Link to our matplotlib fork:
https://github.com/7uner/matplotlib

Changes to Matplotlib
There were minor changes from our original design to implement this feature. Most notably, in addition to the changes we planned for, we had to implement a new method for the **RendererBase** class as well as all of its subclasses called **pixels_to_points()**. This is a method used to convert from display unit (pixels) to points unit (font point size). This method is necessary for our feature because the padding used to separate the **Tick** from the **TickLabels** are in points unit, but the width that we obtained from the **bbox** of each **TickLabel** are in display unit. Furthermore, rather than using the original **set_tick_params()** method already defined for the **Axis** class, we decided to implement a new method called **set_ticklabel_horizontalalignment()**. This is because the original **set_tick_params()** method stores the **Tick** parameter key/value pairs in a dictionary which will be passed down and stored in each **Tick** as they are dynamically created. We did not feel it was necessary for this information to be passed down to each **Tick** individually as that would imply each **Tick** could have a different alignment, which does not make sense for all intents and purposes. As such, we have set a new parameter in the **Axis** class called **_ticklabel_horizontal_alignment** which could have values ranging from **right** (default), **center**, **left**. Furthermore, we have limited this functionality to only the **YAxis**, since it does not make sense to align **TickLabels** in the **XAxis** to the left or right.

Function pixels_to_points(float pixels) => float points
This new method is defined in the parent **RendererBase** class. Similar to the **points_to_pixel()** method that is already available to Matplotlib, our method is the inverse of that, converting from display units to points units. Like its inverse counterpart, this method is a no-op in the **RendererBase** class and must be overridden by the child class.

Function set_ticklabel_horizontal_alignment(alignment=['left','center','right'])
This new method is defined in the **Axis** class as a means for users to set the **TickLabel**'s alignment. The valid options are left, center, and right (default). It will raise an error if the alignment does not match any of the above. A successful call to this method will set the **Axis** artist's **self._ticklabel_horizontal_alignment** to the specified value.

Function get_ticklabel_horizontal_alignment() => self._ticklabel_horizontal_alignment
This new method is defined in the **Axis** class as a corresponding getter method to the setter defined above. This will return the **self._ticklabel_horizontal_alignment** value to the user.

Implementation of the new feature

Similar to what we have discussed in our last deliverable, our problem lies in the fact that the parent **Axis** object calls **draw()** on each of the **Ticks**, but each **Tick** does not communicate with one another as they are drawn. This means that for any **TickLabel** alignment to align properly, we must have the **Axis** class pass this information down to each **Tick** as they are dynamically generated in the **Axis._update_ticks()** method call. After the **Ticks** are created, we then get a list of the bounding boxes (**bboxes**) for all of the **TickLabels**. These **bboxes** represent an invisible rectangle that wraps the **TickLabel** text and contains crucial information for our feature such as the width of this **bbox**. To align all of the **TickLabels** to the left, we get the max width of all **bboxes** and add the difference between each **Tick bbox** width and the max value to the base padding between the **Tick** and the **TickLabels**. However, as previously mentioned, the width of the **bbox** is in pixels and the padding is in font size points. To fix this discrepancy, we had to implement and use the method **Renderer.pixels_to_points()** to convert between the 2 units. After each of the **Tick**'s new padding has been set, we then call the **draw()** method for each **Tick**.