https://trallard.github.io/Talks/Pycon

Down the rabbit hole.

A 101 on reproducible workflows with Python

Tutorial session at Pycon 2018

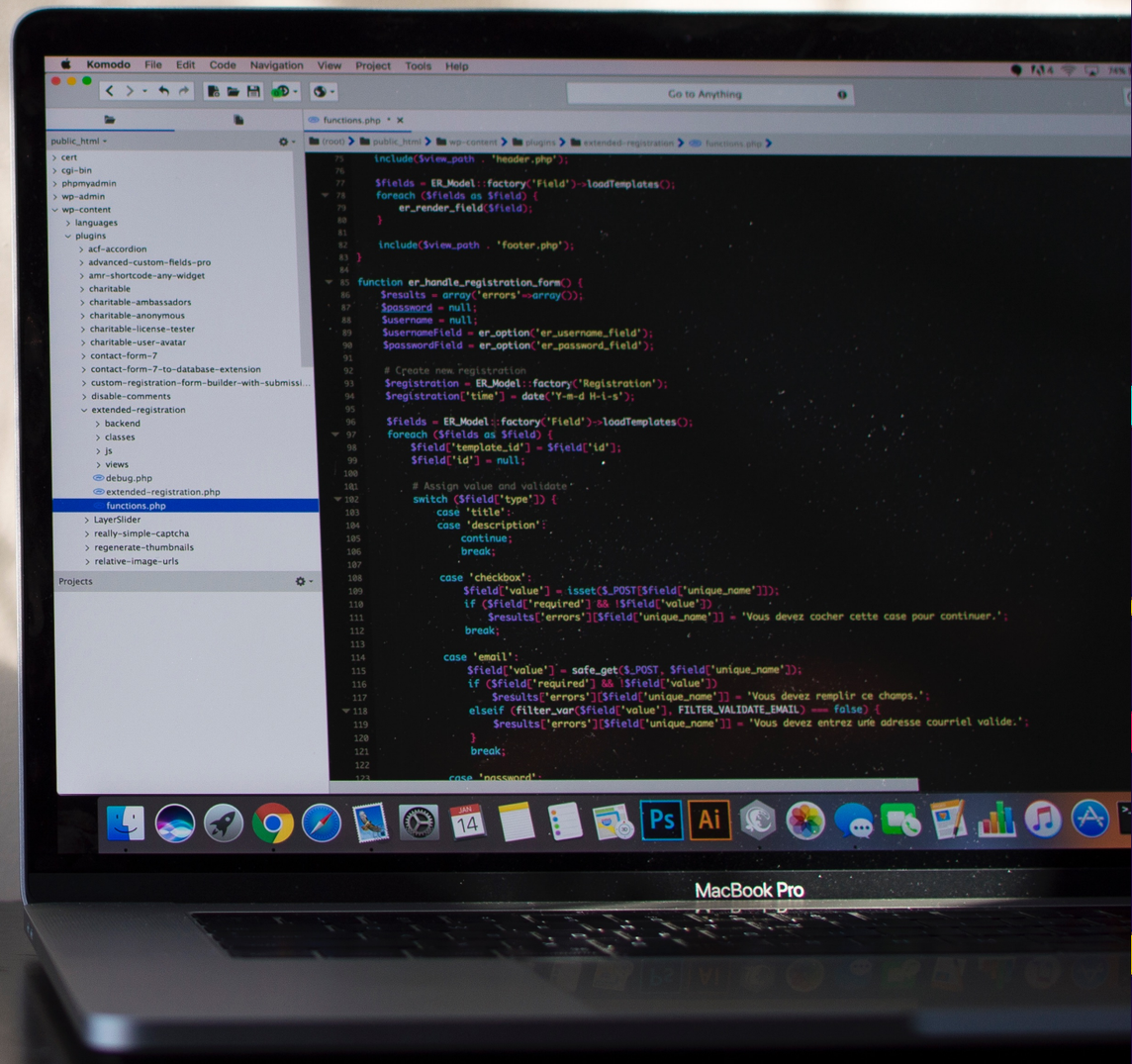
PRESENTED BY

Tania Allard, PhD

Tania Allard

Research Software Engineer
at the University of Leeds

trallard
ixek



What will we do today?

1. Learn moe about reproducibility
2. Use a reproducibility-first approach to set up a new project
3. Set up and use version control to track our changes
4. Add data to our project (and metadata)
5. Set up a reproducible analysis workflow
6. Create a run all script for the project: automate all the things!
7. Learn more about testing
8. Record our project's provenance
9. Learn how to run automated tests
10. Share your analysis with the world!
11. And being credited for it!

Introduction to reproducibility

Everything you always wanted to know but you were too afraid to ask

The father of reproducibility

~150 years ago Pasteur demonstrated how experiments can be conducted reproducibly and the value of doing it that way.

Tartrate double d'Ammoniaque et de soude.

J'ai examiné un très grand nombre de cristaux de tartrate double d'ammon. et de soude et d'autres sans exception lorsqu'on place la base P horizontale, la face G verticale vis à vis, c'est à gauche entre G et P qu'existent les faces hémiédriques on n'en a jamais à droite. On remarquera que cette disposition ne dépend point de l'une ou de l'autre base P. Tous les cristaux faces comme on vient de dire offrent l'hémiédrie ainsi disposée. J'en ai examiné un nombre considérable. Il suffirait qu'il y eut un seul qui offrit à droite de G à P les faces hémiédriques pour que l'hémiédrie n'existât pas ou du moins fut douteuse.

Paratartrate double de Soude et d'Ammon.

Les cristaux sont souvent hémiédres à gauche, souvent à droite, ~~et ils sont la symétrie~~. C'est là qu'est la différence des deux sels

8 gr. Paratarte (hémiédres à droite) dissous dans 55,5 cent. cub. d'eau ont donné dans un tube de 20 cent. une déviation 2,8 = 6°

à 17°. (1)

8 gr. Tartrate de S. et Am. dans les mêmes circonstances m'a donné 7°54' à 17°, à droite

(1) Il est à noter qu'il est difficile malgré le choix des cristaux, de les avoir exactement tous ceux hémitropes à droite de tous ceux hémitropes à gauche. L'hémitropie très distincte en général sur les cristaux un peu gros l'est difficilement sur les petits cristaux implantés sur les gros. Probablement la déviation serait la même pour des cristaux très bien choisis.

Antibiotics

Beers!!!

There *is* a reproducibility (chronic) problem

Rather than a reproducibility crisis

BUSINESS INSIDER MARKETS

Shocking Paper Claims That Microsoft Excel Coding Error Is Behind The Reinhart-Rogoff Study On Debt

Mike Konczal, NewDeal2.0

Apr. 16, 2013, 12:40 PM 92,101

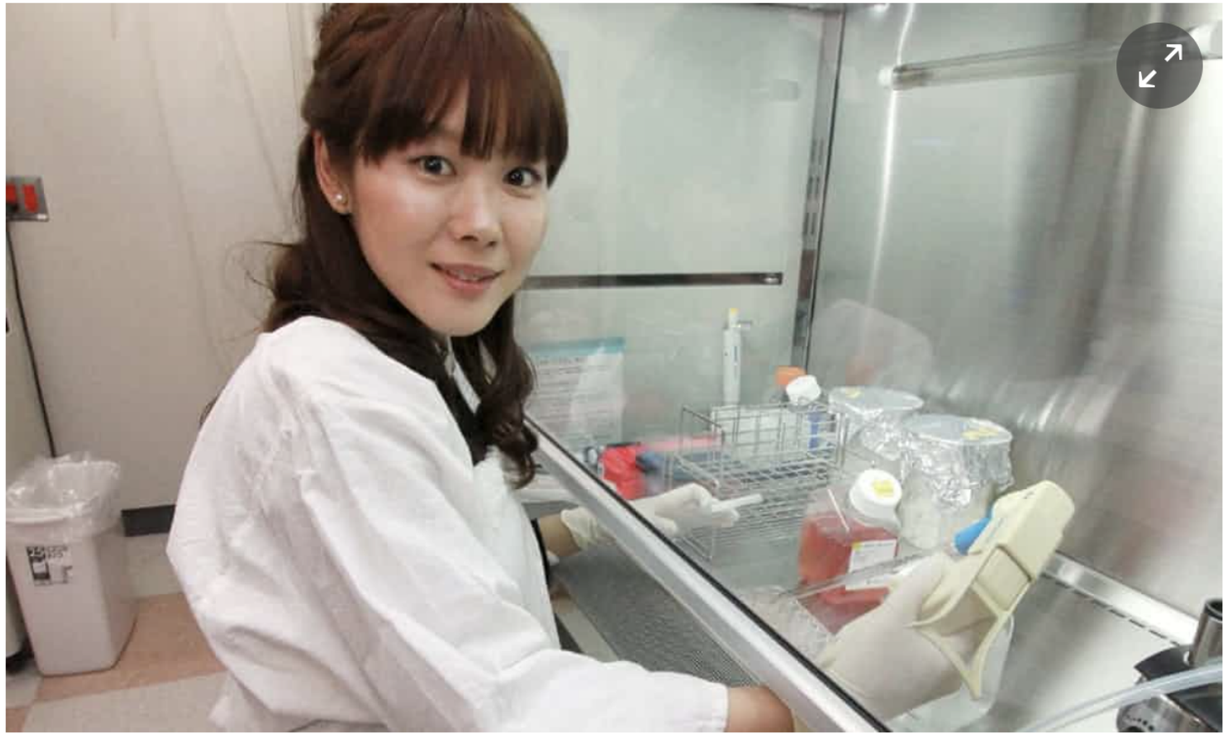
THE WALL STREET JOURNAL.

