How To Put Legend outside of Axes Properly in Matplotlib?

2018-01-23·Python·714 words ·4 mins read ·1030529 times read

When we want to put legend somewhere in a figure using Matplotlib, most of the time, the option `loc='best'` will produce the desired results. However, sometimes, we may want to have finer control over where the legend should be in the image. For example, we may want to put the legend outside of the axes, which is impossible using `loc='best'` .

Put legend in your desired position

According to the documentation of Axes.legend() method, we can use the parameters `loc` and `bbox_to_anchor` to control the position of the legend. Unfortunately, the documentation for the two parameters are not very clear.

In plain words, `bbox_to_anchor` accepts a list of four values: `(x0, y0, width, height)` . It will create a bounding box on the axes, inside which the actual legend will be placed. The lower left coordinate of the bounding box is `(x0, y0)` . Its width and height are given by `width` and `height` , respectively. Often, you will see that a list of only two values are given to `bbox_to_anchor` , which means that the width and height of the bounding box is zero.

The parameter `loc` denotes the alignment relationship between the actual legend box and the bounding box. It means that different position indicated by `loc` in both the legend box and bounding box will be put at the same point. For example, if `loc='center'` , the center of legend box and the bounding box will be at the same point. If `loc='center right'` , the center of the right edge of the legend box and the bounding box will be aligned (i.e., at the same point).

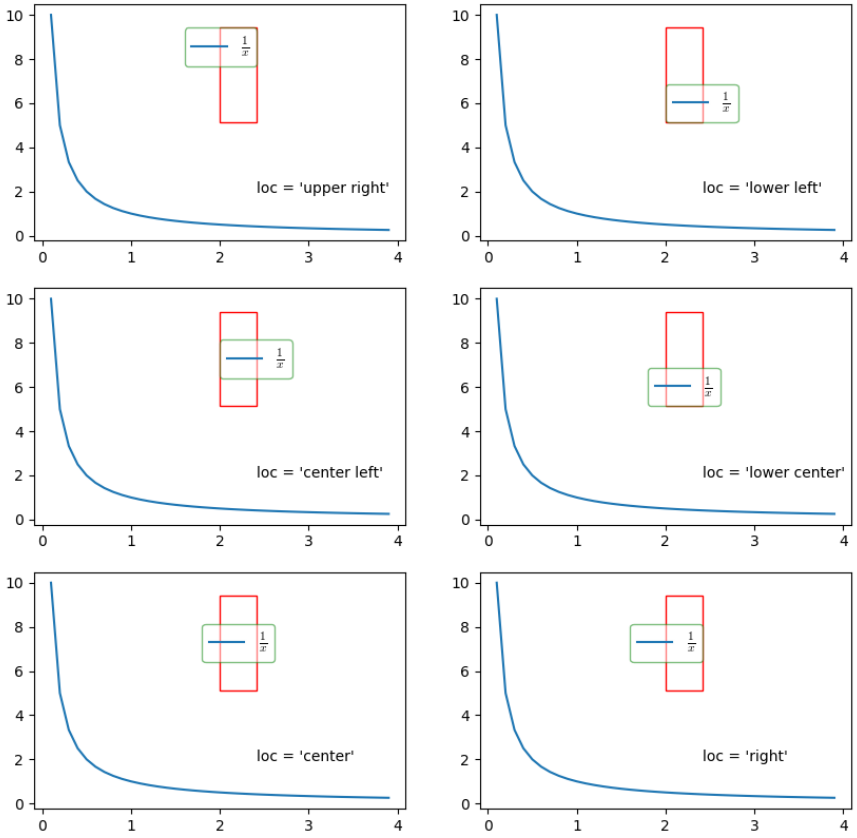
So much for the text. Let's take some examples to better illustrate the idea.

When there are four values for `bbox_to_anchor`

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

locs = ['upper right', 'lower left', 'center left', 'lower center', 'center',
        'right']

x0, y0, width, height = 0.5, 0.5, 0.1, 0.4

x = np.arange(0.1, 4, 0.1)
y = 1.0/x

fig = plt.figure(figsize=(10, 10))

idx = 1
for i in range(0, 2):
    for j in range(0, 3):
        ax = fig.add_subplot(3, 2, idx)
        ax.plot(x, y, label=r'$\frac{1}{x}$')
        ax.legend(loc=locs[idx-1], bbox_to_anchor=(x0, y0, width, height),
            edgecolor='g', fontsize='large', framealpha=0.5,
            borderaxespad=0)
        ax.add_patch(
            patches.Rectangle((x0, y0), width, height, color='r',
                              fill=False, transform=ax.transAxes)
        )
        ax.text(0.6, 0.2, s="loc = '{}'".format(locs[idx-1]),
        transform=ax.transAxes)
```

