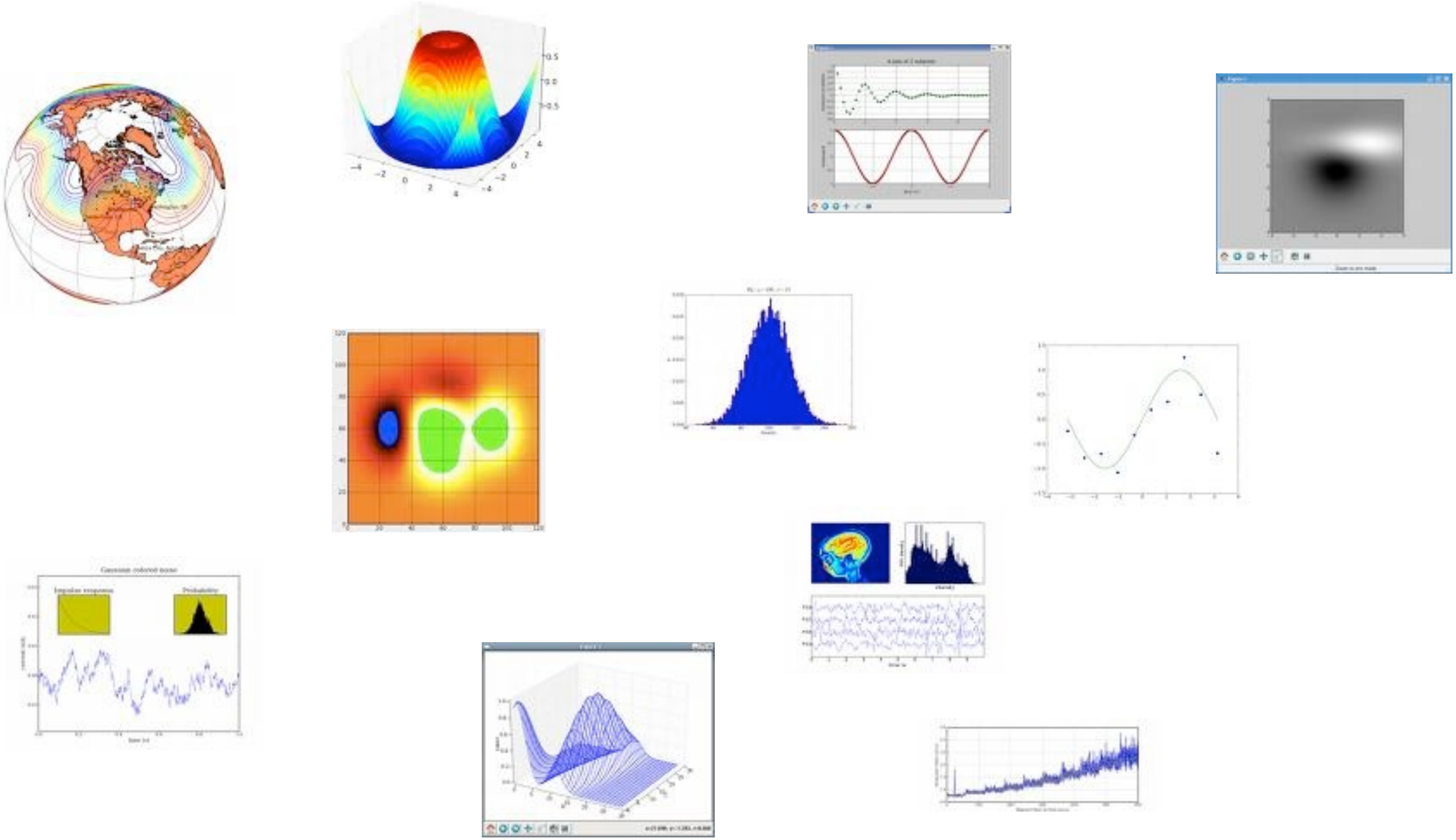
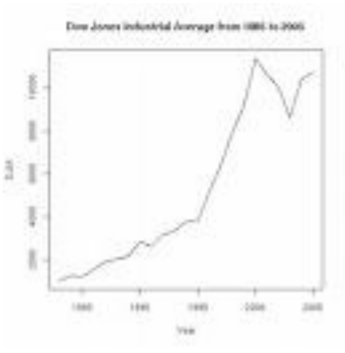


Plotting Data

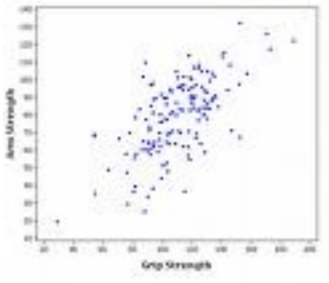
COMP 364 - Lecture 15
February 27th, 2012
Mathieu Perreault

Why plot data programmatically?

