Deliverable 2

Team 28 - Big Software Energy 2.0
Muhammad Farooq, Jadin Luong, Vincent Teng, Aster Wu
March 11th 2020

Table of Contents

List of Potential Issues

Potential Issue #1 (Selected)

GitHub Link: https://github.com/matplotlib/matplotlib/issues/16583

Issue: Matplotlibrc Validates Some Parameters [Incorrectly]

When users want to set a parameter globally, they do by creating a matplotlibrc file (using https://github.com/matplotlib/matplotlib/blob/master/matplotlibrc.template as a template) and set the parameters they need by uncommenting it in their matplotlibrc. To globally set the alignment of the x-axis ticks, we set the xtick.alignment parameter here: https://github.com/matplotlib/matplotlib/blob/master/matplotlibrc.template#L462

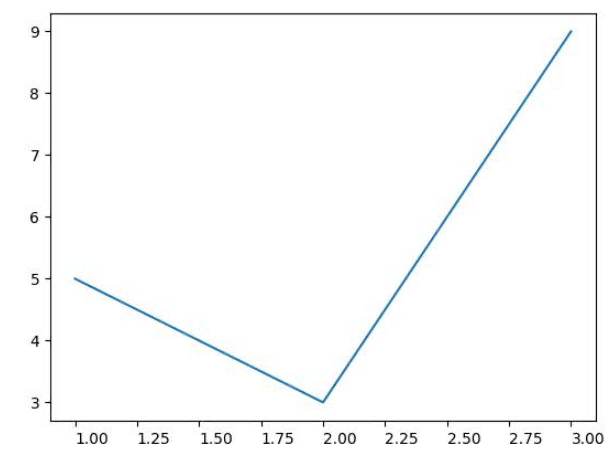
The default values for these parameters come from the rcserup.py file (https://github.com/matplotlib/matplotlib/blob/master/lib/matplotlib/rcsetup.py). This also where validation of the parameters takes place.

The issue here is that the validation for the xtick.alignment parameter does not validate against the correct values. The values xtick.alignment can take are ‘center’, ‘left’, and ‘right’ but as seen here, https://github.com/CSCD01/matplotlib-team28/blob/master/lib/matplotlib/rcsetup.py#L1344-L1345, rcsetup.py is validating against ‘center’, ‘top’, ‘bottom’, and so on which are not correct.

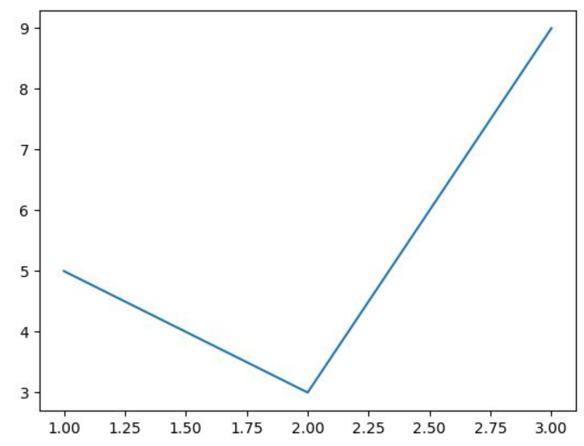
Therefore, whenever a user sets a valid value for this parameter other than ‘center’ (i.e. ‘left’ or ‘right’) and error is thrown as these are not amongst the parameters that are validated against and the x-axis tick alignment is still set to the default, center.

```
xtick.alignment:     left
```

Expected:



Actual



Estimated Work: The issue is well defined and straight forward to reproduce. An error message which provides a bit of code tracing also exists. Though it should be assumed a bit more investigation and testing is required to properly pin-point the source of the issue in code. Estimate 12 hours.

Risks: There may be other teams/open-source developer(s) interested in the issue, some form of compromise or competition may occur as a result. As for actual code base, the risk is minimal as the code for this is mainly restricted to a single file (rcsetup.py). The problem is also well-defined and a potential work-around was given in the comments for the issue.

Rationale: Issue looks reasonable to attempt, further supported by matplotlib developers assigning the issue to a team member that requested assignment of the issue on GitHub. The issue is a confirmed bug meaning there is a real need for development work to enhance the repository. The issue appears to be platform agnostic as opposed to some other issues we've considered which are user OS specific. The issue is also clearly defined, with minimal ambiguity in the task. Testing also appears to be very straightforward and clear for this task, as, after the fix is implemented, we simply need to test that the xtick parameter is set properly, the graphs have the correct xtick alignment, and the appropriate error is thrown when an invalid value is used for the parameter.

