Authoring

(with SageMath and Python notebooks)

James Bond

August 30, 2019

Download Juptyer Notebook files, pdf and html files of this book from
https://github.com/OpenDreamKit/authoring_cookie_cutter

Contents

1 Easy authoring with Jupyter notebooks **3**

1.1 Sagemath Notebook . . . 3
1.2 Key concepts . . . 3
1.3 TESTS . . . 3
1.4 Figure . . . 4
1.5 Passing environment variable to notebooks . . . 4
1.6 References to equations . . . 6
1.7 Problem with SageMath `show` and LaTeX . . . 6
1.8 Plotting . . . 6
1.8.1 Unsolved problem with long output . . . 8

2 Example: Python notebook **9**

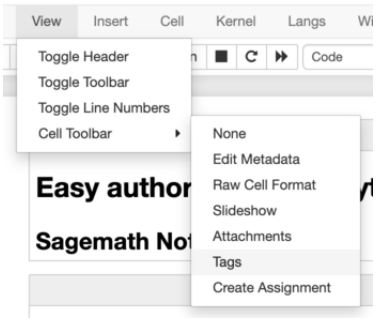


Fig. 4.1: Velocity.

1 Easy authoring with Jupyter notebooks

1.1 Sagemath Notebook

1.2 Key concepts

1. Notebooks are stored in version control system (git) without output.
2. `Makefile` is ised to keep all prescriptions for bulting pdf/html etc.
3. `nbconvert` is used to automatically execute and clean notebooks.
4. It will work with different Jupyter kenrels: we tested wit against SageMath and Python.
5. Notebook will use `PDF` environment variable to distinguish between interactive output or static one. In the `Python` process is it easily accesible as `os.environ['PDF']`. This can be interesting when using interactive features like `@interact` in Sage, which should be replaces by static figures in pdf format.
6. All cells with tags `nbtest` will be removed from pdf or html output. This mechanism can be used for regression testing.
7. We will include `Dockerfile` which allows to run a given repo on mybinder service.

```
In [1]: factor(12345678)
```

```
Out[1]: 2 * 3^2 * 47 * 14593
```

1.3 TESTS

In building process all notebooks are,

- stripped of output,
- executed
- stripped of cells with `nbtest` tags.

Tags can be edites if following mode is activated:

Stripped notebooks can be found in `notebooks4pdf` directory.

It will raise exception in compilation but will be invisible in pdf/html output.

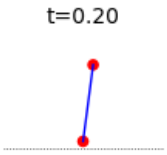


Fig. 4.1: Velocity.

1.4 Figure

- one should use markdown for figures (not html tags):

```
![Fig. 4.1: Velocity.](images/1.png)
```

gives:

1.5 Passing environment variable to notebooks

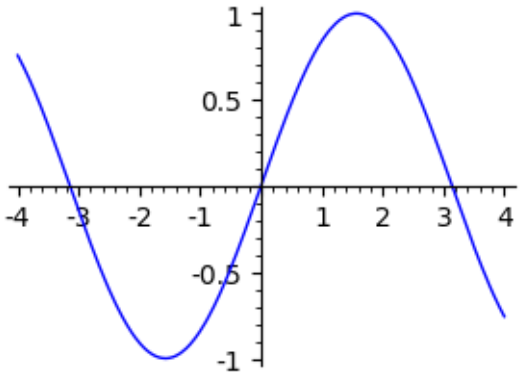
If compiled for pdf, in Makefile we set `PDF=1`. It can be read inside the notebook using following conditional:

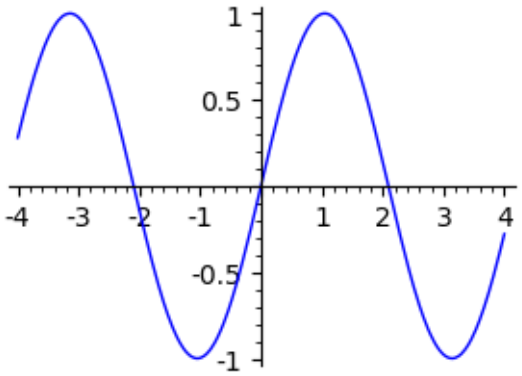
```
if  'PDF' in os.environ.keys():
```

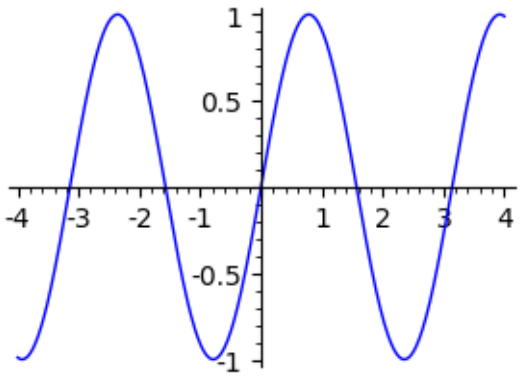
There is an example below:

```
In [3]: import os
        def fun(a):
            plot(sin(a*x),(x,-4,4)).show(figsize=3)

        if  not 'PDF' in os.environ.keys():
            @interact
            def _(a = slider(1,2,0.1)):
                fun(a)
        else:
            for a in [1,1.5,2]:
                fun(a)
```


1
0.5
-4
-3
-2
-1
1
2
3
4
-0.5
-1


1
0.5
-4
-3
-2
-1
1
2
3
4
-0.5
-1


1
0.5
-4
-3
-2
-1
1
2
3
4
-0.5
-1

1.6 References to equations

We use LaTeX labels in equations, they will be used in PDF output:

$$2+2=5 \tag{1}$$

in 1 there is a problem!

