Plotting

March 21, 2017

```
In [1]: #%matplotlib nbagg
        print('-- ')
        print('-- Make plots (the smart way) in python')
        print('-- (much material from presentation by John D West 2015)')
        print('-- (rest from Karen Olsen 2017)')
        print('-- ')
        print('We will go through:')
        print('1) Simple object-oriented plotting')
        print('2) Adding a second axis')
        print('3) Having several plots in one figure (sub-plotting)')
        print('4) Transparency!')
        print('5) Saving plots')
        print('6) ...')
```

```
--
-- Make plots (the smart way) in python
-- (much material from presentation by John D West 2015)
-- (rest from Karen Olsen 2017)
--
We will go through:
1) Simple object-oriented plotting
2) Adding a second axis
3) Having several plots in one figure (sub-plotting)
4) Transparency!
5) Saving plots
6) ...
```

```
In [2]: # Import the modules that we will need for plotting
        import matplotlib.pyplot as plt
        import numpy as np
        from IPython import display
```

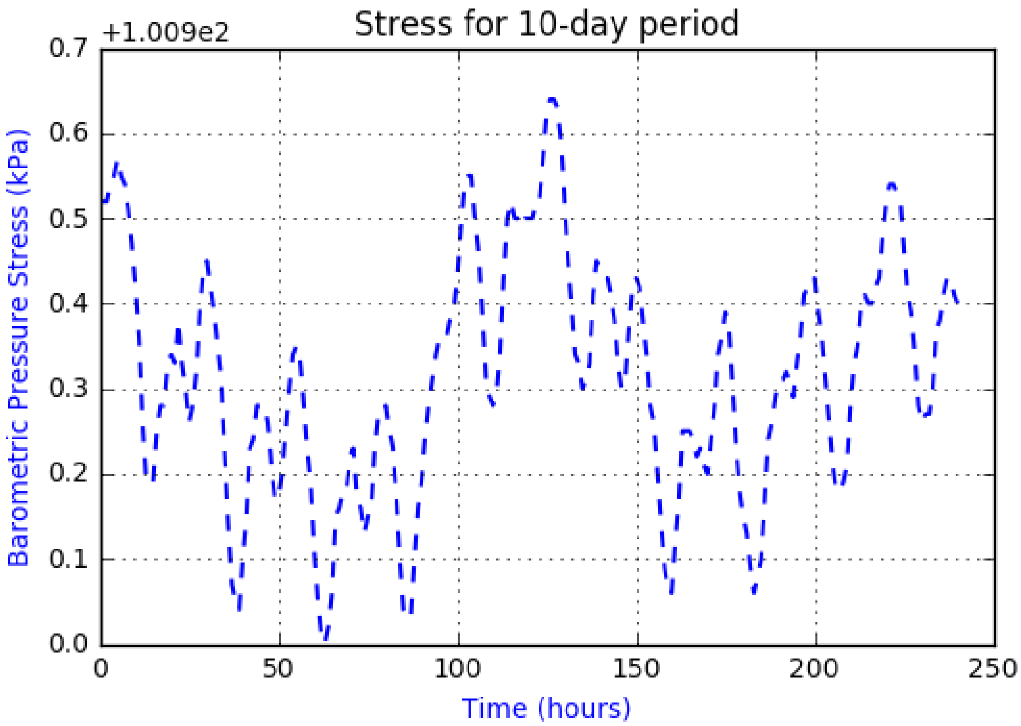
```
In [3]: # Load some data: In this case, tides on the Earth,
        # one measurement per hour
        tide_kPa = np.load('Tide.npy')
        t_hours = np.arange(len(tide_kPa))
        P_kPa = np.load('BP.npy')
```

