

# Plotting a 2D heatmap with Matplotlib


Using Matplotlib, I want to plot a 2D heat map. My data is an n-by-n Numpy array, each with a value between 0 and 1. So for the (i, j) element of this array, I want to plot a square at the (i, j) coordinate in my heat map, whose color is proportional to the element's value in the array.

86

How can I do this?

python numpy matplotlib Edit tags

★  
18

asked Oct 22 '15 at 13:37  
 Karnivaurus  
6,286 31 91 166

2 did you look at the [matplotlib .gallery](#) at all before posting? There are some good examples using `imshow` , `pcolor` and `pcolormesh` that do what you want – tmdavison Oct 22 '15 at 13:57

Possible duplicate of [multi colored Heat Map error Python](#) – jkalden Oct 23 '15 at 6:52

27 In fairness to the "google would have helped" comment, fwiw Google now leads here :) – Owen Nov 24 '16 at 14:26

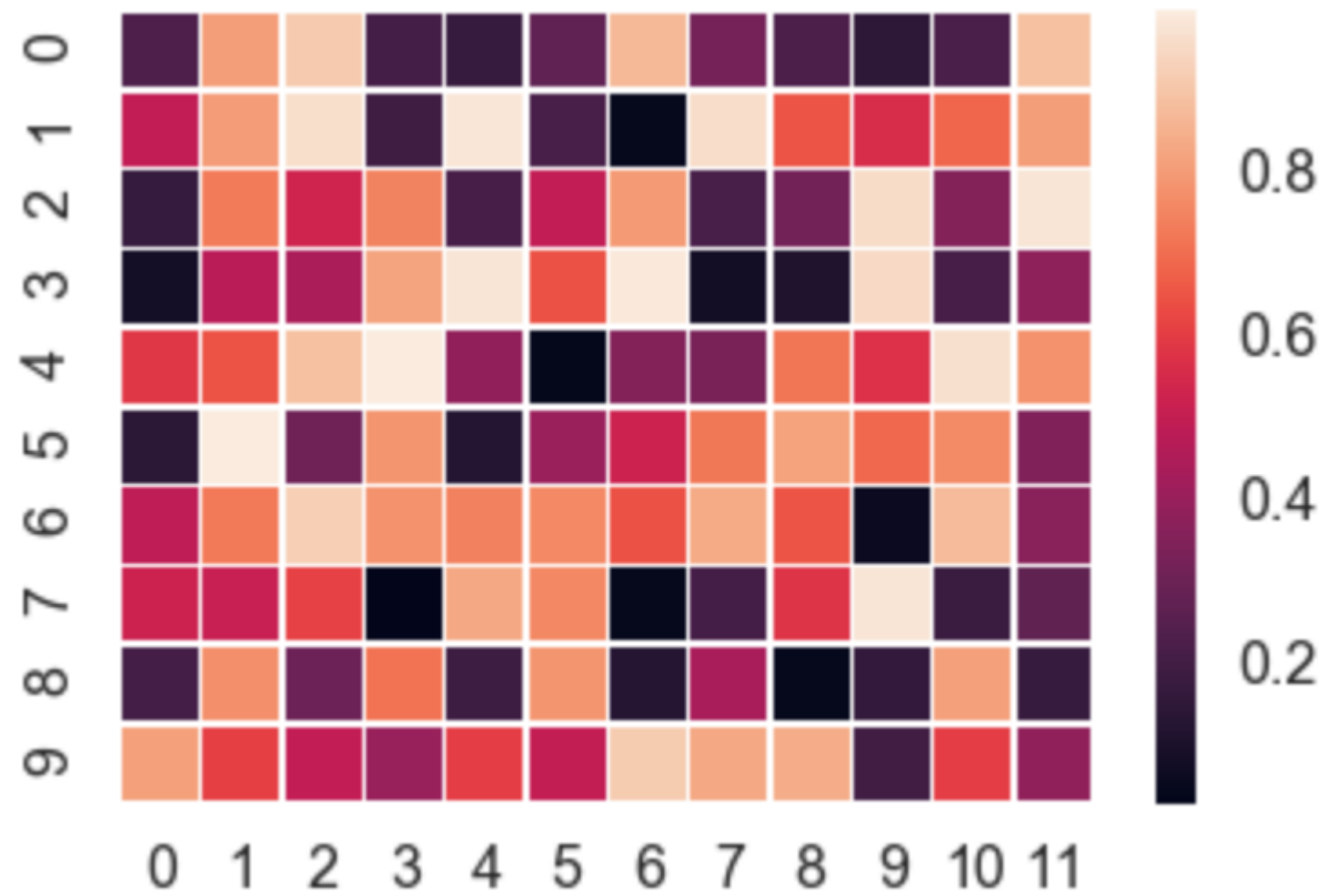
## 6 Answers

[Seaborn](#) takes care of a lot of the manual work and automatically plots a gradient at the side of the chart etc.