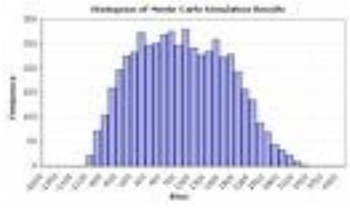
Different kinds of plots...



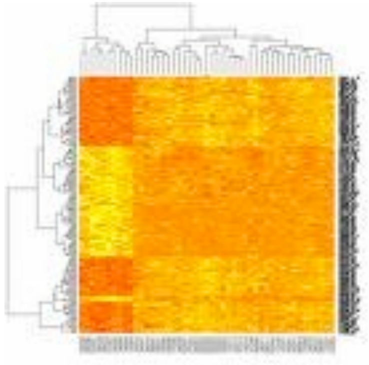
Line plot



Scatter plot



Histogram



Heatmap

Line and scatter plots

Major considerations for line/scatter plotting

- Data consists of numbers
- Each data point has an X and a Y value
  - Data is specified as two lists (X values and Y values)
- *Key issue: we read our data in as strings, but need it to be two lists of numbers.*

Manipulating lists

- x.append(y) - add the object y into list x
- x.remove(y) - remove the first occurrence of y in list x

Exercise: Consider a file containing x-y datapoints - each line has two numbers, separated by a space. Read these points from the file into two lists.

Manipulating lists

```
x = []
y = []
for line in open(‘data.txt’):
  values = line.strip(‘\n’).split()
  x.append(float(values[0]))
  y.append(float(values[1]))
```

Now our two lists contain, **in order,** the data points from the file, where y[i] is the corresponding point for x[i], for all i.

Line plots

- matplotlib is a 3rd party python library that provides MANY plotting functions (http://matplotlib.sourceforge.net)
- *matplotlib.pyplot.figure()* - creates a new blank figure
- *matplotlib.pyplot.plot(X,Y)* - draws a line plot using data points X,Y on the current figure
- *matplotlib.pyplot.show()* - displays the current figure on the screen

Exercise: extend our previous code to plot the data points in a line graph.

Stylizing our plot

- *matplotlib.pyplot.plot(X,Y,fmt)* - fmt is a string that tells matplotlib how our points should be drawn and connected.
    - plot(X,Y,’r’) - draw in red
    - plot(X,Y,’b’) - draw in blue
    - plot(X,Y,’--b’) - draw a dashed blue line
    - plot(X,Y,’g.’) - draw a scatterplot with green points
- *matplotlib.pyplot.hold(True)* - tells matplotlib to combine future plots onto the current plot (rather than replacing it)

Exercise: modify our previous script to draw a scatter plot. It also should take a threshold. All data points with a y-value > threshold should be drawn in green, otherwise blue.

