1.6 Plotting Functions and Histograms

May 9, 2018

1 Plotting Functions and Histograms

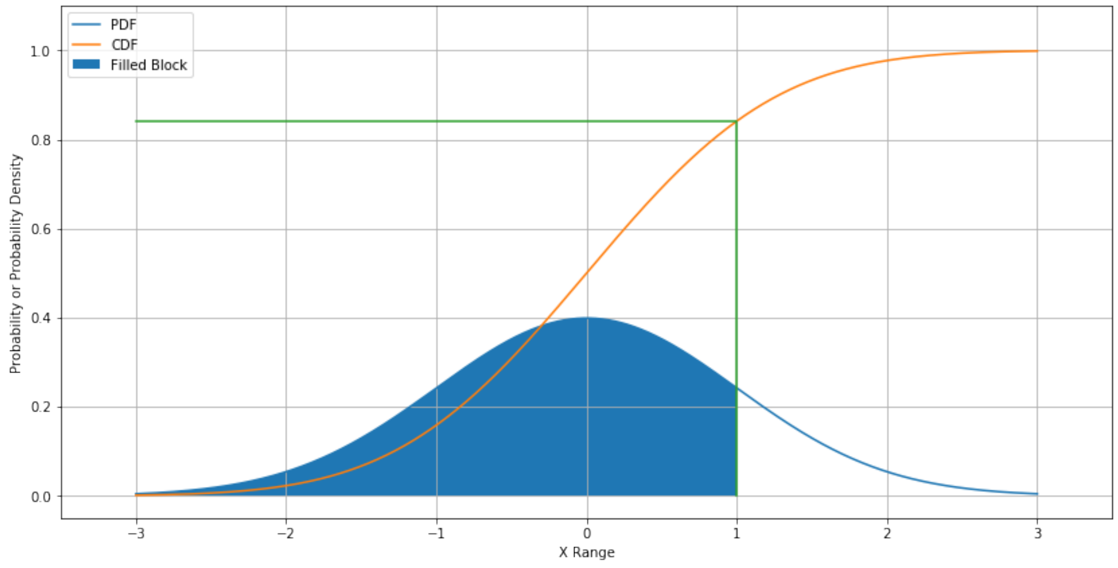
Simple plots can be done directy on the pyplot object, while you will want to define figures within the object for more complex plotting, controlling separate axes, etc. You can leaqrn a whole lot more by googling pyplot and the thing you want to know more about.

```
In [3]: # gets all of numpy but you will still need to refer to e.g. numpy.random.randn() as ran
        from numpy import *                # all of the array capable elements for numerical data
        from scipy.stats import norm       # comprehensive functions for Gaussian normal distribu
        import csv                         # library for reading and writing comma separated valu
        from matplotlib.pyplot import *    # all of pyplot to graph the results of our calculatio

        # put figures inline in the notebook, so you don't have to call show()
        %matplotlib inline

        # This lets you make the default figure size larger (or smaller)
        from IPython.core.pylabtools import figsize
        figsize(14, 7)
```

plot(x,y,label = 'text') draws a plot of y vs x with label 'text'. x and y should be 1D arrays the same size
plot([x1,x2,x3,...],[y1,y2,y3,...]) draws a line from point to point by making 1D arrays right in the function call
fill_between(x,y1,y2,label = 'text') draws a solid area between y1 and y2
hist(data) draws a histogram of the data. Use bins to control the number of data bins. bins = n will give you n bins, auto width. bins = np.linspace(low,high,n) will give you n bins evenly spaced between low and high. Use alpha to control the transparency of the blocks. Use normed = 1 to make it integrate to 1 like a PDF.
axis([x1,x2,y1,y2]) will set the limits on the axes. The [ ] put the values in an array.
xlim(x1,x2) or **ylim(y1,y2)** will do the same for just one of the axes.
legend() draws a legend based on the labels defined so far
xlabel('text') and **ylabel('text')** label the axes
grid() will add grid lines

```
In [4]: x = linspace(-3,3,1000)
        f = norm.pdf(x,0,1)
        g = norm.cdf(x,0,1)
```

