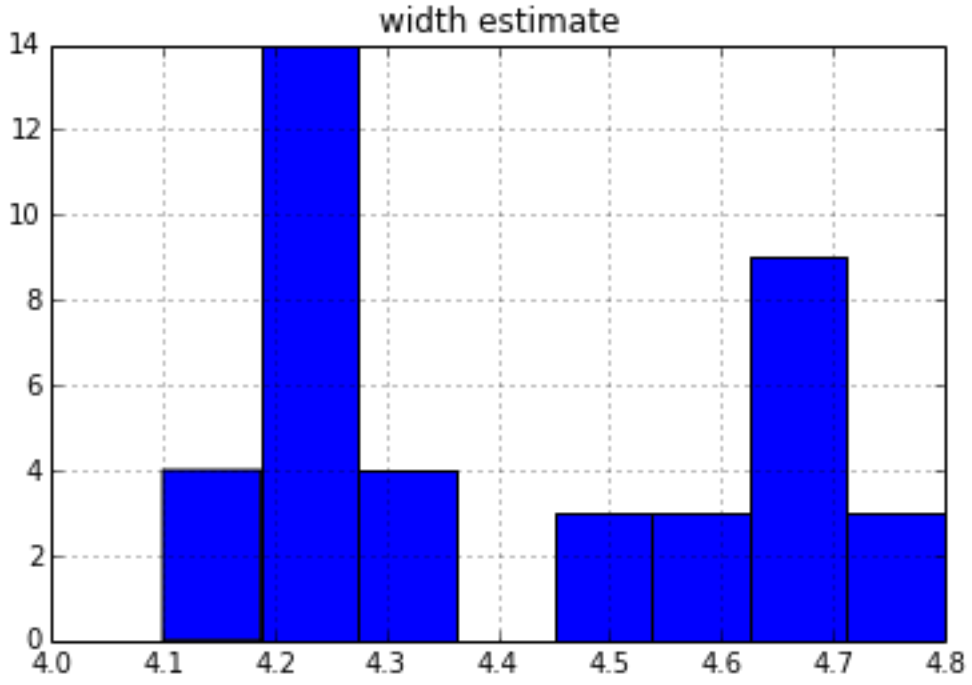
from PDFs to CDFs 2-11

February 15, 2016

In [147]:
```
%matplotlib inline

import numpy as np #matrices and data structures
import scipy.stats as ss #standard statistical operations
import pandas as pd #keeps data organized, works well with data
import matplotlib.pyplot as plt #plot visualization

#import same data as last time:
#https://weather-warehouse.com/WeatherHistory/PastWeatherData_NewYorkCentralPrkObsBelv_NewYor
nyw = pd.read_csv('NYC-CParkWeather.csv')
nyw = nyw.set_index('year') #represents observations
```

In [148]:
```
#inclass measurement data
classX = pd.DataFrame(np.array([4.5, 4.7, 4.6, 4.5,4.1,4.2,4.2,4.2,4.1,4.2,4.3,4.2,4.2,4.3,4.
classX.columns=['width estimate']
classX.hist(bins=8)
```