Annotating a plot

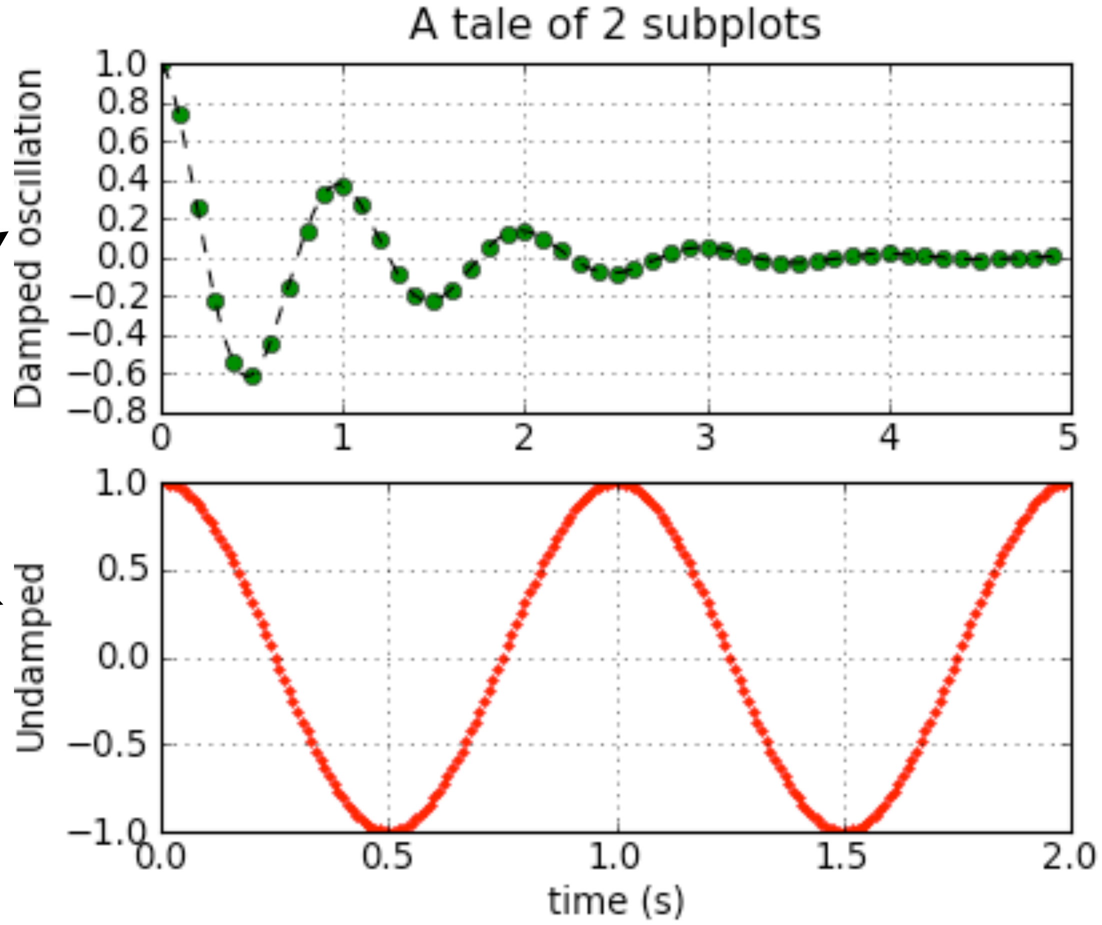
- matplotlib.pyplot.title(s) - set the title of the current plot to s
- matplotlib.pyplot.xlabel(s) - set the label of the x axis to s
- matplotlib.pyplot.ylabel(s) - set the label of the y axis to s
- matplotlib.pyplot.legend([c1,c2,...]) - draw a legend on the figure labeling each curve

Exercise: make the title of our plot the name of the data file, make a legend for the two colors.

Sub plots

matplotlib.pyplot.subplot(# rows, # cols, plot #)

matplotlib.pyplot.subplot(2,1,1)
matplotlib.pyplot.subplot(2,1,2)



Exercise: write a script that makes a figure with 2 subplots: one for sin, one for cos. (plot for x = [0,6])

Dictionaries in Python

- In order to perform more complex analysis, a new datatype is needed: dictionary.
- The “dict” type works the same as a real dictionary. It contains **key-value** mappings: you look up something by **key**, to get the associated **value**.
- Some initialization examples

```
names = {}
mydict = {“Google”: “GOOG”, “Apple”: “AAPL”}
numbers = {1:"one", 2:"two"}
frequencies = {
    “A”: 10,
    “C”:13,
    “G”: 5,
    “T”: 7
    }
```

Dictionaries in Python (2)

- Setting and getting values in a dictionary is similar to how you do with a list.

```
names = {}
names[‘COMP364’] = “Computer Tools” # Setting a value!
print names[‘COMP364’]
print names[‘COMP251’] # Will this work?
```

- Some tricks to avoid a KeyError

```
frequencies = {“A”: 10, “C”:13, “G”: 5}
print frequencies[‘T’]        # Will cause a KeyError
print frequencies.get(‘T’, 0) # Will print 0 by default
```