Potential Issue #2 (Stretch Goal)

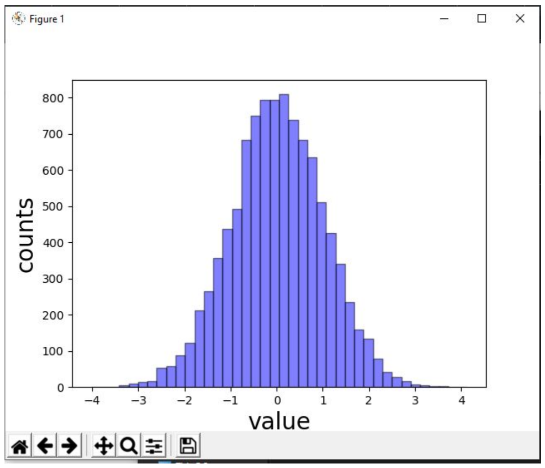
GitHub Link: https://github.com/matplotlib/matplotlib/issues/16389

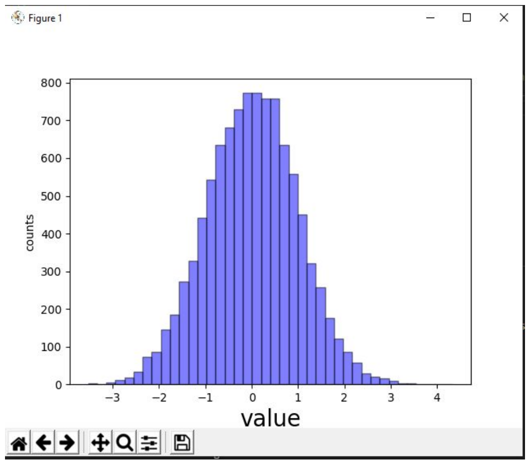
Issue: “Size” ignored if placed before fontproperties

When a method related to displaying and modifying components of a figure are used (axis labels, titles, etc.), the optional size parameter is overwritten when followed by the optional fontproperties parameter:

(*From* https://github.com/matplotlib/matplotlib/issues/16389):

```python
import matplotlib.pyplot as plt

import numpy as np



data = np.random.randn(10000)

plt.hist(data, bins=40, facecolor="blue", edgecolor="black", alpha=0.5)

plt.xlabel("value", fontproperties='SimHei', size=20,  ) # this will work

plt.ylabel("counts", size=20, fontproperties='SimHei', )  # this doesn't

plt.show()
```





(Actual) (Expected)

(Notice how the y-axis label is not set to size 20.)

This is because, when the fontproperties parameter is set, the set_fontproperties method of the FontProperties object is called. This sets the font size back to the default value (\lib\matplotlib\FontManager.py – FontProperties Class). So even if size is explicitly set, it will be overridden by setting the fontproperties parameter.

Setting the size parameter explicitly should take precedence regardless of the other parameters specified.

Estimated Work:

The problem is well defined and preliminary starting points are given in the issue. However, time will be taken in truly understanding the function calls being made. The assignee will need to investigate both the scripting and artist layer, in particular, pyplot.py, figure.py, and artist.py. Estimated work required will be about 16 hours.

Risk:

The problem may be larger than anticipated because the issue occurs on more than one property of the figure (not just the labels for the y axis demonstrated in the images above). As mentioned in the bug summary, the sizing issue may affect fields such as the x axis label, y axis label, title of the figure, etc.

Rationale:

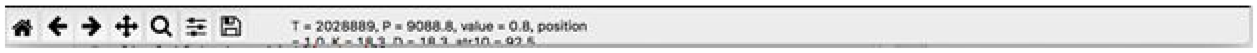
Extensive research may be required to fully understand the essence of the issue and figure out exactly what is being affected/needs to be changed which may increase the time required to solve this issue. Overall, the following issue is clearly stated and seems reasonable to tackle. Since we already know that the main issue relates to the placement of the “size” variable in a parameter and the issue is quite simple to reproduce, we can get a clear sense of how our test cases should be.

Potential Issue #3

GitHub Link: https://github.com/matplotlib/matplotlib/issues/16498

Issue: long string of format_coord in osx backend

When creating a figure, a status bar will also be provided at the bottom of the figure which will show the properties of the graph and allow users to analyze data correspondingly. Users can customize what is shown at the bottom of the status bar, but there is a bug with the text that may be shown within the bar (Users can change the variable “ax.format_coord” to change the text shown). When the customized text is too long, the status bar may cut off the text as shown below.



```
T = 2028889, P = 9088.8, value = 0.8, position
= 1.0, K = 18.3, D = 18.3, atr10 = 92.5
```

Figure 1: Status bar bug (Example from the issue link above)

As you can see from figure 1, the string of text is cut off on the second line of the string. The status bar is not adjusting itself to show the full string.

