DATA SCIENCE, PROGRAMMING, PROJECT MANAGEMENT

Project Management Jupyter Extension for Data Science

A Cookiecutter Jupyter extension that helps easily organise project folders without a terminal

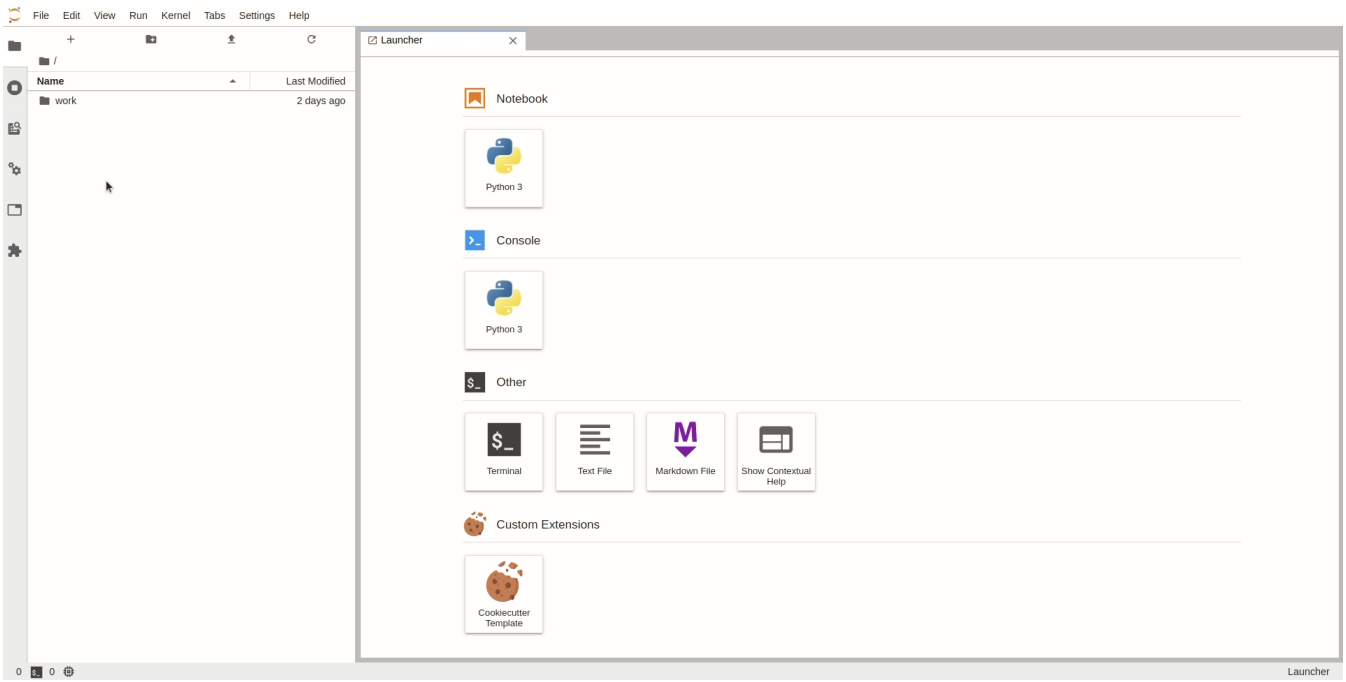
Louis Chan Nov 5, 2020 · 7 min read ★

Before we start

In this blog, we are going to cover one of my personal repositories:

A Cookiecutter-ish extension in Jupyter Launcher

Link to the repository can be found at the end of the blog.



A quick demo

Machine Learning environment, be it Notebook, Lab or Hub. Each of them comes with their own benefits, Notebook is the friendliest for custom plugins, Jupyter Lab makes working with multiple notebooks easy, while Jupyter Hub enables multiple users and more.

> However, one of weaknesses of Jupyter is that it does not come with tools for organising projects.

This is where cookiecutter comes in. For those who has used it before, click here to skip through this section. Otherwise, **cookiecutter is a python package that allows us to create a standard folder structure in just a couple of line**. And the best thing is, you do not need to conform to the one standard that comes with cookiecutter. Take a look here for other formats!

```
$ pip install cookiecutter

$ cd path_to_my_project_folder

# pick your own flavour of cookie
$ cookiecutter https://github.com/drivendata/cookiecutter-data-
science
```

