

PROJECT DELIVERABLE 5

Team 4 (Sky Blue)

CSCD01

March 24, 2016

Team Members:

Chun Cho
Richard Luo
Yuxuan Hu
Geoffrey Hong
Kai Lin

Table of Contents

Section	Page
Feature 2 - Horizontal Stem Plot #5856	
Description	2
User Guide	2-4
Notes	4-5
Acceptance Test	5-6
Bonus Feature - Accepting Figure Arg. #6105	
Description	7
User Guide	-
Notes - N/A	-
Acceptance Test	8

Feature 2 - Horizontal Stem Plot #5856

Description

matplotlib only allows the drawing of vertical stem bar with horizontal base line. However, there are no equivalent of horizontal stem bars at the moment. The requester indicated that he wanted a way to implement a horizontal version of stem bars using a keyword like:

```
orientation = {"vertical" | "horizontal"}
```

We have implemented it as requested and the usage and resulting horizontal stem plot will be shown below.

User Guide

```
matplotlib.pyplot.stem(*args, **kwargs)
```

Create a stem plot.

Call signatures:

```
stem(y, linefmt='b-', markerfmt='bo', basefmt='r-')
stem(x, y, linefmt='b-', markerfmt='bo', basefmt='r-')

#Orientation indicate the newly added feature.
stem(x, y, linefmt='b-', markerfmt='bo', basefmt='r-', orientation = "vertical")
stem(x, y, linefmt='b-', markerfmt='bo', basefmt='r-', orientation = "horizontal")
```

By default, if the parameter orientation was not passed, the function stem plot will use vertical stem with horizontal baseline.

A vertical stem plot is a stem plot that plots vertical lines (using linefmt) at each x location from the baseline to y, and places a marker there using markerfmt. A horizontal line at 0 is plotted using basefmt.

A horizontal stem plot is a stem plot that plots a horizontal lines at each x location from the baseline to y.

If no x values are provided, the default is $(0, 1, \dots, \text{len}(y) - 1)$

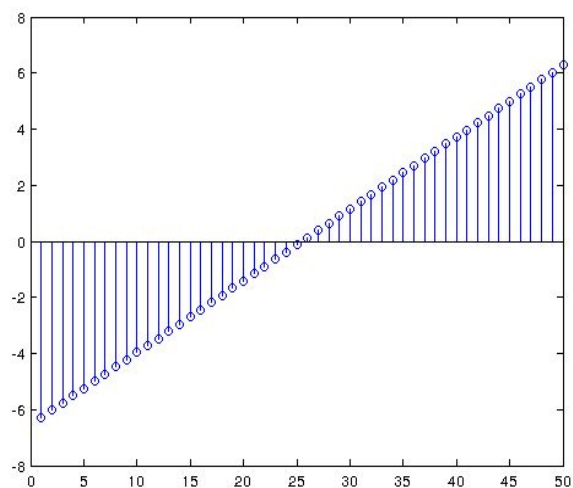
Return value is a tuple (*markerline*, *stemlines*, *baseline*).

See also

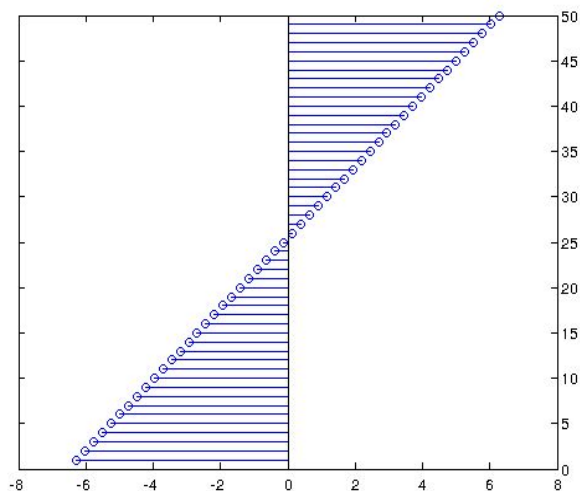
http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.stem for more detail on vertical stem with horizontal baseline.