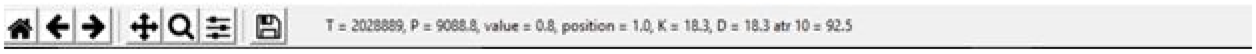

T = 2028889, P = 9088.8, value = 0.8, position = 1.0, K = 18.3, D = 18.3 atr 10 = 92.5

Figure 2: Potential expected result 1

In figure 2, we can see that the string of text is not cut off at all and shows the information clearly. This may be one of the results the users may expect.

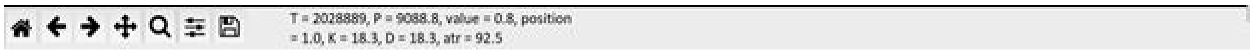

T = 2028889, P = 9088.8, value = 0.8, position
= 1.0, K = 18.3, D = 18.3, atr = 92.5

Figure 3: Potential expected result 2

Text in both figure 1 and figure 3 are very similar, but in figure 3, the string of text is fully shown and is not cut off which shows another result that may be expected. In this result, the text is automatically shifted upwards. Another expected result (similar to figure 3) is the status bar automatically adjusted to the size of the text.

Amount of work needed:

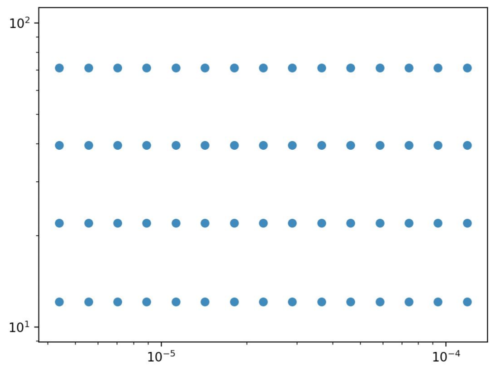
Overall, this bug seems quite simple but does require some research. It seems to be a lot more focused on the user interface. More specifically, the items that seem to be affected are the dimensions of the status bar and the dimensions of the dynamically generated text strings. Time estimation: 8 hours.

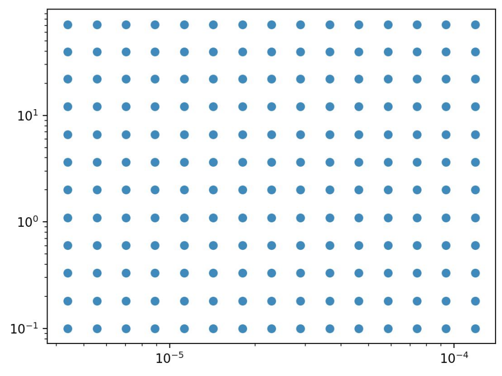
Potential Issue #4 (Selected)

GitHub Link: https://github.com/matplotlib/matplotlib/issues/16552

Issue: Scatter autoscaling still has issues with log scaling and zero values

The scatter plot scales differently compared to other types of plots when using log scaling and the set of x or set of y values contain a 0. The reproduced plots from the GitHub issue are shown below.





Scatter Plot *Manually Created Plot*

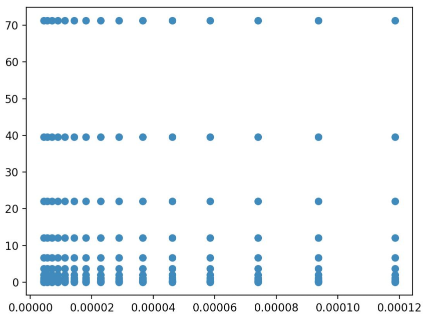
The plot on the left was created using the scatter() function like so:

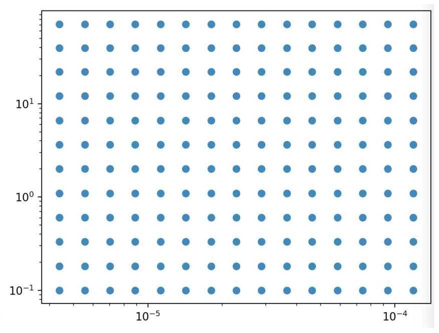
```
ax.scatter(pts[:,0], pts[:,1])
```

While the plot on the right was created manually using the plot() function like so:

```
ax.plot(pts[:,0], pts[:,1], marker="o", ls="")
```

Since the set of y values contains the 0, the amount of rows is much less than it should be. If the set of x values contains the 0, the number of columns would be much less than should be. We know they should be the same plot because when the log scaling or the 0 values are removed, the correct plots look like so:





Log Scaling Removed *0 Values Removed*

<u>Estimated Work:</u> The bug is very easy to reproduce and tests can be easily validated by comparing a manually created plot to the one created by scatter(). We will need to look into the scatter() method in the Axes class to determine what changes need to be made. It seems to be an issue related to how scatter() uses values of 0 when combined with log scaling or how show() . In order to fix it, the team will need to do some research into the methods and logic used by scatter() and determine the best possible fix. The estimated completion time is 10+ hours.