And, voila! You should now have a **standard folder structure** ready to go. In my case, the directory looks like:

```
├── README.md          <- Front page of the project. Let everyone
│                         know the major points.
│
├── models             <- Trained and serialized models, model
│                         predictions, or model summaries.
│
├── notebooks          <- Jupyter notebooks. Use set naming
│                         E.g. `1.2-rd-data-exploration`.
│
├── reports            <- HTML, PDF, and LaTeX.
│   └── figures        <- Generated figures.
│
├── requirements.txt   <- File for reproducing the environment
│                         `$ pip freeze > requirements.txt`
```

```
    ├── processed      <- The final data sets for modelling.
    └── raw            <- The original, immutable data.
│
└── src                <- Source code for use in this project.
    ├── __init__.py    <- Makes src a Python module.
    │
    ├── custom_func.py <- Various custom functions to import.
    │
    ├── data           <- Scripts to download or generate data.
    │   └── make_dataset.py
    │
    ├── features       <- Scripts raw data into features for
    │   │                 modeling.
    │   └── build_features.py
    │
    ├── models         <- Scripts to train models and then use
    │   │                 trained models to make predictions.
    │   │
    │   ├── predict_model.py
    │   └── train_model.py
    │
    └── viz            <- Scripts to create visualizations.
        └── viz.py
```

Why do we need a Cookiecutter-ish Jupyter extension?

Imagine if you are working on a project in JupyterLab.

Inspiration hits; and you want to start a new project. What do you do?

> *A. Start a terminal; go through Cookiecutter to create another standard folder directory; and launch another JupyterLab at port 8890 to work on this project coz you have 8888 and 8889 running two different JupyterLab sessions already.*
>
> *B. Within the same JupyterLab session, start working on Untitled29.ipynb*
>
> *C. Forget about it; it's way too much trouble to start another project, meh!*

We all know the answer, it's either B, or C.

So what is wrong with Cookiecutter?

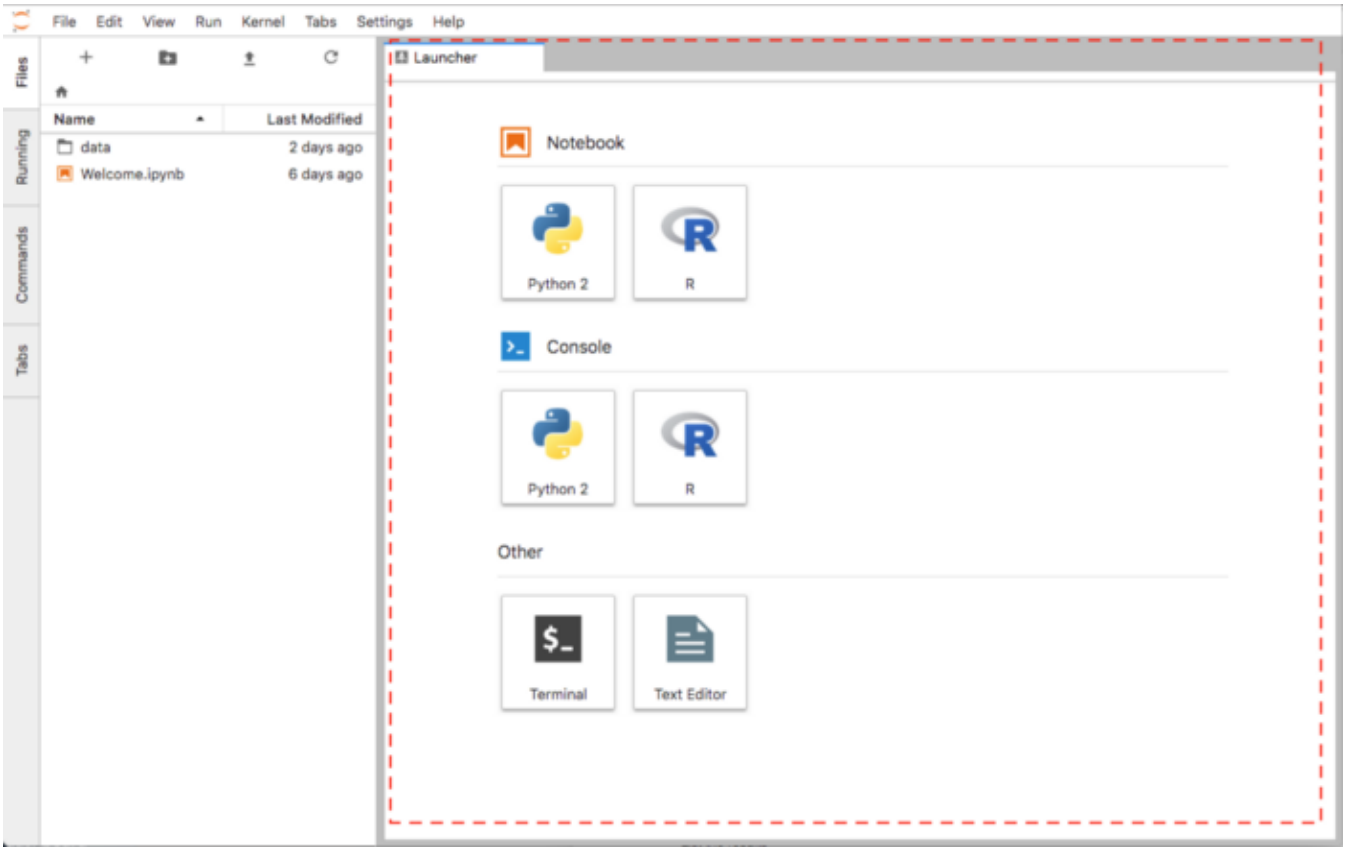
very-very-very-well-disciplined data scientists to start a new project