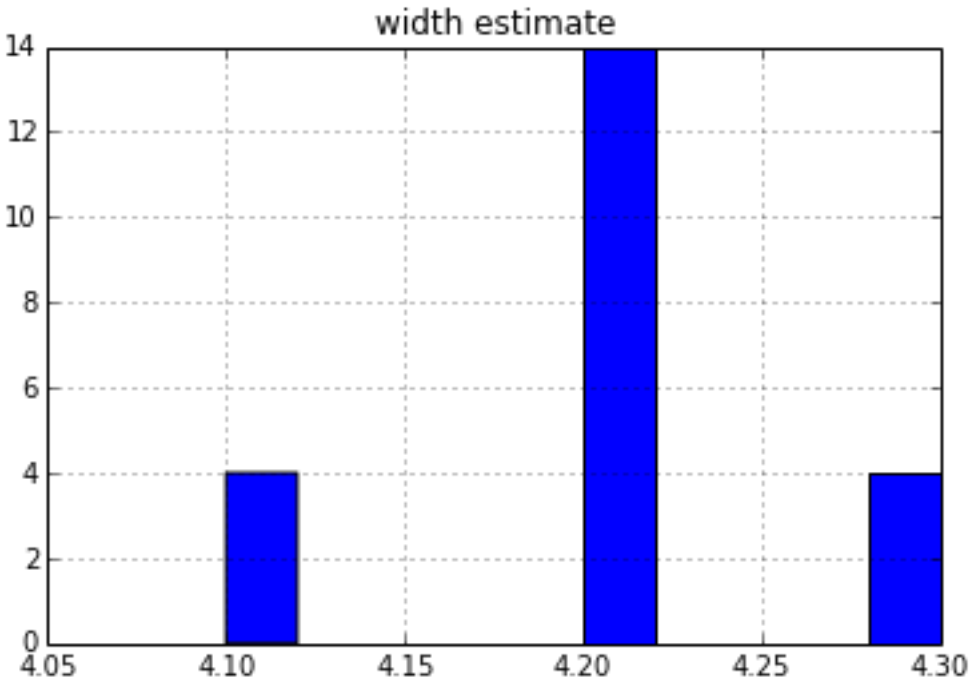
Out[148]:
```
array([[<matplotlib.axes.AxesSubplot object at 0x7f40cf4b2590>]], dtype=object)
```



In [149]:
```
#X.describe()
newX = classX.loc[classX['width estimate']<4.4] #loc returns elements at the given locations
newX.hist()
newX.describe()
```

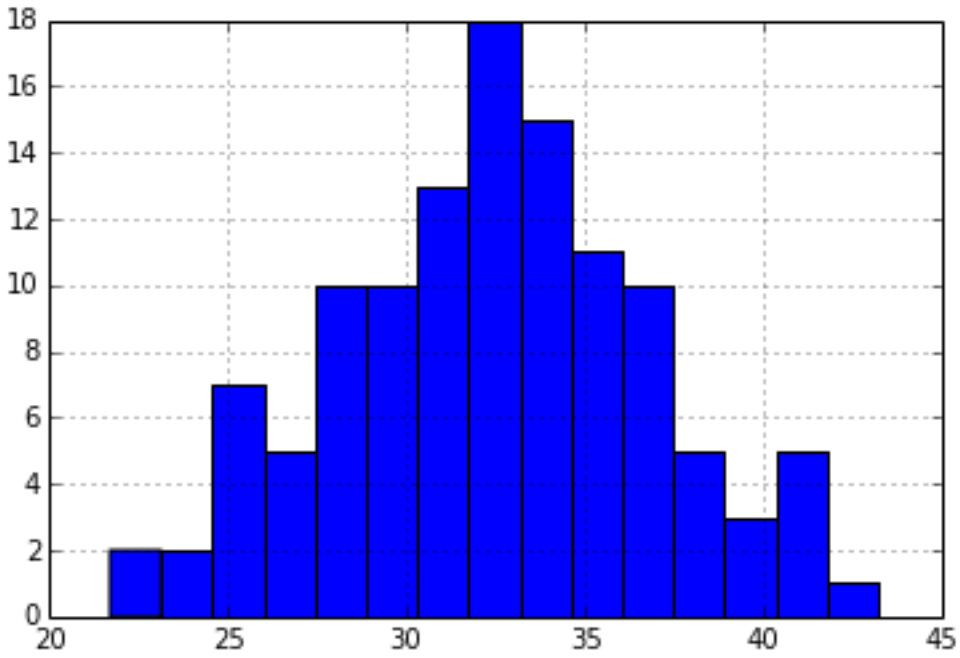
Out[149]:

	width estimate
count	22.000000
mean	4.200000
std	0.061721
min	4.100000
25%	4.200000
50%	4.200000
75%	4.200000
max	4.300000