Example:

1. Vertical Stem with Horizontal Baseline.



2. Horizontal stem with Vertical Baseline



Notes

The rest of the functionality are the same when plot using a vertical stem with horizontal baseline.

The only difference is horizontal stem and vertical stem. The rest functionality are the same.

User Acceptance Tests

Regression tests will be run in order to ensure that the introduction of the feature does not break pre-existing functionality.

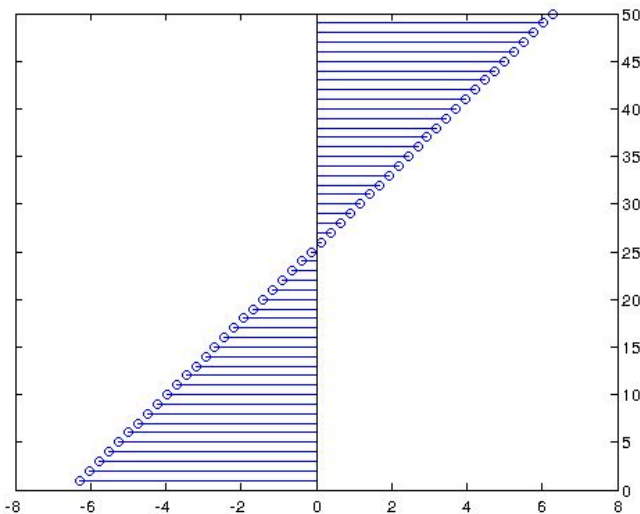
```
> python tests.py
```

Additionally, there will be tests to see if the below features functions as expected (by running `pyplot.stem()` to create stem plots):

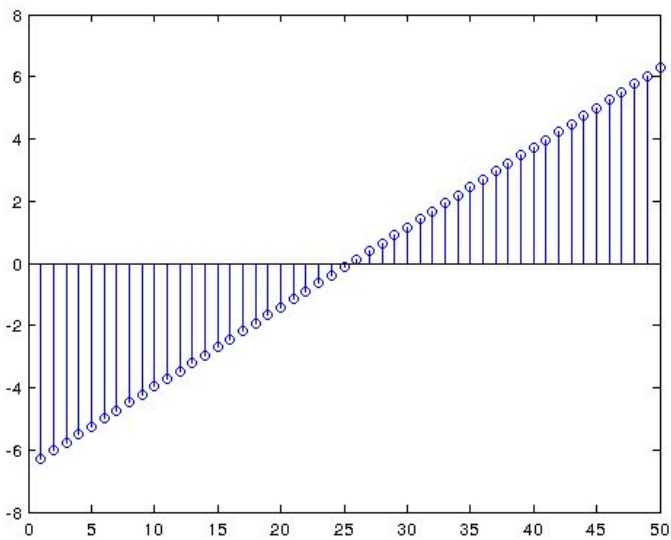
- New feature: horizontal stem with vertical baseline
- Existing feature: vertical stem with horizontal baseline

Expected outputs:

Horizontal stem with vertical baseline(New feature)



Vertical Stem with Horizontal baseline (Existing feature)