REAL TIME ECONOMICS

Reinhart, Rogoff Admit Excel Mistake, Rebut Other Critiques

What pushes scientists to lie? The disturbing but familiar story of Haruko Obokata

The spectacular fall of the Japanese scientist who claimed to have triggered stem cell abilities in regular body cells is not uncommon in the scientific community. The culprit: carelessness and hubris in the drive to make a historic discovery



Riken researcher Haruko Obokata working at her laboratory in Kobe. Photograph: Jiji Press/AFP/Getty Images

The year 2014 was one of extremes for Haruko Obokata. A year of high highs and even lower lows. Barely 30 years old, she was head of her own laboratory at the Riken Center for Developmental Biology (CDB) in Kobe,

http://www.bitss.org/2015/12/31/science-is-show-me-not-trust-me/

Simon Hettrick
@sjh5000

The reproducibility crisis is like climate change. It's an issue of vital importance to the general population. It relies completely on a small group of experts to warn people of the dangers. It's largely overlooked by policymakers in their rush to deal with short-term problems.

11:35 AM - Apr 5, 2018

17 See Simon Hettrick's other Tweets

🤔 But what is reproducibility?

Glad you asked

Data Replication & Reproducibility

PERSPECTIVE

Reproducible Research in Computational Science

Roger D. Peng

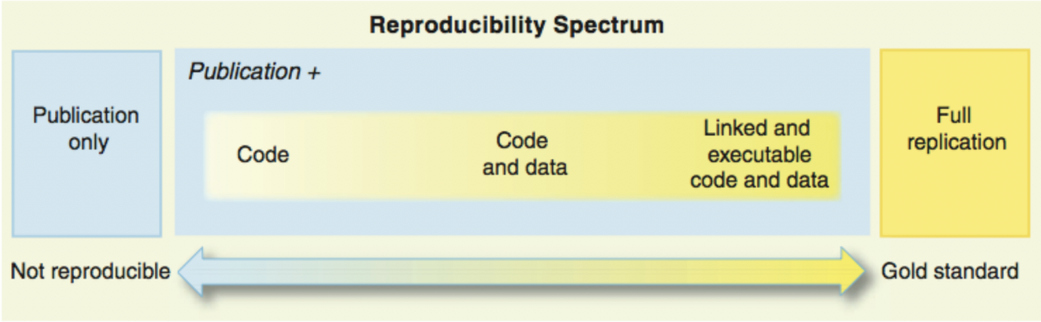
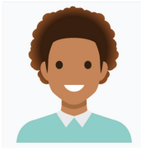


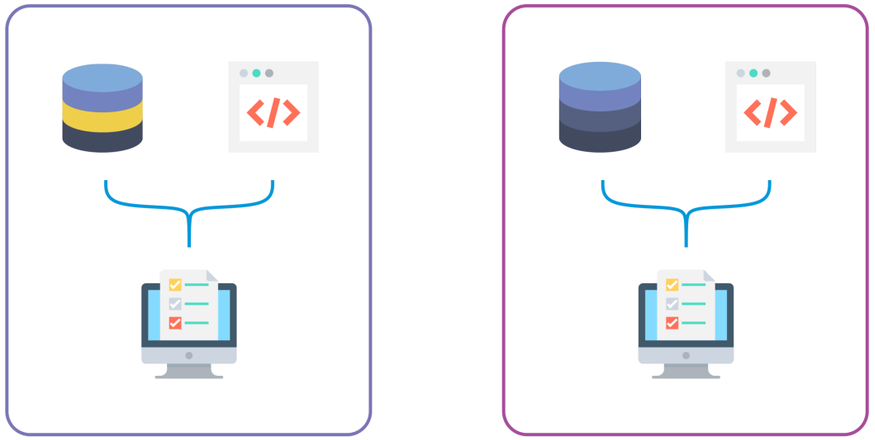
Fig. 1. The spectrum of reproducibility.

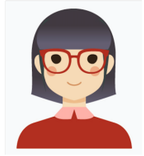
Peng(2011) http://dx.doi.org/10.1126/science.1213847

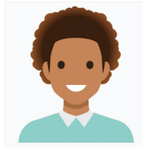
Reproducible



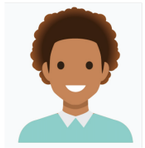
Replicable







Robust



And why should I care?

Convention

"It is like agreeing that we will all drive on the left or the right. A hallmark of civilization is following conventions that constrain your behaviour a little, in the name of public safety"

- Jenny Brian on *Project oriented workflows*

Technical scenario

Your objective is to have a complete chain of custody (provenance) from your raw data to your finished results and figures.

Always be able to figure out what code and data were used to generate which result.

If using version control you can also refer to specific versions of your study (i.e. manuscript, first quarter report, Nobel Prize committee version)

Practical scenario

Imagine someone manages to sneak into your lab at night AND deletes everything except for your code and data ('cause these are in safe repositories)


PASSWORD
01

Imagine being able to run a single command to generate everything including results, tables, and figures in their final polished form.

Wouldn't it be great?

Better yet if someone completly unfamiliar with your project could be able to look at these files and understand what you did and why (i.e. readers, collaborators, your replacement, you in 6 months time).