In other words, there are too many clicks between you and your next project with a well-defined standard file structure.

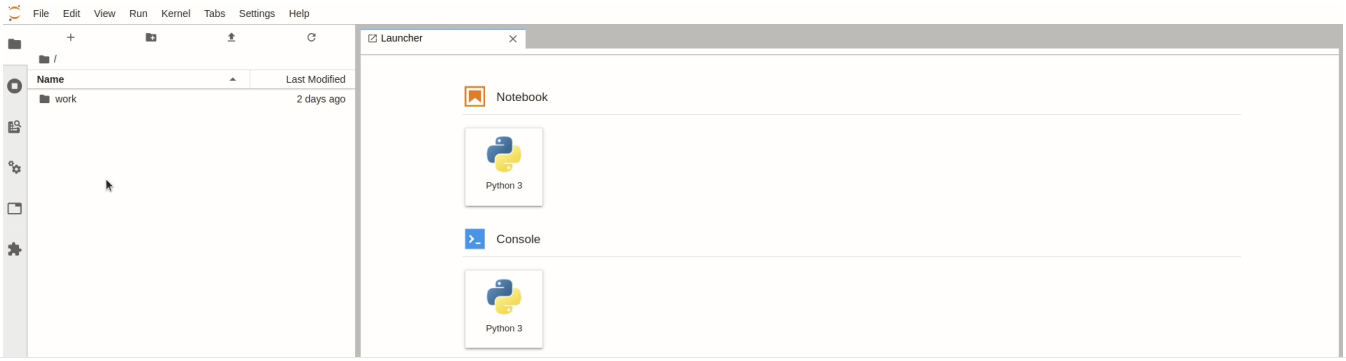
Cookiecutter Plugin for JupyterLab and JupyterHub

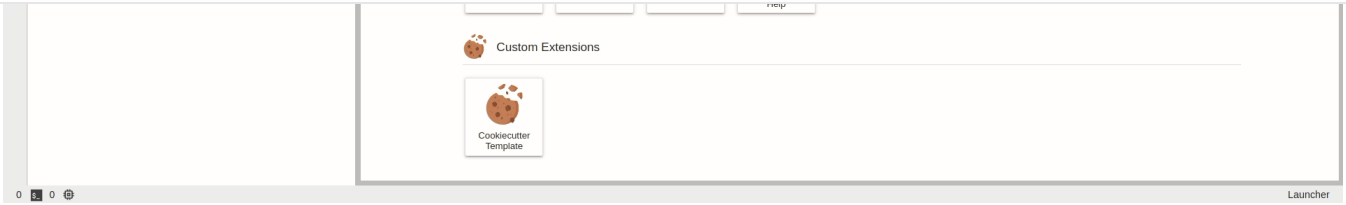
What if there is a magic button in Jupyter Launcher that can create a Cookiecutter file structure in the current folder?

Introducing the Jupyter Cookiecutter Extension:



A typical JupyterLab / JupyterHub Launcher. Credit: Ellip





A magical JupyterLab / JupyterHub Launcher! Credit: Louis Chan

To start with, we are going to use the official JupyterLab services example which already comes with the necessary components to make a custom extension in typescript. This blog will be focusing on how this plugin can improve our workflow, plus that you will have access to the repository, so we won't dive into the code. But feel free to drop me a note if you have any questions!

In essence, what this plugin does is that it would first track your current folder. Say we create a new folder called `example_project` and double-click to enter it, you should be able to see `example_project` above the Notebook section in your Launcher.

