Version control with Jupyter Notebook

A Step-by-Step Guide to Jupytext

Shinichi Okada

Feb 21 · 5 min read ★

Photo by Mimi Garcia on Unsplash

Table of Contents

Introduction

Jupyter notebook generates files that contain metadata, source code, formatted text, and rich media. Only one word of change results in thousands of letters in `git diff`.

Jupytext can save Jupyter Notebook to a git-friendly and human-friendly file format, including Markdown, Python, Julia, Bash, Clojure, Matlab, TypeScript, Javascript, etc.

It also converts these documents into Jupyter Notebooks. In this article, I am going through a step-by-step guide to version control for Jupyter Notebook using Jupytext.

Supported extensions are:

Supported Extensions						
.ipynb	.md	.markdown	.Rmd	.py	.R	.r
.jl	.cpp	.ss	.clj	.scm	.sh	.ps1
.q	.m	.pro	.js	.ts	.scala	.rs
.robot	.auto	.cs	.fsx			

. . .

. . .

Creating a demo repo

First, let's create a new Jupyter Notebook file with the following codes.

```
x = np.arange(-3, 3, 0.1)
y = np.sin(x)
plt.plot(x, y)
plt.show()
```

Please create a Github repo.

```
echo "# jupyter_notebook_version_control" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin
git@github.com:username/jupyter_notebook_version_control.git
git push -u origin master
```

I change the word `sin` to `cos` in a ipynb file.

```
y = np.cos(x)
```

This link is the result of ` git diff` . It generated thousands of letters for three letters.

```
diff --git a/Version_control.ipynb b/Version_control.ipynb
index 85a47b8..df2eab9 100644
--- a/Version_control.ipynb
+++ b/Version_control.ipynb
@@ -2,12 +2,12 @@
  "cells": [
   {
    "cell_type": "code",
-   "execution_count": 2,
+   "execution_count": 1,
    "metadata": {},
    "outputs": [
     {
      "data": {
```

```
-      "image/png": "iVBORw0KGgoAAAANSUhEUgAAAYIAAAD4CAYAAADhNOGaAAAABHNCSVQICAgIfAhkiAAAAAlwSF
        lzAAALEgAACxIB0t1+/AAAADh0RVh0U29mdHdhcmUAbWF0cGxvdGxpYiB2ZXJzaW9uMy4xLjMsIGh0dHA6Ly9tY
        XRwbG90bGliLm9yZy+AADFEAAAgAElEQVR4nO3dd3hUddrG8e+TDgECIQECCYFAaFIlNDsCghWxgq4vVtSVta0F
        de3r2huWVdayqCh2RcRFQLAhQoJ0SAg9tCRACAnped4/MuzGmwpz+e65s rMmXNm7tEwd2bOOb+fqCrGGGP
        8V4DTAYwxxjjLisAYY/ycFYExxvg5KwJjjPFzVgTGGOPngpwOcCyioqK0Q4cOTscwhivkpKSkq2q0VWXe2URd0
        jQgeTkZKdjGGOMVxGRLdUtt6+GjDHGz1kRGGOMn7MiMMYYP2dFYIwxfs6KwBhj/JwVgTHG+DkrAmOM8XNWBMbh/
        SRdRFaIyPGV7hsvIutdl/HuyGOMWebIuesTwb+BUYe5/0gwXWZAPwTQTQEQigQeBQcBA4EERaeGmTMYYY46AW84j
        UNUfRKTKDR5iiSA5wGzFHVvQAiMoeKQvnAHbmMd6hvFzFzAHxQ0+fb2VevUEr8fQmedJb2PS9Of/Yh/BdTZ+ff
        2kNMY0iKLSMpZtzSF5yz5JXf9BSVHtG
        1MRBidWzXh9G6tOLtXDK2ahdVzWt/nNdWqqlOAKQBJSUk2m44xXmjb3o089+sWPlyyyjZyDFW/8nVs14cyebegf3
        4LuMc0IDw2iUXAgjYIDCQsJQBU2ZeezKTufjVl5bMzKZ9W0/Tz81RoeIjrDr9A7NVQR
        bAfiKt2OdS3bTsXXQ5WXL2igTMaYBlBervyYns27v2xm3rpMAkQ4o0drxg4GjfvjHN6B7T7HfL0jM
        P8NXyncxcsYP7Pl/FA1+uZky/dtw6JnHXVJWufQQzVbVnNfedDUwEzqjo9izcSF4Oefea6v+HTfF4BuPWSLqHLDjuWDc
        yaqSajbnscXiEiKqib9Ybk7ikBEPqDiL/AoEckAAAwFxjayIFPYAdEpn2XcSSoBXgjmgKZg0QlxiIiJ7mgCCdBm
        GCGx4RBhBcbQb+vYxM/Fb3o0FrPmXydDlQ2T4tBPnnRT7sEJBEnXvdo6uAfzjmt+iJKPWWNRYgUhJI5VzYKPK
        YIxny8wt5LFZa/ly2Q04axeMYQGBdbr8+7IKWDWhPPHXhy/H7PfZvG1yt3Oq5SI14Y25cBHSIbNIMnsiIwxtSrjV
        WBMZ4ptKycqb+soXn6aRXFrODacm8OehnQkLrt8CqGpDVh7PfZvGG1yt3Oq55I14Y25cBHSIbNIMnsiIwxtSrjV
        l5THz/N9bszOXIgKdzRTypa93PbhcjL2HeSmoZ25eVgiXG1FuM3v1Lka//DM79
        xfwz8uP599XDXC8BAD6x0cy65aTueD4WF76LZ2dn5TsfyOFYExphjVlauPPmdfdVz/bgodo8OZefPJNkrh
        ordgp6hSWgQz1zch1cu055NWXmcNflHPv8tw+lYHsVrziMwxniWPXlF3DJ9GT+lZ5uYBwPnntcg+8L0Bpn946hX
        /vm3PrhMm77cDnb9hbwXL9M7e1RpOcWKwBhz1NbvsPsD4txfa+MxfaS24a0KqJHD9u1yP4Wp3u2mQLl7snXOQe0cnO/a/+A
```