From a sustainability point of view

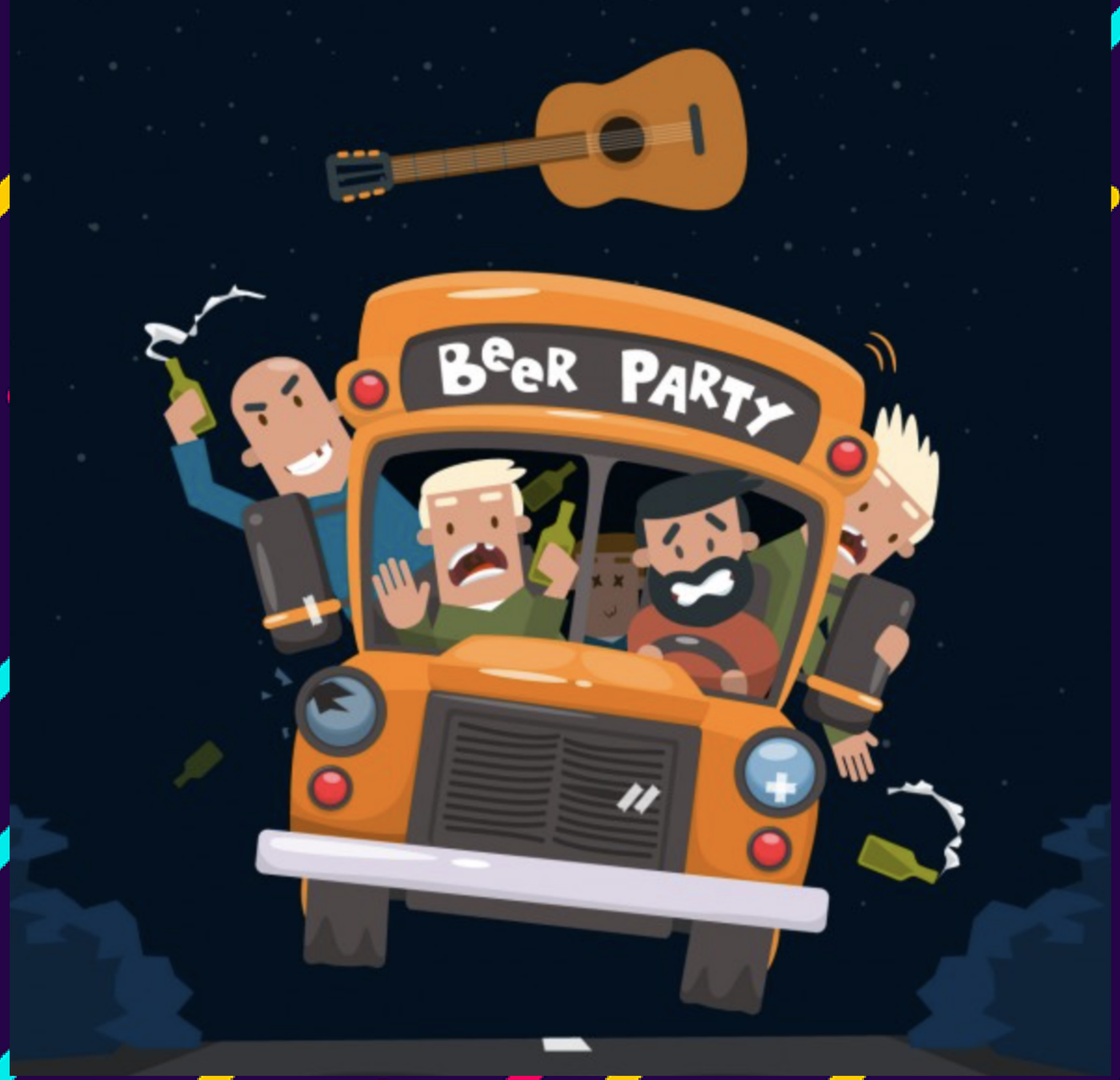
Speed scientific progress

Contribute to open source and our beloved community

Acquire more varied, highly valuable skills

And my all time favourite... increase the bus factor

ixek


BEER PARTY

Discussion time

- What measures do you take to ensure your analyses are: reproducible, replicable or robust?
- Have you ever encountered any barriers to reproducibility? Of what sort?
- Write your thoughs on this Etherpad

How can we make our analyses more reproducible?

We can share (as and when possible):

- Well documented codes
- Data used to produce the results
- The details of the workflows used
- Information on how to cite our work

Before we move on

I know you are eager to get on with the hands-on session but there are some requirements that need to be satisfied. Make sure you have the following installed:

- Python 3.x
- Jupyterlab
- shell

- recipy
- pytest
- cookiecutter

- matplotlib
- pandas
- nbval and nbdime

Also you need to have a GitHub account and a Travis CI account (make sure to get it from travis-ci.org so that it is for free!)


Ready to get your reproducibility skills on?


Every Adventure
Requires
a First Step
- Cheshire Cat


PROJECT PLAN
NOW THAT'S A TERM I'VE
NOT HEARD IN A LONG TIME

Setting yourself for success

Questions to ask	Possible scenarios	Tools and helpers
What is your MVP? Final product?	Thesis, article, internal report	Latex, Jupyter notebooks, scripts
Who will use my data and code?	Only me, colleagues, other groups	Git (private/public) , License, documentation
Who and how long will this be maintained for?	6 months, until studies compleition	Collaborators, OSS community
What about my other assets? (data, slides, workflows)	Can be shared, published, deposited in a repository	Figshare, institutional database, web page

Have you got a minute to talk about open source?

"Open-source licenses allow people to coordinate their work freely, within the confines of copyright law, while making access and wide distribution a priority I've always thought that this is fundamentally aligned with the method of science, where we value academic freedom and wide dissemination of scientific findings."

- Lorena barba *(Casting out nines Interview)*

Quick guide to licensing

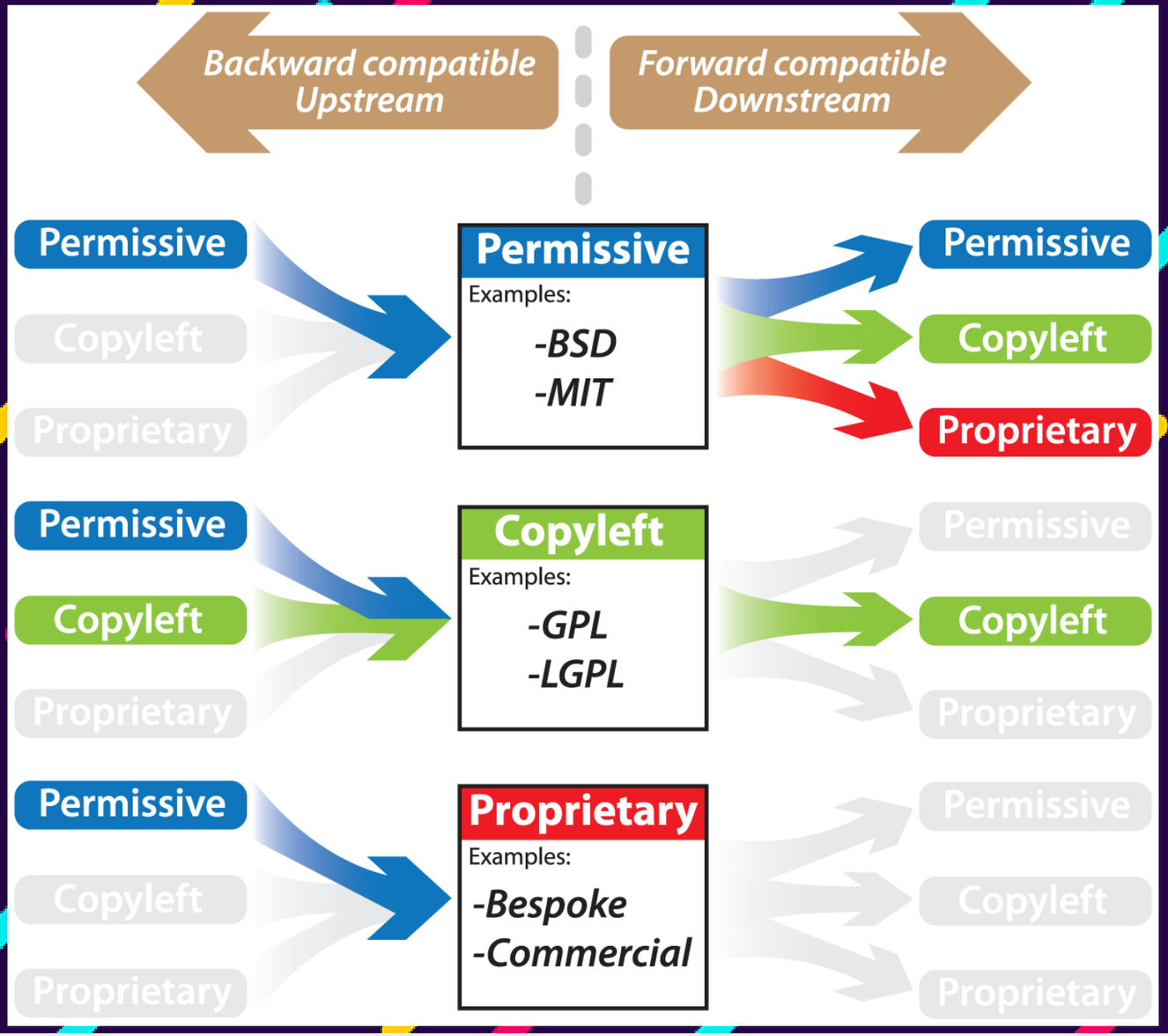
Open data and content can be freely used, modified, and shared by anyone for any purpose (The Open Definition)

- Simply making the source code public does not make your project open source
- Code has copyright, and without a license others don't know if they can use it or not (always add a license)

- Permissive licenses give more freedoms: authors need to be credited (MIT, BSD, Apache License)
- Copyleft (share-alike) licenses restrict the use of software by requiring that any derivative works be *also* under the license of the original (GPL).

