Save plot to image file instead of displaying it using Matplotlib (so it can be used in batch scripts for example)

I am writing a quick-and-dirty script to generate plots on the fly. I am using the code below (from Matplotlib documentation) as a starting point:

```
from pylab import figure, axes, pie, title, show

# Make a square figure and axes
figure(1, figsize=(6, 6))
ax = axes([0.1, 0.1, 0.8, 0.8])

labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'
fracs = [15, 30, 45, 10]

explode = (0, 0.05, 0, 0)
pie(fracs, explode=explode, labels=labels, autopct='%1.1f%%', shadow=True)
title('Raining Hogs and Dogs', bbox={'facecolor': '0.8', 'pad': 5})

show()  # Actually, don't show, just save to foo.png
```

I don't want to display the plot on a GUI, instead, I want to save the plot to a file (say foo.png) - how do I do that?

python matplotlib

edited Dec 23 '15 at 5:52

Martin Thoma
13.3k 14 104 221

asked Mar 8 '12 at 17:38

Homunculus Reticulli
9,476 36 106 183

40 Looks like I found the answer: its pylab.savefig('foo.png') – Homunculus Reticulli Mar 8 '12 at 17:42

1 Link should maybe link to somewhere in matplotlib.org? – A.Wan Dec 10 '15 at 19:40

5 Also if not using pylab, the figure object has a `savefig` method too. So you can call `fig = plt.figure()` then `fig.savefig(...)` . – A.Wan Dec 10 '15 at 19:43

10 Answers

While the question has been answered, I'd like to add some useful tips when using savefig. The file format can be specified by the extension:

```
savefig('foo.png')
savefig('foo.pdf')
```

Will give a rasterized or vectorized output respectively, both which could be useful. In addition, you'll find that `pylab` leaves a generous, often undesirable, whitespace around the image. Remove it with:

```
savefig('foo.png', bbox_inches='tight')
```

edited Feb 5 '14 at 14:08

lorenzo
820 7 18

answered Mar 27 '12 at 13:35

Hooked
27.2k 12 91 143

14 `bbox_inches=0` does not work on my 64-bit Windows 7 system. Instead I used: `bbox_inches='tight'` , which does the trick. – Zhubarb Sep 13 '13 at 14:00

5 bbox_inches='tight' also worked for me on ubuntu 12.04 – Matt Klein Sep 19 '13 at 13:49

Is it possible to change the dimensions of the resulting image? – Llamageddon Oct 28 '13 at 21:15

6 @Asmageddon In `plt.savefig` you can change the dpi, see the link in the answer. The dimensions can be controlled when creating the figure, see `figsize` in

matplotlib.org/api/figure_api.html#matplotlib.figure.Figure – Hooked Oct 29 '13 at 0:46

@Hooked plt.savefig saves the figure but is does not prevent displaying it. Even when I leave out plt.show() the figure is displayed. How can I prevent that? – MoTSCHIGGE Aug 20 '14 at 11:46

The solution is:

```
pylab.savefig('foo.png')
```

edited Oct 19 '14 at 7:37
Peter Mortensen
9,913 12 68 100

answered Mar 27 '12 at 11:36
Lukasz Czerwinski
2,444 4 19 34

As others have said, `plt.savefig()` or `fig1.savefig()` is indeed the way to save an image.

However I've found that in certain cases (eg. with Spyder having `plt.ion()` : interactive mode = On) the figure is always shown. I work around this by forcing the closing of the figure window in my giant loop, so I don't have a million open figures during the loop:

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots( nrows=1, ncols=1 )  # create figure & 1 axis
ax.plot([0,1,2], [10,20,3])
fig.savefig('path/to/save/image/to.png')   # save the figure to file
plt.close(fig)    # close the figure
```

edited Oct 20 '15 at 6:12

answered Apr 28 '15 at 22:35
Demis
685 8 16

You could also set `plt.ioff() # turn of interactive plotting mode` , but that might disable behaviour you would want to use should your code exit with an error. – Demis Dec 14 '15 at 19:04

If you don't like the concept of the "current" figure, do:

```
import matplotlib.image as mpimg

img = mpimg.imread("src.png")
mpimg.imsave("out.png", img)
```

edited Oct 19 '14 at 7:38

Peter Mortensen
9,913 12 68 100

answered Jan 30 '14 at 18:30

wonder.mice
2,624 2 16 26

Doesn't this just copy `src.png` to `out.png` ? – gerrit May 26 at 11:44

That's just an example, that shows if you have an image object (`img`), then you can save it into file with `.imsave()` method. – wonder.mice May 26 at 23:29

Just found this link on the MatPlotLib documentation addressing exactly this issue: http://matplotlib.org/faq/howto_faq.html#generate-images-without-having-a-window-appear

They say that the easiest way to prevent the figure from popping up is to use a non-interactive backend (eg. Agg), via `matplotib.use(<backend>)` , eg:

```
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
plt.plot([1,2,3])
plt.savefig('myfig')
```

I still personally prefer using `plt.close( fig )` , since then you have the option to hide certain figures (during a loop), but still display figures for post-loop data processing. It is probably slower than choosing a non-interactive backend though - would be interesting if someone tested that.

answered Jan 4 at 0:35

Demis
685 8 16

```
import datetime
import numpy as np
from matplotlib.backends.backend_pdf import PdfPages
import matplotlib.pyplot as plt

# Create the PdfPages object to which we will save the pages:
# The with statement makes sure that the PdfPages object is closed properly at
# the end of the block, even if an Exception occurs.
with PdfPages('multipage_pdf.pdf') as pdf:
    plt.figure(figsize=(3, 3))
    plt.plot(range(7), [3, 1, 4, 1, 5, 9, 2], 'r-o')
    plt.title('Page One')
    pdf.savefig()  # saves the current figure into a pdf page
    plt.close()

    plt.rc('text', usetex=True)
    plt.figure(figsize=(8, 6))
    x = np.arange(0, 5, 0.1)
    plt.plot(x, np.sin(x), 'b-')
    plt.title('Page Two')
    pdf.savefig()
    plt.close()

    plt.rc('text', usetex=False)
    fig = plt.figure(figsize=(4, 5))
    plt.plot(x, x*x, 'ko')
    plt.title('Page Three')
    pdf.savefig(fig)  # or you can pass a Figure object to pdf.savefig
    plt.close()

    # We can also set the file's metadata via the PdfPages object:
    d = pdf.infodict()
    d['Title'] = 'Multipage PDF Example'
    d['Author'] = u'Jouni K. Sepp\xe4nen'
    d['Subject'] = 'How to create a multipage pdf file and set its metadata'
    d['Keywords'] = 'PdfPages multipage keywords author title subject'
    d['CreationDate'] = datetime.datetime(2009, 11, 13)
    d['ModDate'] = datetime.datetime.today()
```

answered Jun 30 '15 at 8:38

Victor Juliet
214 2 16

The Solution :

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
matplotlib.style.use('ggplot')
ts = pd.Series(np.random.randn(1000), index=pd.date_range('1/1/2000', periods=1000))
ts = ts.cumsum()
plt.figure()
ts.plot()
plt.savefig("foo.png", bbox_inches='tight')
```

If you do want to display the image as well as saving the image use:

```
%matplotlib inline
```

after `import matplotlib`

edited Mar 27 at 8:59

answered Mar 27 at 8:49

Durgesh satam
41 3

I used the following:

```
import matplotlib.pyplot as plt

p1 = plt.plot(dates, temp, 'r-', label="Temperature (celsius)")
p2 = plt.plot(dates, psal, 'b-', label="Salinity (psu)")
plt.legend(loc='upper center', numpoints=1, bbox_to_anchor=(0.5, -0.05),        ncol=2,
fancybox=True, shadow=True)

plt.savefig('data.png')
plt.show()
f.close()
plt.close()
```

I found very important to use plt.show after saving the figure, otherwise it won't work.figure exported in png

edited Apr 5 at 13:58

answered Apr 5 at 13:34

Cristian Muñoz
68 5

If, like me, you use Spyder IDE, you have to disable the interactive mode with :

```
plt.ioff()
```

(this command is automatically launched with the scientific startup)

If you want to enable it again, use :

```
plt.ion()
```

answered Aug 29 '15 at 14:50

Covich
395 1 9

The other answers are correct. However, I sometimes find that I want to open the figure *object* later. For example, I might want to change the label sizes, add a grid, or do other processing. In a perfect world, I would simply rerun the code generating the plot, and adapt the settings. Alas, the world is not perfect. Therefore, in addition to saving to PDF or PNG, I add:

```
with open('some_file.pkl', "wb") as fp:
    pickle.dump(fig, fp, protocol=4)
```

Like this, I can later load the figure object and manipulate the settings as I please.

I also write out the stack with the source-code and `locals()` dictionary for each function/method in the stack, so that I can later tell exactly what generated the figure.

answered May 26 at 11:48

gerrit
3,105 1 16 51