Risks: Since the issue is related to input values, any change in logic may result in changes to other types of scatter plots as well. To mitigate these risks, the specific consequences of changing logic must be clearly understood and test cases must be made for all types of scatter plots.

Rationale: The issue is simple to identify and reproduce. The issue is also very specific and the general consequences of modifying the logic can be understood. Since there is a method that will reproduce the right plot, testing the plot can be done easily by comparing the project plot with the correct plot. The issue also appears on all platforms so all of the developers on the team can work on the issue. In addition, no one has commented on the issue on GitHub which may indicate that no one is working on it.

Potential Issue #5

GitHub Link: https://github.com/matplotlib/matplotlib/issues/15039

Issue: NonUniformImage wrong image when using large values for axis

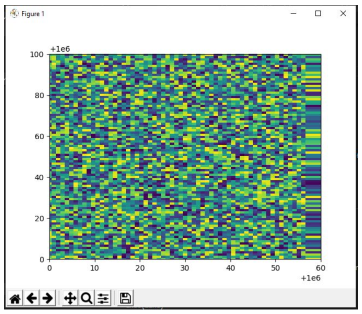
Using large values for the axis when plotting matplotlib images using `matplotlib.image.NonUniformImage` results in a broken graph.

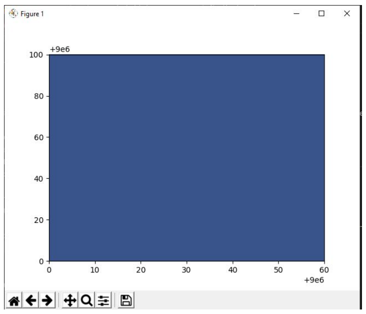
(From https://github.com/matplotlib/matplotlib/issues/15039)

```
import numpy
import matplotlib.pyplot as plt
from matplotlib.image import NonUniformImage

offset = 1.*10**6 # XXX change this XXX
nx = 60
ny = 100
x = numpy.arange(offset, offset+nx)
y = numpy.arange(offset, offset+ny)
z = numpy.random.rand(ny, nx)

ax = plt.subplot(111)
im = NonUniformImage(ax, extent=(offset, offset+nx, offset, offset+ny))
im.set_data(x, y, z)
ax.images.append(im)
ax.set_xlim(offset, offset+nx)
ax.set_ylim(offset, offset+ny)
```

```
plt.show()
```

(In offset variable, when set to 1*10^6) (In offset variable, when set to 9*10^6)





Estimated Work:

This will require significant work to debug as there are several possible locations to begin debugging. This could be an issue with pyplot not mapping data properly, or with dimensions of the Figure, or possibly with the rendering backend itself. Estimated time to be complete would be 30 hours.

Chosen Issues

Chosen Issue #1 - 16583

See above for a full description of the issue.

How we fixed it:

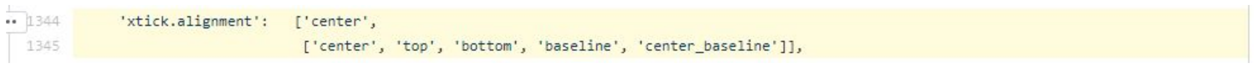
In matplotlib, all global parameters for plots are set at a default value in a file called rcsetup.py: https://github.com/CSCD01/matplotlib-team28/blob/d2/lib/matplotlib/rcsetup.py. This is also where parameter validation is done.

If users want to set their own defaults globally, they simply need to create a matplotlibrc file (based off of matplotrc.template: https://github.com/CSCD01/matplotlib-team28/blob/d2/matplotlibrc.template) and uncomment and set the values they want or they can use:

```
mpl.rcParams['xtick.alignment'] = <alignment-style>
```

In this case, the issue was with the xtick.alignment parameter:

Notice the xitck.alignment parameter in rcsetup.py before our code fix:

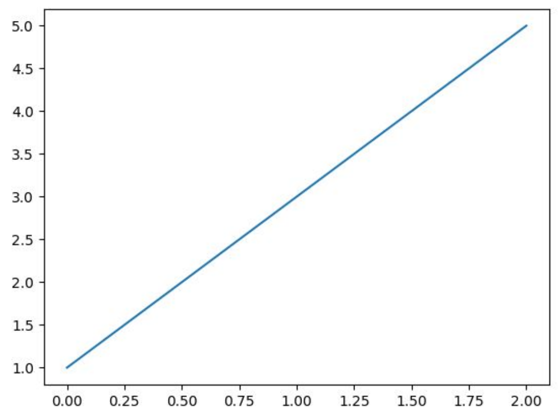


```
'xtick.alignment':   ['center',
                      ['center', 'top', 'bottom', 'baseline', 'center_baseline']],
```

The second argument is the list of possible values the xtick.alignment parameter can take. However, these validation parameters are not correct. The actual parameters that xtick.alignment can take are ‘center’, ‘left’, or ‘right’. So when the user was setting xtick.alignment parameter to something other than these values, like ‘top’ for example, it would throw a Value Error. But if the user tried to set it to one of the correct values, it would still throw a Value Error because it would be validating against these incorrect validation parameters.