> The website http://choosealicense.com/ is a good starting point.
> Also make sure to check their Appendix with a table of all FOSS licenses and features.

Not all licenses are compatible! Also compatibility is directional.



Morin et al (2012). PLOS

The White Rabbit put on his spectacles. **"Where shall I begin, please your Majesty?"** he asked.

"Begin at the beginning," the King said gravely, **"and go on till you come to the end: then stop."**

Quick poll: 5 mins

🤔 How do you start a new project? Do you have any preferred files, directory, data structure or naming conventions?

📝 Write your ideas **on this Etherpad**

Project structure

Good project layout ensures:

- Integrity of the data
- Portability of the project
- Easier to pick the project back up after a brak

There is no single way to organise a project.... but we need to take advantage of the power of convention.

> "A place for everything, everything in its place"
>
> - Benjamin Frankin

```
.
├── LICENSE
├── README.md
├── bin                <- Compiled codes or binaries*
├── config             <- Configuration files*
├── data
│   ├── external       <- Data from third party sources*
│   ├── interim        <- Intermediate data that has been transformed
│   ├── processed      <- The clean data set
│   └── raw            <- The original, immutable data dump
├── docs               <- Package documentation
├── notebooks          <- Jupyter or Rmarkdown noteboks
├── reports            <- For a manuscript source, e.g., LaTeX, md
├── figures            <- Figures for the manuscript or reports
├── output             <- The results of your analysis
└── src                <- Source code for this project
    ├── data           <- scripts and programs to process data
    ├── models         <- Source code for your own model
    ├── tools          <- Any helper scripts go here (utils)
    └── visualization  <- Scripts for visualisation of your outputs
```

☆The sections marked with a * are optional

You can do this manually, from the shell or use whatever method you prefer. But we want an opinionated approach for this.

We are going to use Cookie cutter to create our base filesystem and some support documents.

Using our command line we will first activate our conda environment.

```
$ source activate reproPython
```

Now we can create our project structure

```
$ cookiecutter gh:mkrapp/cookiecutter-reproducible-science
```

☆ Note that there is a Data science cookiecutter version

Version control from the begining

We will initialize a Git repository within this directory.
From your shell:

```
$ cd reproProject  # go to the newly created project
$ git init         # initialize a repository here
$ git status       # check the status (new/modified files)
```

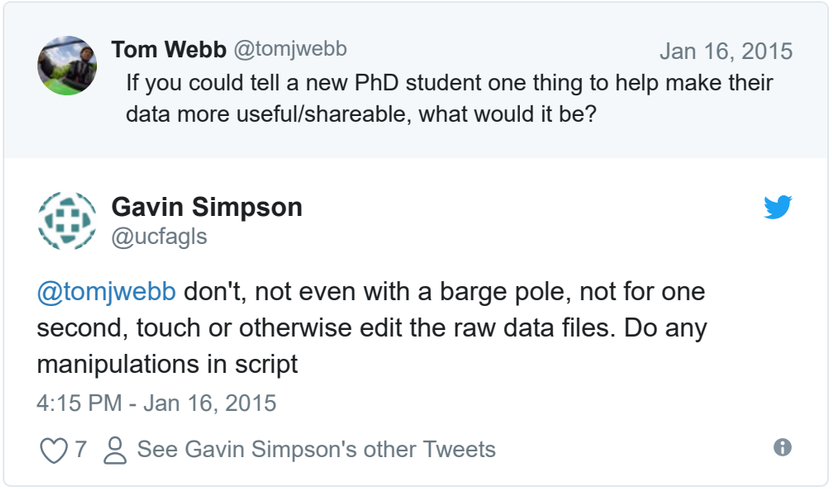
If you are happy with the LICENSE.md you can do your first commit now:

```
$ git add .          # Adding all the changes made
$ git commit -m 'Create initial structure'  # Commit and comment
```

☆ Remember a commit is like a snapshot of your project at a specific time

If at any point you need to know if you are in a repository type `git status` in your shell

Raw data are sacrosanct



Getting started with your data

Should I version control my raw data?

> The raw data *should* never change. All of the processing or data wrangling must be done in copies of your raw data.

However, it is important to consider how you are going to share your data. The best practices for sharing data on the Web summary is a good place to get started.

You can also explore alternatives like gitannex to version control your data, or FigShare to share your data.

Adding raw data

We will be using the Kaggle wine review dataset. The first step is to download the data and store it in our `data/raw` directory.

You should have a copy of the datasets that we will be using already if you followed the installation instructions. Othwerwise you can get a copy using the link in the Etherpad.

You got data... is it enough?

Data without documentation has no value

☆ metadata = data about data ☆

Information that describes, explains, locates or makes it easier to find, access, and use a resource (in this case data)

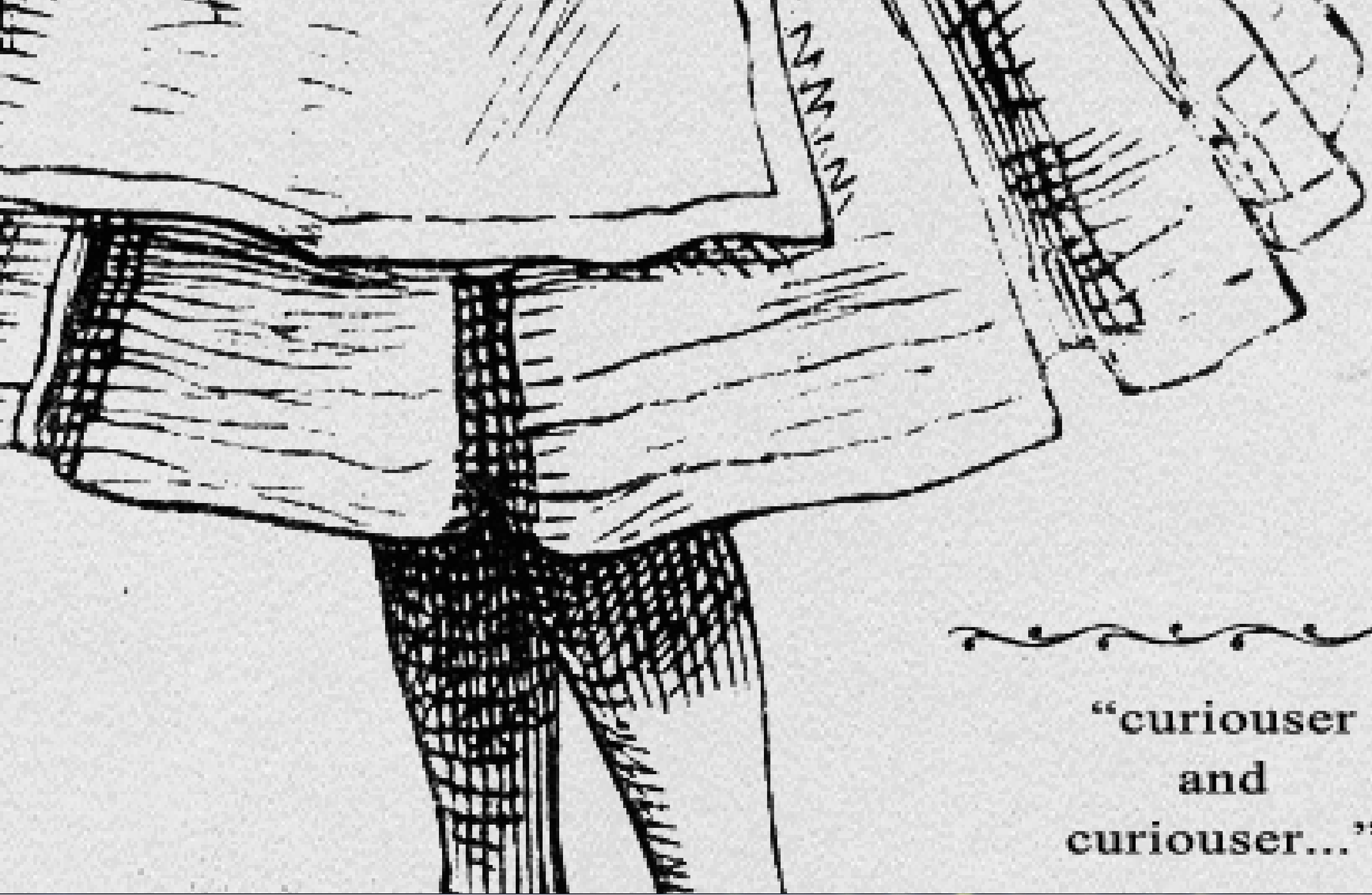
Adding metadata

You want to make sure that all your data has information describing how you got the data, the meaning of the columns, etc.