1.7 Problem with SageMath `show` and LaTeX

There is an issue in Sagemath which prevent from corect display of formulas in nbconvert. We circumvent it by using showmath function whereaver applicable

```
In [4]: expr = (x^2+1)/x^3*sin(x)+2^x
```

```
In [5]: show(expr)
```

```
2^x + (x^2 + 1)*sin(x)/x^3
```

```
In [6]: from IPython.display import Math
        def showmath(expr):
            return Math(latex(expr))
```

```
In [7]: showmath(expr)
```

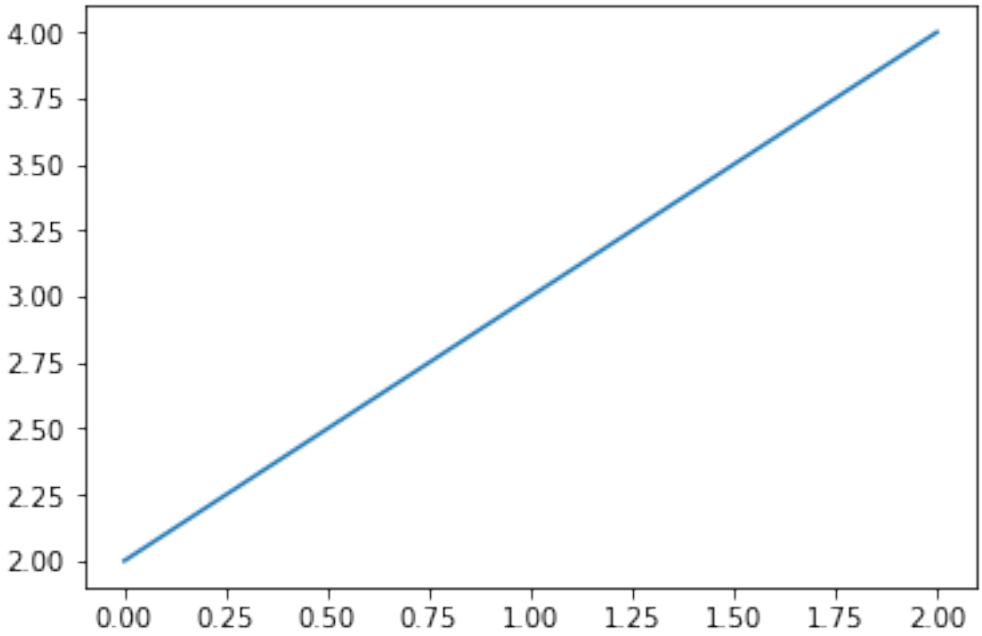
Out[7]:

$$2^{x}+\frac{\left(x^{2}+1\right)\sin\left(x\right)}{x^{3}}$$

1.8 Plotting

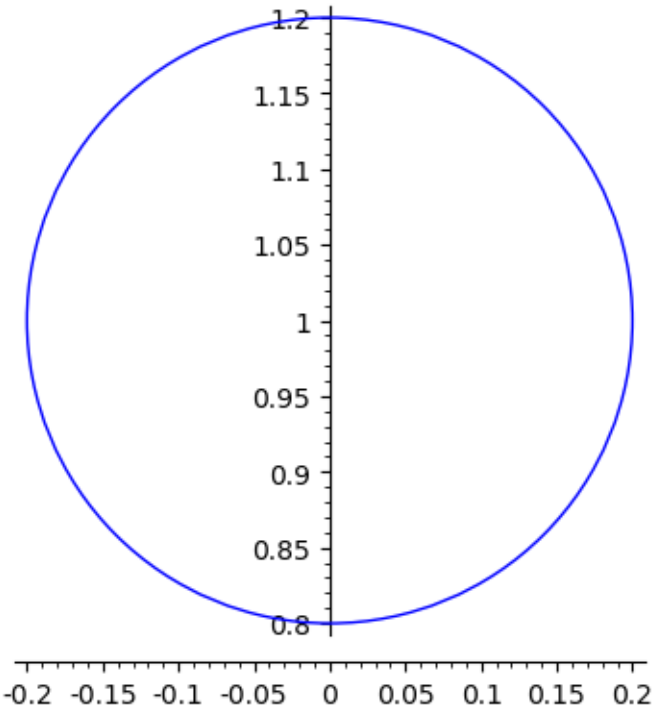
- if using matplotlib, it is recommended to call `plt.show()`

```
In [8]: %matplotlib inline
        import matplotlib.pyplot as plt
        plt.plot([2,3,4])
        plt.show()
```



In [9]: `circle((0,1),0.2)`

Out[9]:



1.8.1 Unsolved problem with long output

```
In [10]: expr = expand((1+x)^21)
```

```
In [11]: showmath( expr )
```

Out[11]:

$$x^{21} + 21\,x^{20} + 210\,x^{19} + 1330\,x^{18} + 5985\,x^{17} + 20349\,x^{16} + 54264\,x^{15} + 116280\,x^{14} + 203490\,x^{13} + 293930\,x^{12} + 352$$

One can force newpage in LaTeX by:

2 Example: Python notebook

```
In [1]: import matplotlib.pyplot as plt
        plt.plot([1,2,1],[2,1.2,1],'o')
        plt.show()
```

```
<Figure size 640x480 with 1 Axes>
```

```
In [2]: print(2+2)
```

```
4
```