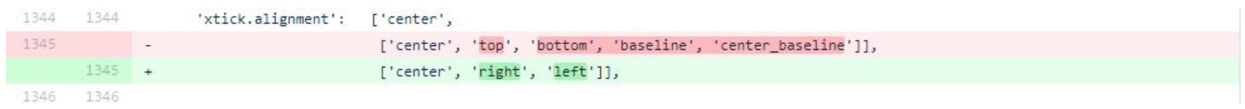
Example of plot when trying to set xtick.alignment in our matplotlibrc to anything:

```
xtick.alignment : left
```



Notice how the x-axis ticks are still centered. A Value Error is also thrown.

After our code fix:



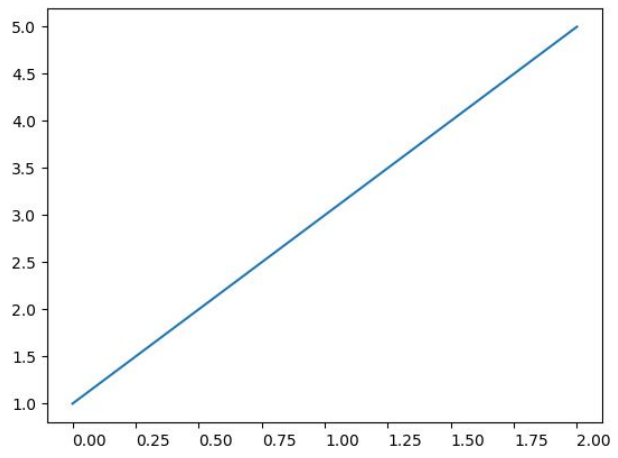
```
1344 1344       'xtick.alignment':   ['center',
1345      -                            ['center', 'top', 'bottom', 'baseline', 'center_baseline']],
     1345 +                            ['center', 'right', 'left']],
1346 1346
```

Code changes are located at https://github.com/CSCD01/matplotlib-team28/blob/98081a81721009e8f2053a52df2f7526be04d231/lib/matplotlib/rcsetup.py#L1329-L1330.

Now, if a user enters ‘center’, ‘left’, or ‘right’ in their matplotlibrc file or manually, no error is thrown and the x-axis ticks are aligned properly. If a user enters a value other than these, a Value Error is thrown as expected.

Example of plot when trying to set xtick.alignment in our matplotlibrc:

```
xtick.alignment : left
```



Notice how the x-axis ticks are now shifted to the left and no Value Error is thrown.

How we tested it:

Prerequisites

Python 3.6+
Pytest

If not done so already, install and build matplotlib from source (from our fork):

```
git clone https://github.com/CSCD01/matplotlib-team28.git
cd matplotlib-team28
git checkout d2
python -mpip install -ve .
```

Acceptance Testing

Check if x-axis ticks are shifted to the left of the x axis labels:

1) Create a plot however you like. You can also use the code snippet below:

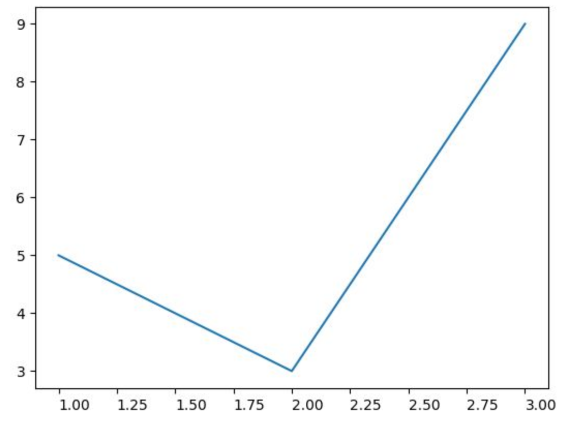
```
import matplotlib as mpl
import matplotlib.pyplot as plt

plt.plot([1,2,3], [5,3,9])
plt.show()
```

2) In the same directory as the file with the above code, create a file called matplotlibrc. Copy the contents from https://github.com/CSCD01/matplotlib-team28/blob/master/matplotlibrc.template and place it in matplotlibrc.