For your own use make sure to create at least a README file describing the data as best as you can. Create a README.txt (or .md) file inside the data directory and add the following content or something similar.

```
Title: Reproducible Python
keywords: wine, reviews, magazine, kaggle
Data collected from Kaggle winemag reviews
URL:  https://www.kaggle.com/residentmario/renaming-combining-data/data
Collected on: 09/05/2018 by Tania Allard
```

Commit the README to git



"curiouser
and
curiouser..."

Adding the code that performs the analysis

We want to do the following:

- Create a Jupyter notebook for exploratory analysis
- Generate the following outputs using python scripts:
  - Generate a subset of `winemag-130k-v2.csv` containing only the following columns: `country, designation, points, price (in GBP)`. Save in a .csv file
  - Generate and save a table of wines only produced in Chile
  - Save a scatterplot of the wines points vs price and a distribution plot of wine scores

Don't worry you do not have to generate all of the scripts... we have provided some scripts for you to get started. You should now have a directory called `SupportScripts`

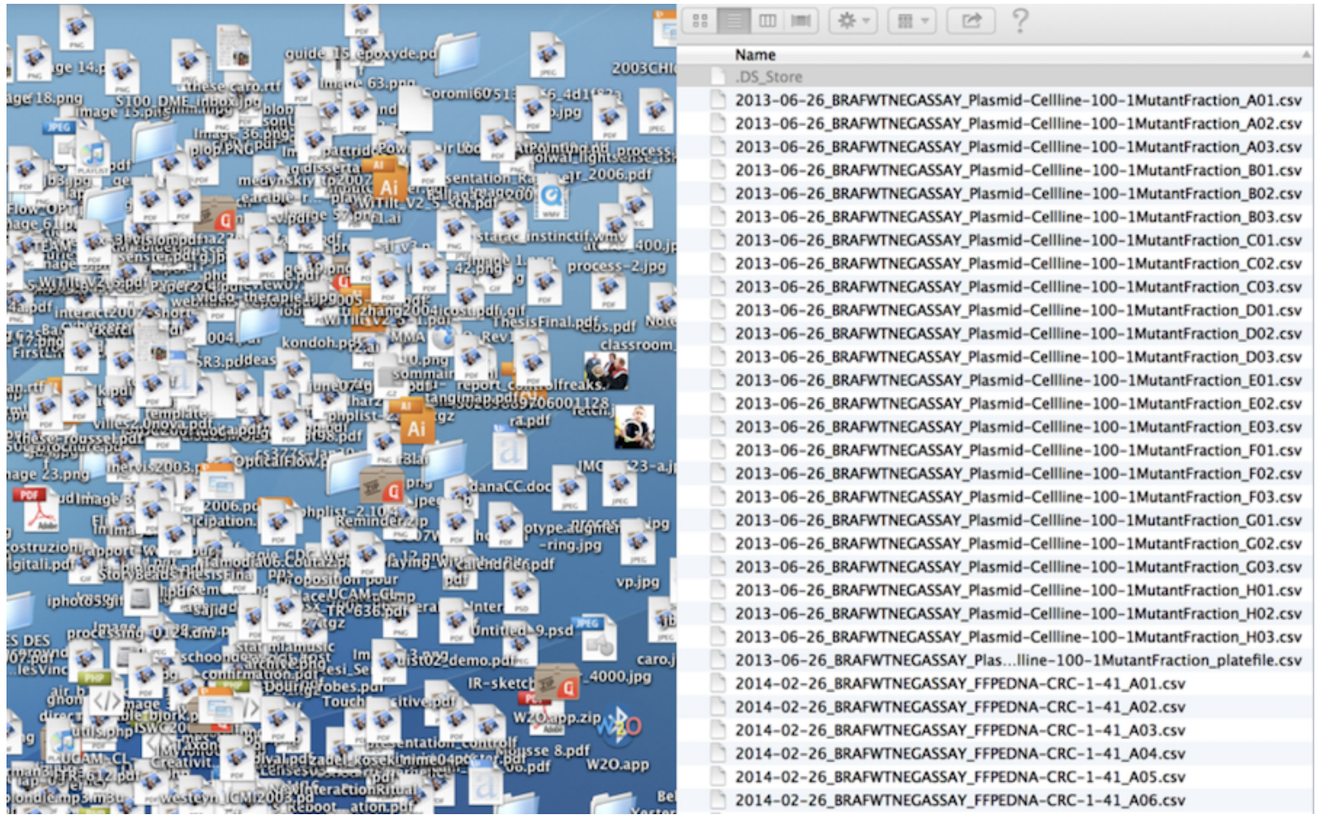
You need to make sure that they are in the appropriate directory inside your newly created project.

- Noteboks
- src/data
- src/visualization

Once this is done commit your changes to git

```
$ git add .
$ git commit -m "Add processing scripts"
```

Let's face it.... there are going to be files LOTS of files



The art of naming

The three principles for (file) names:

- Machine readable : regex and globbing friendly, deliberate use of delimiters *
- Human readable: contains info on content, connects to concept of slug from semantic URLs
- Plays well with default ordering: put something numeric first, use ISO 8601 for dates YYYY-MM-DD

* Avoid spaced, accented characters, files 'foo' and 'Foo'


PUBLIC SERVICE ANNOUNCEMENT:
OUR DIFFERENT WAYS OF WRITING DATES AS NUMBERS CAN LEAD TO ONLINE CONFUSION. THAT'S WHY IN 1988 ISO SET A GLOBAL STANDARD NUMERIC DATE FORMAT.
THIS IS THE CORRECT WAY TO WRITE NUMERIC DATES:
2013-02-27
THE FOLLOWING FORMATS ARE THEREFORE DISCOURAGED:
02/27/2013 02/27/13 27/02/2013 27/02/13
20130227 2013.02.27 27.02.13 27-02-13
27.2.13 2013. II. 27. 27½-13 2013.158904109
MMXIII-II-XXVII MMXIII LVII/CCCLXV 1330300800
((3+3)×(111+1)-1)×3/3-1/3^3 2013
10/11011/1101 02/27/20/13 2 3 1 4 0 1 2 3 7 5 6 7 8
hisssss
2-27-13

What works and what doesn't

⃠ NO	✓ YES
report.docx	2018-02-03_report-for-sla.docx
Joey's filename has spaces and punctuation.xlsx	joeys-filenames-are-getting-better.xlsx
fig 1.png	fig01_scatterplot-talk-length-vs-interest.png

Running Jupyter Lab

We will be using Jupyter lab to write, execute, and modify our scripts and notebooks. You should have this installed already. We are going to start an instance by typing on the shell:

```
$ jupyter lab
```

The scripts

Let's start by checking the scripts and notebooks:

- 00_explore-data.ipynb: exploratory analysis
- 01_subset-data-GBP.py: subset of winemag-130k-v2.csv containing only the following columns: country, designation, points, price (in GBP). Save in a .csv file
- 02_visualize-wines.py
- 03_country-subset.py

> The best file names are self exploratory

You can run them from your shell like so:

```
$ python src/data/01_subset-data-GBP.py data/raw/winemag-data-130k-v2.csv
$ python src/visualization/02_visualize-wines.py data/interim/2018-05-09-winemag_priceGBP.csv
$ python src/data/03_country-subset.py data/interim/2018-05-09-winemag_priceGBP.csv Chile
```

Make sure to be at the root of your directory e.g. `reproPython-test`

☹ What problems did you encounter?