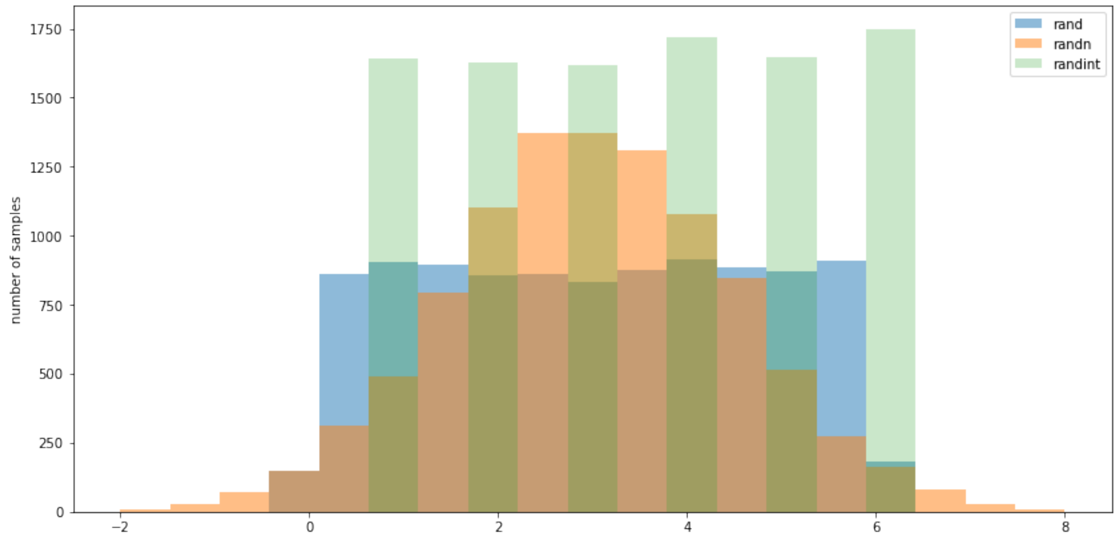
```
         plot(x,f,label = 'PDF')
         plot(x,g,label = 'CDF')
         xto1 = linspace(-3,1,1000)
         fto1 = norm.pdf(xto1,0,1)
         fill_between(xto1,0,fto1,label = 'Filled Block')
         plot([1,1,-3],[0,norm.cdf(1,0,1),norm.cdf(1,0,1)])
         legend()
         xlabel('X Range')
         ylabel('Probability or Probability Density')
         grid()
         axis([-3.5,3.5,-0.05,1.1])

Out[4]: [-3.5, 3.5, -0.05, 1.1]
```



So the interval containing 95% of the samples isn't exactly ±two standard deviations:

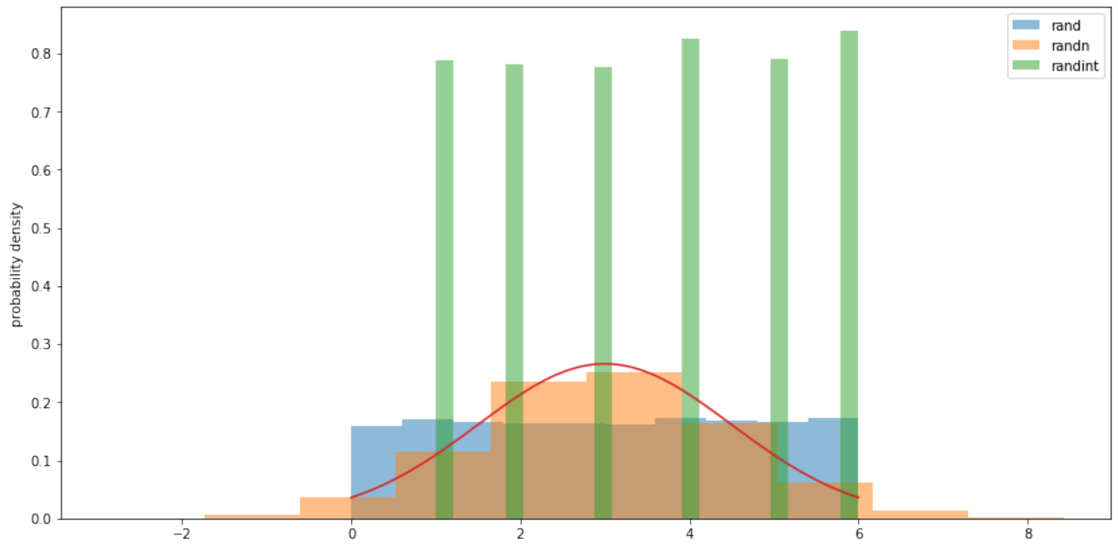
```
In [4]: norm.interval(0.95,192,10)

Out[4]: (172.40036015459947, 211.59963984540053)

In [5]: n = 10000
        r1 = random.rand(n) * 6
        r2 = random.randn(n) * 1.5 + 3
        r3 = random.randint(1,7,n)
        hist(r1,alpha=0.5, bins = linspace(-2,8,20), label = 'rand')
        hist(r2,alpha=0.5, bins = linspace(-2,8,20), label = 'randn')
        hist(r3,alpha=0.25, bins = linspace(-2,8,20), label = 'randint')
        ylabel('number of samples')
        legend()
```

```
Out[5]: <matplotlib.legend.Legend at 0x1174af978>
```



```
In [6]: x1 = linspace(0,6,100)
        f1 = norm.pdf(x1,3,1.5)
        hist(r1,alpha=0.5, label = 'rand', normed=1)
        hist(r2,alpha=0.5, label = 'randn', normed=1)
        hist(r3,alpha=0.5,bins = 24, label = 'randint', normed=1)
        plot(x1,f1)
        ylabel('probability density')
        legend()
```

```
Out[6]: <matplotlib.legend.Legend at 0x1177bd2e8>
```