3) Locate the xtick.alignment parameter (https://github.com/CSCD01/matplotlib-team28/blob/master/matplotlibrc.template#L462) in your matplotlibrc file and change it to left.

```
xtick.alignment:     left
```

4) Run the file from above (python <filename>) and verify that the x-axis ticks are to the left of the labels. If using the above code snippet, the output should look like the following:

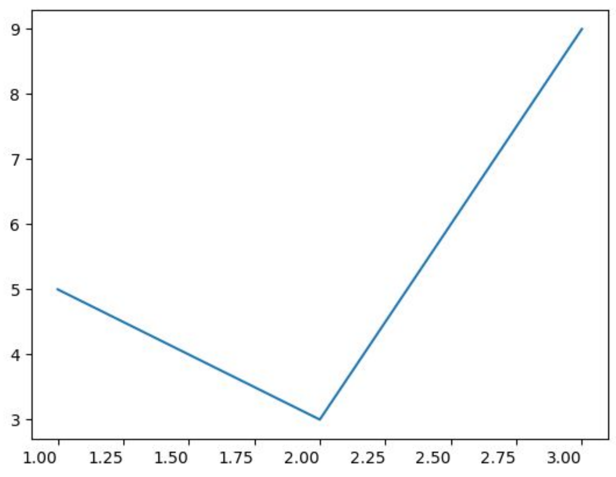


Check if x-axis ticks are shifted to the right of the x axis labels:

The steps for this are identical to the above. In step 3, simply set xtick.alignment to right not left.

```
xtick.alignment:     right
```

Verify that the x-axis ticks are to the left of the labels. If using the above code snippet, the output should look like the following:



Check if setting xtick.alignment to anything besides ‘left’, ‘right’, or ‘center’ produces an error:

Again, like above, in step 3, set xtick.alignment to anything (other than ‘left’, ‘right’, or ‘center’):

```
xtick.alignment:     top
```

Verify the following error appears:

ValueError: Key xtick.alignment: Unrecognized xtick.alignment string 'top': valid strings are ['center', 'right', 'left']

Unit Testing

Test cases are located at https://github.com/CSCD01/team_28-project/blob/master/deliverable_2/tests/bugfix-16583/test_xtick_alignment.py

To run the test cases:

```
git clone https://github.com/CSCD01/team_28-project.git
cd deliverable_2/tests/bugfix-16583
pip install pytest (if not done so already)
pytest test_xtick_alignment.py
```

We have seven test cases in total for this bug:

```
def test_xtickalign_validation_rcparams():
    with pytest.raises(ValueError):
        mpl.rcParams['xtick.alignment'] = 'top'
```

This test checks to see if Value Error is thrown when a value other than ‘center’, ‘left’, or ‘right’ is set for xtick.alignment parameter.

```
@check_figures_equal()
def test_left_xtick(fig_test, fig_ref):
    labels = range(0, 100, 10)

    fig_test.subplots()
    ax1 = fig_test.axes
    ax1[0].set_xticklabels(labels, ha='left')

    mpl.rcParams['xtick.alignment'] = 'left'
    fig_ref.subplots()
    ax2 = fig_ref.axes
    ax2[0].set_xticklabels(labels)
```

This test checks to see if the x-axis ticks are properly shifted to the left when the xtick.alignment parameter is set to left. We use a proven method of aligning x-axis ticks in set_ticklabels and compare the figures, one using set_ticklabels and one using the rc params variable.

```
@check_figures_equal()
def test_right_xtick(fig_test, fig_ref):
    labels = range(0, 100, 10)

    fig_test.subplots()
    ax1 = fig_test.axes
    ax1[0].set_xticklabels(labels, ha='right')

    mpl.rcParams['xtick.alignment'] = 'right'
    fig_ref.subplots()
    ax2 = fig_ref.axes
    ax2[0].set_xticklabels(labels)
```

This test checks to see if the x-axis ticks are properly shifted to the right when the xtick.alignment parameter is set to right.

```
@check_figures_equal()
def test_center_xtick(fig_test, fig_ref):
    labels = range(0, 100, 10)

    fig_test.subplots()
    ax1 = fig_test.axes
    ax1[0].set_xticklabels(labels, ha='center')

    mpl.rcParams['xtick.alignment'] = 'center'
    fig_ref.subplots()
    ax2 = fig_ref.axes
    ax2[0].set_xticklabels(labels)
```

This test checks to see if the x-axis ticks are properly centered when the xtick.alignment parameter is centered.

