

CSCD01 - Team I/O

Initial Bug Report

Table of Contents

Pyplot hlines and vlines do not use the 'lines.color' property in rcParams by default · Issue #16482 · matplotlib/matplotlib -Jinming Zhang	3
Large markers escape legend · Issue #8705 · matplotlib/matplotlib - Omar Chehab	5
Bug Description	5
Code for reproduction	5
Result	5
Estimated Effort	6
“Size” ignored if placed before fontproperties · Issue #16389 · matplotlib/matplotlib - Robert Augustynowicz	7
https://github.com/matplotlib/matplotlib/issues/13799 - jinyanghu	9
Setting a thetalim > 2pi gives odd results · Issue #16501 · matplotlib/matplotlib - Dennis Tismenko	11
Bug Description	11
Code for reproduction	11
Result	11
Estimated Effort	12

[Pyplot hlines and vlines do not use the 'lines.color' property in rcParams by default · Issue #16482 · matplotlib/matplotlib](#) -Jinming Zhang

Terminologies:

- rcParams (runtime configuration parameters):
 - Used to specify a list of configurations for plotting the group of components(lines, axes, ticks...)
 - can be set using
 - Matplotlib.pyplot.rc, ie. plt.rc('lines', color='blue', linewidth=10)
 - plt.rc_context({dictionary of configurations}), ie. plt.rc_context({'lines.linewidth': 10})
- hlines / vlines: functions in matplotlib.pyplot to draw lines in a 2D coordinate space

Defect:

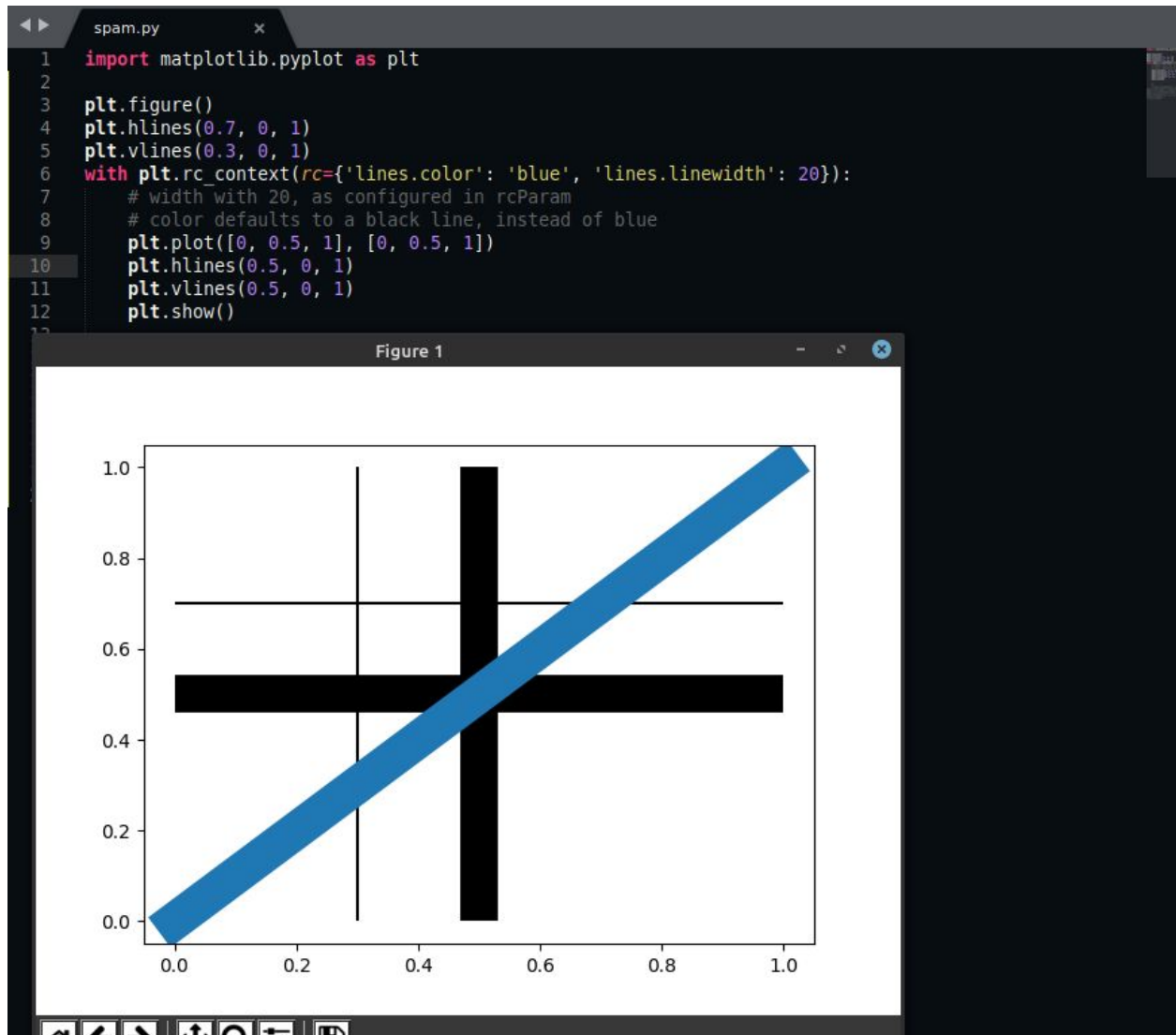
Currently, hlines and vlines do not use the 'color' property set by rcParams when 'color' is not provided in its parameters (other properties set by rcParams are applied properly).