31

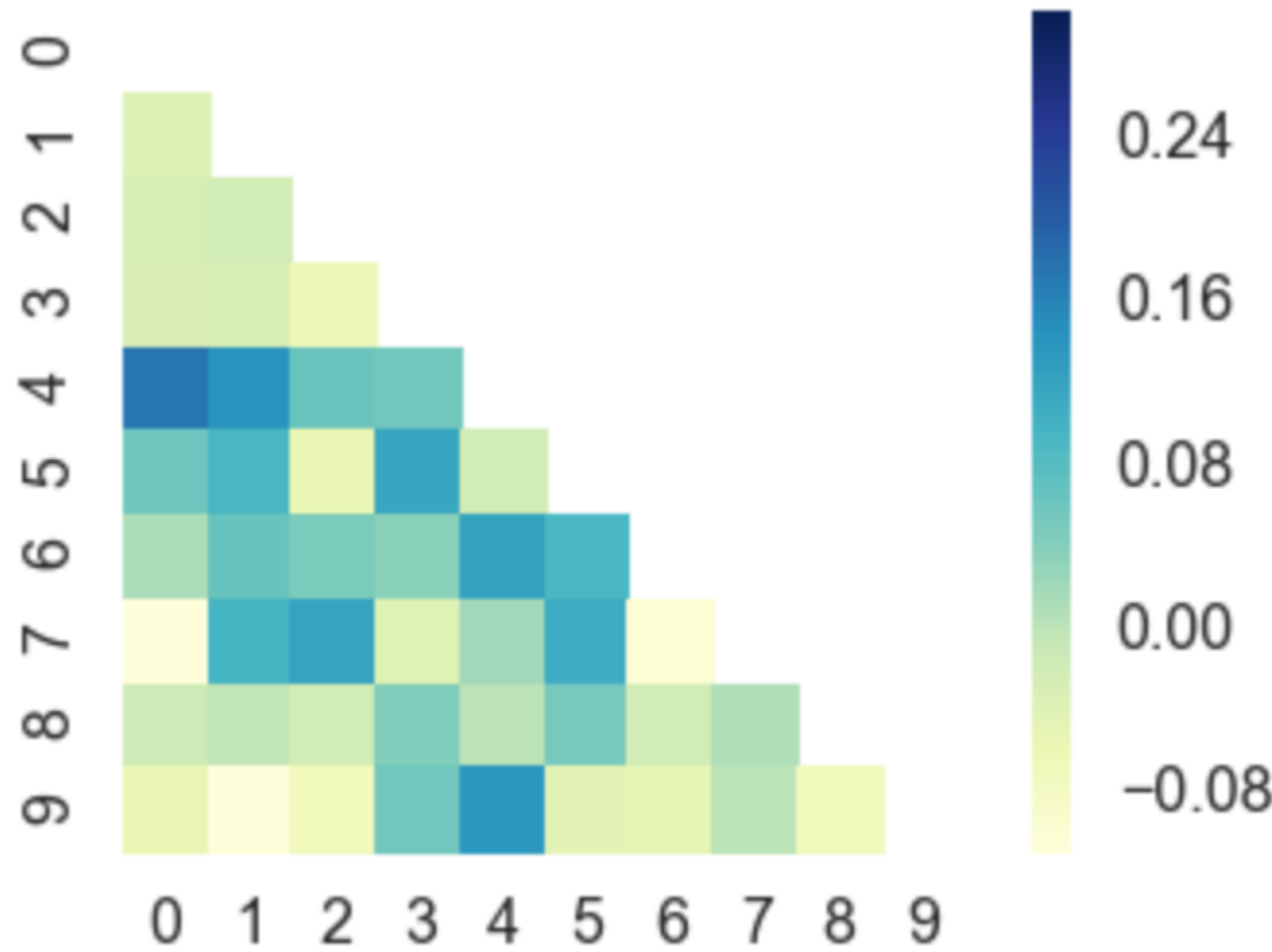
```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

uniform_data = np.random.rand(10, 12)
ax = sns.heatmap(uniform_data, linewidth=0.5)
plt.show()
```