The next three following test cases are meant to fail because they essentially check that each xtick.alignment value (‘center’, ‘right’, ‘left’) produce different figures. Example: a figure with an xtick.alignment value of ‘center’ should have it’s x-ticks centered whereas a figure with an xtick.alignment value of ‘left’ should have it’s x-ticks shifted to the left. Note that when running the next few test cases, they should result in “xfailed” rather than “passed” since we DO NOT want the figures being compared to be equal (we use @pytest.mark.xfail to help assert failure). If the test cases result in “xpassed” it means that the figures are equal to one another (not what we want).

```
@pytest.mark.xfail
```

```
@check_figures_equal(extensions=['png'])
def test_center_left_xtick_alignment(fig_test, fig_ref):
    mpl.rcParams['xtick.alignment'] = 'center'
    fig_test.subplots().plot([1, 2, 3])
    mpl.rcParams['xtick.alignment'] = 'left'
    fig_ref.subplots().plot([1, 2, 3])
```

This test checks to see that the two figures are different from one another where one figure has an xtick.alignment of ‘center’ and another with xtick.alignment of ‘left’.

```
@pytest.mark.xfail
@check_figures_equal(extensions=['png'])
def test_center_right_xtick_alignment(fig_test, fig_ref):
    mpl.rcParams['xtick.alignment'] = 'center'
    fig_test.subplots().plot([1, 2, 3])
    mpl.rcParams['xtick.alignment'] = 'right'
    fig_ref.subplots().plot([1, 2, 3])
```

This test checks to see if the two figures are different where it compares xtick.alignment values of ‘center’ with ‘right’.

```
@pytest.mark.xfail
@check_figures_equal(extensions=['png'])
def test_left_right_xtick_alignment(fig_test, fig_ref):
    mpl.rcParams['xtick.alignment'] = 'left'
    fig_test.subplots().plot([1, 2, 3])
    mpl.rcParams['xtick.alignment'] = 'right'
    fig_ref.subplots().plot([1, 2, 3])
```

This test checks to see if the two figures are different, comparing xtick.alignment values of ‘left’ with ‘right’.

How These Changes Affect matplotlib:

Throughout our research in this issue, we found there is no easy way to set x-axis alignment globally for all plots in matplotlib. If a user wants a consistent way to do this for all their plots without setting it globally, they would have to set per graph using the matplotlib axis library. In a large research project, with dozens of graphs, this would become tedious and would lead to human error and inconsistencies in the plots for the project. The ability to set the xtick.alignment allows users to easily have consistency throughout their plots. Although this is not critical, or even a medium priority bug fix, it helps in the overall goal of matplotlib in making plotting of graphs and data easy, readable, and professional.

Files Changed:

Code:
https://github.com/CSCD01/matplotlib-team28/blob/98081a81721009e8f2053a52df2f7526be04d231/lib/matplotlib/rcsetup.py#L1329-L1330

Test cases added:
https://github.com/CSCD01/matplotlib-team28/blob/d2/lib/matplotlib/tests/test_rcparams.py#L539-L580

Chosen Issue #2 - 16552

How we fixed it:

The issue occurs only for the scatter plot so the issue must have been caused by the method. The method used to create the scatter plot could be found in the file _axes.py. Since the issue was with scatter plots not autoscaling when the value 0 was used and log scales don't show coordinates with the value 0 in them, we solved the issue by filtering out 0 coordinates from the array.

How we tested it:

Prerequisites

Python 3.6+
Pytest

If not done so already, install and build matplotlib from source (from our fork):

```
git clone https://github.com/CSCD01/matplotlib-team28.git
cd matplotlib-team28
python -mpip install -ve .
```

Acceptance Testing:

Under each instruction, there is an example code snippet.

1) In a new Python file, copy and paste the following code snippet to create a matplotlib file

```
import itertools
import numpy as np
from matplotlib import pyplot as plt
```

2) Create 2 arrays for x values and y values and insert any values you'd like

```
x_vals = [4.38462e-06, 5.54929e-06, 7.02332e-06, 8.88889e-06]
y_vals = [0.10000000000000002, 0.182, 0.332, 0.604]

pts = np.array(list(itertools.product(x_vals, y_vals))) # This converts
the values to a numpy array
```

3) If you chose not to insert the value 0 in one or both of the arrays, insert the value 0 into one or both of the arrays

```
y_vals = [0, 0.10000000000000002, 0.182, 0.332, 0.604] # I will insert a 0
into the y values
```

4) Create a figure and set the x-scale and/or y-scale of the axes to use log scaling depending on which arrays contain the value 0

```
fig = plt.figure("Scatter plot")
ax = fig.gca()
ax.set_yscale('log')
```

5) Display the graph as a scatter plot

```
ax.scatter(pts[:,0], pts[:,1])
```

6) Create another figure and set x-scale and/or y-scale of the axes to use log scaling depending on which arrays contain the value 0

```
fig = plt.figure("Regular plot")
ax = fig.gca()
```

```
ax.set_yscale('log')
```

7) Display the graph as a regular plot

```
ax.plot(pts[:,0], pts[:,1], marker="o", ls="")
```

8) Run the matplotlib code
9) Check if the newly created graphs are the same.

