Matplotlib: python plotting

Informatics 1 for Biomedical Engineers
Tutor Session 7

KTI, Knowledge Technologies Institute

27. November 2016

Today’s Topics

1. What is matplotlib
2. Drawing lines
3. Plotting charts using numpy arrays
4. Basic plotting commands
    - Standard plot
    - Histogram
    - Pie Chart, Bar Chart
    - Scatter Plots

Student Goals

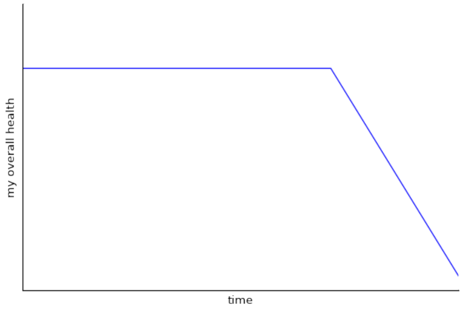
- Be able to create simple plots
- Understanding matplotlib examples (online) and adapting them [1]
- Knowing where to look up to be able to beautify your plots

[1] http://matplotlib.org/examples/

What is matplotlib?

- A plotting library for Python
- External library
    - Doesn’t come with Python’s Standard Library
    - But: Included in Anaconda, SciPy
- Open Source

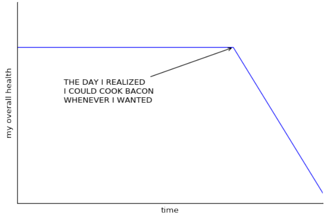
Why matplotlib?



"matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc, with just a few lines of code."

— http://matplotlib.org

Why matplotlib?

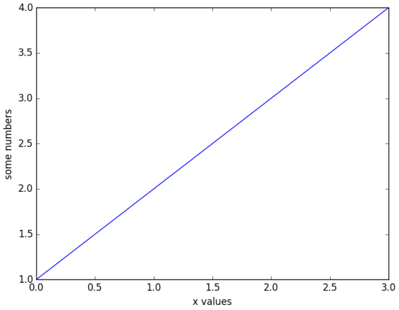


```python
plt.annotate(
  'THE␣DAY␣I␣REALIZED\nI␣COULD␣COOK..',
  xy=(70, 1),
  arrowprops=dict(arrowstyle='->'),
  xytext=(15, -10) )
```

"matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc, with just a few lines of code."

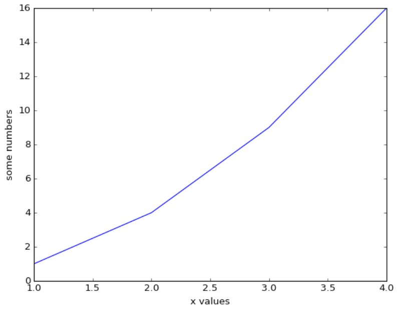
— http://matplotlib.org

Drawing our first line



```
import matplotlib.pyplot as plt

y_axis = [1,2,3,4]
plt.plot(y_axis)
plt.ylabel('some numbers')
plt.xlabel('x values')
plt.show()
```

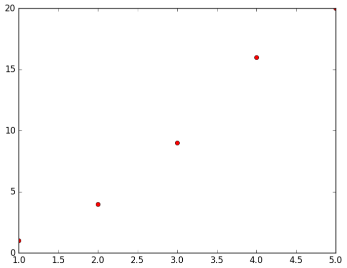
Drawing our first line



- If a single list of array is given to *plot()*, matplot automatically generates the x values for you (starting at 0.0)
- providing both axes:

```
plt.plot(
  [1, 2, 3, 4],
  [1, 4, 9, 16])
```

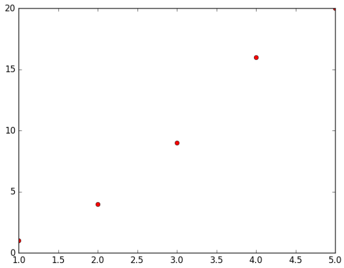
Controlling line properties



```python
plt.plot(x, y,
  marker='o', linestyle='',
  color='red')
```

- different ways to control line properties
  - keyword args
  - setter methods
  - format strings
  - setp() command

Controlling line properties



```
plt.plot(x, y,
  marker='o', linestyle='',
  color='red')
```

- different ways to control line properties
  - keyword args 👍
  - setter methods 👍
  - format strings 👎
  - setp() command