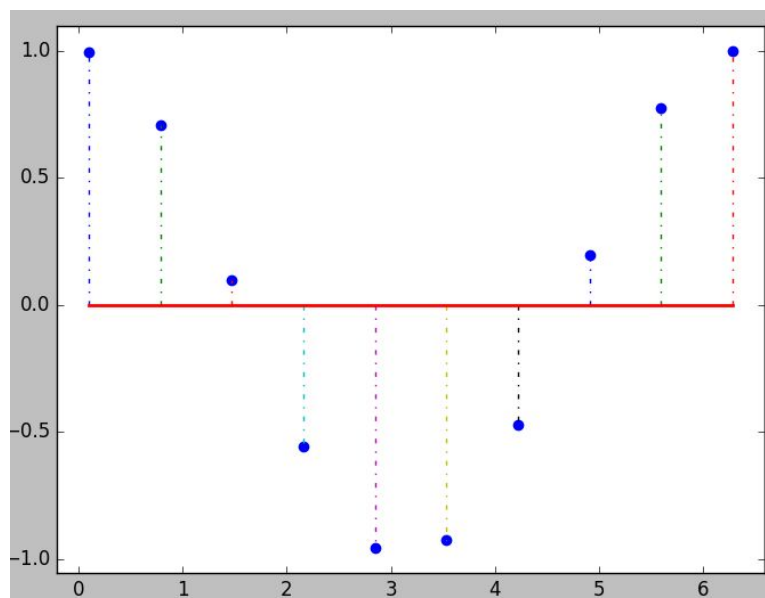
Unit-test Suite

Test case #1: Testing existing vertical stem plot

>python python vert_stem_horo_base_existing.py

Expected Result: There should be a vertical stem plot with horizontal baseline

Purpose: Ensure that additional augmentation does not break pre-existing functionality

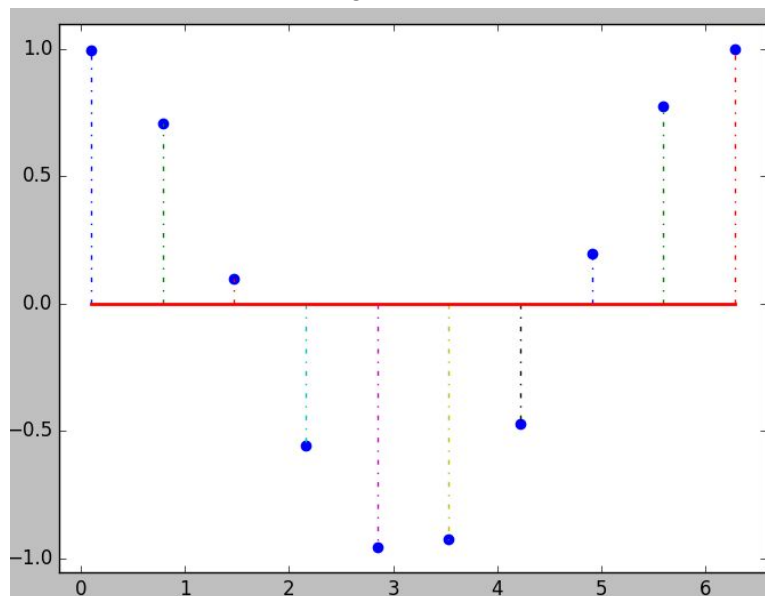


Test case #2: Testing vertical stem plot with orientation='vertical'

```
>python vert_stem_horo_base_new.py
```

Expected Result: There should be a vertical stem plot with horizontal baseline

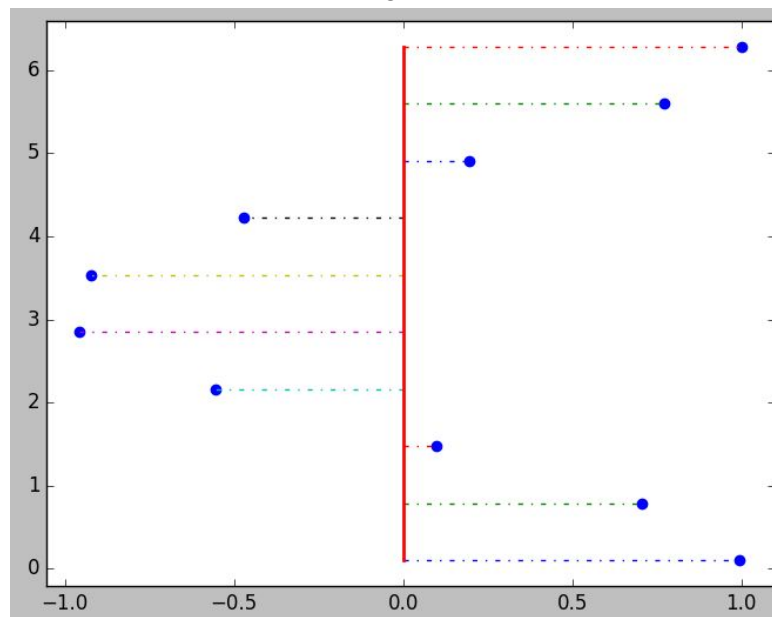
Purpose: Ensure user is able to create vertical stem plot with horizontal baseline while giving the orientation='vertical' argument.