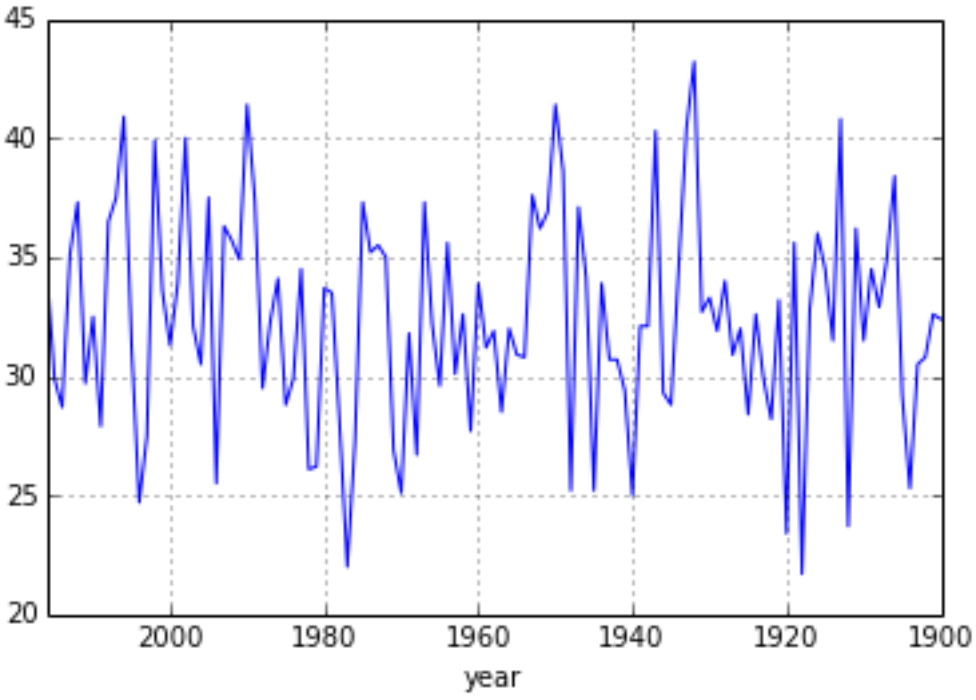
In [150]:
```
nyw_m=nyw['mean']
nyw_m.hist(bins=15)
```