. . .

Jupytext setup

Let's install and setup the Jupytext.

```
pip install jupytext --upgrade
```

Or for conda users

```
conda install -c conda-forge jupytext
```

RESTART Jupyter Notebook.

. . .

Converting to a python file

You can convert a ipynb file to one of the supported file. I will use a python file in this article.

In your terminal, you can run like this.

```
jupytext --to py <your-file-name>.ipynb
```

For my case:

```
jupytext --to py Version_control.ipynb
```

Outputs:

```
[jupytext] Reading ./Version_control.ipynb
[jupytext] Writing ./Version_control.py
```

Converting multiple files

Let's convert all ipynb files at once. Please create more files in your directory.

```
jupytext --to py *.ipynb
```

Output:

```
[jupytext] Reading Version_control.ipynb
[jupytext] Writing Version_control.py
[jupytext] Reading sine.ipynb
[jupytext] Writing sine.py
[jupytext] Reading tangent.ipynb
[jupytext] Writing tangent.py
```

You can convert a file into a directory. Jupytext will create a new directory if it does not exist.

```
jupytext --to destination_folder//py *.ipynb
```

Notes:

If you prefer you can run jupytext in one of the cells. But this cell will be in your converted file as well.

```
!jupytext --to py <your-file-name>.ipynb
```

Converted file

Let's see the converted file in your terminal.

```
cat Version_control.py
```

My output:

```
# ---
# jupyter:
#   jupytext:
#     text_representation:
#       extension: .py
#       format_name: light
#       format_version: '1.5'
#       jupytext_version: 1.3.3
#   kernelspec:
```

```
#     display_name: Python 3
#     language: python
#     name: python3
# ---

x = np.arange(-3, 3, 0.1)
y = np.cos(x)
plt.plot(x, y)
plt.show()
```

It is very compact and the file size is very small. Nice 😃 👏👏👏👏.

. . .