Or, you can even plot upper / lower left / right triangles of square matrices, for example a correlation matrix which is square and is symmetric, so plotting all values would be redundant anyway.

```
corr = np.corrcoef(np.random.randn(10, 200))
mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask)] = True
with sns.axes_style("white"):
    ax = sns.heatmap(corr, mask=mask, vmax=.3, square=True, cmap="YlGnBu")
    plt.show()
```



edited yesterday



Fermi paradox

3,029 5 31 54

answered Apr 2 '18 at 9:27



PyRsquared

1,767 2 22 34

- 1

▲

▼

I'm very fond of the plot type, and the half matrix is useful. Two questions: 1) in the first plot the little squares are separated by white lines, could they be joint? 2) the white line width seem to vary, is this an artefact? – [P. Camilleri](#) Apr 28 '18 at 22:23
- 1

▲

▼

You can use the 'linewidth' argument I used in the first plot for any other plot (in the second plot for example), to get spaced out squares. The line widths only appear to vary in the first plot due to screen shot issues, they don't actually vary in reality, they should stay at the constant you set them. – [PyRsquared](#) Apr 29 '18 at 0:44

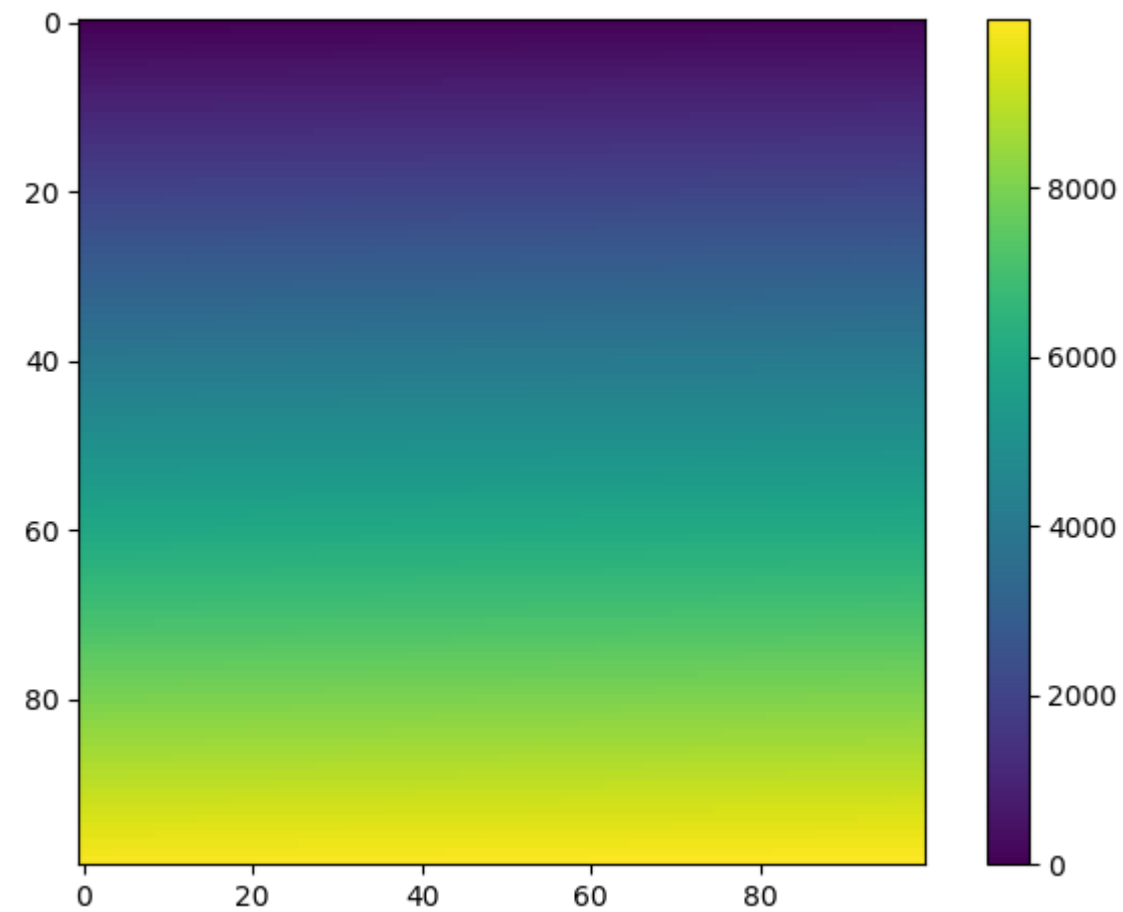
▲ For a 2d numpy array, simply use imshow() may help you:

4

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

def heatmap2d(arr: np.ndarray):
    plt.imshow(arr, cmap='viridis')
    plt.colorbar()
    plt.show()

test_array = np.arange(100 * 100).reshape(100, 100)
heatmap2d(test_array)
```



This code produces a continuous heatmap.

You can choose another built-in colormap from [here](#).

answered Apr 27 at 10:04



Huang Yuheng

350 3 19

I would use matplotlib's [pcolor/pcolormesh](#) function since it allows nonuniform spacing of the data.

8

Example taken from [matplotlib](#):

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

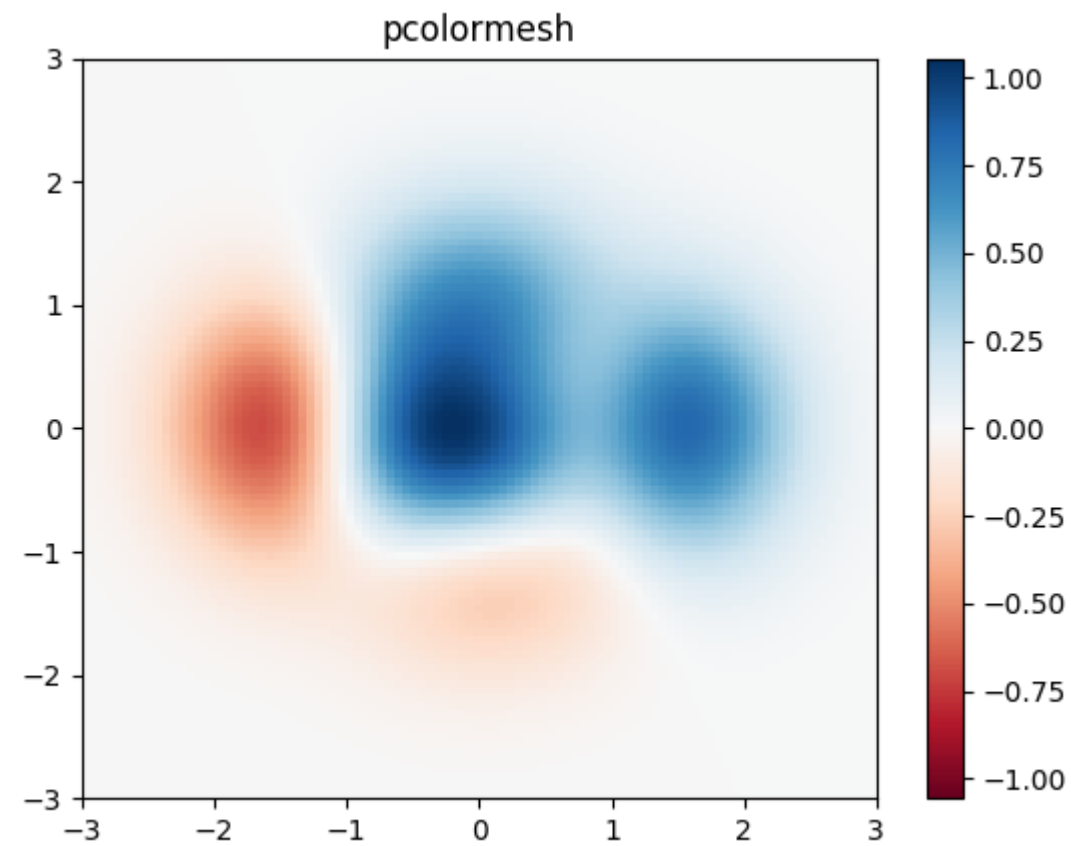
# generate 2 2d grids for the x & y bounds
y, x = np.meshgrid(np.linspace(-3, 3, 100), np.linspace(-3, 3, 100))

z = (1 - x / 2. + x ** 5 + y ** 3) * np.exp(-x ** 2 - y ** 2)
# x and y are bounds, so z should be the value *inside* those bounds.
# Therefore, remove the last value from the z array.
z = z[:-1, :-1]
z_min, z_max = -np.abs(z).max(), np.abs(z).max()

fig, ax = plt.subplots()

c = ax.pcolormesh(x, y, z, cmap='RdBu', vmin=z_min, vmax=z_max)
ax.set_title('pcolormesh')
# set the limits of the plot to the limits of the data
ax.axis([x.min(), x.max(), y.min(), y.max()])
fig.colorbar(c, ax=ax)
```

```
plt.show()
```



answered Jan 8 at 9:36



Erasmus Cedernaes

597 9 9

▲ I'm trying to a similar plot with X, Y and Z (intensity) specified as lists. It generate the output: [stackoverflow.com/questions/54167159/...](https://stackoverflow.com/questions/54167159/...) – tandem Jan 13 at 8:48