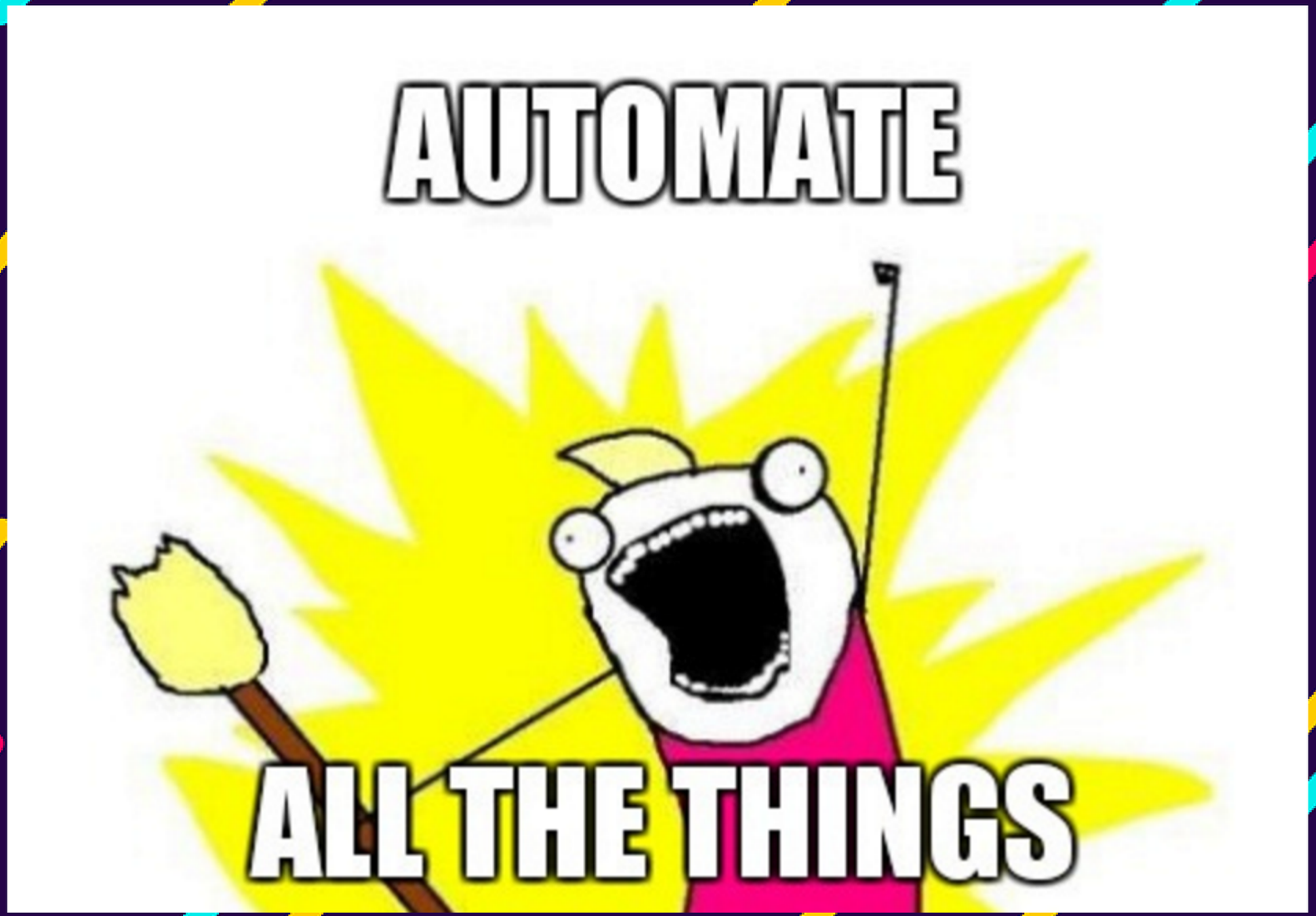
Besides, this gets quite boring pretty soon and still depends a lot on the user.

Documentation

Documentation is an important part of a reproducible workflow.

Take 5 minutes and identify which scripts/notebook have the best documentation. What makes it a good documentation?

A good point to start is checking the Google Python style guidelines


AUTOMATE
ALL THE THINGS

Packaging

We used a modular approach here, so we can use and reuse the functions more efficiently. The next step it to make a `runall` script to minimize the user interaction.

First, we need to make sure that Python recognizes our scripts as a package so we can call functions from the multiple modules.

From the shell:

```
$ touch src/data/__init__.py            # Ensures Python understands
$ touch src/visualization/__init__.py   # that we are creating a package
$ touch src/__init__.py
```

Creating the run all script

> We will run everything from the root directory.
> As such all the paths will be relative to the top level of your project

Since our modules start with digits (i.e. `01`, `02`) we cannot do the import as we'd normally do

```
from mypackage import myAwesomeModule
```

Instead we need to do it like so:

```
subset = importlib.import_module('.data.01_subset-data-GBP', 'scripts')
plotwines = importlib.import_module('.visualization.02_visualize-wines', 'scripts')
country_sub = importlib.import_module('.data.03_country-subset', 'scripts')
```

Also, we need to make sure that the other subpackages/modules are imported correctly. Add the following to `src/__init__.py`

```
from . import data
from . import visualization
```

TO DO:

How would you do to run the analysis from step 01 (process the data) to 03 subset for a country and plot the results?

Once you have done this and make sure you can run it from your shell and commit the changes to git.

Note you might need to run this from the shell like so

```
python -m scripts.runall-wine-analysis
```

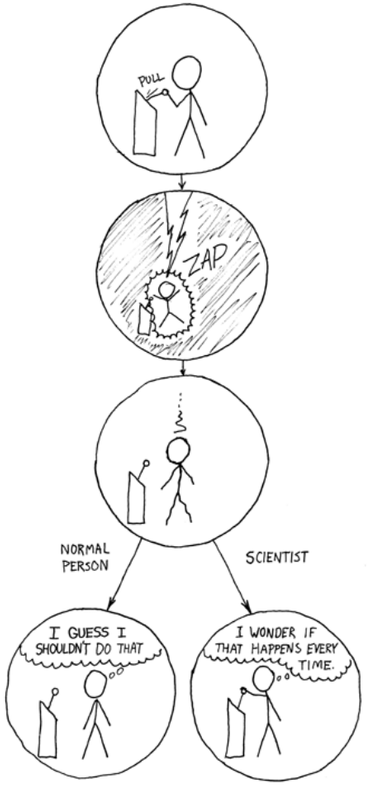
Well it is more like testing time

Testing

We now have a fully automated script! 🎉👏🦄

😕 Such a shame we still cannot guarantee the results are correct... or that there are no bugs.

The next step is to include tests... in fact testing should be a core part of our development process. In fact all of our reproducible workflows are analogous to experimental design in the scientific world



https://xkcd.com/242/

There are various approaches to test software:

- Assertions: 🐴 == 🐴
- Exceptions: (within the code) serve as warnings ⚠
- Unit tests: investigate the behaviour of units of code (e.g functions)
- Regression tests: defends against 🐛
- Integration tests: ⚙ checks that the pieces work together as expected

Exceptions

Remember when you tried to run `02_visualize-wines.py`? It woud not work unless you had created a figures directory beforehand.

We can catch this kinds of errors by adding this piece of code:

```
try:
        # try to save the figure
        fig.savefig(fname, bbox_inches = 'tight')
except OSError as e:
    # wowza! the directory does not exist
    os.makedirs('figures')
    print('Creating figures directory')
    fig.savefig(fname, bbox_inches='tight')
```

Now our `runall script` should work!!! 🎉 🎉

```
$ python src.runall-wine-analysis
```

Unit testing

Open `03_country-subset.py` and add the following function:

```
def get_mean_price(filename):
    """ function to get the mean price of the wines
    rounded to 4 decimals"""
    wine = pd.read_csv(filename)
    mean_price = wine['price'].mean()
    return round(mean_price, 4)  # note the rounding here
```

An modify the get_country function

```
def get_country(filename, country):
    # Load table
    wine = pd.read_csv(filename)

    # Use the country name to subset data
    subset_country = wine[wine['country'] == country ].copy()

    # Constructing the fname
    today = datetime.datetime.today().strftime('%Y-%m-%d')
    fname = f'data/processed/{today}-winemag_{country}.csv'

    # Saving the csv
    subset_country.to_csv(fname)
    print(fname)  # print the fname from here

    return(subset_country)  #returns the data frame
```

Create the testing suite

To run the tests we are going to use pytest. You can find more information in the following resources:

- Pytest usage examples can be found here
- Rules for test discovery

Now we can create our tests:

```
$ mkdir tests                     # Create tests directory
$ touch tests/__init__.py         # Help find the test
$ touch test_03_country_subset.py # Create our first test
```

☆ Your test scripts name must start with: `test`

Modifying `test_03_country_subset.py`

```
import importlib

country = importlib.import_module('.data.03_country-subset', 'src')

interim_data = "data/interim/2018-05-09-winemag_priceGBP.csv"
processed_data = "data/processed/2018-05-03-winemag_Chile.csv"

def test_get_mean_price():
    mean_price = country.get_mean_price(processed_data)
    assert mean_price == 20.7865
```

Run from the shell using `pytest`

What if we want to consider all the decimal numbers?

```
import importlib
import numpy.testing as npt

country = importlib.import_module('.data.03_country-subset', 'src')

interim_data = "data/interim/2018-04-30-winemag_priceGBP.csv"
processed_data = "data/processed/2018-04-30-winemag_Chile.csv"

def test_get_mean_price():
    mean_price = country.get_mean_price(processed_data)
    assert mean_price == 20.7865
    npt.assert_allclose(country.get_mean_price(processed_data),
                                    20.787, rtol = 0.01)
```

Run from the shell using `pytest`

What else could go wrong?