```
In [4]: # 1) Simple object-oriented plotting
        # In this presentation we will use an object-oriented way to make plots.
        # That means we make an axes 'object' that we add stuff to:
        plt.close('all') # to close all open windows
        fig = plt.figure(figsize=(6,4))
        ax1 = fig.add_subplot(111)
```

```
In [5]: # Now, we can use a lot of matplotlib functions on the 'ax1' object
        # (in ipython hit tab after 'ax1.' to see all your options)
        ax1.plot(t_hours, P_kPa, color='blue', linewidth = 1.5, \
                 linestyle='--', label='Barometric Pressure')
        # Or, in short-hand notation:
        # ax1.plot(t_hours, P_kPa, c='blue', lw = 1.5, ls='--', label='Barometric
        # Let's take a look
```

```
Out[5]: [<matplotlib.lines.Line2D at 0x1101160d0>]
```

```
In [6]: # Let's put some axis labels and a title on there:
        ax1.set_ylabel('Barometric Pressure Stress (kPa)', color='blue')
        ax1.set_xlabel('Time (hours)', color='blue')
        ax1.set_title('Stress for 10-day period')
        # Add grid lines
        ax1.grid(True)
        plt.show()
```

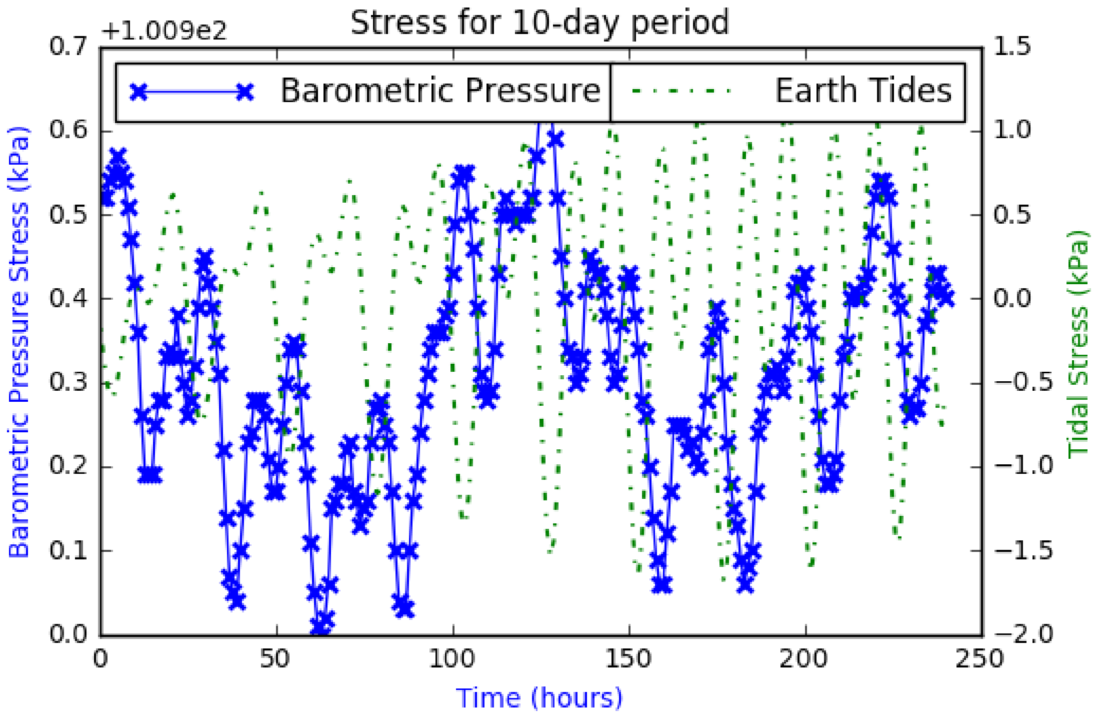


```
In [7]: # 2) Adding a second axis
        # We can also add a SECOND axis by creating another object:
        fig = plt.figure(figsize=(6,4))
        ax1 = fig.add_subplot(111)
```

```
In [8]: # Let's first plot what we had before, now with crosses:
        ax1.plot(t_hours, P_kPa, color='blue', marker='x', lw=1, mew=2, label='Barc
        ax1.set_ylabel('Barometric Pressure Stress (kPa)', color='blue')
        ax1.set_xlabel('Time (hours)', color='blue')
        ax1.set_title('Stress for 10-day period')
```

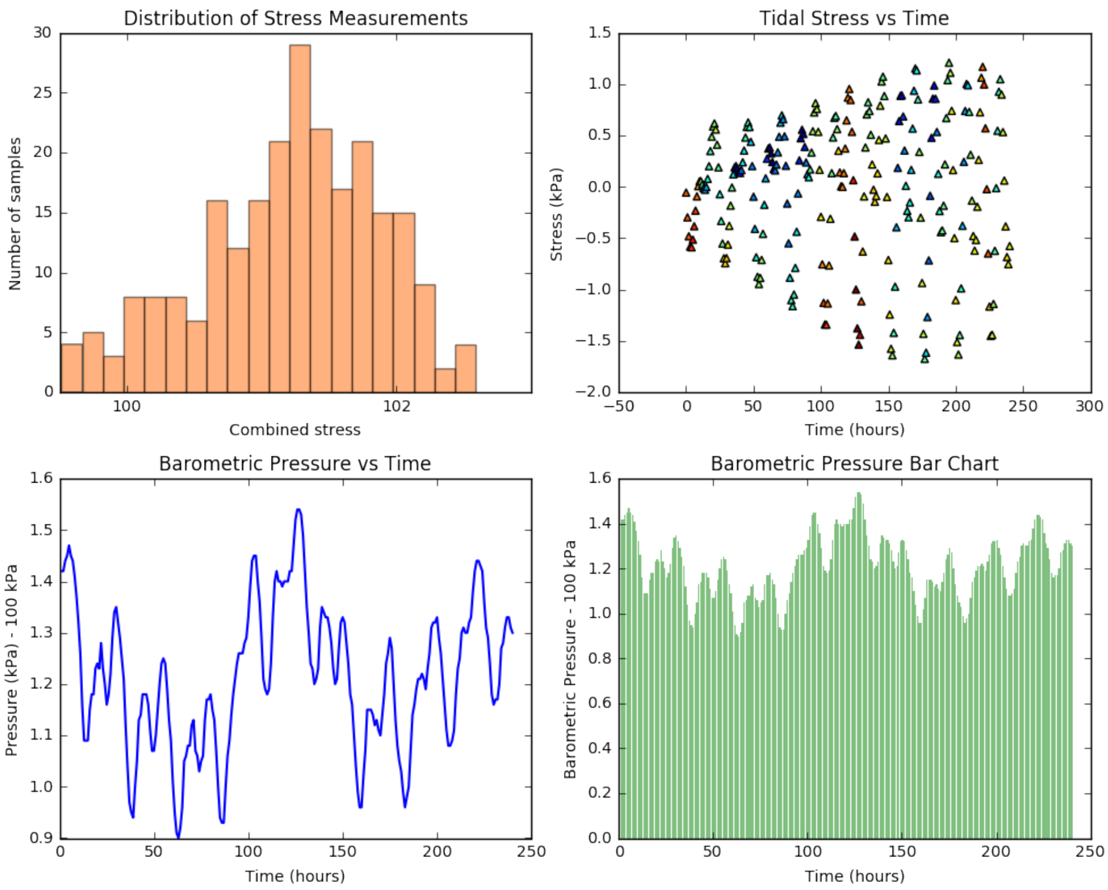
```