▲ Here's how to do it from a csv:

12

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import griddata

# Load data from CSV
dat = np.genfromtxt('dat.xyz', delimiter=' ', skip_header=0)
X_dat = dat[:,0]
Y_dat = dat[:,1]
Z_dat = dat[:,2]

# Convert from pandas dataframes to numpy arrays
X, Y, Z, = np.array([]), np.array([]), np.array([])
for i in range(len(X_dat)):
    X = np.append(X, X_dat[i])
    Y = np.append(Y, Y_dat[i])
    Z = np.append(Z, Z_dat[i])

# create x-y points to be used in heatmap
xi = np.linspace(X.min(), X.max(), 1000)
yi = np.linspace(Y.min(), Y.max(), 1000)
```

```
# Z is a matrix of x-y values
zi = griddata((X, Y), Z, (xi[None,:], yi[:,None]), method='cubic')

# I control the range of my colorbar by removing data
# outside of my range of interest
zmin = 3
zmax = 12
zi[(zi<zmin) | (zi>zmax)] = None

# Create the contour plot
CS = plt.contourf(xi, yi, zi, 15, cmap=plt.cm.rainbow,
                  vmax=zmax, vmin=zmin)
plt.colorbar()
plt.show()
```

where `dat.xyz` is in the form

```
x1 y1 z1
x2 y2 z2
...
```

edited Dec 10 '18 at 0:36

answered Sep 8 '16 at 22:46

 [kilojoules](#)

3,201 10 38 87

- 1
- ▲

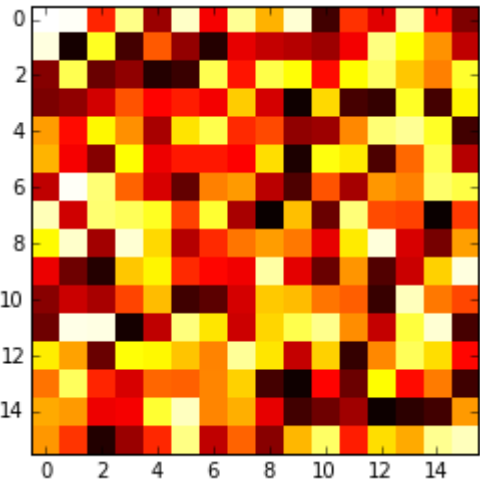
Just a short heads up: I had to change the method from cubic to either nearest or linear because the cubic resulted in a lot of NaNs since I'm working with rather small values between 0..1 – [Maikefer](#) Feb 9 '18 at 18:08

▲ The `imshow()` function with parameters `interpolation='nearest'` and `cmap='hot'` should do what you want.

139

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

a = np.random.random((16, 16))
plt.imshow(a, cmap='hot', interpolation='nearest')
plt.show()
```



edited Dec 6 '17 at 21:13

answered Oct 22 '15 at 13:44

 [Coquelicot](#)

3,720 4 23 31

 [P. Camilleri](#)

6,799 4 22 49

- ▲ I don't think specifying interpolation is necessary. – [miguel.martin](#) Mar 28 '17 at 6:12  
▼
- 2 ▲ @miguel.martin as per pyplot's doc: "If interpolation is None (its default value), default to rc image.interpolation". So I think it is necessary to include it. – [P. Camilleri](#) Mar 28 '17 at 7:11  
▼

▲ Here's how to do it from a csv:

0

```
fig = matplotlib.pyplot.gcf()
dat = np.genfromtxt('dat.xyz', delimiter='\t', skip_header=0)
X_dat = dat[:,0]
Y_dat = dat[:,1]
Z_dat = dat[:,2]
X, Y, Z, = np.array([]), np.array([]), np.array([])
for i in range(len(X_dat)):
    X = np.append(X,X_dat[i])
    Y = np.append(Y,Y_dat[i])
    Z = np.append(Z,Z_dat[i])

xi = np.linspace(X.min(),X.max(),100)
yi = np.linspace(Y.min(),Y.max(),100)

zi = griddata((X, Y), Z, (xi[None,:], yi[:,None]), method='cubic')
plt.imshow(zi)
plt.show()
```

where `dat.xyz` is in the form

```
x1 y1 z1
x2 y2 z2
...
```

deleted by owner Sep 8 '16 at 22:16

answered Sep 8 '16 at 22:13



[kilojoules](#)

3,201 10 38 87

Got a question that you can't ask on public Stack Overflow? [Learn more](#) about sharing private information with Stack Overflow for Teams.