Out[150]:
```
<matplotlib.axes.AxesSubplot at 0x7f40cf1b4dd0>
```

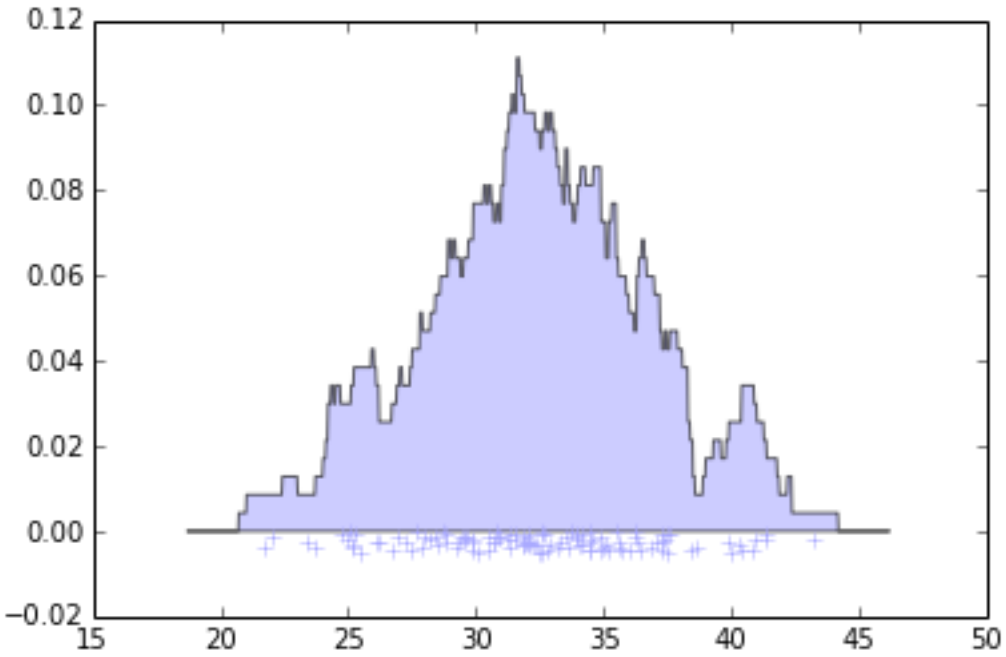


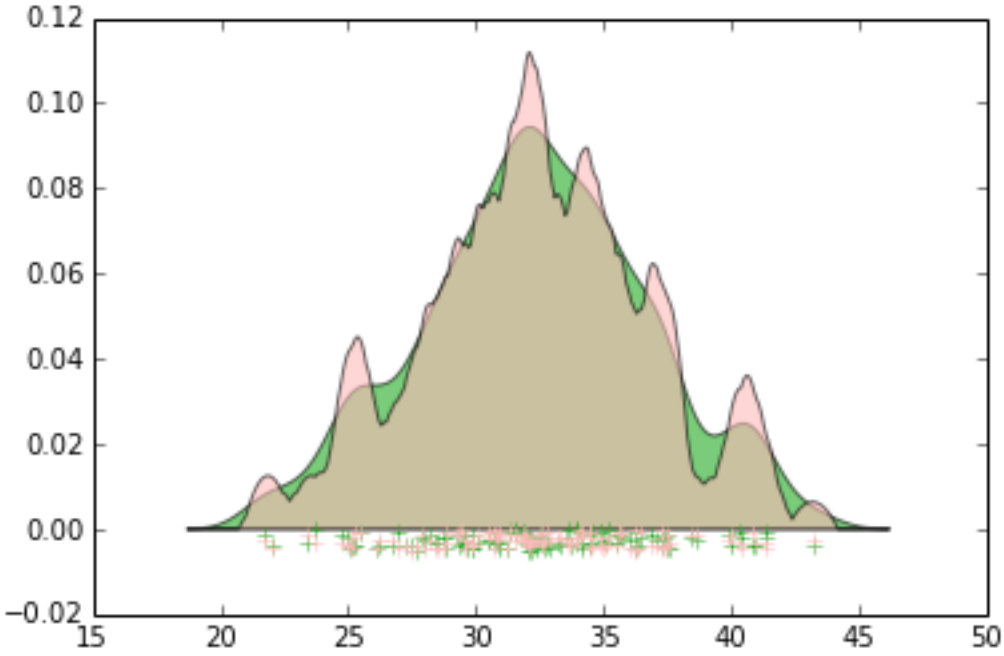
In [151]: `nyw_m.plot()`

Out[151]: `<matplotlib.axes.AxesSubplot at 0x7f40cef17e90>`

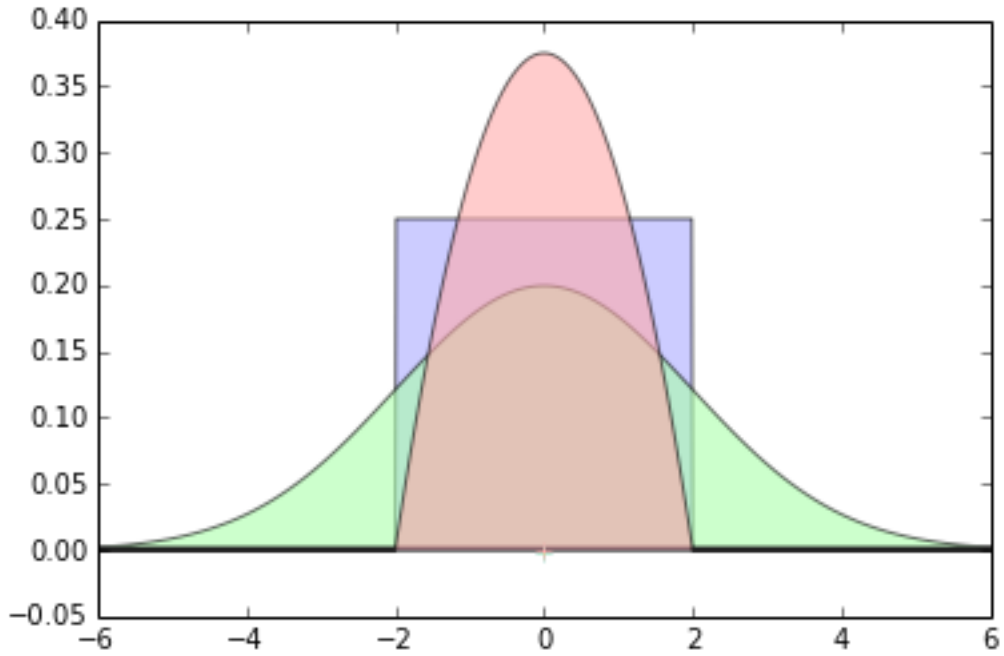


In [152]:
```
#import KDE
from sklearn.neighbors import KernelDensity

def kde_plot(kernel, X, color="#aaaaff", bw = 1):
    #create the estimator:
    kde_X = KernelDensity(kernel=kernel, bandwidth=bw).fit(X)

    #setup range:
    range = np.linspace(X.min()-bw*3, X.max()+bw*3, 1000)[:,np.newaxis]

    #plot:
    plt.fill(range[:,0], np.exp(kde_X.score_samples(range)), fc=color, alpha=.6)
    dots = [y-np.random.rand()*.005 for y in np.zeros(X.shape[0])] #all points, randomly jitt
    plt.plot(X[:,0], dots, '+k', color=color)

Xnyw_m = nyw_m.reshape(-1,1)
kde_plot('tophat', Xnyw_m)
plt.show()
kde_plot('gaussian', Xnyw_m, '#22aa22')
kde_plot('epanechnikov', Xnyw_m, '#ffbbbb')
```



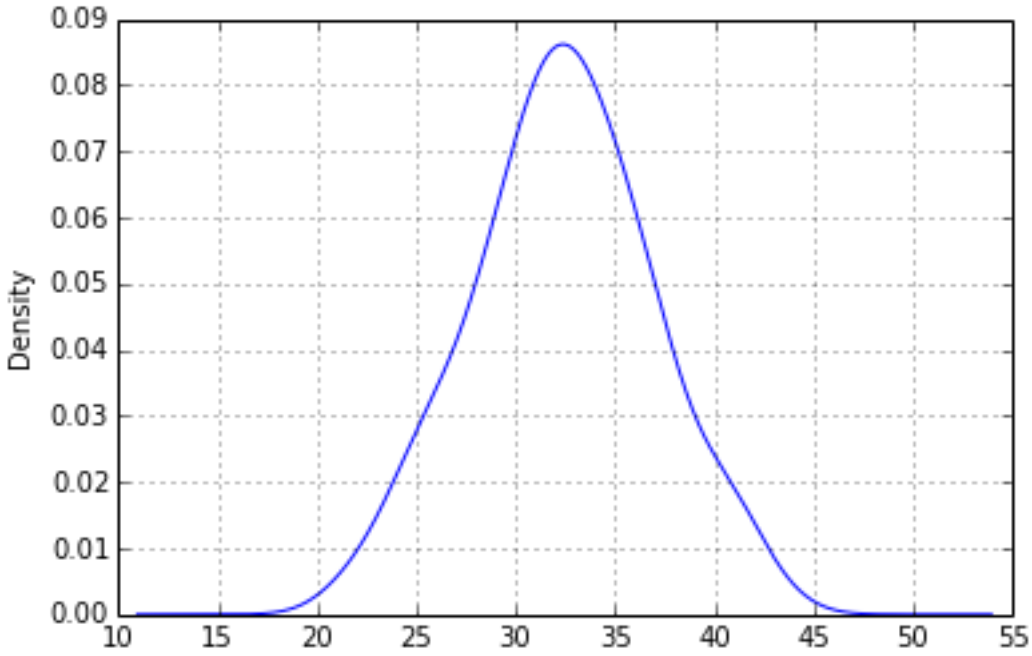


In [153]:
```
#lets see what each kernel looks like:
X = np.array([[0]])
kde_plot('tophat', X, bw=2)
kde_plot('gaussian', X, '#aaffaa', bw=2)
kde_plot('epanechnikov', X, '#ffaaaa',bw=2)
```