What if we created a data set and we want to make sure that my interim or raw data has not changed? -> What about my dataframes?

```
import pandas.testing as pdt
import pandas as pd


interim_data = "data/interim/2018-05-09-winemag_priceGBP.csv"
processed_data = "data/processed/2018-05-09-winemag_Chile.csv"

def test_get_country():
    # call the function
    df = country.get_country(interim_data, 'Chile')

    # load my previous dataset
    base = pd.read_csv(processed_data)

    # check if I am getting a dataframe
    assert isinstance(df, pd.DataFrame)
    assert isinstance(base, pd.DataFrame)

    # check that they are the same dataframes
    pdt.assert_frame_equal(df, base)
```

See what we just did?

We tested each of the functions in our module.
Notice something in the functions we just wrote?

- Set-up: `mean = country.get_mean(interim_data)`
- Assertions: `assert mean_price == 20.786`

Now don't forget to commit your code:

```
$ git add .
$ git commit -m "Add unit test suite"
```

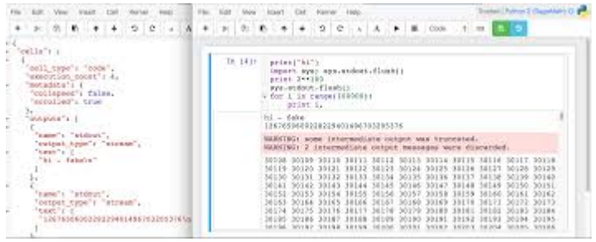
Past as truth

Regression tests assume that the past is “correct.” They are great for letting developers know when and how a code base has changed. They are not great for letting anyone know why the change occurred. The change between what a code produces now and what it computed before is called a regression.

How many times have you tried to run a script or a notebook you found online just to realize it is broken?

Let's do some regression testing on the Jupyter notebook using nbval

We first need to understand how a Jupyter notebook works. All the data is stored in a .json like format (organised key, data values)... this includes the results, code, and markdown.



Nbval checks the stored values while doing a *mock run* on the notebook and compares the saved version of the notebook vs the results obtained from the mock run

Try it on your shell

```
$ pytest --nbval src/data/00_explore-data.ipynb
```

What would happen if you were to have a cell like this one?

```
import time
print('This notebook was last run on: ' + time.strftime('%d/%m/%y') + ' at: ' + time.strftime('%H:%M:%S'))
```

Provenance

Image you created a beautiful graph and some results that makes your research Nobel worthy. Of course you ran the workflow multiple times doing minimal changes every single time. But now, 6 months later you need that **one** plot for you Nobel!!

We can use the package recipy to log each run of your code to a database, keeping track of the input files, output files and the version of your code, and then let you query this database to find out how you actually did create graph.png

Make sure everything is commited to git before carrying on.

Add the following line to your runall-wine-analysis script

```
import recipy
```

Note that this has to be the firts import Run the script again `python -m src.runall-wine-analysis`

Try using the `recipy latest` and `recipy gui` commands

“This is impossible.”

Alice

“Only if you believe it.”

The Mad Hatter

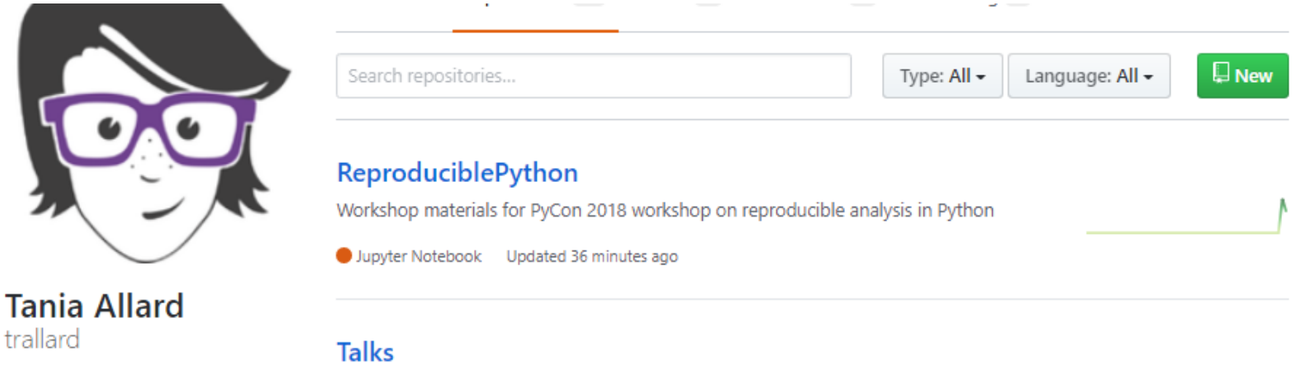
Sharing your code with the world

We now have a fully automated and tested workflow!!! 😁🎉😇

And we are ready to share our awesomeness with the world.

Head to https://github.com/ and login to your account

In your repository section click on the green new repository button



Make sure to make this a public repository and do not add a README or a License since we aleady have these.

We need to link your local project to your GitHub repository In your repository section click on the green new repository button

type those command on your shell (or copy and paste). Use your own details

Refresh your web browser and... ta dah! Your project is online

Continuous integration

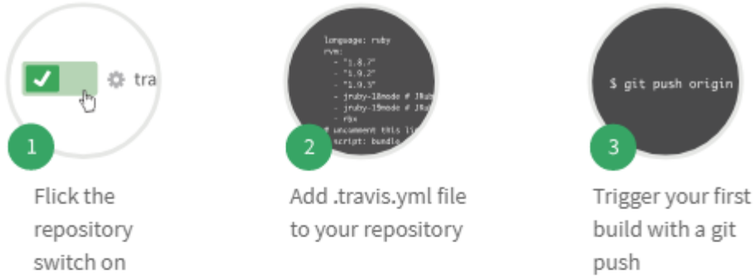
Now, instead of running our tests manually every time we want for this to be tested every time we push something from our local computer to our GitHub account.

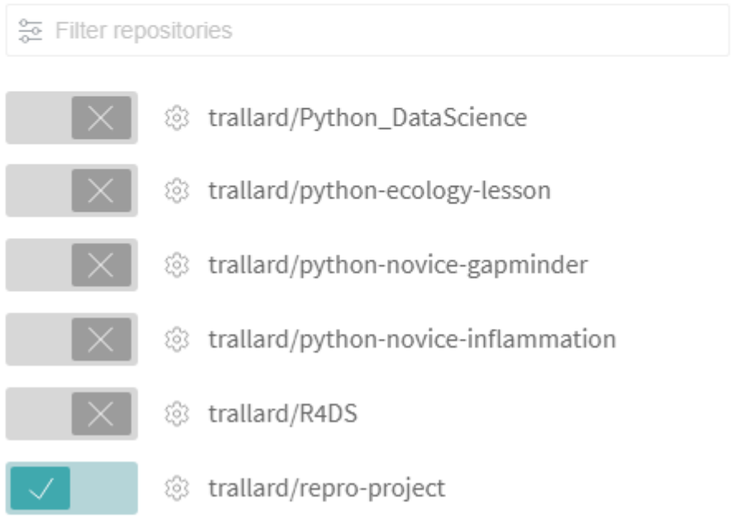
Some of the advantages of doing this are:

- check every version of your code
- check for errors continuosly
- report the results of the tests
- identify when things stop working

Activate Travis CI

Travis CI is a continuous integration server hosting platform. All you need is an account. Now let's go to https://travis-ci.org/ and activate CI for your project





travis.yml

A .travis.yml script tells Travis CI what steps are needed to test your project

```
language: python

python:
  - 3.6

before_install:
  # Here we download miniconda and create our conda environment
  - export MINICONDA=$HOME/miniconda
  - export PATH="$MINICONDA/bin:$PATH"
  - hash -r
  - wget http://repo.continuum.io/miniconda/Miniconda-latest-Linux-x86_64.sh -O miniconda.sh
  - bash miniconda.sh -b -f -p $MINICONDA
  - conda config --set always_yes yes
  - conda update conda
  - conda info -a
  # Create the environment from a yml file
  # Is this familiar?
  - conda env create -f testenv.yml -v
  - source activate testenv

script:
```

Does your code run in a colleague's computer?

