Cookiecutter Data Science - VC Edition Documentation

Release 0.0.1

Victor Calderon <victor.calderon@vanderbilt.edu>

Nov 06, 2018

Contents

Part I

Author

- Victor Calderon (victor.calderon@vanderbilt.edu)
- Driven Data Team

Part II

Description

> **A logical, reasonably standardized, but flexible project structure for doing and sharing data science work.**
>
> —Data Science Team

Making your project **reproducible** is one of the key elements when doing research in pretty much any field, including Data Science. This package aims at making it easier to be transparent about what *steps* and *commands* are used when producing tangible results for a research project.

In the following sections, I will provide instructions on how to to use this project tempalte, as well as how to make the most out of this template.

This documentation is part of the repository cookiecutter-data-science-vc, and has been adapated from the Cookiecutter Data Science Project template by Driven Data organization.

Part III

Contents

CHAPTER 1

Getting Started

Author: Victor Calderon (victor.calderon@vanderbilt.edu)

Description: A logical, reasonably standardized, but flexible project structure for doing and sharing data science work.

1.1 Structuring your Project

Author: Victor Calderon (victor.calderon@vanderbilt.edu)

Description: A logical, reasonably standardized, but flexible project structure for doing and sharing data science work.

Now that your have a *working* version of **python** on your computer, you can start doing research.

One of the key elements of a project is for it to be **reproducible** by others. Having this in mind when you're structuring your project will allow others to look at your code, understand it well enough to be able to **recreate** your results.

This is a short guide on 2 ways to structure your code, without having to do much of creating documets, etc.

Table of Contents

- *Documentation for your new project*
- *Continuous Integration for your Project*

1.1.1 Cookiecutter and Folder structure

Cookiecutter is a command-line utility that creates projects from cookiecutters (project templates), e.g. Python package projects, LaTeX documents, etc.

Cookiecutter has been widely used for many projects, and each team and organization can create their own *template*. For more information, visit the cookiecutter documentation.

As the famous say goes:

> Don't reinvent the wheel!

You can always create your own folder and file structures, and organize your documents the old-fashioned way. The problem with this is that it may **vary** from project to project, and it will be more difficult to be consistent and effective throught your projects.

For this reason, I rely on `cookiecutter` templates to create the file and folder structure of a project.

There are many different `cookiecutter` templates out there, but after trying to find the best one that suits my needs in **research** and **programming**, I found one that works great! And after some modifications, I came up with a *version* of this template.

These two templates are shown in *Data Science - Cookiecutter* and *Personal version - Cookiecutter*. But first, let's make sure you have `cookiecutter` installed correctly.

1.1.2 Requirements to use `cookiecutter` templates

The minimum rquirements for creating `cookiecutter` templates are:

- Python 2.7 or 3.5
- Cookiecutter Python package >= 1.4.0: This can be installed with `pip` or `conda` depending on how you manage your Python packages.

You can install it by typing this on the terminal

```
pip install cookiecutter
```

or via Anaconda:

```
conda config --add channels conda-forge
conda install cookiecutter
```

Now you can use `cookiecutter` to create new templates for projects and papers!

1.1.3 Data Science - Cookiecutter

Cookiecutter Data Science is best described as

> A logical, reasonably standardized, but flexible project structure for doing and sharing data science work.

This folder structure allows everyone looking at your code to understand it right away. It also provides many different functions (as part of a `Makefile`) that simplify the workflow of your project.

In a nutshell, this cookiecutter includes:

- A **Makefile** file with **useful functions.
- **Documentation** to make your project easily accessible and readable
- And more!

In order to use this template, you follow the documentation in Cookiecutter Data Science.

1.1.4 Personal version - Cookiecutter

If you need more than the *normal* Data Science Cookiecutter template, you can use my version. Some of the differences are:

- It includes and easy-to-use `environment.yml` file that makes it easy to install dependencies.
- Extra functions in the `Makefile`.
- Choice of what kind of documenation to use. One has the option choose from *traditional* Read The Docs style or the Astropy Sphinx Theme.

You can check how these two styles look like:

- - **Read The Docs Version**
- - **Astropy Version**

Next, you can create your own Project based on this *cookiecutter* version

To start a new project and prompts

To start a new project, type the following:

```
$ cookiecutter https://github.com/vcalderon2009/cookiecutter-data-science
```

If you want the **default** project scheme from *DrivenData* (see above), run:

```
cookiecutter https://github.com/drivendata/cookiecutter-data-science
```

Depending on what kind of folder structure you want, you might want to choose from the different types.

After running this command, **you will be prompted some questions** regarding the parameters for the project. This will prompt you to answer a few questions like:

Question	Description
`project_name`	Name of the project. This can be similar to one on Github. Examples: • SDSS_analysis • Lung_cancer_analysis
`repo_name`	Name of the directory/repository, the project will be saved. This field *should not contain spaces should not contain spaces* Examples: • Calderon_Victor_Astro_PhD_Thesis • Szewciw_Adam_Astro_PhD_Thesis
`author_name`	Author's first name. It can include spaces Examples: • Adam Sanchez • Rose Roserberg
`author_email`	Author's email address. Examples: • some_email@gmail.com • another_email@yahoo.com
`short_description`	A short description of the project. This can be a *brief* overview of the project. Examples: • Repository for lung cancer analysis • Analysis on galaxies and cosmology
`long_description`	A longer version of `short_description` Examples: • Repository for lung cancer analysis • Analysis on galaxies and cosmology
`open_source_license`	Type of License for the paper. Without this, one cannot use any of. Options: • MIT • BSD 3-Clause • GNU GPL v3+ • Apache Software Licence 2.0 • BSD 2-Clause*
`s3_bucket`	Path to AWS storage. This is **temporarily** disabled!!
`aws_profile`	AWS profile name. This is **temporarily** disabled!!
`conda_python_env`	Name of the project's anaconda environment. In order to use the packages of this project, you need to first `activate` this environment. Examples: • lung_cancer_env • SDSS_galaxies_env
`github_username`	Author's Github username. This will be use tolink the project to the Github repository. Examples: • username • username2018

Using the Template

Now that one has answered the questions from *To start a new project and prompts*, you just need to populate the project with scripts, notebooks, and **of course**, documentation!!

