ipywidgets_demo

July 17, 2017

1 Interactive widgets for the Jupyter notebook (ipywidgets)

- Maarten Breddels - Kapteyn Astronomical Institute / RuG - Groningen
- Material on github https://github.com/maartenbreddels/ewass-2017
- for anaconda:
- `$ conda install -c conda-forge ipyvolume bqplot`

```
In [1]: 1+3
```

```
Out[1]: 4
```

```
In [2]: import IPython
```

```
In [3]: #IPython.display.Image(url="https://apod.nasa.gov/apod/image/1703/OrionTrapezium_Hubble_
        IPython.display.Image(url="demo.png")
```

```
Out[3]: <IPython.core.display.Image object>
```

Are you for real or $\sqrt{-1}$?

```
In [4]: %%html
        <button>click</button>
```

```
<IPython.core.display.HTML object>
```

```
In [5]: %%html
        <button onclick='alert("hi")'>click</button>
```

```
<IPython.core.display.HTML object>
```

2 ipywidgets: Interactive HTML Widgets

- github: https://github.com/jupyter-widgets/ipywidgets/
- documentation (live): https://ipywidgets.readthedocs.io/en/latest/

```
In [6]: import ipywidgets as widgets
```

```
In [7]: button = widgets.Button(description="hi")

In [8]: button
```

```
hi
hi
hi
```

```
In [9]: def say_hi(the_button):
            print("hi")
        button.on_click(say_hi)

In [10]: button.description = "hello"

In [11]: slider = widgets.FloatSlider(min=0, max=10, step=0.5, value=3)
         slider

In [12]: slider.value

Out[12]: 7.5

In [13]: slider.value = 2

In [14]: text = widgets.FloatText(value=1)
         text

In [16]: text.value

Out[16]: 1.0

In [17]: widgets.jslink((text, 'value'), (slider, 'value'))

In [18]: %matplotlib inline
         import matplotlib.pylab as plt
         import numpy as np

In [19]: @widgets.interact(period=(0.5, 2, 0.1), phase=(0, np.pi, 0.1))
         def plotxy(period=1.0, phase=0):
             x = np.linspace(0, 4*np.pi, 100)
             y = np.sin(period*x+phase)
             plt.plot(x, y)
             plt.ylim(-1, 1)
             plt.xlim(0, 4*np.pi)
             plt.show()
```

3 Making your own Widget

- http://ipywidgets.readthedocs.io/en/latest/examples/Widget%20Custom.html

```
In [20]: import ipywidgets as widgets
         import traitlets

In [21]: class HelloWidget(widgets.DOMWidget):
             _view_module = traitlets.Unicode('hello').tag(sync=True)
             _view_name = traitlets.Unicode('HelloView').tag(sync=True)
             value = traitlets.Unicode('Hello World!').tag(sync=True)

In [22]: %%javascript
         require.undef('hello'); // if we execute this cell twice, we want to lose the old defin

         // define the module, which has the same name as _view_module above
         define('hello', ["jupyter-js-widgets"], function(widgets) {
             var HelloView = widgets.DOMWidgetView.extend({
                 render: function() {
                     this.update_value()
                     this.model.on('change:value', this.update_value, this);
                 },
                 update_value: function() {
                     this.el.textContent = this.model.get('value');
                 },
             });

             return {
                 HelloView : HelloView // the key has the same name as _view_name above
             };
         });

<IPython.core.display.Javascript object>


In [23]: hi = HelloWidget()

In [24]: hi

In [25]: hi.value = "Ola!"

In [26]: text = widgets.Text()
         widgets.jslink((hi, 'value'), (text, 'value'))
         text
```

Other widgets * gmaps (google maps) * ipyleaflet (open streetview and more) * pythreejs (3d visualization / WebGL wrapper) * ipyvolume (3d visualization / scientific) * bqplot (2d plotting) * Soon? aladin lite (Thomas Boch/CDS, sky atlas)

