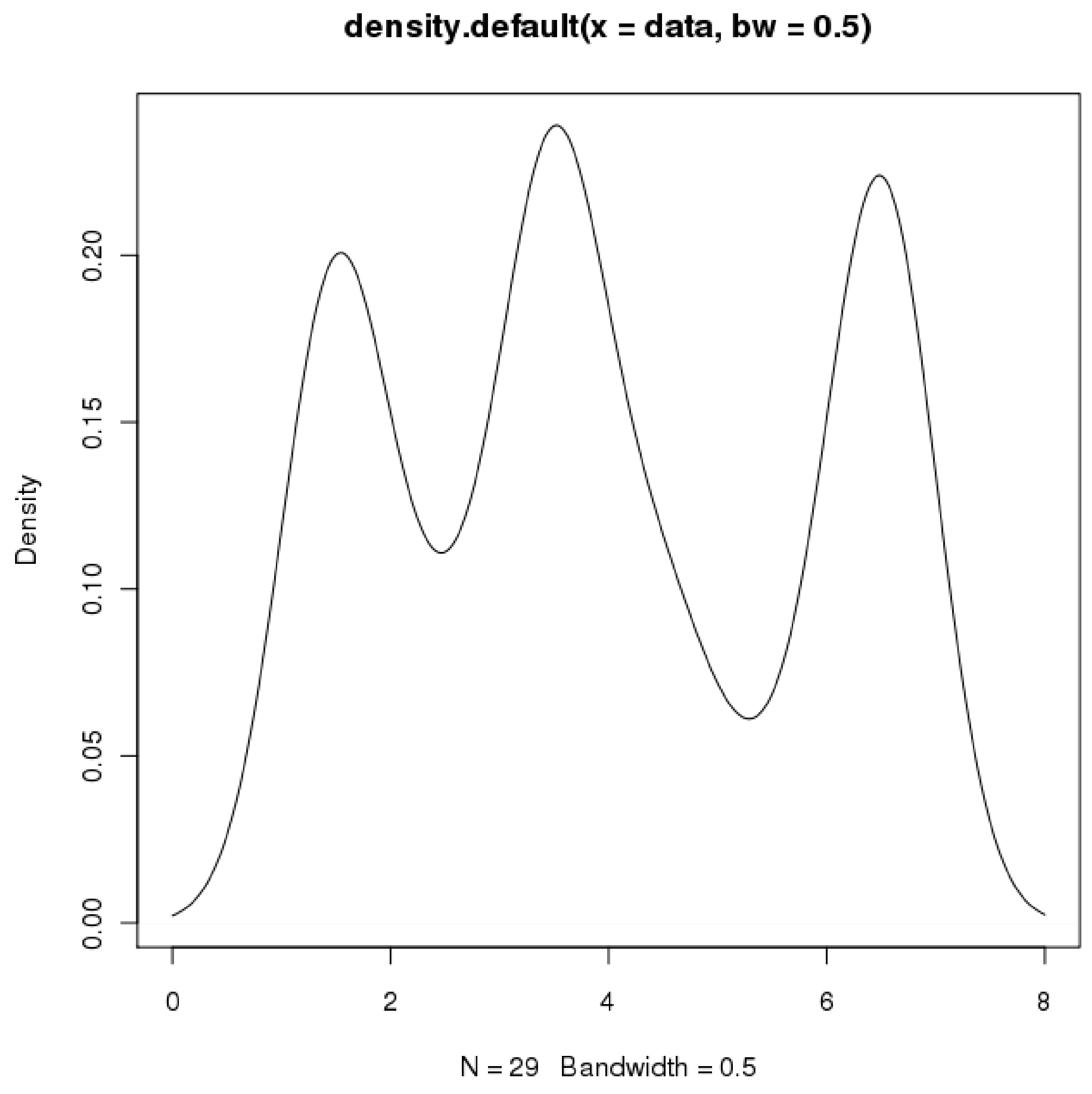
# How to create a density plot in matplotlib?

