MATPLOTLIB: PROJECT DELIVERABLE 3

Team 16 (Do It Tomorrow)

CSCD01

March 5th, 2017

Team Members:
Mengzhe Lu
Jubin Patel
Derek Chow
Daniel Karas
Gauravjeet Kala

Table of Contents

Demonstration of Bugs

As a team, we have discussed and decided to look into the following issues found on the issue list of matplotlib:

GitHub Issue Link	Description
Issue 7523	When the text instances have the same parameters (coordinates, text string, alpha) except for the linespacing, the second one is displayed over the first one. The linespacing of the first instance is used for the second instance as well. ● Add self._linespacing to matplotlib.text.Text.get_prop_tup under text.py in lib folder
Issue 1656	If you create an animation and do not store it in a persistent variable, it will fail. Since the animation is not stored, python's garbage collection removes it since it is not explicitly stored. ● Save the animation in a parameter in Animation class under figure.py so that the garbage collector does not remove it
Issue 8089	When plotting time sequenced data, using matplotlib.dates.MinuteLocator(interval=1) to set the x-axis ticks results in a Runtime Error. ● Either increase locator.MAXTICKS or remove it altogether in ticker.py in the Locator class
Issue 7840	The fill_between color / facecolor / edegcolor resolution are not deterministic. Hatches in the fill_between function are not rendered with the Agg backend. ● Use the method normalize_kwargs in the __init__.py under the cbooks folder to eliminate all of the aliases and to resolve the colour/edgecolour/facecolour consistently
Issue 8059	When adding a BboxConnectorPatch, the edges are displayed correctly but the facecolor is not shown. The face is empty. ● Need to have a check in inset_locator.py to identify when face color is passed in and not passed as fill=False.
Issue 8128	The autofmt_xdate of the figure.Figure class by default only formats the major tick labels. Increased functionality including the minor ticks as well would be nice. ● Add which kwarg to the method figure.autofmt_xdate under figure.py to control which tick label gets rotated
Issue 6921	Having numpoints less than 1 isn't handled as it isn't supposed to allow numpoints less than 1. ● Raise an error to check if numpoints parameter passed into method get_xdata is an integer and not a float
Issue 6284	Create a figure and axis, and add a vspan to that axis. If you twinx it, and apply a plot to the twin, the x_limits can adjust, potentially making the vspan not visible. ● The autoscaling of the figure is currently not twin axis aware enough. When the plot is shown, only one of the graph is shown. Would start investigating the _base.py file under the axes folder as it deals with autoscaling of axes.

Issue 1656: Animations & Object Persistence

Description:

In issue 1656 of matplotlib, when creating an animation, if it is not stored in a persistent variable of the Animation class, the animation will not render. For example, entering the following code will work:

```
import matplotlib.pyplot as plt
from matplotlib import animation

fig = plt.figure()

def animate(i):
        # ... do something here

anim = animation.FuncAnimation(fig, animate)
plt.show()
```

However, if we were to change the final lines to just:

```
animation.FuncAnimation(fig, animate)
plt.show()
```

(i.e. not creating a variable to store the animation)

This change leads to the animation not rendering. This problem occurs because the figure does not store a reference to the animation object. Since there is no reference to the object, python's garbage collector will remove the animation.

Solution:

Our fix applies changes to two files in the libraries of matplotlib:

- In the branch issue-1656, under the file: matplotlib-master/lib/matplotlib/animation.py on line 871 and 872 we added the following lines:

```
# Save the animation so that garbage collector does not destroy it
fig.animations.append(self)
```

- Also in the branch issue-1656, under the file: matplotlib-master/lib/matplotlib/figure.py on line 360 we added the following lines:

```
self.animations = []
```

A description of our solution is that, we now maintain a list of animations inside the figure class while also appending any new animations which are created to the list of animations maintained in the figure class. This solution ensures that since there is a reference to the object, python's garbage collection will not remove the animations.

Technical Commentary:

From a coding stand point our changes have virtually no effect on the performance of matplotlib because as previously mentioned we introduced a new list inside the figure class. This change has no effect on any of the currently functioning components because it was never made use of. Similarly, the process of appending the animation to the list of animations associated with the current figure has no impact due to the line of code not being used or dependent on anything beforehand. If anything the only time we can see an issue occurring is if the user attempts to create more animations than what the system's memory can handle.

Now if we look at the design of the system, our code does affect the design of the system. Prior to our bug fix there was a one way coupling present between Animation and Figure. Since our code stored a list of animations in figure now there is a two way coupling present. Now a figure knows about an animation and similarly an animation knows about a figure.