Out[8]: <matplotlib.text.Text at 0x1102995d0>
```

```
In [9]: # Then we create a new axes object and plot another type of data sharing tl
        ax2 = ax1.twinx()
        ax2.set_ylabel('Tidal Stress (kPa)', color='green')
        ax2.plot(t_hours, tide_kPa, color='green', linewidth=1.5, \
                 linestyle = '-.', label='Earth Tides')
        # And add legends to tell the two datasets apart:
        ax1.legend(loc='upper left',handlelength=4)
        ax2.legend(loc='upper right',handlelength=4)
        plt.show()
```

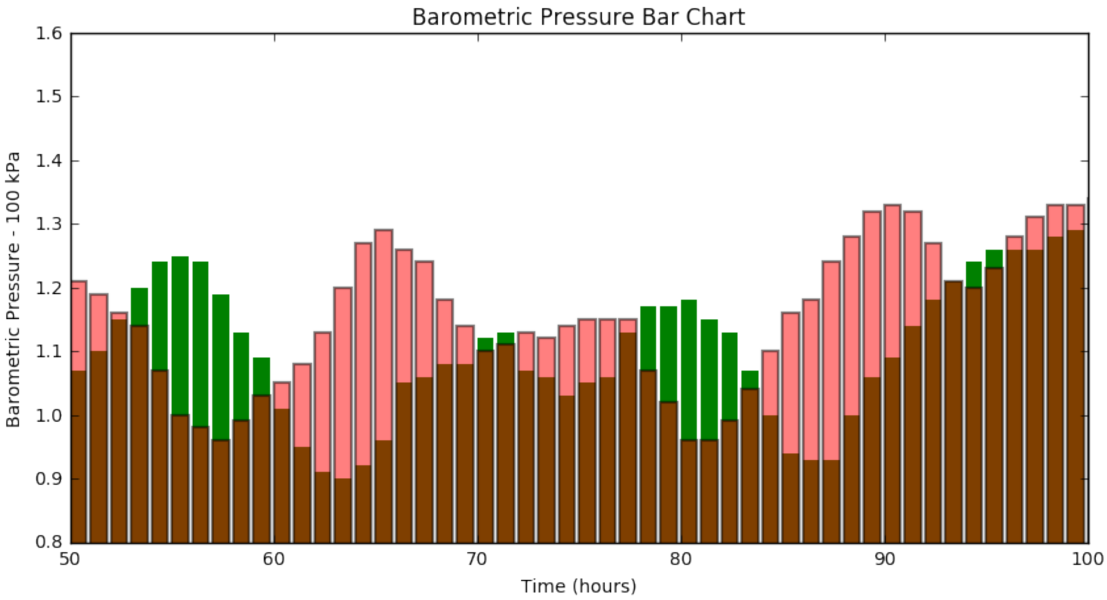


```
In [10]: # 3) Having several plots in one figure (sub-plotting)
         # A quick way to set up all axes in one line:
```

```
         fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2,2,figsize=(10,8))
         # which is the same as:
         # fig = plt.figure()
         # ax1 = fig.add_subplot(221)
         # ax2 = fig.add_subplot(222)
         # ax3 = fig.add_subplot(223)
         # ax4 = fig.add_subplot(224)
         # Matplotlib has a routine for making histograms:
         ax1.hist(tide_kPa + P_kPa, bins=20, color='#ff6600', alpha=0.5)
         ax1.set_ylabel('Number of samples')
         ax1.set_xlabel('Combined stress')
         ax1.set_title('Distribution of Stress Measurements',fontsize=12)

Out[10]: <matplotlib.text.Text at 0x11050d9d0>

In [11]: # The x-axis got a little bit crammed, we can change that:
         ax1.xaxis.major.locator.set_params(nbins=3)
         # ax1.set_xticks([])

In [12]: # Matplotlib can also do a simple scatter plot:
         ax2.scatter(t_hours, tide_kPa, c=P_kPa, marker='^')
         ax2.set_xlabel('Time (hours)')
         ax2.set_ylabel('Stress (kPa)')
         ax2.set_title('Tidal Stress vs Time')