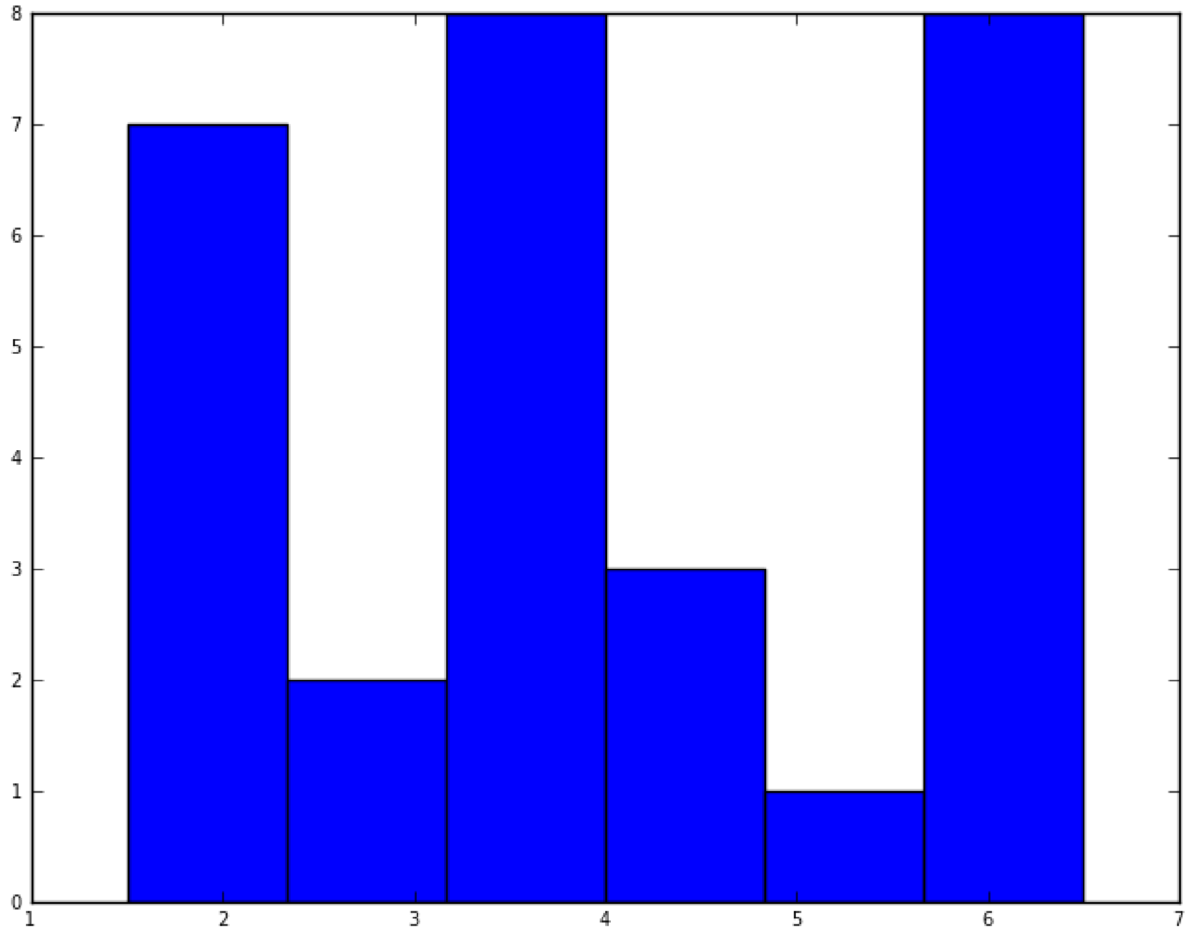
In R I can create the desired output by doing:

```
data = c(rep(1.5, 7), rep(2.5, 2), rep(3.5, 8),
         rep(4.5, 3), rep(5.5, 1), rep(6.5, 8))
plot(density(data, bw=0.5))
```



In python (with matplotlib) the closest I got was with a simple histogram:

```
import matplotlib.pyplot as plt
data = [1.5]*7 + [2.5]*2 + [3.5]*8 + [4.5]*3 + [5.5]*1 + [6.5]*8
plt.hist(data, bins=6)
plt.show()
```



I also tried the normed=True parameter but couldn't get anything other than trying to fit a gaussian to the histogram.

My latest attempts were around `scipy.stats` and `gaussian_kde`, following examples on the web, but I've been unsuccessful so far.