**Test case #3:** Testing horizontal stem plot with orientation='horizontal'

```
>python horo_stem_vert_base_new.py
```

Expected Result: There should be a horizontal stem plot with vertical baseline

Purpose: Ensure user is able to create horizontal stem plot with vertical baseline while giving the orientation='horizontal' argument.



Bonus Feature - Accepting figure arg. in subplot2grid

#6105

Description:

Currently, the subplot2grid function in matplotlib forces users to use the current figure with `fig = gcf()` in the function's code. This results in various graphing limitations, such as the requirement to change the current figure every time in order to use subplot2grid. There is an easy way to remove this limitation. Adding a new argument in the subplot2grid function to allow users to pass in a figure, which the function will use.

```
def subplot2grid(shape, loc, rowspan=1, colspan=1, fig=None,
**kwargs):
```

Make `fig` to be `None` initially and assign it to current figure if user does not give any specific figure.

```
    if fig == None:
        fig = gcf()
```

User Guide

User can simply create a new figure and pass it as an argument when calling the subplot2grid function.

```
ax1 = plt.subplot2grid((3,3), (0,0), colspan=3, fig=plt.figure(0))
```

```
ax2 = plt.subplot2grid((3,3), (0,0), colspan=3, fig=plt.figure(1, facecolor='red'))
```

These functions will produce two different figures.

User Acceptance Tests

Regression tests will be run in order to ensure that the introduction of the feature does not break pre-existing functionality.

```
> python tests.py
```

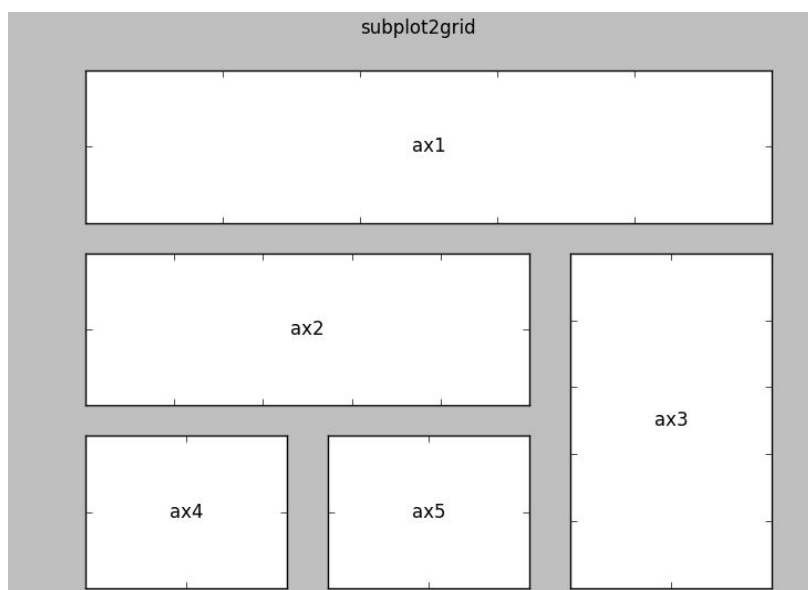

Unit-test Suite

Test case #1: Testing subplot2grid without phrasing figure argument

>python normal_subplot2grid_with_default_fig.py

Expected result: Produce an image of the current figure and have five axes with correct colspan and rowspan.

Purpose: To test previous existing functionality of subplot2grid, ensure that it uses current figure if no figure argument is passed into subplot2grid.



Test case #2: Testing subplot2grid with two different figures

>python subplot2grid_two_fig_argument.py

Expected result: Produce two different figures with title **Figure 0** and **Figure 5**

Purpose: Ensure new feature and it's function works, when passing a figure as an arg.