Adding ipynb to .gitignore

Since we are not tracking `ipynb` files, we can add it to a `.gitignore` file. Please create a `.gitignore` in your project root directory where you have .git directory.

```
touch .gitignore
```

Please add `*.ipynb` and ` .ipynb_checkpoints` to ignore all Jupyter Notebook files. Or add this complete list to your gitignore.

```
# for Jupytext ignoring ipynb files
*.ipynb
```

At this stage, git will still track changes in `.ipynb` files. To fix this you need to remove git cache and add all files again.

```
git rm -r --cached .
git add .
git commit -m "fixed untracked files"
```

After changing a line in your Jupyter Notebook to see if `.gitignore` is working.

```
# change whatever you want
y = np.arange(-2,2,0.1)
```

Check it in your terminal:

```
git status
```

It should not return a modifed file. Let's run Jupytext one more time to reflect our change. Please run the following in your terminal.

```
jupytext --to py Version_control.ipynb
```

The converted file will be replaced. 😃

```
[jupytext] Reading ./Version_control.ipynb
[jupytext] Writing ./Version_control.py (destination file replaced)
```

Let's check the git status.

```
git status

On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working
directory)

    modified:   Version_control.py

no changes added to commit (use "git add" and/or "git commit -a")
```

It tracked only the python file, not ipynb. Please run `git diff`.

```
git diff
diff --git a/Version_control.py b/Version_control.py
index 02d91ea..6522717 100644
--- a/Version_control.py
+++ b/Version_control.py
@@ -14,6 +14,7 @@
 # ---

 x = np.arange(-3, 3, 0.1)
+y = np.arange(-2,2,0.1)
 y = np.cos(x)
 plt.plot(x, y)
 plt.show()
```

Please add, commit and push the change.

```
git add .
git commit -m "Update"
git push
```

. . .

Converting to ipynb files

We are going to clone this repo to another directory and convert it to a ipynb file.

```
cd ..
git clone
git@github.com:shinokada/jupyter_notebook_version_control.git my-
new-dir
```

I cloned my repo to a directory called my-new-dir.

```
cd my-new-dir
ls
README.md    Version_control.py sine.py    tangent.py
```

Or if you have the `tree`.

```
tree
.
├── README.md
├── Version_control.py
├── sine.py
└── tangent.py

0 directories, 4 files
```

We have all the files we need. Let's convert it to ipynb file.

From your terminal:

```
jupytext --to ipynb *.py
```

Output:

```
[jupytext] Reading Version_control.py
[jupytext] Writing Version_control.ipynb
[jupytext] Sync timestamp of 'Version_control.py'
[jupytext] Reading sine.py
[jupytext] Writing sine.ipynb
[jupytext] Reading tangent.py
[jupytext] Writing tangent.ipynb

ls
README.md      Version_control.py    sine.py    tangent.py.
Version_control.ipynb sine.ipynb   tangent.ipynb
```

. . .

Other commands

These are other command you can use.

```
# convert notebook.md to an .ipynb file and run it
jupytext --to notebook --execute notebook.md

# update the input cells in the .ipynb file and preserve outputs and
metadata
jupytext --update --to notebook notebook.py
```

```
# Turn notebook.ipynb into a paired ipynb/py notebook
jupytext --set-formats ipynb,py notebook.ipynb

# Update all paired representations of notebook.ipynb
jupytext --sync notebook.ipynb
```

. . .

Paired notebooks

Jupytext can write a given notebook to multiple files. In addition to the original notebook file, Jupytext can save the input cells to a text file — either a script or a Markdown document. Please read more details if you are interested.

. . .

Conclusion

Jupytext is easy to use and create human-friendly files which you can edit in another editor as well. If you are using `git diff` , this is an excellent tool to have. I think this is the most complete open-source tool for the version control with Jupyter Notebook at the moment.

. . .

References

- https://github.com/mwouts/jupytext
- https://jupytext.readthedocs.io/en/latest/index.html

- https://jupytext.readthedocs.io/en/latest/introduction.html#demo-time

Thanks to Marc Wouts.

Jupyter Notebook Git Programming Version Control Towards Data Science