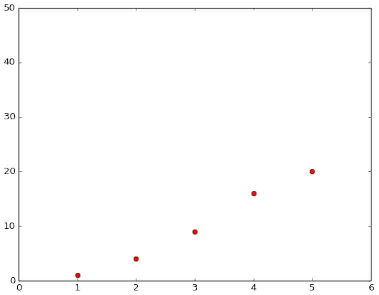
Some Line Properties

character	description
label	a label for auto legend
linestyle	solid, dashed, datted, ...
marker	marker style (e.g. point ’.’, circle ’o’)
color	red, blue, etc.
alpha	float (0.0 transparent through 1.0 opaque)
linewidth	float value in points

... and many more, see doc: [2]

[2] http://matplotlib.org/api/lines_api.html#matplotlib.lines.Line2D

Zooming in and out - axis()



- xlim() and ylim() get or set the axis limits

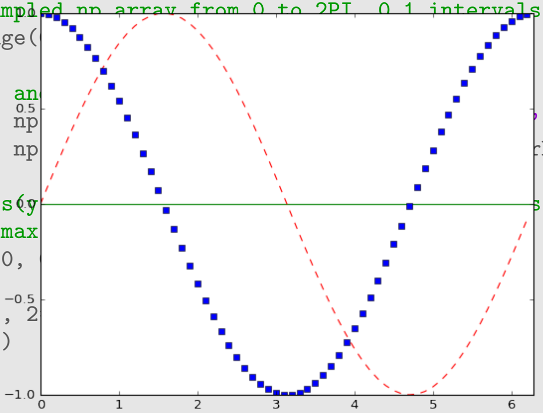
```
plt.plot(
  [1,2,3,4,5],
  [1,4,9,16,20], marker='o', ..)
plt.xlim(0, 6)
plt.ylim(0, 50)
```

Using numpy arrays

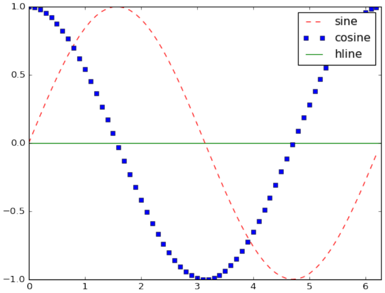
```python
# evenly sampled np array from 0 to 2PI, 0.1 intervals
t = np.arange(0., 2 * np.pi, 0.1)

# plot sine and cosine
plt.plot(t, np.sin(t), color='red', linestyle='dashed', label='sine')
plt.plot(t, np.cos(t), color='blue', linestyle='', marker='s', label='cosin

# plt.hlines(y, xmin, xmax, ..): Plot horizontal lines at each y from
# xmin to xmax.
plt.hlines(0, 0, 2*np.pi, color='green')

plt.axis([0, 2*np.pi, -1, +1])
plt.legend()
plt.show()
```

Using numpy arrays

```
# evenly sampled np array from 0 to 2PI, 0.1 intervals
t = np.arange(

# plot sine an
plt.plot(t, np                                        ', label='sine')
plt.plot(t, np                                        rker='s', label='cosin

# plt.hlines(y                                        s at each y from
# xmin to xmax
plt.hlines(0,

plt.axis([0, 2
plt.legend()
plt.show()
```



Labelling our plots



- Use the label keyword
- Call plt.legend() to uncover the label box

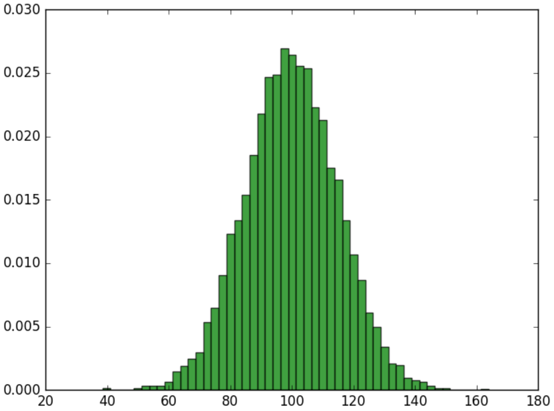
```
plt.plot(..., label='sine')
plt.plot(..., label='cosine')
plt.hlines(..., label='hline')
plt.legend()
```

Plotting a Histogram - plt.hist()