For example, if rcParams is set as plt.rc('lines', color='blue', linewidth=10), then lines draw by hlines and vlines should have color blue and linewidth 10, if these two parameters are not provided when calling hlines and vlines. Currently, the two functions will draw lines of width 10 and default color of black, instead of blue.

Reproduce the defect:

Consider the following screenshot.

- The thin black cross is produced by the first two lines of vlines and hlines call, before the setting of rcParams
- Defect: After setting the rcParams (of blue line color and 20 line width), the similar vlines and hlines calls (at different start position) produce a thicker cross, which shows that width in rcParams correctly takes effect, whereas color did not.
- To further demonstrate the defect, the line produced by 'plot' function applied both color and width set by rcParams



Defect scope:

Although it is a relatively simple defect, it involves the fundamental usage of matplotlib and we will need to track the code through different layers to figure out the proper fix. Considering the short time we have, the fact that we are all new to matplotlib and open source projects, I think this is a great defect to start with to contribute and get ourselves familiar with the process of contributing to matplotlib project.

Time estimation:

Less than 10 hours.

[Large markers escape legend · Issue #8705 · matplotlib/matplotlib](#) - Omar Chehab

Bug Description

Increasing marker size leads to markers that escape the legend boundaries (or even the figure). Additionally, when there are multiple legend entries over each other, then they can overlap.

Code for reproduction

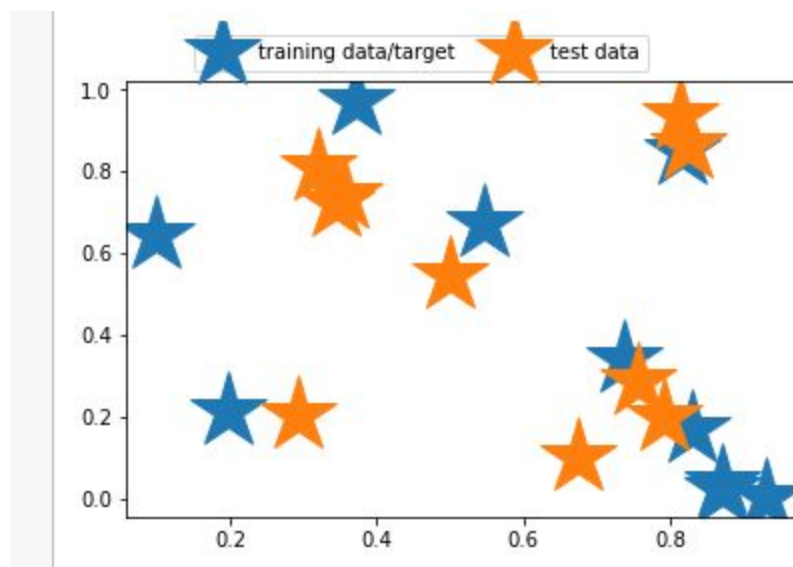
```
import numpy as np
import matplotlib.pyplot as plt

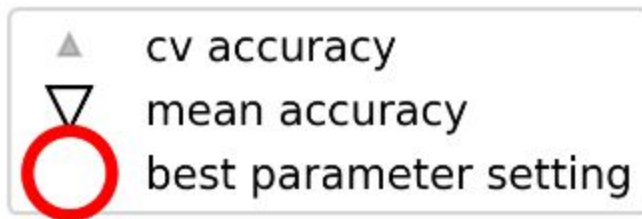
a, = plt.plot(np.random.uniform(size=10), np.random.uniform(size=10), '*',
markersize=40)

b, = plt.plot(np.random.uniform(size=10), np.random.uniform(size=10), '*',
markersize=40)

plt.legend([a, b],
           ["training data/target", "test data", "test prediction"],
           ncol=3, loc=(.1, 1.025))
```

Result





Estimated Effort

An attempt to close this issue due by a previous D01 team in 2018, there was a lot of discussion on their pull request <https://github.com/matplotlib/matplotlib/pull/10765>

We would need to read the discussion and understand what the maintainers expect from a solution to this problem.