The structure of the finalized project can be found in the *Project Structure* section.

Editing your environment

Now that you have a working proect from **cookiecutter**, you can start by editing the *environment* of your project.

If you downloaded **my version of cookiecutter**, you should be able to edit the `environment.yml` file. This file states which packages need to be installed by Anaconda and `pip` in order to run the scripts of the package.

The `environment.yml` file looks like the following:

```
name: name_of_environment

channels:
  - defaults

dependencies:
  - python>=3.6
  - ipython
  - anaconda
  - astropy
  - h5py
  - numpy
  - pandas
  - scipy
  - seaborn
  - pip
  - pip:
    - GitPython
    - progressbar2
```

You can edit the `environment.yml` file to include/exclude packages needed by your project.

After having edited the list of packages needed by your project, you can execute the command

```
$ make environment
```

to **create the environment**.

If you have done this step before, and you want to **update the environment**, you need to run

```
$ make update_environment
```

instead.

1.1.5 Adding your Project repository to Github

If you follow the instructions from above, you should have

- Downloaded the repository
- Created your own project with the desired file and folder structure
- Created your working **environment** for you project

The next step is to add it to Github and make it accessible.

To do this, your should do the following:

1. Create a Github repository with the **same name** as the repository.
2. Type `git add remote origin git@github.com:<username>/<project_name>.git`. In here you need to **replace** `<username>` and `project_name` with your details.
3. `git push origin master` - This will push your project to Github.

To check that you did this correctly, type

```
git remote -v
```

and you should get something that looks like this:

```
origin  https://github.com/<username>/<project_name>.git (fetch)
origin  https://github.com/<username>/<project_name>.git (push)
```

where `username` and `project_name` pertain to your repository on Github.

Now all of the files are online on Github, and should be ready to integrate them with Read The Docs.

1.1.6 Documentation for your new project

Now that you have both a working local and online copy of your code, the next step is to create the documentation for the project.

For this, you can easily use Read The Docs (RTD).

You need to do the following:

- Create an account on "Read the Docs"
- Go to your `Profile` and select `My Projects`
- From there, you should import the repository *manually* (it's easier). Click on `Import a Project` and follow the instructions.
- You should add the project with the **same name** as the Github Repo if possible. Otherwise, you might need to **change** the links to the *badges* on the `README.md` files in the project, among others.
- Make sure that the repository was correctly built by looking at the `Builds` and see that it compiled correctly. If not, it should tell you if there was an error and what the error was.
- Now you go and change the documentation depending on the project's needs.

1.1.7 Continuous Integration for your Project

Continuous integration deals with testing your code for possible errors, and making sure that everything is working as expected. Depending on your project's needs.

This template includes a `.travis.yml`, which the files used by Travis CI. Travis CI is a *Continuous integration* platform for testing your code, and checking the functionality of your project.

Project based on the modified version of the cookiecutter data science project template.

Project based on the modified version of the cookiecutter data science project template.

CHAPTER 2

Commands

The Makefile contains the central entry points for common tasks related to this project.

This section is dedicated towards the functions used through the analysis.

Project based on the modified version of the cookiecutter data science project template.

CHAPTER 3

Project Structure

The organization of the project is the following:

```
├── LICENSE
├── Makefile           <- Makefile with commands like `make data` or `make train`
├── README.md          <- The top-level README for developers using this project.
├── data
│   ├── external       <- Data from third party sources.
│   ├── interim        <- Intermediate data that has been transformed.
│   ├── processed      <- The final, canonical data sets for modeling.
│   └── raw            <- The original, immutable data dump.
│
├── docs               <- A default Sphinx project; see sphinx-doc.org for details
│
├── models             <- Trained and serialized models, model predictions, or model summaries
│
├── 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`.
│
├── references         <- Data dictionaries, manuals, and all other explanatory materials.
│
├── reports            <- Generated analysis as HTML, PDF, LaTeX, etc.
│   └── figures        <- Generated graphics and figures to be used in reporting
│
├── requirements.txt   <- The requirements file for reproducing the analysis environment, e.g.
│                         generated with `pip freeze > requirements.txt`
│
├── environment.yml    <- The Anaconda environment requirements file for reproducing the analysis␣
↪environment.
│                         This file is used by Anaconda to create the project environment.
│
├── src                <- Source code for use in this project.
│   ├── __init__.py    <- Makes src a Python module
│   │
```

(continues on next page)

(continued from previous page)

```
    ├── data           <- Scripts to download or generate data
    │   │
    │   └── make_dataset.py
    │
    ├── features       <- Scripts to turn 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
    │
    └── visualization  <- Scripts to create exploratory and results oriented visualizations
        └── visualize.py
    │
└── tox.ini            <- tox file with settings for running tox; see tox.testrun.org
```

Project based on the modified version of the cookiecutter data science project template.

Project based on the modified version of the cookiecutter data science project template.