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

mu, sigma = 100, 15
x = mu + sigma * np.random.randn(10000)

plt.hist(x, 50, normed=1, facecolor='g', alpha=0.75)
plt.show()
```

Plotting

```
import n
import m

mu, sigm
x = mu +

plt.hist
plt.show
```

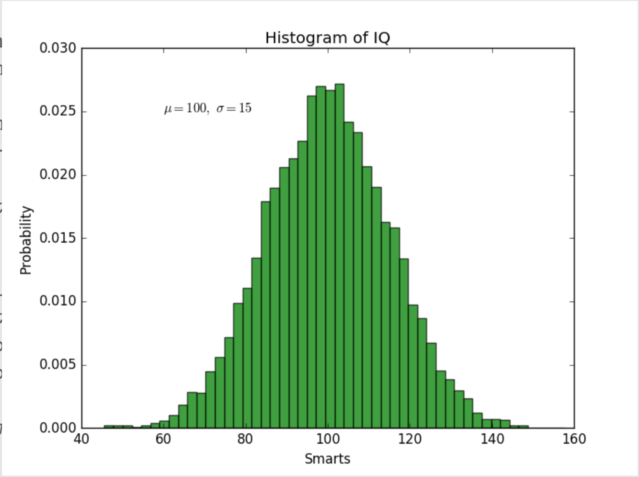


Adding text to our Histogram

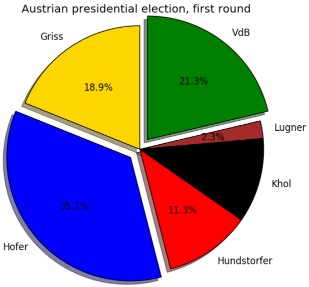
```
import numpy as np
import matplotlib.pyplot as plt

mu, sigma = 100, 15
x = mu + sigma * np.random.randn(10000)

plt.hist(x, 50, normed=1, facecolor='g', alpha=0.75)

#Adding text to our histogram:
plt.title('Histogram␣of␣IQ')
plt.text(60, .025, r'$\mu=100,\␣\sigma=15$')
plt.xlabel('Smarts')
plt.ylabel('Probability')

plt.show()
```

Adding text to our Histogram

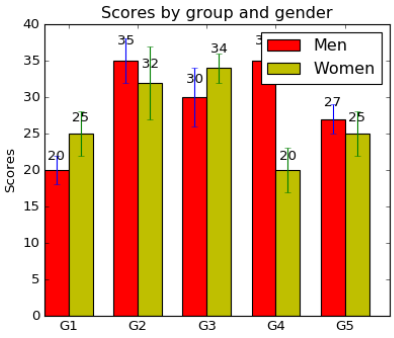
```
import n
import m

mu, sigm
x = mu +

plt.hist

#Adding
plt.titl
plt.text
plt.xlab
plt.ylab

plt.show
```



Gallery: A few more examples



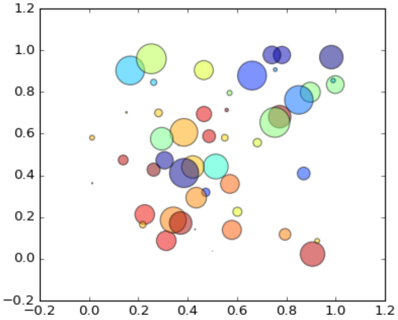
Pie Chart - plt.pie()



Bar Chart - plt.bar()

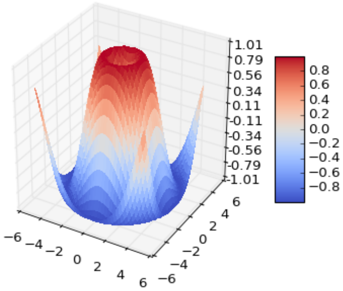
http://matplotlib.org/examples/api/barchart_demo.html

Gallery: A few more examples



Scatter plot - plt.scatter()

http://matplotlib.org/examples/shapes_and_collections/scatter_demo.html



Surface 3D - plt.plot_surface()

http://matplotlib.org/examples/mplot3d/surface3d_demo.html

Complex Example – Plotting a retirement fund

- Look back at the task of unit 5: Calculate a retirement fund
- Someone already solved the task and wrote her solution into the source file *retirement_model.py*
- We want to use the functions *calc_savings()* and *calc_retirement_account()* and plot their returned values.

```python
# importing the functions:
from retirement_model import calc_savings, calc_retirement_account
help(calc_savings) #or calc_savings?
```

Student Task

Task: Plot historical stock prices of the ATX

- Go to Google Finance and search for ATX
- Click on *Historical prices* and download the spreadsheet or try this link: [3]
- Plot the prices in column *Close*.
- Bonus: Add a *fill_between* plot - between each day's *High* and *Low* Price

[3] `https://goo.gl/acl8nh`