Issue 6921: Value Check of ‘numpoints’

Description:

In issue 6921 of matplotlib, when creating a legend, a variable, ‘xdata’, of method get_xdata in legend_handler.py was being referenced before assignment due to invalid arguments being passed.
For example, when entering the following code:

```
1.    import matplotlib as mpl
2.    import matplotlib.pyplot as plt
3.    import numpy as np
4.    line_up, = plt.plot([1,2,3], label='Line 2')
5.    line_down, = plt.plot([3,2,1], label='Line 1')
6.    plt.legend(numpoints = 0.5)
```

Line 6, “plt.legend(numpoints = 0.5)” will cause the variable ‘xdata’ in legend_handler.py to not be assigned, in which will cause the following error since ‘xdata’ is being returned.

```
"UnboundLocalError: local variable 'xdata' referenced before assignment"
```

Solution:

Our fix required a change to one file in matplotlib:

- In the branch issue-6921, under the file: matplotlib-master/lib/matplotlib/legend_handler.py on lines 151 to 163 was where we added our fix. In our fix, we created the following check to make sure the arguments, specifically ‘numpoints’, that was being passed in was valid. We made changes to the if statement, and added:

```
if ((numpoints < 1) or not (isinstance(numpoints, int) or numpoints.is_integer())):
        raise ValueError("numpoints must be a whole number and greater" +
        " than or equal to 1; it was", numpoints)
```

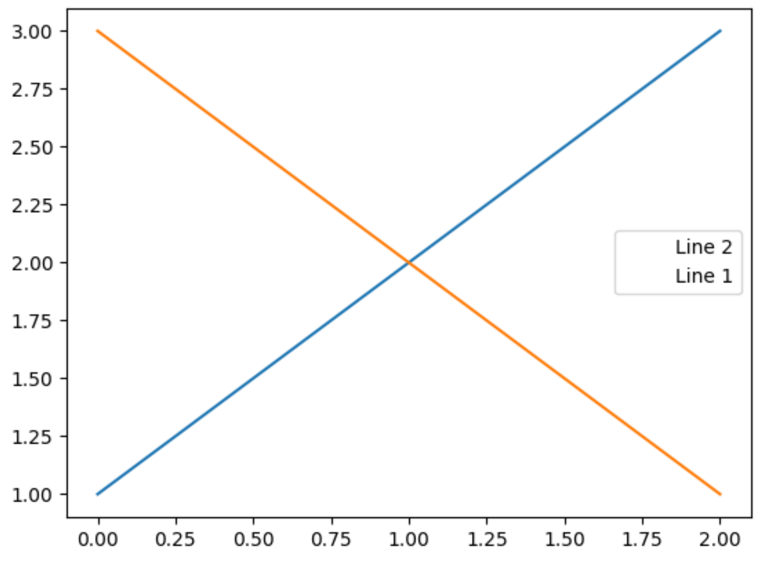
A description of our solution is that, we are now checking the arguments being passed in, specifically ‘numpoints’, is a whole number (integer). Also we made sure to handle the case where a user might enter ‘1.0’ or ‘2.0’ etc. which are whole numbers but are not recorded as an instance of *int*. Therefore we used the method is_integer() to handle such a case.
By doing this, we ensure that an error is properly raised when the argument passed in is invalid.

Technical Commentary:

From a coding stand point our changes carry a slight risk of being unable to use the method 'isinstance(---)' because the dependency, numpy, is configured differently when using integers. Other than this, our changes should not have any effect on the overall performance of matplotlib because as previously stated we are only adding in a check to see if the arguments passed are valid, and if they are not we would raise the appropriate error.

The changes should also have no effect on the current working functionalities of matplotlib and its components, as the code we've provided does not use these components to fix the issue at hand. Therefore we don't see our code changes affecting matplotlib in any major ways currently or in the future.

At first our team wanted to make sure the 'numpoints' parameter was greater than 1 and throw an error if it wasn't. However when we were testing and implementing the changes, we noticed that values between 1 and 2 caused a peculiar bug. When the parameter 'numpoints' was assigned a value between 1 and 2, the legend did not display the color indicators for each line. See image below for example:

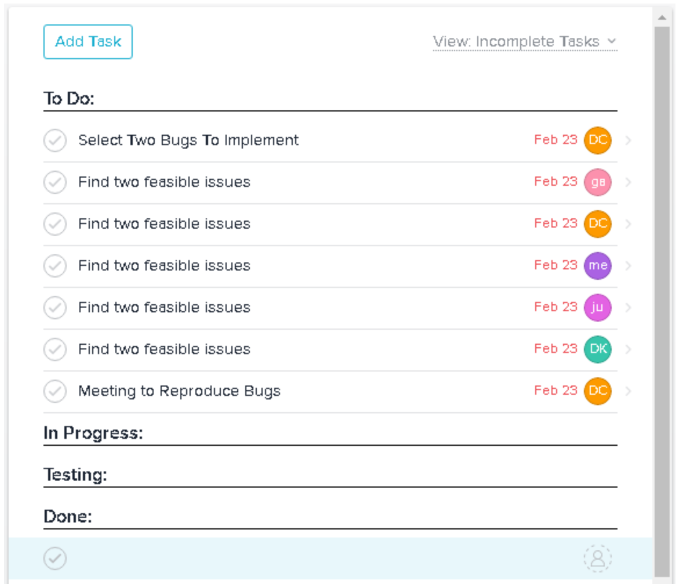


As you can see, a user is unable to tell which line is Line 1 or Line 2, since the legend isn't displaying the color associated to each line. When we discovered this, we decided to only accept whole numbers as we couldn't find any reason as to why someone would enter decimal values for the parameter 'numpoints', aside from 1.0 or 2.0 etc.

Project Management

We used Asana as our task board for our sprint and took screenshots at the beginning, middle and end of our sprint:

Beginning of Sprint:



Add Task

View: Incomplete Tasks

To Do:

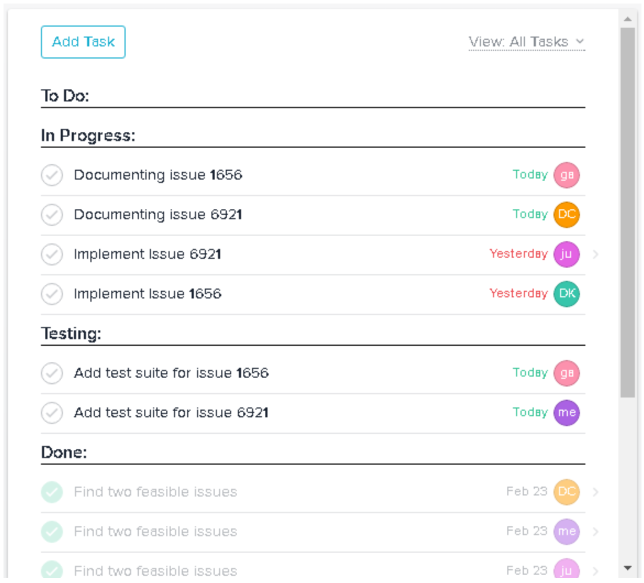
Select Two Bugs To Implement	Feb 23	DC
Find two feasible issues	Feb 23	ga
Find two feasible issues	Feb 23	DC
Find two feasible issues	Feb 23	me
Find two feasible issues	Feb 23	ju
Find two feasible issues	Feb 23	DK
Meeting to Reproduce Bugs	Feb 23	DC

In Progress:

Testing:

Done:

Middle of Sprint:



Add Task

View: All Tasks

To Do:

In Progress:

Documenting issue 1656	Today	ga
Documenting issue 6921	Today	DC
Implement Issue 6921	Yesterday	ju
Implement Issue 1656	Yesterday	DK

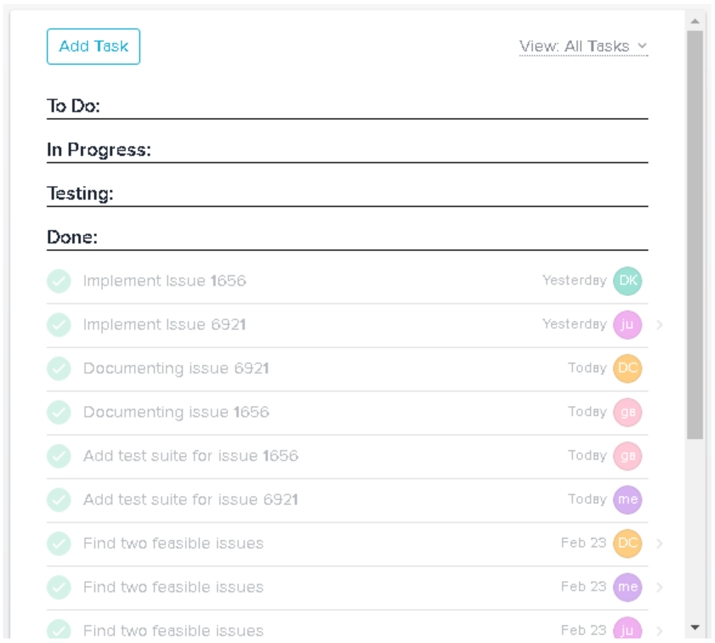
Testing:

Add test suite for issue 1656	Today	ga
Add test suite for issue 6921	Today	me

Done:

Find two feasible issues	Feb 23	DC
Find two feasible issues	Feb 23	me
Find two feasible issues	Feb 23	ju

End of Sprint:



Add Task

View: All Tasks

To Do:

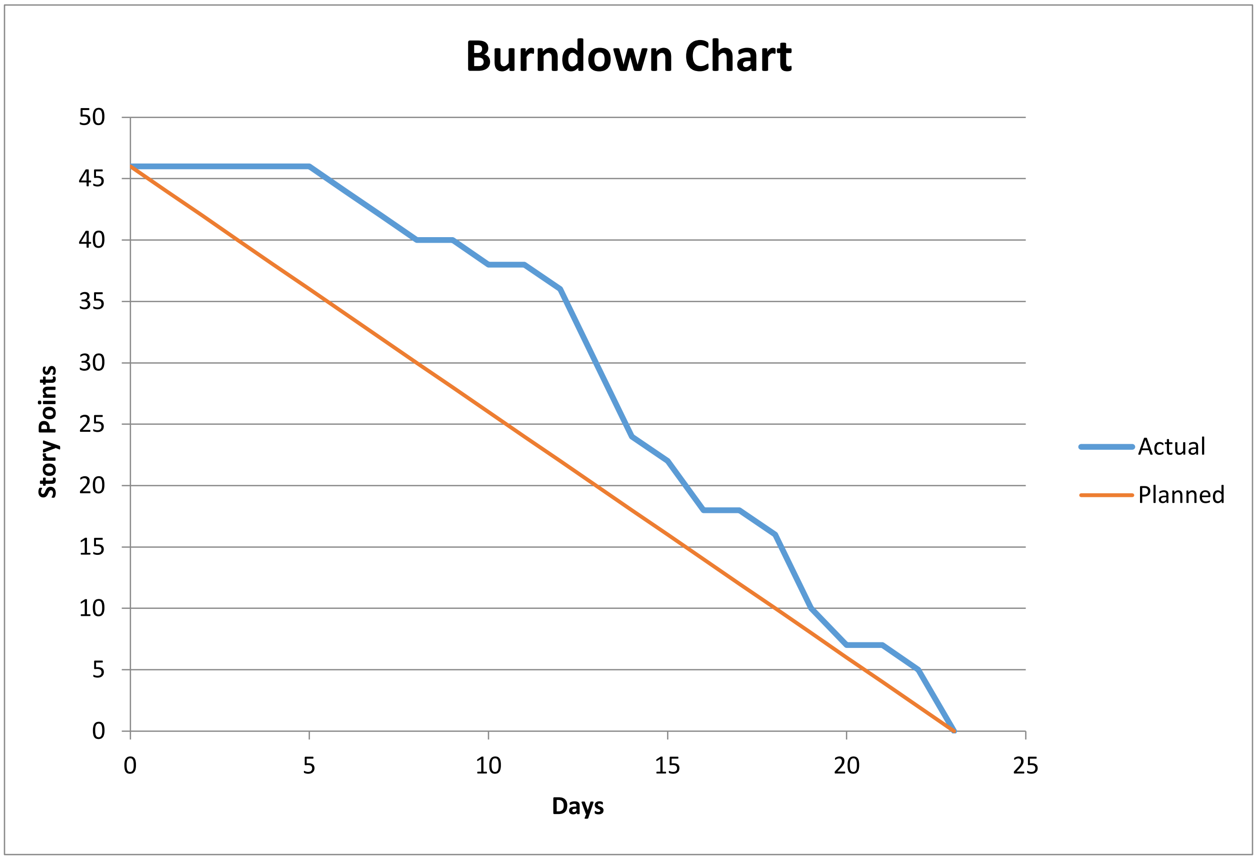
In Progress:

Testing:

Done:

Implement Issue 1656	Yesterday	DK
Implement Issue 6921	Yesterday	ju
Documenting issue 6921	Today	DC
Documenting issue 1656	Today	ga
Add test suite for issue 1656	Today	ga
Add test suite for issue 6921	Today	me
Find two feasible issues	Feb 23	DC
Find two feasible issues	Feb 23	me
Find two feasible issues	Feb 23	ju

Burndown Chart:



Note: Each story point is associated to one developer hour