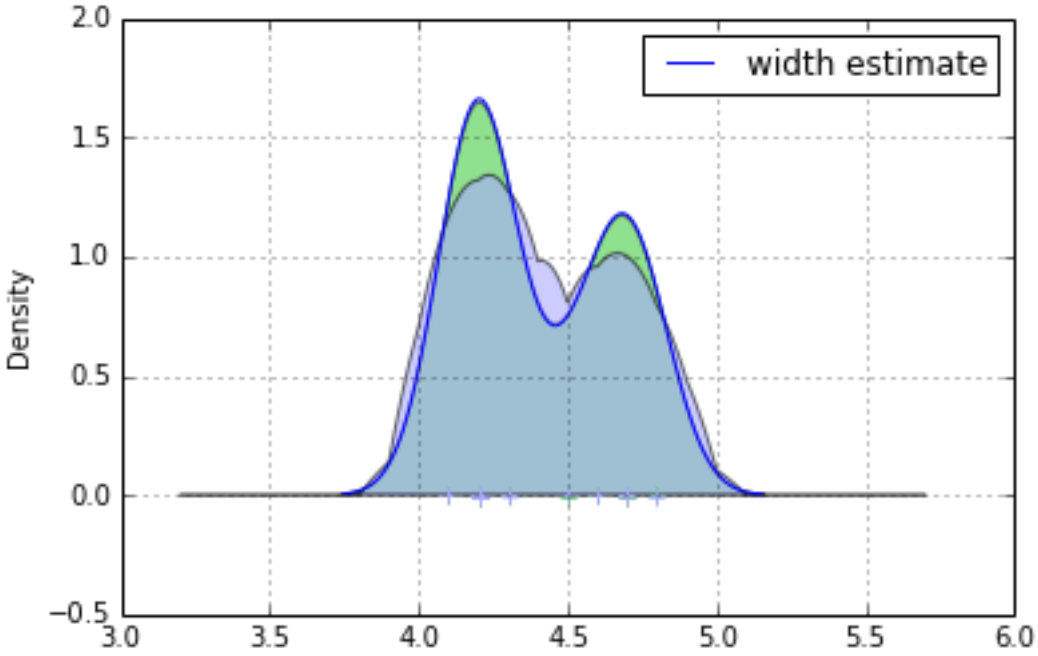


In [154]:
```
#Pandas also has KDE built in:
nyw_m.plot(kind='kde')
```

Out[154]:
```
<matplotlib.axes.AxesSubplot at 0x7f40cf09a910>
```

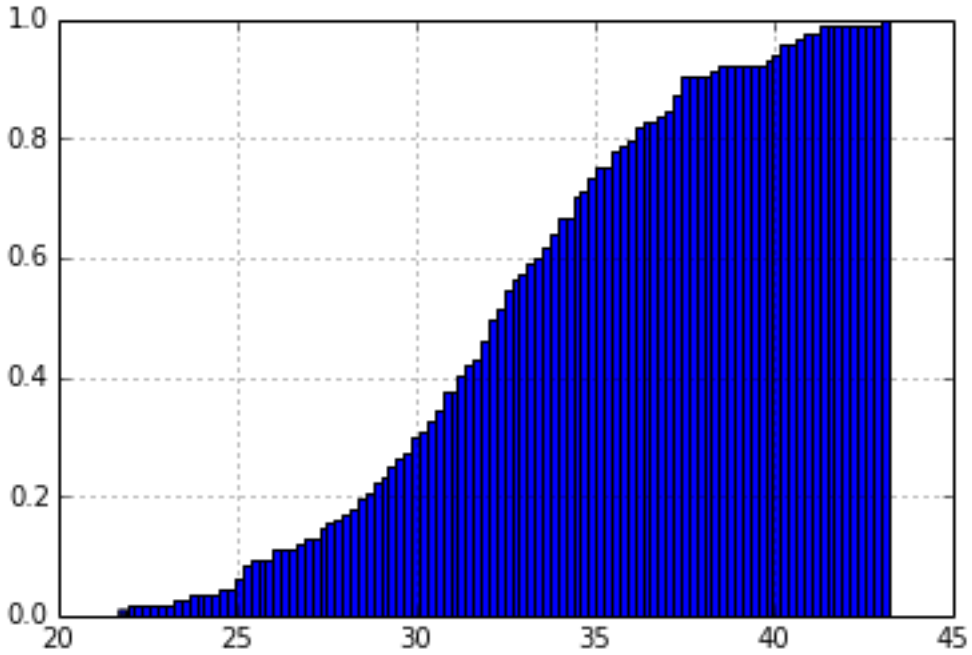


In [155]:
```
#KDE on class measurement data.
# let's see whether pandas' KDE matches Gaussian or Epan
classX.plot(kind='kde')
kde_plot('gaussian', classX.as_matrix(), bw=0.12, color='#44cc44')
kde_plot('epanechnikov', classX.as_matrix(), bw=0.3)
#pandas seems to be using a Gaussian kernel with bandwidth optimized to .12
```



In [156]:
```
#classX.hist(cumulative=True)
nyw_m.hist(cumulative=True, normed=True, bins=100)
```

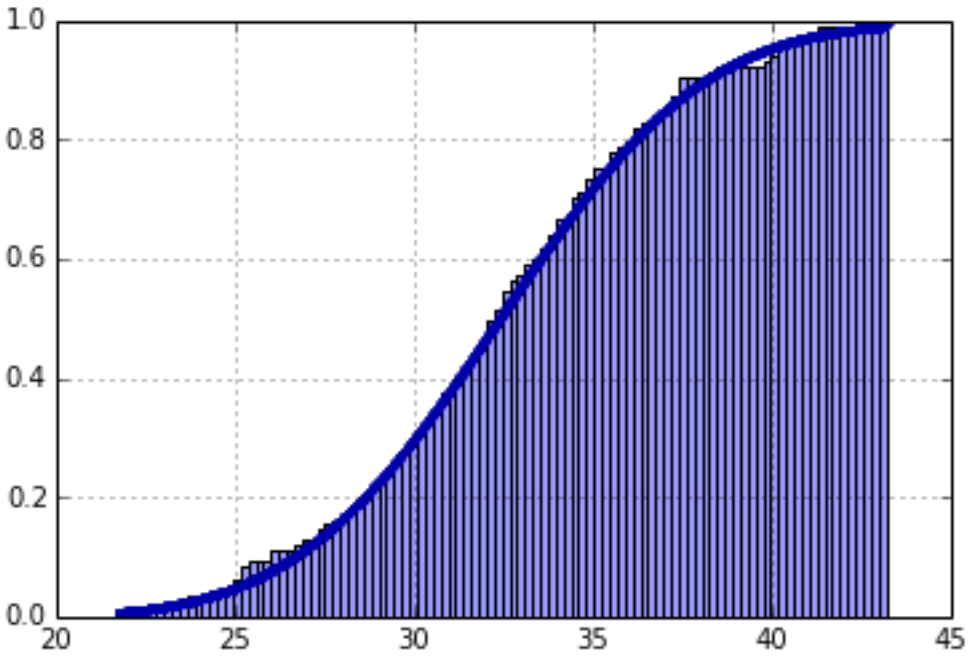
Out[156]:
```
<matplotlib.axes.AxesSubplot at 0x7f40cf3f89d0>
```



In [157]:
```
#no "elementary" form solution for normal cdf (due to antiderivative of e being sum of logs)
#however, it can be approximated with a measurement error function; i.e. a sigmoid
import math
def normal_cdf(x, mu= 0, sigma = 1):
    return (1 + math.erf((x - mu) / math.sqrt(2) / sigma)) / 2 #(Grus, 2015)

def lighten_color(c, amount=9):
    return ''.join([chr(min(ord(c[i])+amount,ord('f')) ) if c[i] != '#' else '#' for i in rang

[(x, normal_cdf(x)) for x in np.linspace(-1,1,10)]
```

Out[157]:
```
[(-1.0, 0.15865525393145702),
 (-0.7777777777777779, 0.21835001536137888),
 (-0.5555555555555558, 0.289257360753972),
 (-0.33333333333333337, 0.36944134018176367),
 (-0.1111111111111116, 0.45576411895468855),
 (0.1111111111111116, 0.5442358810453114),
 (0.33333333333333326, 0.6305586598182363),
 (0.5555555555555536, 0.7107426392460279),
 (0.7777777777777768, 0.7816499846386211),
 (1.0, 0.841344746068543)]
```