Complex analysis depend on a number of packages and libraries native to your OS, packages the user installes, environmental variables and so on. Manually maintaining these dependencies is a rather tedious task.

That is why we use package managers, such as Conda. Do you remember we started this workshop by installing all the packages we needed and one of the steps was to use an environment.yml file? This ensures we all have the same packages.

Then at the beginnign of the course we activated our reproPython environment and have been using the packages installed in this environment.

> conda is only one package manager, there are many more alternatives for you

Let's create out testenv.yml

```
name: testenv
      channels:
      - conda-forge
      - defaults
      dependencies:
      - python>3.6
      - pytest
      - pandas
      - matplotlib
      - pip:
        - nbval
```

Note we are not specifying versions of the packages so by default conda will install the latest versions available

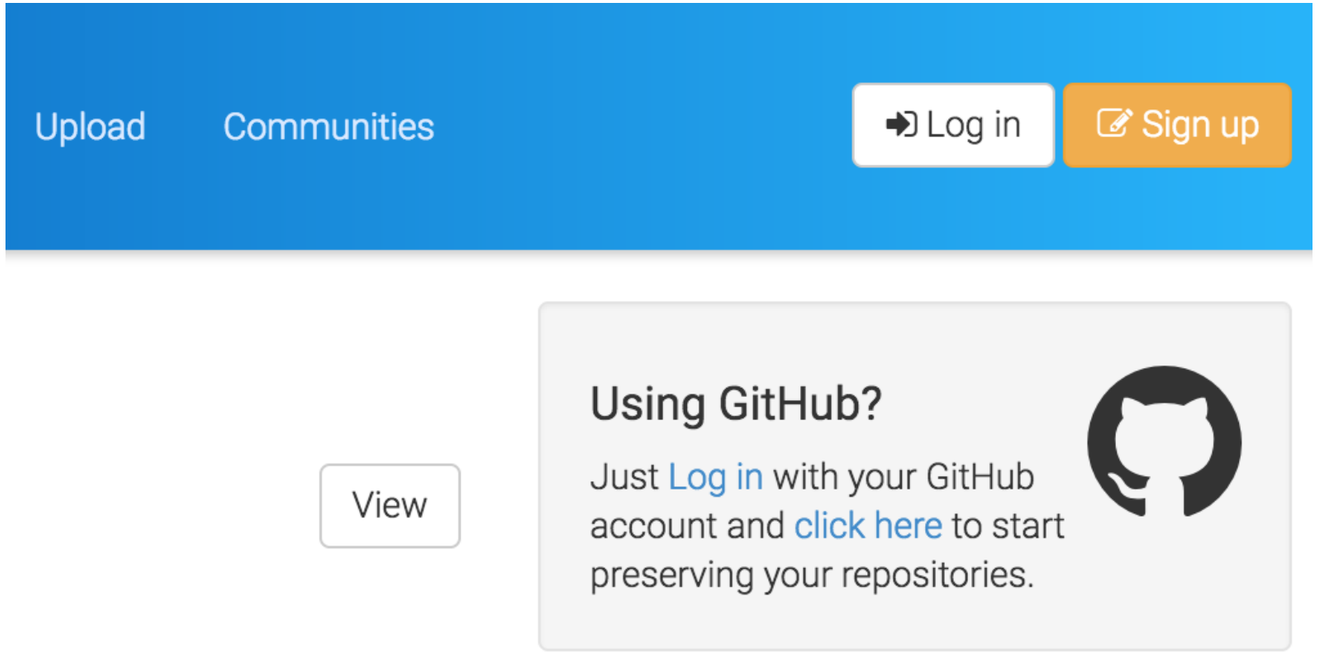
Commit your changes and push to GitHub:

```
$ git add .
$ git commit -m "Add files for CI"
$ git push
```

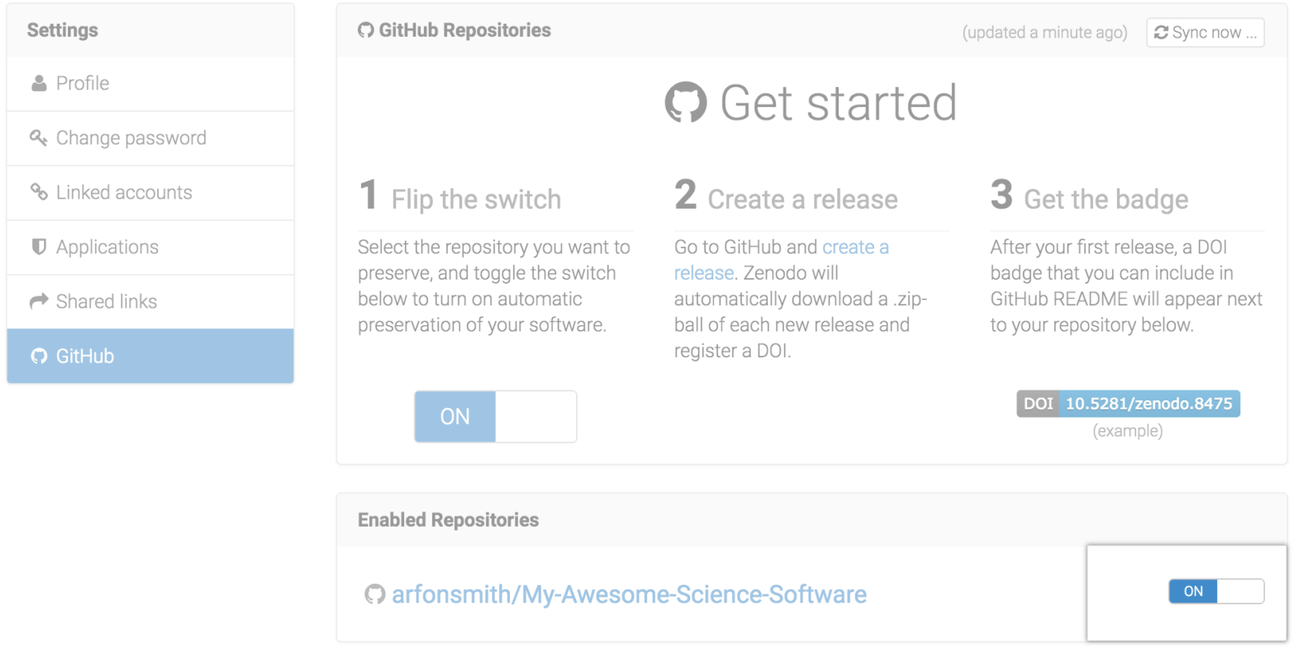
This will trigger an automatic test of your project

Making your code citable

Head to https://zenodo.org/ and login using your GitHub account



Find your repository on Zenodo and toggle Zenodo on



We will be redirected to GitHub to create a release of the repository.

After creating the release, go back to Zenodo and refresh the page. You should now see the newly created DOI 😁

Click on the DOI and copy the markdown text, add this to your README.md and push to GitHub!

Creating a citation file

```
cff-version: 1.0.3
message: If you use this software, please cite it as below.
authors:
  - family-names: Allard
    given-names: Tania
    orcid: https://orcid.org/0000-0003-4925-7248
title: My reproducibel python workflow
version: 0.1
doi: 10.5281/zenodo.1241049
date-released: 2018-05-09
```

Save as CITATION.cff and commit and push to GitHub

Additional resources

- A quick guide to software licensing
- Metadata standards
- Introduction to metadata
- DataCite Metadata Schema
- Data Reuse Checklist
- Semantic versioning info on creating tags for software
- Make your code citable
- Citation File Format
- Codemeta: metadata for your software

ixek