- Use the trick to increment values

```
frequencies[‘T’] = frequencies.get(‘T’, 0) + 1
```

Dictionaries in Python (3)

- Sometimes we are interested in just the **values,** or just the **keys,** or both at the same time (**items**)!

```
>>> frequencies = {“A”: 10, “C”:13, “G”: 5}
>>> frequencies.values()
[10, 13, 5]
>>> frequencies.keys()
[‘A’, ‘C’, ‘G’]
>>> frequencies.items()
[(‘A’,10), (‘C’,13), (‘G’, 5)]
```

- Knowing this, you can also iterate over a dictionary’s **keys, values** or **items**

```
frequencies = {“A”: 10, “C”:13, “G”: 5}
for k in frequencies.keys():    # Iterating over the keys!
  print k, frequencies[k]
```

Dictionaries in Python (4)

- Exercise: Suppose that you know your dictionary contains only **integer values**, how do you calculate the average of those values?

```
frequencies = {“A”: 10, “C”:13, “G”: 5, “T”: 20}
???
```

Dictionaries in Python (4)

- Exercise: Suppose that you know your dictionary contains only **integer values**, how do you calculate the average of those values?

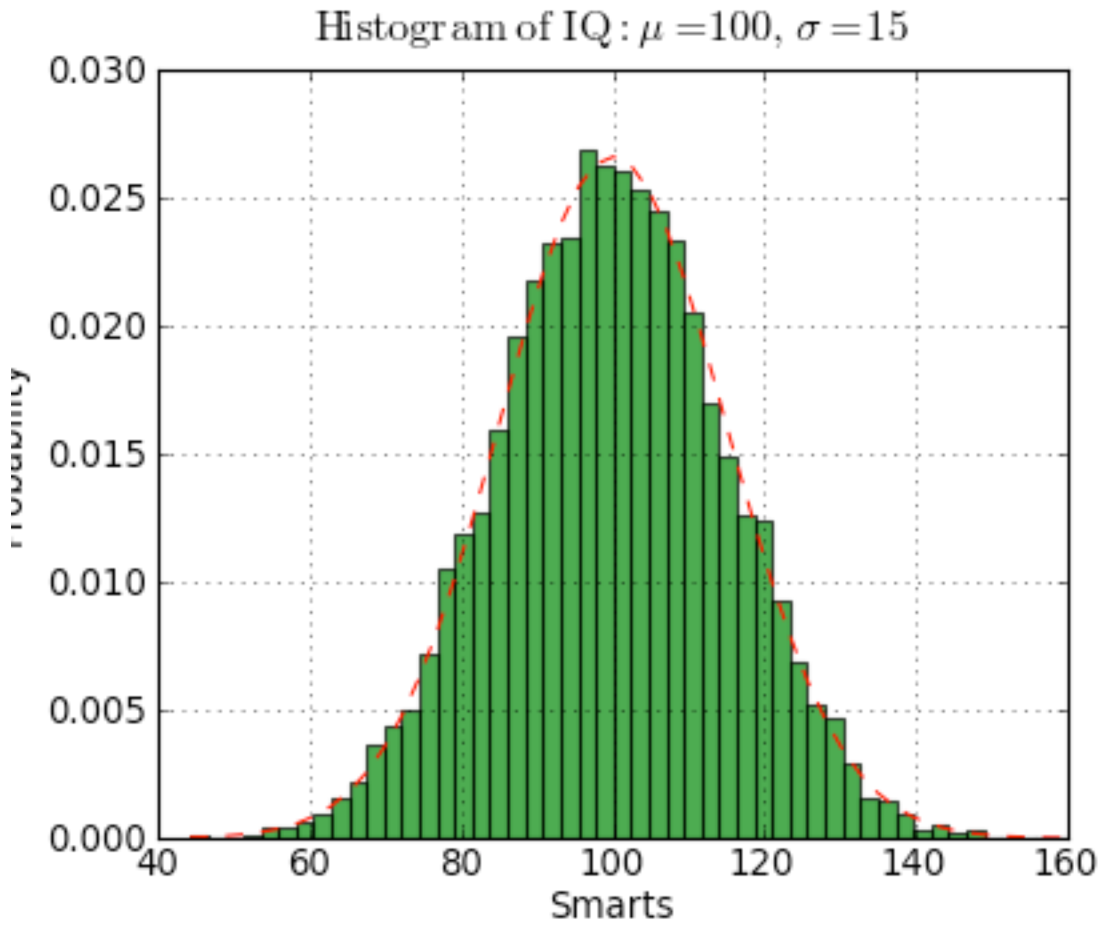
```
frequencies = {“A”: 10, “C”:13, “G”: 5, “T”: 20}
v = frequencies.values()
print sum(v)/float(len(v))
```

