CSC D01

11/10 (aka 110%)

Deliverable 4

By: Ben Cooper, Kirisanth Ganeshamoorthy, Matthew Kewarth, and Viola Ou

Table of Contents

Feature #1 - abLine

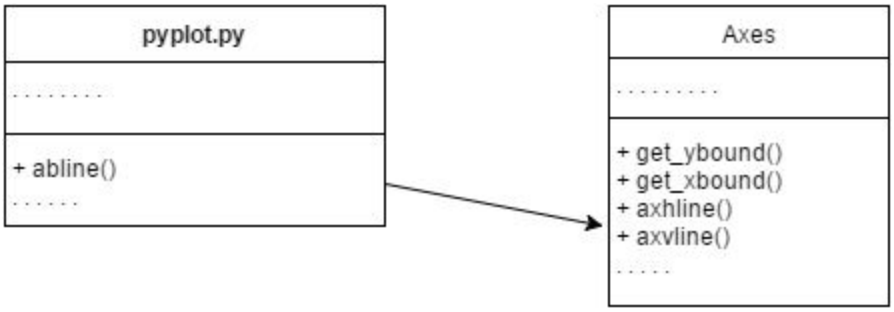
Link: https://github.com/matplotlib/matplotlib/issues/5253

Description

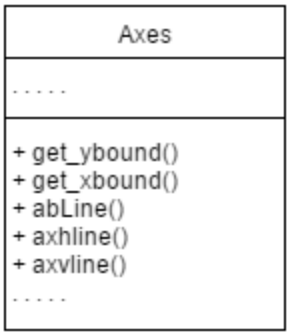
Currently, matplotlib does not allow the user to to create a straight line of arbitrary angle and interest for this feature has also been acknowledged on Stack Overflow as well. Users would like a method, `abLine`, that creates a straight line a variable angle based on the intercept, slope, and/or points along the line that the user inputs.

Implementation Plan

Currently in `Axes` two unique methods exists, `axhline` and `axvline`, for creating horizontal and vertical line along the axes. These two have provided useful hints into working towards creating a line spanning across the axes through their use of the `get_xbound` and `get_ybound` to acquire the limits of the axes object as well as how we could perform the required transformation onto the line object.



Our initial idea was to implement `abLine` inside of pyplot but after taking into consideration the existence of `axvline` and `axhline` this plan was scrapped to follow the developers' example.



Following the pattern `abLine` would be implemented inside the Axes class with `axhline` and `axvline`. `_axes.py` already imports the `lines` and `transformation` classes so we will not be increasing any dependencies. It should be noted that `axvline` and `axhline` can be seen as specific cases of `abline` and could later be rewritten to call upon `abline (a = [0,0], b = [0,1])` and `abline(a = [0,0], slope = 0)` respectively. Had we followed our initial plan this would have created an unnecessary dependency on pyplot and go against the 3-layer design of matplotlib.

Feature #2 - pyav based movie writer

Link: https://github.com/matplotlib/matplotlib/issues/4416

Description

A new Python wrapper for `ffmpeg` has been created to allows users to work with video files in python. Including reading them into a series of images and converting a series of images into videos. It has been requested that matplotlib could take advantage of this new library by allowing the user to export a series of images (created by matplotlib) into a movie file.

Implementation Plan

Our current plan is to implement this feature into pyplot, where we can use this library to take a series of images generated by `pyplot` and pass them the output to `pyav` and produce a video file. Since pyav handles most of the work, we only need to make a bridge between the two systems. `Pyplot.figure()` could be modified to keep track of all the figures being created and then, upon the user's request, all of the figures in the plot could be put into a movie file. There is an example of turning a series of images into a video file linked here: https://github.com/soft-matter/pims/blob/master/pims/display.py#L21

Our implementation will likely follow this example closely.

Feature #3 - Accepting figure argument in subplot2grid

Link: https://github.com/matplotlib/matplotlib/issues/6105

Description

The current method `subplot2grid` method cannot accept a figure argument into its constructor. A user wants to be able to pass their figure as an argument to `subplot2grid`. Right now, the function calls `gcf()` to simply get the current figure.

Implementation Plan

The implementation of this feature is straightforward, with adding the argument “gcf()” into the function declaration line of `subplot2grid` as an optional parameter. This will be done on line 1152 of `pyplot.py`.

Acceptance Testing

Acceptance Testing - abLine

After consideration we decided to select abLine as our primary goal, not only was abLine a highly desired and approachable feature with ample tips in the thread's comments and `axhline` and `axvline` making good examples for how to tackle this beast.

Following the request many suggestions were made for the input of abLine: two points in the line, a point and the slope, and the y-intercept and the gradient were the most commonly suggested. Implementing the y-intercept could result in the interesting case of providing, for example, `abLine(a= [0,0], y-intercept = 4)` to produce and vertical line and for each valid combination of characteristics the user could provide for the line we will need a test. At this point in time the inputs a, b, slope, and y-intercept are considered and the following tests would be required:

- `abLine(a = [0,0], b = [2,3])`
- `abLine(a = [0,0], slope = 3)`
- `abLine(b = [2,3], slope = 3)`
- `abLine(y-intercept = 4, b  = [5,0])`
- `abLine(y-intercept = 4, a = [5,0])`
- `abLine(y-intercept = 2, slope = 5)`
- `abLine(y-intercept = 2, a = [0,0])`
- `abLine(y-intercept = 2, b = [0,0])`

Each of these tests will require visual confirmation to ensure abLine is correctly computing each line.

Acceptance Testing - pyav based movie writer

As this is an animation feature we are working on, the acceptance testing stage will heavily rely on inputs and outputs of various tester plots and their supposed animated counterparts. The most sensible way to test this feature is with visual confirmation from a human. Testing this feature would involve creating a script that produces multiple figures before trying to output all of them into a video file.

Once the video file has been created, we will visually confirm that each figure is there and matches the ones plotted using pyplot.

Acceptance Testing - accepting figure argument in subplot2grid

Since this feature is an addition to `subplot2grid` it will not affect any current implementations out there. If we were to run the current test cases associated with `subplot2grid` it will return the expected output and prove that regression testing passes. We can test the addition of the feature by using `pyplot` to create output with multiple figures and then using `subplot2grid` on a figure which is no longer the current figure.