Files Changed:

Code:
https://github.com/CSCD01/matplotlib-team28/commit/8a2e0417b14c6011209e2c01ad60143467e0486d#diff-7e79bc5b4cdd21de353697e9ada248b7

Test cases added:

https://github.com/CSCD01/team_28-project/blob/master/deliverable_2/tests/bugfix-16552/test_log_zero_scaling.py

Unit Testing

To run the test cases:

```
git clone https://github.com/CSCD01/team_28-project.git
cd deliverable_2/tests/bugfix-16552
pip install pytest (if not done so already)
pytest test_log_zero_scaling.py
```

We have four test cases in total for this bug:

```
git clone https://github.com/CSCD01/team_28-project.git
cd deliverable_2/tests/bugfix-16552
pip install pytest (if not done so already)
pytest test_log_zero_scaling.py
```

We have four test cases in total for this bug:

Test checks if two figures would be equal if both x and y axis contained a zero value

```
@check_figures_equal()
def test_scatter_autoscaling_w_zero_xy(fig_test, fig_ref):
   x_vals = [0, 4.38462e-06, 5.54929e-06, 7.02332e-06, 8.88889e-06]
   y_vals = [0, 0.10000000000000002, 0.182, 0.332, 0.604]

   pts = np.array(list(itertools.product(x_vals, y_vals))) # This converts the
values to a numpy array

   ax_test = fig_test.gca()
   ax_test.set_yscale('log')
   ax_test.plot(pts[:,0], pts[:,1], marker="o", ls="")

   ax_ref = fig_ref.gca()
   ax_ref.set_yscale('log')
   ax_ref.plot(pts[:,0], pts[:,1], marker="o", ls="")
```

Test checks if two figures would be equal if only x axis contained a zero value

```
@check_figures_equal()
def test_scatter_autoscaling_w_zero_x(fig_test, fig_ref):
   x_vals = [0, 4.38462e-06, 5.54929e-06, 7.02332e-06, 8.88889e-06]
   y_vals = [0.10000000000000002, 0.182, 0.332, 0.604]

   pts = np.array(list(itertools.product(x_vals, y_vals))) # This converts the
values to a numpy array

   ax_test = fig_test.gca()
   ax_test.set_yscale('log')
   ax_test.plot(pts[:,0], pts[:,1], marker="o", ls="")

   ax_ref = fig_ref.gca()
   ax_ref.set_yscale('log')
   ax_ref.plot(pts[:,0], pts[:,1], marker="o", ls="")
```

Test checks if two figures would be equal if only y axis contained a zero value

```
@check_figures_equal()
def test_scatter_autoscaling_w_zero_y(fig_test, fig_ref):
   x_vals = [4.38462e-06, 5.54929e-06, 7.02332e-06, 8.88889e-06]
   y_vals = [0, 0.10000000000000002, 0.182, 0.332, 0.604]

   pts = np.array(list(itertools.product(x_vals, y_vals))) # This converts the
values to a numpy array

   ax_test = fig_test.gca()
   ax_test.set_yscale('log')
   ax_test.plot(pts[:,0], pts[:,1], marker="o", ls="")

   ax_ref = fig_ref.gca()
   ax_ref.set_yscale('log')
   ax_ref.plot(pts[:,0], pts[:,1], marker="o", ls="")
```

Test checks if two figures would be equal if neither axis contained a zero value

```
@check_figures_equal()
def test_scatter_autoscaling_wo_zero(fig_test, fig_ref):
   x_vals = [4.38462e-06, 5.54929e-06, 7.02332e-06, 8.88889e-06]
   y_vals = [0.10000000000000002, 0.182, 0.332, 0.604]

   pts = np.array(list(itertools.product(x_vals, y_vals))) # This converts the
values to a numpy array

   ax_test = fig_test.gca()
   ax_test.set_yscale('log')
   ax_test.plot(pts[:,0], pts[:,1], marker="o", ls="")

   ax_ref = fig_ref.gca()
   ax_ref.set_yscale('log')
   ax_ref.plot(pts[:,0], pts[:,1], marker="o", ls="")
```

How These Changes Affect matplotlib:

No changes were made to architecture. These changes will affect how log scaling is affected by 0 values. Although previous graphs were valid, they were not accurate. With this change, users of matplotlib will now be able to see all of the points they chose to plot rather than a quarter. Without this change, users would have to resort to using a regular plot to create their graph. This issue was not a high priority bug fix as there were numerous workarounds that did not require extra effort. However, this bug will allow the scatter function to be more consistent and reliable