Out[12]: <matplotlib.text.Text at 0x1105b2610>

In [13]: # Let's do another line plot:
         ax3.plot(t_hours, P_kPa-100., color='blue', linewidth = 1.5)
         ax3.set_xlabel('Time (hours)')
         ax3.set_ylabel('Pressure (kPa) - 100 kPa')
         ax3.set_title('Barometric Pressure vs Time')

Out[13]: <matplotlib.text.Text at 0x11060bc50>

In [14]: # And a bar plot
         ax4.bar(t_hours,P_kPa-100.,facecolor='green',lw=0,alpha=0.5)
         ax4.set_xlabel('Time (hours)')
         ax4.set_ylabel('Barometric Pressure - 100 kPa')
         ax4.set_title('Barometric Pressure Bar Chart')

Out[14]: <matplotlib.text.Text at 0x1106985d0>

In [15]: # A smart way to adjust fontsizes that works (most of the time):
         plt.tight_layout()
         plt.show()
```



```
In [16]: # 4) Transparency!
         fig = plt.figure(figsize=(10,5))
         ax1 = fig.add_subplot(111)
         ax1.bar(t_hours,P_kPa-100.,facecolor='green',lw=0,alpha=1)
         ax1.bar(t_hours,(P_kPa-100.)[::-1],facecolor='red',lw=1.5,alpha=0.5)
         ax1.set_ylim([0.8,1.6])
         ax1.set_xlim([50,100])
         ax1.set_title('Barometric Pressure vs Time')
         ax1.set_xlabel('Time (hours)')
         ax1.set_ylabel('Barometric Pressure - 100 kPa')
         ax1.set_title('Barometric Pressure Bar Chart')
         plt.show()
```



```
In [17]: # 5) And finally to save the plot, python has different formats available
         # Default is png:
         plt.savefig('plots/subplot')

In [18]: # For papers, postscript is typically used:
         plt.savefig('plots/subplot.eps', format='eps', dpi=200)

In [19]: # PDF is better at handling transparency:
         plt.savefig('plots/subplot.pdf', format='pdf', dpi=200)

In [ ]:
```