Histograms

hist(...)

hist(x,bins=10)

All available options:
http://matplotlib.sourceforge.net/api/pyplot_api.html#matplotlib.pyplot.hist



Exercise: plot the distribution of gene lengths in a genome file

Exercise: use subplot to plot (1) the distribution of gene lengths in a genome file and (2) the length of genes along the genome (in order)

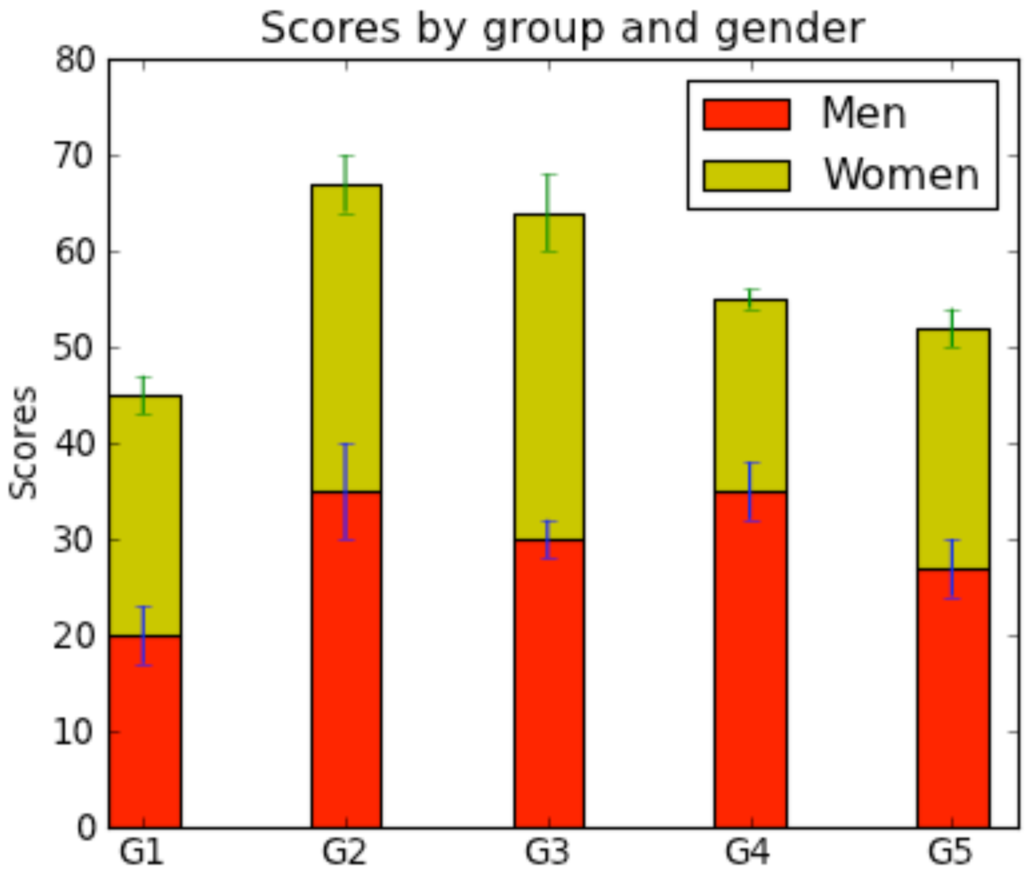
Bar Charts

bar(...)

bar(left,height)

Argument	Description
left	the x coordinates of the left sides of the bars
height	the heights of the bars

All available options:
http://matplotlib.sourceforge.net/api/pyplot_api.html#matplotlib.pyplot.bar



Bar charts are useful to print frequencies or scores. More generally, data that is categorized.

e.g. Print the frequency of each nucleotide in a file.