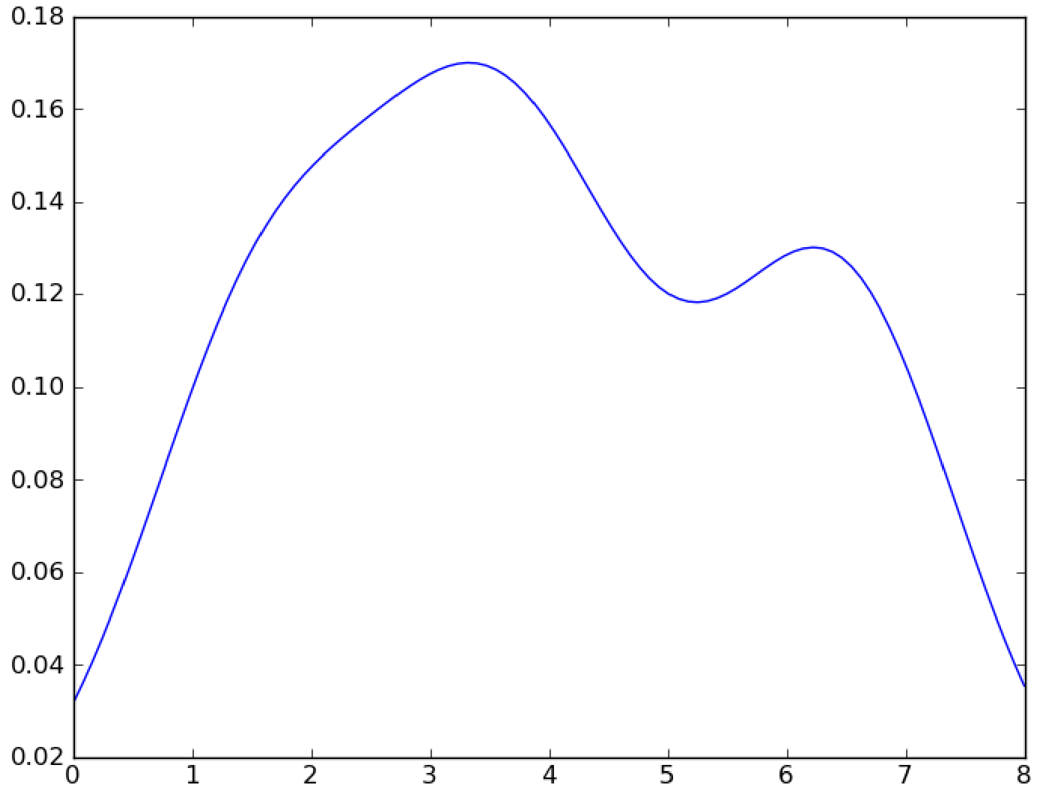
python r numpy matplotlib scipy

asked Nov 10 '10 at 23:39

Unode
**3,932** 2 19 38

## 4 Answers

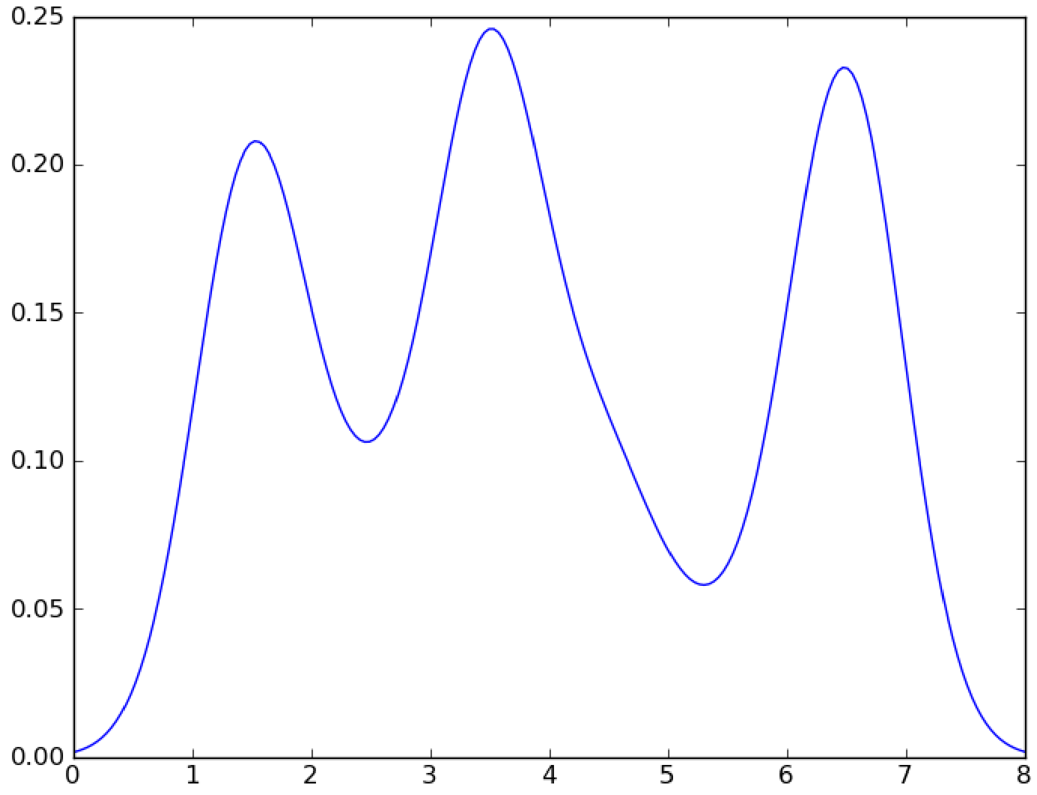
Sven has shown how to use the class `gaussian_kde` from Scipy, but you will notice that it doesn't look quite like what you generated with R. This is because `gaussian_kde` tries to infer the bandwidth automatically. You can play with the bandwidth in a way by changing the function `covariance_factor` of the `gaussian_kde` class. First, here is what you get without changing that function:



However, if I use the following code:

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import gaussian_kde
data = [1.5]*7 + [2.5]*2 + [3.5]*8 + [4.5]*3 + [5.5]*1 + [6.5]*8
density = gaussian_kde(data)
xs = np.linspace(0,8,200)
density.covariance_factor = lambda : .25
density._compute_covariance()
plt.plot(xs,density(xs))
plt.show()
```

I get



which is pretty close to what you are getting from R. What have I done? `gaussian_kde` uses a changable function, `covariance_factor` to calculate it's bandwidth. Before changing the function, the value returned by covariance_factor for this data was about .5. Lowering this lowered the bandwidth. I had to call `_compute_covariance` after changing that function so that all of the factors would be calculated correctly. It isn't an exact correspondence with the bw parameter from R, but hopefully it helps you get in the right direction.

answered Nov 11 '10 at 6:49

Justin Peel
**26.5k** 4 34 57

4

@Justin Nice answer (+1) and not wanting to start any Python v R flame wars or anything, but I am loving the way R works with data much more succinctly that python and other languages. I'm sure python has lots of good points over R (I'm not a Python user so I'm so totally uniformed to possibly comment) and can be used for lots more work than analysing data, but as a long-time R user I do forget how succinct a language it is for such tasks until examples like this crop up. – Gavin Simpson Nov 11 '10 at 12:38

4 (still fighting with editing comments) Here is a subclass of gaussian_kde that allows to set the bandwidth as an argument and more examples: mail.scipy.org/pipermail/scipy-user/2010-January/023877.html and there is an enhancement ticket at projects.scipy.org/scipy/ticket/1092 . Note, gaussian_kde is designed for n-dimensional data. – user333700 Nov 11 '10 at 14:53

8 @Gavin Simpson, yes, R is more succinct because it has a narrower scope. It is made for statistical computation and graphics. Python is a general programming language that can do pretty much whatever you want it to do. Because of that, the syntax might not be as succinct. Part of that is a different design in Numpy/Scipy, but part of it is just the modular set-up on Python. R is great if you only need to do computations and graphics, but if you need to use those computations in some brader applicatoin, then you might want something like Python. However, you can also use R from Python... – Justin Peel Nov 11 '10 at 19:09

1 @Justin - Yes indeed! That was partly my point - it was a pleasant surprise on the R side rather than a dig at the python side. – Gavin Simpson Nov 11 '10 at 19:14

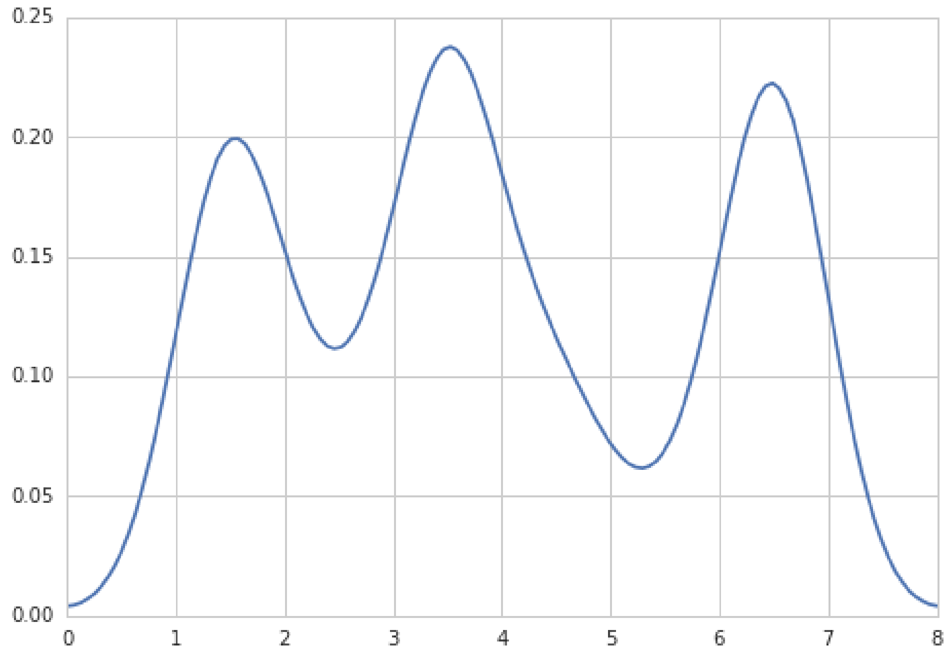
5 A `set_bandwidth` method and a `bw_method` constructor argument were added to gaussian_kde in scipy 0.11.0 per issue 1619 – eddygeek Jan 22 '15 at 14:46

|

Five years later, when I Google "how to create a kernel density plot using python", this thread still shows up at the top!

Today, a much easier way to do this is to use seaborn, a package that provides many convenient plotting functions and good style management.

```python
import numpy as np
import seaborn as sns
data = [1.5]*7 + [2.5]*2 + [3.5]*8 + [4.5]*3 + [5.5]*1 + [6.5]*8
sns.set_style('whitegrid')
sns.kdeplot(np.array(data), bw=0.5)
```



answered Sep 26 '15 at 23:57

Xin
**553** 5 13

Thank you so much .. Been searching for something like this since days .. can u pls explain why the `bw=0.5` is given? – Sitz Blogz Apr 19 at 15:00

1 @SitzBlogz The `bw` parameter stands for bandwidth. I was trying to match OP's setting (see his original first code example). For a detailed explanation of what `bw` controls, see en.wikipedia.org/wiki/…. Basically it controls how smooth you want the density plot to be. The larger the bw, the more smooth it will be. – Xin Apr 19 at 19:26

Thank you so much :) – Sitz Blogz Apr 19 at 19:29

I have another query to ask my data is discrete in nature and I am trying to plot the PDF for that, after reading through scipy doc I understood that PMF = PDF any suggestions on that how to plot it? – Sitz Blogz Apr 19 at 19:31

Maybe try something like:

```
import matplotlib.pyplot as plt
import numpy
from scipy import stats
data = [1.5]*7 + [2.5]*2 + [3.5]*8 + [4.5]*3 + [5.5]*1 + [6.5]*8
density = stats.kde.gaussian_kde(data)
x = numpy.arange(0., 8, .1)
plt.plot(x, density(x))
plt.show()
```

You can easily replace `gaussian_kde()` by a different kernel density estimate.

answered Nov 11 '10 at 0:40

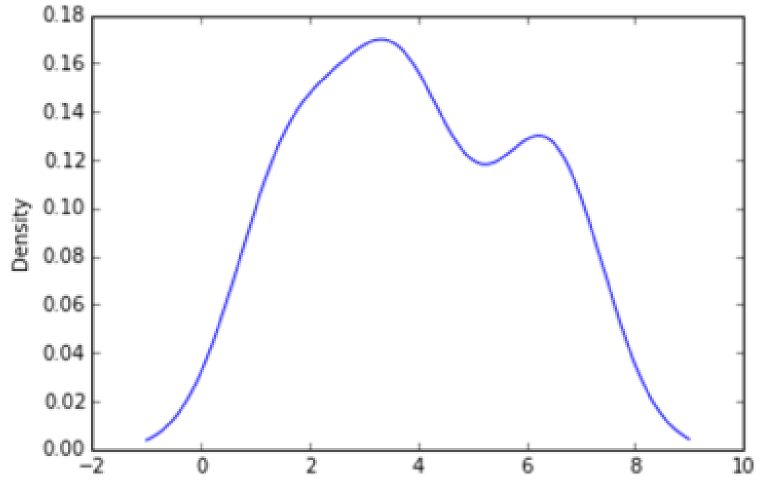
Sven Marnach
**223k** 39 571 591

1 +1 for the working example and already getting close to the desired output – Unode Nov 11 '10 at 15:58

**Option 1:**

Use `pandas` dataframe plot (built on top of `matplotlib` ):

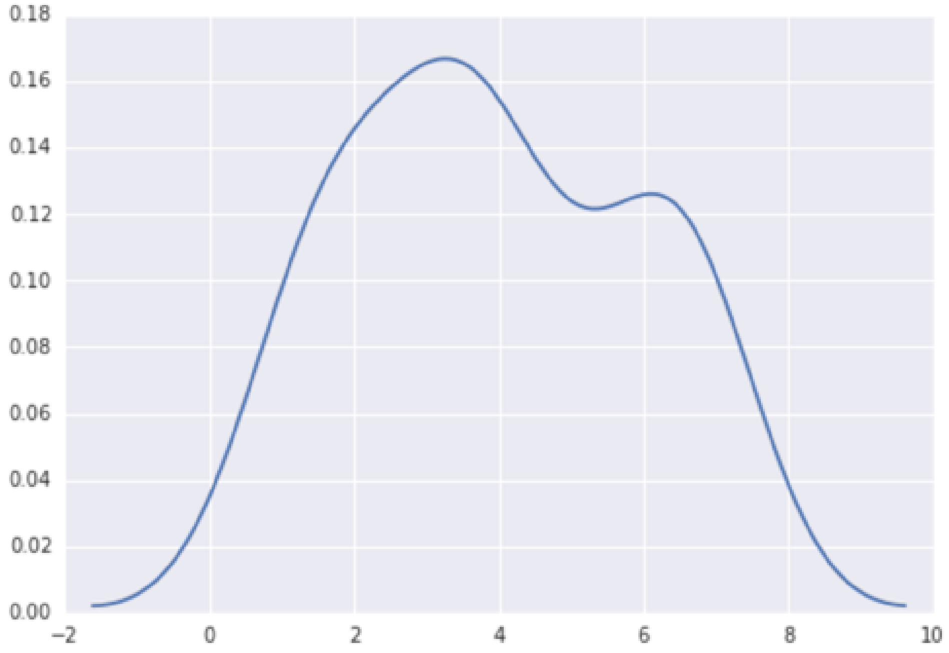
```
import pandas as pd
data = [1.5]*7 + [2.5]*2 + [3.5]*8 + [4.5]*3 + [5.5]*1 + [6.5]*8
df = pd.DataFrame(data)
df.plot(kind='density')
```



## Option 2:

Use `distplot` of `seaborn` :

```
import seaborn as sns
data = [1.5]*7 + [2.5]*2 + [3.5]*8 + [4.5]*3 + [5.5]*1 + [6.5]*8
sns.distplot(data, hist=False)
```



answered Nov 2 '15 at 9:28

Aziz Alto
**2,435** 1 18 21

1 To add the bandwidth parameter: df.plot.density(bw_method=0.5) – Anake Aug 25 at 13:41