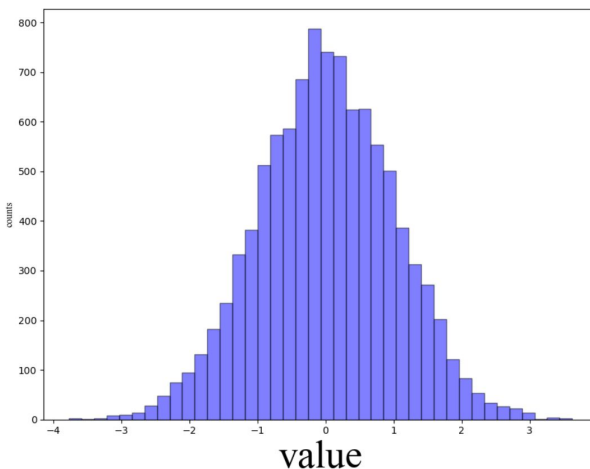
It's definitely a hairy problem that would require more than 20 hours to resolve and settle concerns.

[“Size” ignored if placed before fontproperties · Issue #16389 · matplotlib/matplotlib](#) - Robert Augustynowicz

When using the text class functionality which is a child to artist in matplotlib, a bug has been found in its implementation. The order of the given values which affects the text of such labels makes a difference, even though the variables themselves have been specified. It was found that if the font is specified first through fontproperties, then its size set without an issue, but if the font properties is specified after then the size argument will be ignored. This is exemplified by the code below which produces the graph shown, where the axis tests are different sizes while the only difference is order of parameters in the code.

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

data = np.random.randn(10000)
plt.hist(data, bins=40, facecolor="blue", edgecolor="black", alpha=0.5)
plt.xlabel("value", fontproperties='Times New Roman',size=40 ) # working command
plt.ylabel("counts", size=40, fontproperties='Times New Roman') # non working
plt.show()
```



The estimated time to complete would be approximately 4 hours. The first section would be documentation and specification which also requires careful line by line debugging as to find the issue so it can be planned to be properly fixed. This process should take up to 2 hours. Then implementation of the fix must be done which could be done in an estimated hour as much of the forethought has already been done in the design and specification phase. Finally a testing phase must occur which would take another hour to ensure no other bugs occurred due to the fix put into place.

<https://github.com/matplotlib/matplotlib/issues/13799> - jinyanghu

Bug description

When using variable colors for error bars, the colors are incorrect/shifted if `np.nan(None)` value is provided in x-axis and y-axis.

The estimated work need to complete

Need to modify the way `ErrorBars` object working with nan values. The correct fix is keeping all the Nan values instead of removing them from the axis. The previous PR removed Nan values before drawing and was rejected because the author simply removed the corresponding colour of invalid points before passing values to `Errorbar` project. The Estimated time is 5 hours.

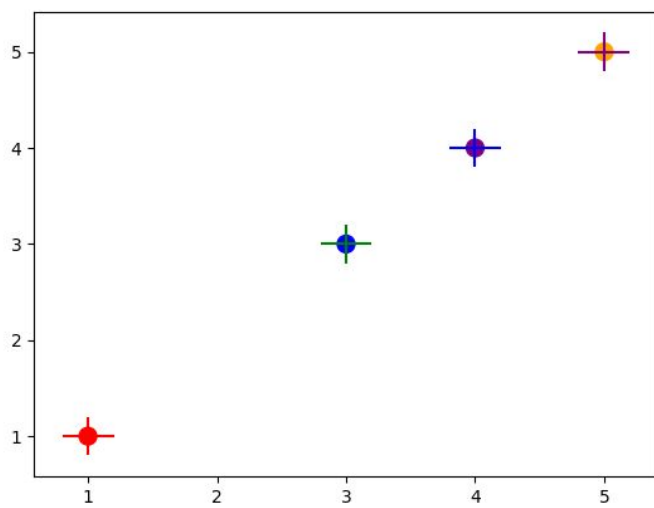
Code to reproduce bug

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

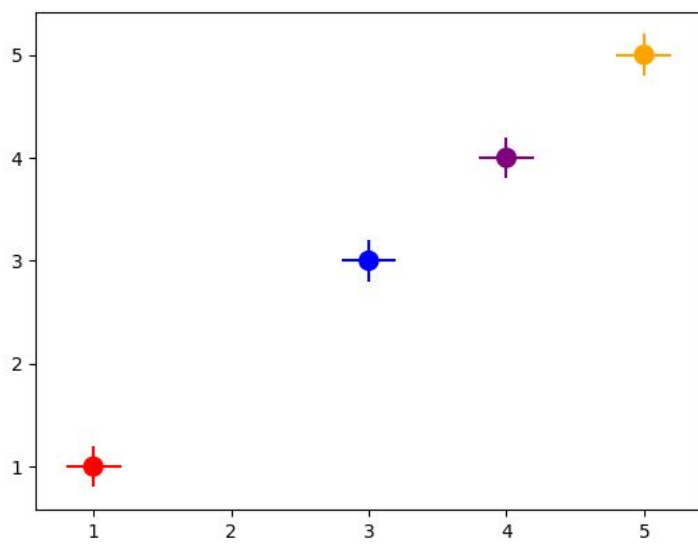
x = [1, np.nan, 3, 4, 5]
colors = ['red', 'green', 'blue', 'purple', 'orange']

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.scatter(x, x, c=colors, s=100)
ax.errorbar(x, x, xerr=0.2, yerr=0.2, ecolor=colors, fmt='none')
fig.savefig('errorbar_colors.png')
```

