Replacing Jupyter with Orgmode

Rohit Goswami · 2020-02-13 · tools · emacs · workflow · orgmode

Background

- I dislike Jupyter notebooks (and JupyterHub) a lot
- EIN is really not much of a solution either

In the past I have written some posts on TeX with JupyterHub and discussed ways to use virtual Python with JupyterHub in a more reasonable manner.

However, I personally found that EIN was a huge pain to work with, and I mostly ended up working with the web-interface anyway.

It is a bit redundant to do so, given that at-least for my purposes, the end result was a LaTeX document. Breaking down the rest of my requirements went a bit like this:

What exports well to TeX?

Org, Markdown, anything which goes into pandoc

What displays code really well?

LaTeX, Markdown, **Org**

What allows easy visualization of code snippets?

Rmarkdown, RStudio, JupyterHub, **Org** with babel

Clearly, orgmode is the common denominator, and ergo, a perfect JupyterHub alternative.

Setup

Throughout this post I will assume the following structure:

```
tree tmp
mkdir -p tmp/images
touch tmp/myFakeJupyter.org
```

tmp
├── images
└── myFakeJupyter.org
1 directory, 1 file

As is evident, we have a folder `tmp` which will have all the things we need for dealing with our setup.

Virtual Python

Without waxing too eloquent on the whole reason behind doing this, since I will rant about virtual python management systems elsewhere, here I will simply describe my preferred method, which is using poetry.

```
# In a folder above tmp
poetry init
poetry add numpy matplotlib scipy pandas
```

The next part is optional, but a good idea if you figure out using direnv and have configured `layout_poetry` as described here:

```
# Same place as the poetry files
echo "layout_poetry()" >> .envrc
```

Note:

- We can nest an arbitrary number of the `tmp` structures under a single place we define the poetry setup
- I prefer using `direnv` to ensure that I never forget to hook into the right environment

Orgmode

This is not an introduction to org, however in particular, there are some basic settings to keep in mind to make sure the set-up works as expected.

Indentation

Python is notoriously weird about whitespace, so we will ensure that our export process does not mangle whitespace and offend the python interpreter. We will have the following line at the top of our `orgmode` file:

```
# -*- org-src-preserve-indentation: t; org-edit-src-content: 0; -*-
```

Note:

- this post is actually generating the file being discussed here by

tangling the file

- You can get the whole file here

TeX Settings

These are also basically optional, but at the very least you will need the following:

```
#+author: Rohit Goswami
#+title: Whatever
#+subtitle: Wittier line about whatever
#+date: \today
#+OPTIONS: toc:nil
```

I actually use a lot of math using the `TeX` input mode in Emacs, so I like the following settings for math:

```
# For math display
#+LATEX_HEADER: \usepackage{amsfonts}
#+LATEX_HEADER: \usepackage{unicode-math}
```

There are a bunch of other settings which may be used, but these are the bare minimum, more on that would be in a snippet anyway.

Note:

- rendering math in the `orgmode` file in this manner requires that we use `XeTeX` to compile the final file

Org-Python

We essentially need to ensure that:

- Babel uses our virtual python
- The same session is used for each block

We will get our poetry python pretty easily:

```
which python
```

Now we will use this as a common `header-arg` passed into the property drawer to make sure we don't need to set them in every code block.

We can use the following structure in our file:

```
\* Python Stuff
  :PROPERTIES:
  :header-args:    :python /home/haozeke/.cache/pypoetry/virtualenvs/test-2aLV_5DQ-py3.8/bin/python :session
One :results output :exports both
  :END:
Now we can simply work with code as we normally would
\#+BEGIN_SRC python
print("Hello World")
\#+END_SRC
```

Note:

- For some reason, this property needs to be set on **every** heading (as of Feb 13 2020)
- In the actual file you will want to remove extraneous `\` symbols:
  - * → *
  - \#+BEGIN_SRC → #+BEGIN_SRC
  - \#+END_SRC → #+END_SRC

Python Images and Orgmode

To view images in `orgmode` as we would in a JupyterLab notebook, we will use a slight trick.

- We will ensure that the code block returns a file object with the arguments
- The code block should end with a print statement to actually generate the file name

So we want a code block like this:

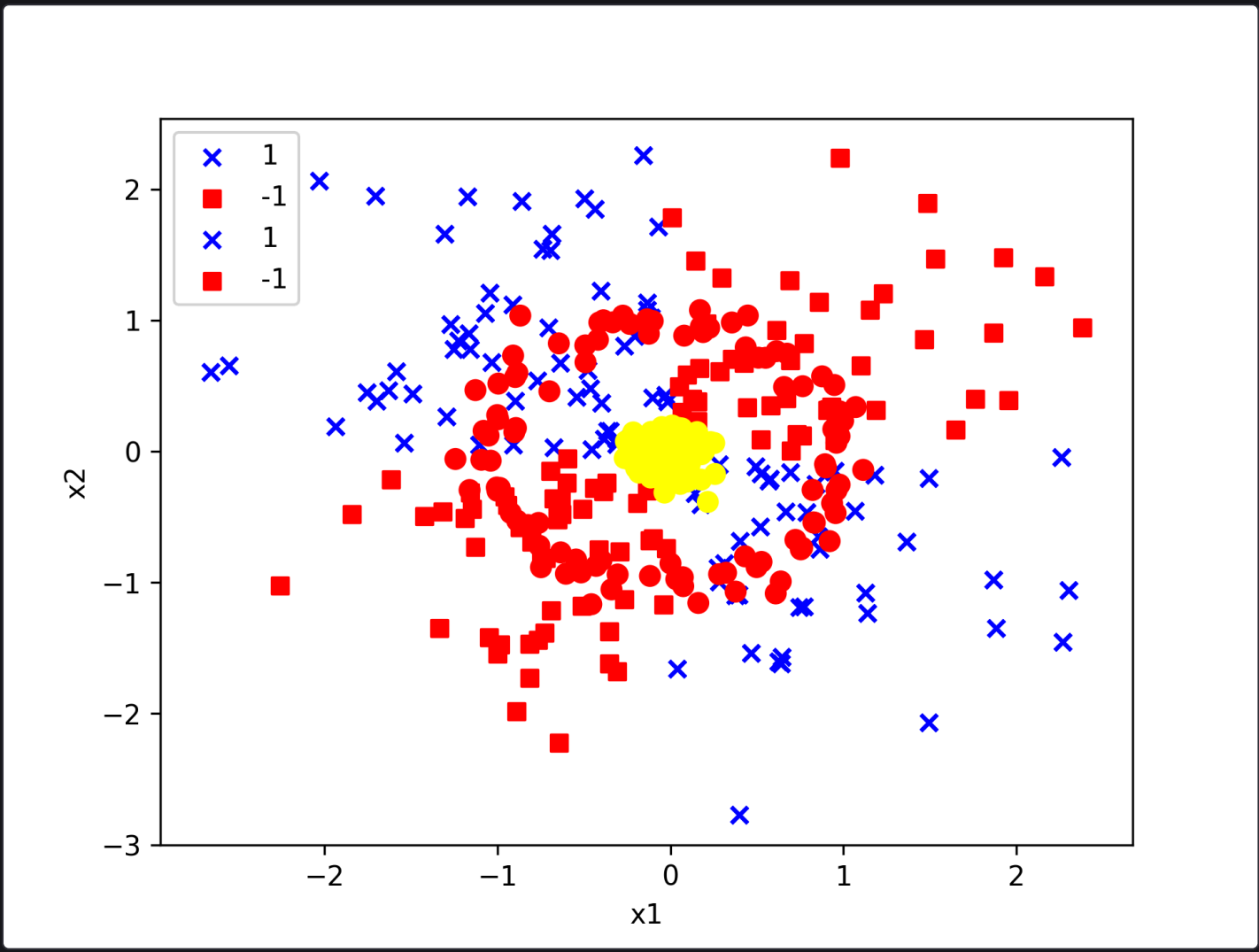
```
#+BEGIN_SRC python :results output file :exports both
import matplotlib.pyplot as plt
from sklearn.datasets.samples_generator import make_circles
X, y = make_circles(100, factor=.1, noise=.1)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='autumn')
plt.xlabel('x1')
plt.ylabel('x2')
plt.savefig('images/plotCircles.png', dpi = 300)
print('images/plotCircles.png') # return filename to org-mode
#+end_src
```

Which would give the following when executed:

```
#+RESULTS:
[[file:images/plotCircles.png]]
```

Since that looks pretty ugly, this will actually look like this:

```
import matplotlib.pyplot as plt
from sklearn.datasets.samples_generator import make_circles
X, y = make_circles(100, factor=.1, noise=.1)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='autumn')
plt.xlabel('x1')
plt.ylabel('x2')
plt.savefig('images/plotCircles.png', dpi = 300)
print('images/plotCircles.png') # return filename to org-mode
```



Bonus

A better way to simulate standard `jupyter` workflows is to just specify the properties once at the beginning.

```
#+PROPERTY: header-args:python :python /home/haozeke/.cache/pypoetry/virtualenvs/test-2aLV_5DQ-
py3.8/bin/python :session One :results output :exports both
```

This setup circumvents having to set the properties per sub-tree, though for very large projects, it is useful to use different processes.

Conclusions

- The last step is of course to export the file as to a `TeX` file and then compile that with something like `latexmk -pdfxe -shell-escape file.tex`

There are a million and one variations of this of course, but this is enough to get started.

The whole file is also reproduced here.