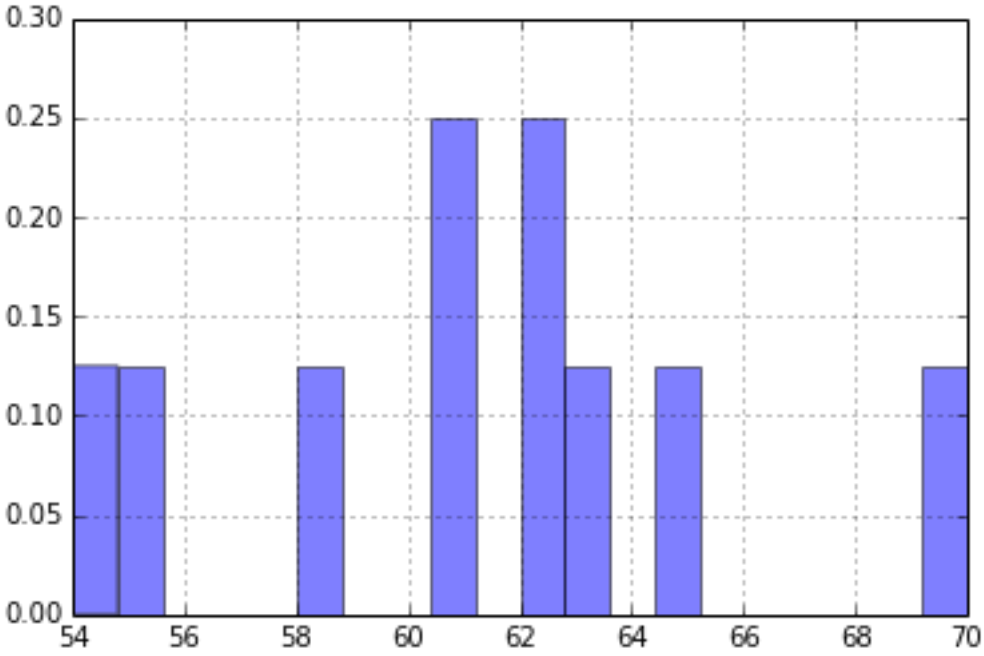
In [159]:
```
def normal_cdf_plot(X, color="#0000aa"):
    # X: a pandas dataframe with one column
    #first plot the data with a cumulative histogram:
    X.hist(cumulative=True, normed=True, bins=100, color=lighten_color(color))

    #calculate the mean and sigma of the data:
    (mu, sigma) = X.mean(), X.std()

    #plot
    range = np.linspace(X.min(), X.max(), 1000)[:,np.newaxis]
    dots = [normal_cdf(x,mu,sigma) for x in range]
    plt.plot(range, dots,'.',color=color,label=X.name)

normal_cdf_plot(nyw_m)
#normal_cdf_plot(classX)
```




```
In [165]: #discrete random variables:
          def bernoulli_trial(p):
              return 1 if np.random.rand()<p else 0
          bernoulli_trial(.6)
```

```
Out[165]: 0
```

```
In [166]: def binomial(p, n):
              trials = [bernoulli_trial(p) for _ in range(n)]
              return sum(trials), trials

          #"draw" a binomial:
          count, x = binomial(.6, 100)
          #draw a series of 10 bonomials
          X = pd.Series([binomial(.6, 100)[0] for _ in range(10)])
          X.hist(bins=20, normed=True, alpha=.5)
          plt.show()
          #draw a series of 100 bonomials
          X = pd.Series([binomial(.6, 100)[0] for _ in range(100)])
          X.hist(bins=20, normed=True, alpha=.5)
          #draw a series of 10000 bonomials
          X = pd.Series([binomial(.6, 100)[0] for _ in range(10000)])
          X.hist(bins=20, normed=True, alpha=.5)
```



Out[166]: <matplotlib.axes.AxesSubplot at 0x7f40ce786950>