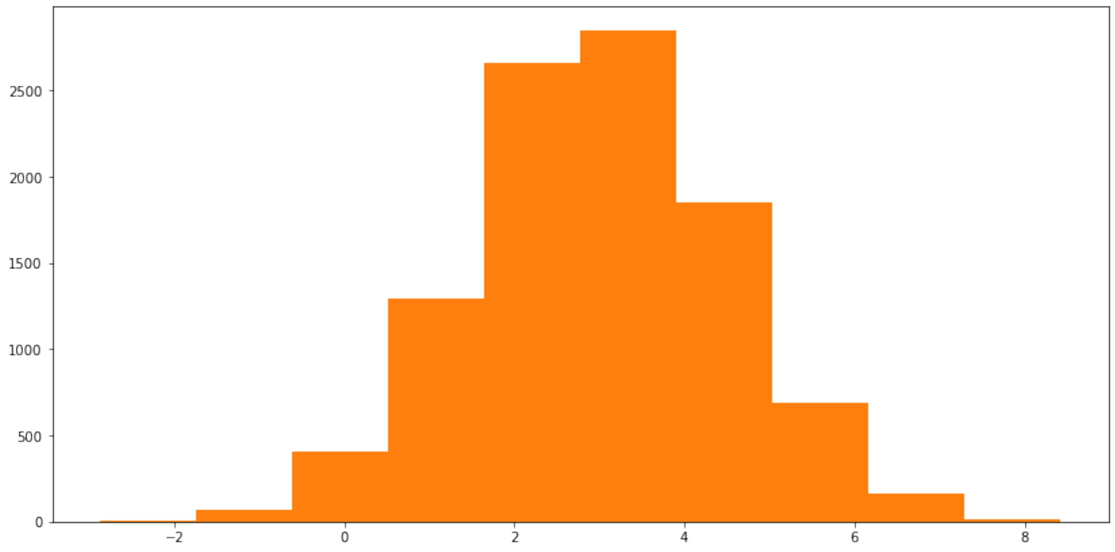


1.1 Customize the Bin Widths

This sample shows several other more advanced things besides customizing bin widths. Ask yourself which plots actually represent the data clearly. Which mark is given to more students, A or B?

```
In [7]: N,bins,patches = hist(r2)   # the line from most examples
        hist(r2)    # the first array is a list of counts and the second is the bin boundaries
```

```
Out[7]: (array([    5.,    68.,   404.,  1294.,  2664.,  2846.,  1851.,   689.,
                  163.,    16.]),
         array([-2.86856549, -1.73915053, -0.60973557,  0.51967938,  1.64909434,
                 2.7785093 ,  3.90792426,  5.03733922,  6.16675418,  7.29616913,
                 8.42558409]),
         <a list of 10 Patch objects>)
```

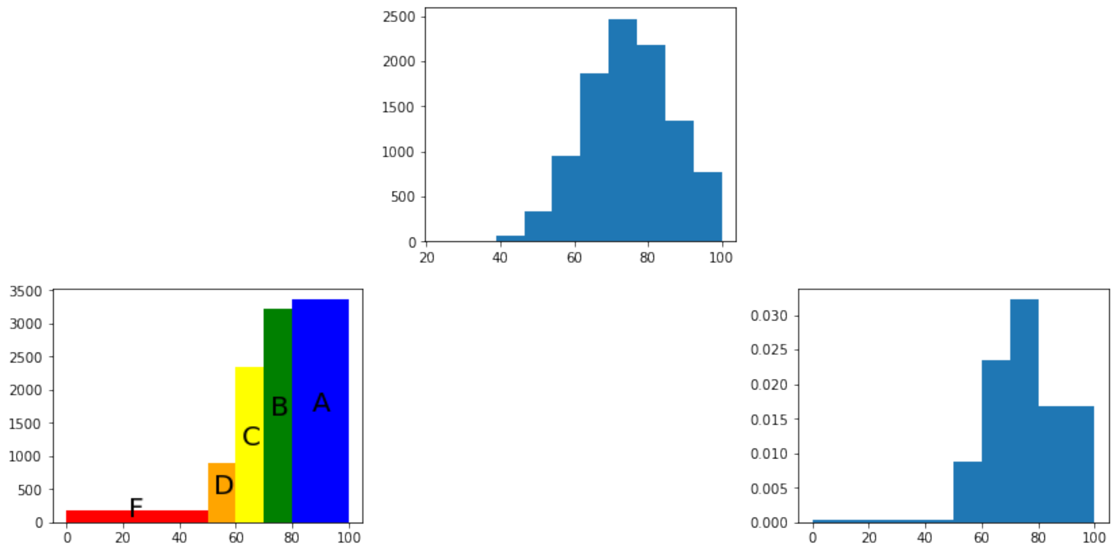


```
In [8]: marks = random.randn(n) * 12 + 75   # an imaginary range of student percentage marks
        for i in range(0,n):
            marks[i] = min(marks[i],100)       # that can't be larger than 100
            marks[i] = max(marks[i],0)         # or smaller than 0
        mean(marks)
```

