2, linewidth=2, color='k')
22 ax2.tick_params(labelcolor='w', top='on',
   bottom='on', left='off', right='off')
23 ax3 = fig.add_subplot(223)
24 ax3.scatter(xdata, ydata,)
25 ax3.plot(xdata, ydata2, linewidth=2,
   color='m')
26 plt.xlim([0, 4])
27 plt.ylim([0, 3.0])
29 ax3.set_xticks(np.arange(0, 5, 1))
30 ax3.set_yticks(np.arange(0, 4, 1))
```

15-16 Set axes limits

17 Clear the x, set the y

19 Second plot

20-21 Add the same data (being lazy!)

22 Clear axes labels

23 Third plot

24-25 Add the same data (being lazy!)

26-30 Set axes labels

Subplots

```
31 ax4 = fig.add_subplot(224)
32 ax4.scatter(xdata, ydata,)
33 ax4.plot(xdata, ydata2, linewidth=2, color='g')
35 plt.xlim([0, 4])
36 plt.ylim([0, 3.0])
37 ax4.set_xticks(np.arange(0, 5, 1))
38 ax4.set_yticks([])
```

15-16 Set axes limits

17 Clear the x, set the y

19 Second plot

20-21 Add the same data (being lazy!)

22 Clear axes labels

23 Third plot

24-25 Add the same data (being lazy!)

26-30 Set axes labels

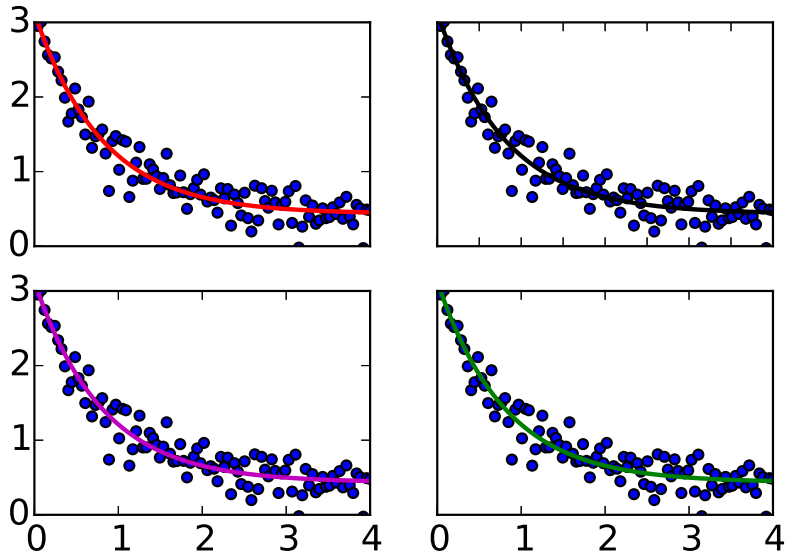
31 Fourth plot

32-33 Add the same data (being lazy!)

34-38 Set axes labels

Subplots

```
31 ax4 = fig.add_subplot(224)
32 ax4.scatter(xdata, ydata,)
33 ax4.plot(xdata, ydata2, linewidth=2, color='g')
35 plt.xlim([0, 4])
36 plt.ylim([0, 3.0])
37 ax4.set_xticks(np.arange(0, 5, 1))
38 ax4.set_yticks([])
```



QUESTIONS?