```
        idx += 1

plt.show()
```

The above code will produce the image below



In the above image, the red box is the bounding box, and the green box is the legend box. `loc` in each subplot indicates the alignment between the two boxes.

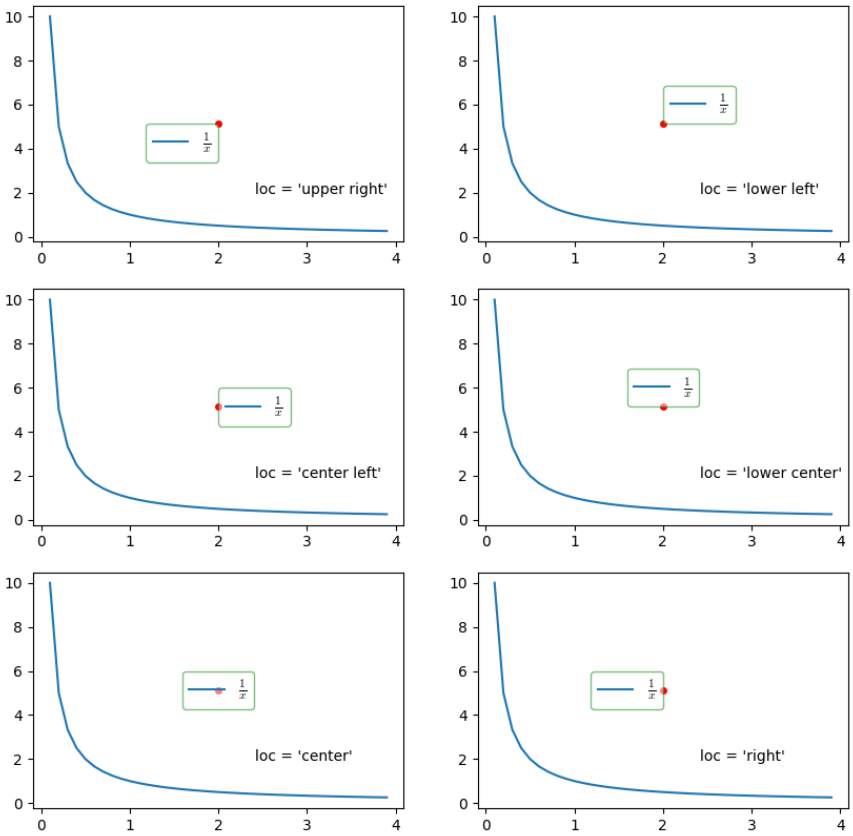
When there are only two values for `bbox_to_anchor`

When there are only two values given to `bbox_to_anchor` , the bounding box width and height are set to zero. So the bounding box will become a tiny little point. Let's modify the above code slightly,

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

locs = ['upper right', 'lower left', 'center left', 'lower center', 'center',
        'right']

x0, y0, width, height = 0.5, 0.5, 0, 0

x = np.arange(0.1, 4, 0.1)
y = 1.0/x

fig = plt.figure(figsize=(10, 10))

idx = 1
for i in range(0, 2):
    for j in range(0, 3):
        ax = fig.add_subplot(3, 2, idx)
        ax.plot(x, y, label=r'$\frac{1}{x}$')
        ax.legend(loc=locs[idx-1], bbox_to_anchor=(x0, y0, width, height),
            edgecolor='g', fontsize='large', framealpha=0.5,
            borderaxespad=0)
        ax.add_patch(
            patches.Rectangle((x0, y0), width, height, color='r',
                              fill=False, transform=ax.transAxes)
```

```
        )
        ax.text(0.6, 0.2, s="loc = '{}'".format(locs[idx-1]),
        transform=ax.transAxes)
        ax.plot(x0, y0, 'r.', markersize=8, transform=ax.transAxes)
        idx += 1

plt.show()
```

Now, the produced image becomes

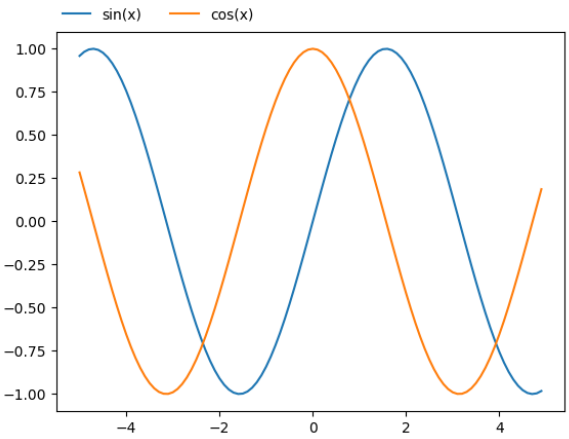


Put legend outside of the axes box

Now that we know how to position the legend, it is trivial to put it outside of the axes. See the example code below on how to do it,

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

x = np.arange(-5, 5, 0.1)
y1 = np.sin(x)
y2 = np.cos(x)

fig, ax = plt.subplots()
ax.plot(x, y1, label='sin(x)')
ax.plot(x, y2, label='cos(x)')

ax.legend(loc='lower left', bbox_to_anchor= (0.0, 1.01), ncol=2,
           borderaxespad=0, frameon=False)

plt.show()
```

The produced image is



Caveat

There is also a `fig.legend()` method which does similar things as the `ax.legend()` method, i.e., it can also put the legend outside of the axes. But, if you try to save the figure with its legend produced by `fig.legend()` using the option `bbox_inches='tight'`, the legend may not be present in the generate image file. This is a bug of Matplotlib. Right now, just stick to the `axes.legend()` method.

References

- https://stackoverflow.com/questions/39803385/what-does-a-4-element-tuple-argument-for-bbox-to-anchor-mean-in-matplotlib/48405821#48405821
- https://stackoverflow.com/questions/4700614/how-to-put-the-legend-out-of-the-plot?noredirect=1&lq=1
- https://stackoverflow.com/questions/48128546/why-is-the-legend-not-present-in-the-generated-image-if-i-use-tight-for-bbox-i

Author : jdhao
LastMod : 2020-03-18
License : CC BY-NC-ND 4.0

#Matplotlib

‹ LaTeX Equation Numbering Done Right in Hexo

Working with Fonts in Matplotlib ›