Actual output:



Expected output:



[Setting a thetalim > 2pi gives odd results · Issue #16501 · matplotlib/matplotlib](#) - Dennis Tismenko

Bug Description

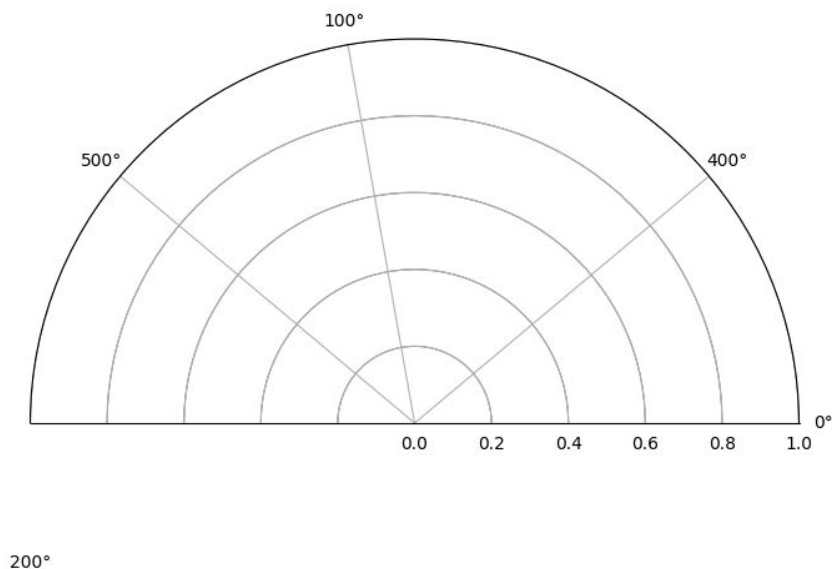
When plotting a graph using a polar projection (polar coordinate system) and modifying the viewing bounds of the PolarAxes with `set_thetalim`, out of bounds values are not being handled properly and produce an undesirable figure instead (see Result).

Code for reproduction

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

ax = plt.subplot(111, projection='polar')
ax.set_thetalim(0, 3 * np.pi)
plt.show()
```

Result



Estimated Effort

Resolving this issue would first require an understanding of the desired behaviour. In theory, there are many solutions to this problem, including:

- Displaying an error message to the user for exceeding the boundary range of 2π for a polar coordinate graph
- Using modulo to interpret and scale the input values to be within the boundary range [with a warning message]

Depending on the desired behaviour, the issue can take between 1-6 hours to resolve, with the lower bound of the estimate representing a simple solution to the issue (such as throwing an error message for illegal arguments), and the upper bound of the estimate corresponding to a more complex solution involving multiple classes (ie. PolarAxes, Scale).