4 Bqplot

- Interactive plotting in the Jupyter notebook
- Documentation: https://bqplot.readthedocs.io
- Github: https://github.com/bloomberg/bqplot

```
In [27]: import bqplot.pyplot as plt
         import numpy as np

         x = np.linspace(0, 2, 50)
         y = x**2

         fig = plt.figure()
         scatter = plt.scatter(x, y)
         plt.show()

In [28]: fig.animation_duration = 500
         scatter.y = x**0.5

In [29]: scatter.selected_style = {'stroke':'red', 'fill': 'orange'}
         plt.brush_selector()

Out[29]: <bqplot.interacts.BrushSelector at 0x11606c908>

In [30]: scatter.selected

Out[30]: [7, 8, 9, 10, 11]

In [31]: scatter.selected = [1,2,10,40]
```

5 Ipyvolume - 3d plotting widget using WebGL

- 3d plotting for Python in the Jupyter notebook based on IPython widgets using WebGL
- ~6 months old
- Glyphs + Volume rendering (surfaces/meshes/lines are underway)
- Live documentation http://ipyvolume.readthedocs.io/en/stable/ (since and thanks to ipywidgets 6)
- http://github.com/maartenbreddels/ipyvolume - MIT License
- Notebook can be at a supercomputer on the other side of the earth, rendering is done locally (in your browser)
- Hackday tomorrow: install/contribute

```
In [32]: import ipyvolume.pylab as p3

In [33]: from astropy.io import fits
         hdulist = fits.open('ngc6946.fits')
         cube = hdulist[0].data
         #print(np.nanmin(cube), np.nanmax(cube))
         cube[np.isnan(cube)] = 0
```

```
        import scipy.ndimage
        cube = scipy.ndimage.gaussian_filter(cube, 2.5)
        cube.shape

Out[33]: (101, 100, 100)

In [34]: p3.figure()
        p3.volshow(cube, level=[0.3, 0.11, 0.19],
                         opacity=[0.2, 0.1, 0.06], level_width=0.05, data_min=0.03, data_max=0
        p3.show()

In [35]: x, y, z = np.loadtxt('galaxies.dat', unpack=True)[:,:1000]

In [36]: p3.figure()
        scatter = p3.scatter(x, y, z, marker='sphere')
        p3.show()

In [37]: scatter.size = 1

In [38]: scatter.color = "yellow"

In [39]: size_slider = widgets.FloatSlider(min=0.1, max=10, step=0.01)
        widgets.jslink((scatter, 'size'), (size_slider, 'value'))
        size_slider

In [40]: scatter.color = "magenta"

In [41]: scatter.selected = np.random.randint(0, len(x), 100)
        scatter.color_selected = "blue"

In [42]: p3.save("example.html")
        !open example.html

In [43]: fig = plt.figure()
        scatter2d = plt.scatter(x, y)
        fig

In [44]: box = widgets.HBox([p3.gcc(), fig])
        box

In [45]: scatter2d.selected_style = {'stroke':'red', 'fill': 'orange'}
        plt.brush_selector()

Out[45]: <bqplot.interacts.BrushSelector at 0x11939c2b0>

In [46]: widgets.jslink((scatter2d, 'selected'), (scatter, 'selected'))

In [47]: import ipyvolume.embed

In [48]: ipyvolume.embed.embed_html("ipyvolume_bqplot.html", box)
        !open ipyvolume_bqplot.html
```

```
In [49]: p3.figure()
         p3.style.use('dark')
         quiver = p3.quiver(*ipyvolume.datasets.animated_stream.fetch().data[:,::,::4], size=5)
         p3.animate_glyphs(quiver, interval=200)
         p3.show()

In [50]: p3.style.use('light')

In [51]: quiver.geo = "cat"

In [52]: marker = widgets.Select(options=['arrow', 'cat'])
         marker

In [53]: widgets.jslink((marker, 'value'), (quiver, 'geo'))

In [54]: IPython.display.Image(url="poster-billion-stars.png")# S14.3 (2nd floor)

Out[54]: <IPython.core.display.Image object>

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:
```

6 Conclusions

- Jupyter notebook is an interactive environment, to mix code, text, equations, plots
- With ipywidget you can build small GUI's, e.g. to play with variables (model parameter for instance)
- ipyvolume brings 3d visualization, volume rendering, scatter, quiver, and animations
- Widgets can be rendered outside the notebook, live documentation, share with collegues, bring to a meeting, press material

```
In [ ]:
```