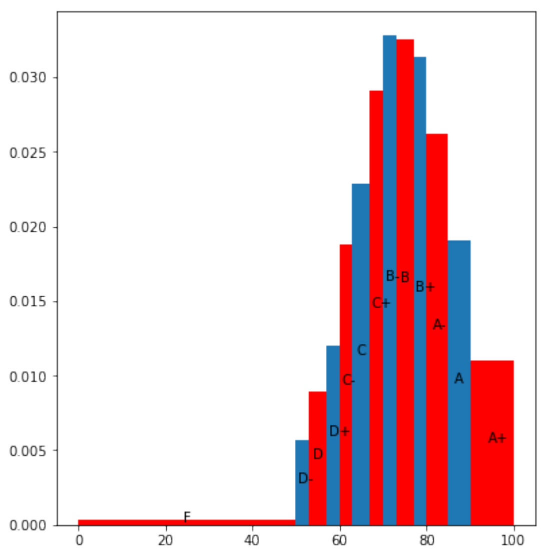
```
Out[8]: 74.87166992211526
```

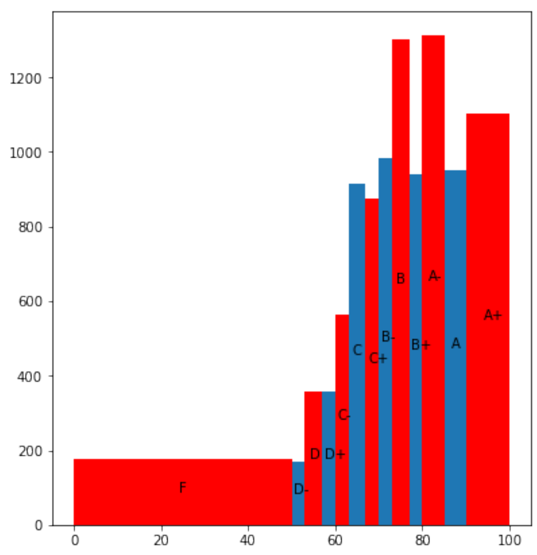
```
In [9]: b=[0,50,60,70,80,100]
        l=['F','D','C','B','A']
        # you need to create a figure to do some of the more complex elements in plotting
        fig = figure()
        ax = fig.add_subplot(234) # ax will be the fourth set of axes in a fig layout of 2 rows
        N,bins,patches = ax.hist(marks,bins = b)
        patches[0].set_facecolor('red')
        patches[1].set_facecolor('orange')
        patches[2].set_facecolor('yellow')
        patches[3].set_facecolor('green')
        patches[4].set_facecolor('blue')
        for i in range(0,5):
            ax.text((b[i+1]+b[i])/2-3,N[i]/2,l[i],size = 20)
        ax2 = fig.add_subplot(232)
        ax2.hist(marks)
        ax6 = fig.add_subplot(236)
        ax6.hist(marks,bins=b,normed = 1)
```

```
Out[9]: (array([ 0.000352,  0.00885 ,  0.02351 ,  0.03224 ,  0.01682 ]),
         array([  0,  50,  60,  70,  80, 100]),
         <a list of 5 Patch objects>)
```



```
In [10]: b=[0,50,53, 57, 60,  63,  67, 70,  73,  77, 80,  85,  90, 100]
         l=['F','D-','D','D+','C-','C','C+','B-','B','B+','A-','A','A+']
         fig = figure()
         ax = fig.add_subplot(121)
         N,bins,patches = ax.hist(marks,bins=b,normed = 1)
         for i in range(0,13):
             ax.text((b[i+1]+b[i])/2-1,N[i]/2,l[i])
             if (int(i/2)*2 == i): patches[i].set_facecolor('red')
         axc = fig.add_subplot(122)
         N,bins,patches = axc.hist(marks,bins=b)
         for i in range(0,13):
             axc.text((b[i+1]+b[i])/2-1,N[i]/2,l[i])
             if (int(i/2)*2 == i): patches[i].set_facecolor('red')
```


0.000
0.005
0.010
0.015
0.020
0.025
0.030
0
20
40
60
80
100
F
D-
D
D+
C-
C
C+
B-
B
B+
A-
A
A+


0
200
400
600
800
1000
1200
0
20
40
60
80
100
F
D-
D
D+
C-
C
C+
B-
B
B+
A-
A
A+