Once you click the Cookiecutter Template button, it would then create a folder structure as below. Give it a couple of seconds and refresh if nothing shows up; magic needs time.

```
├── README.md          <- The top-level README for
│                         developers using this project.
│
├── data
│   ├── external       <- Third party sources.
│   │   └── .gitkeep
│   ├── interim        <- In-progress intermediate data.
│   │   └── .gitkeep
│   ├── processed      <- The final data sets for modelling.
│   │   └── .gitkeep
│   └── raw            <- The original, immutable data.
│       └── .gitkeep
│
├── docs               <- A default Sphinx project; see
│   │                     sphinx-doc.org for details
│   └── .gitkeep
│
├── environment.yml    <- Conda environment file
│
├── logs               <- Folder for storing logging outputs
│   └── .gitkeep
│
```

```
│
├── notebooks          <- Jupyter notebooks. Naming convention
│   │                     is a number (for ordering), the
│   │                     creator's initials, and a short `-`
│   │                     delimited description, e.g.
│   │                     `1.0-jqp-initial-data-exploration`.
│   └── .gitkeep
│
├── references         <- Data dictionaries, manuals, and all
│   │                     other explanatory materials.
│   └── .gitkeep
│
├── reports            <- HTML, PDF, and LaTeX.
│   └── figures        <- Generated figures.
│       └── .gitkeep
│
├── requirements.txt   <- The requirements file for reproducing
│                         the analysis environment, e.g.
│                         generated with `pip freeze >
│                         requirements.txt` or `pipreqs src`
│
├── setup.py           <- Fill in the file to make this project
│                         pip installable with `pip install -e`
│
├── sql                <- SQL scripts, stored procedures etc.
│   └── .gitkeep
│
├── src                <- Source code for use in this project.
│   └── __init__.py    <- Makes src a Python module
│
├── tests              <- Scripts for unit testing
│   └── .gitkeep
│
└── .gitignore
```

There are a couple of key features to this specific folder structure:

1. Extended file structure based on Cookiecutter Data Science template with additional folders like `tests` for promoting test-driven development, `sql` for encouraging the use of databases over `pickle` (credit to my colleague who insisted to have this `sql` subfolder in place!), `logs` folder for tracking all the logging outputs etc.

2. Comes with `.gitkeep` files to make sure that all the folders will be tracked by git even if you don't have any files say under `tests` folder (remember, git by default would not track empty folders)

All of the above, within a single button!

Now you don't have any excuse not to start organising your projects folders! (at least at the very beginning of the project)

Limitations & Points to Note

1. This plugin has only been tested on `JupyterLab >= 2.0.0` and `JupyterHub >= 1.2.0`.
2. Again, magic takes time. So try not to create empty files until all the files and folders have been generated.
3. There will be an `Invalid response: 409 Conflict` error if the file / folder already exists. You will end up getting an `Untitled Folder` or `Untitled.py`.
4. The plugin has a hard-coded file structure right now. If you need a different structure, you will need to update `jupyter-cookiecutter-extension/src/index.ts` before install the plugin.
5. This is a Cookiecutter-**ish** plugin, and it does not come with other configurations like aws, s3, licenses, etc.
6. This plugin works best when you start your Jupyter Lab or Jupyter Hub in the parent directory of all your projects. For example, my local parent directory of my project folders is `/projects`, which means I should launch Jupyter at `/projects`, but not the actual project directory.
7. The repository comes with a Docker installation guide. If you would like to have it your Docker development environment, the Dockerfile should have you covered!

Repository

```
$ git clone https://gitlab.com/louis.chanloyuet/jupyter-
cookiecutter-extension
$ cd jupyter-cookiecutter-extension
$ git fetch && git checkout 1.0.0
```

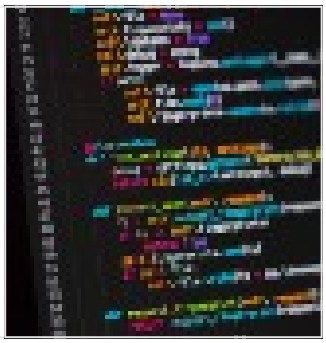
Louis Chan / Jupyter Cookiecutter Extension

You may also like this

Efficient Conditional Logic on Pandas DataFrames

Time to stop being too dependent on .iterrows() and .apply()

towardsdatascience.com



Efficient Root Searching Algorithms in Python

Implementing efficient searching algorithms for finding roots, and optimisation in Python

towardsdatascience.com

Before you go

Thanks for reading this blog! Being a disciplined data scientist is never easy; and I must confess I have never been one myself (***APOLOGIES***). Hopefully this plugin would lower the barrier for everyone to pick up some good habits to make our codebases better organised and more maintainable. Let me know if you have learnt something new from this! Feel free to drop your comments and questions in the comments. Until next time, stay safe, keep coding, and be my guest forking the repository.

Adios!

Louis Chan — Director, Data Science — FTI Consulting | LinkedIn

Ambitious, curious and creative individual with a strong belief in inter-connectivity between branches knowledge and a...

www.linkedin.com