

Toward Reproducible Data Science

Tips and tricks of making data science projects more reproducible.

Valentina Staneva

Senior Data Scientist, eScience Institute

Reliable data science studies?



Subscribe

NEWS • 05 JUNE 2020

High-profile coronavirus retractions raise concerns about data oversight

Retracted studies had relied on health-record analyses from a company that declined to share its raw data for an audit.

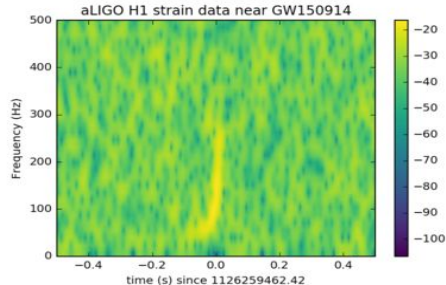
“Since we do not have the ability to verify the primary data or primary data source, I no longer have confidence in the origination and veracity of the data, nor the findings they have led to,” said Mandeep Mehra, a cardiologist at

[Many more retractions!](#)

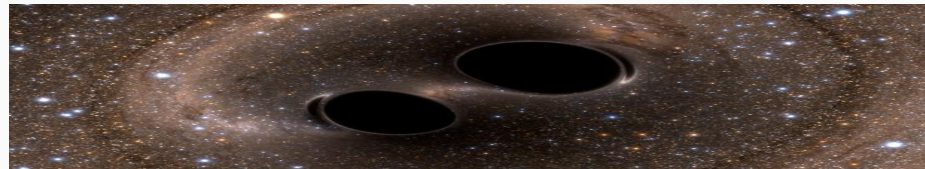
LIGO experiment

Jupyter Notebooks analyzing the data:

```
noverlap=NOVL, cma
at))
plt.xlabel('time (s) since '+str(tevent))
plt.ylabel('Frequency (Hz)')
plt.colorbar()
plt.axis([-0.5, 0.5, 0, 500])
plt.title('aLIGO L1 strain data near GW150914')
plt.savefig('GW150914_L1_spectrogram_whitened.png')
```



https://losc.ligo.org/s/events/GW150914/GW150914_tutorial.html



Is the experiment reproducible?

Quora

Ask or Search Quora

Ask Question

Detection of Gravitational Waves (February 2016)

+2



Gravitational waves discovery - is LIGO experiment reproducible or is it just a lucky timing that it caught a signal?

It took so many years for LIGO to detect the waves - is it because the instrumentation improved a lot recently? Will this enable us to detect these waves every day or will it require signals from massive black hole collisions?

general relativity - Why didn't LIGO wait for a second observation of a ...
physics.stackexchange.com/.../246611 Stack Exchange

Apr 1, 2016 - My whole life I have been taught that the very hallmark of scientific experiment are reproducible results. So why didn't LIGO wait for a second ...

Second Gravitational Wave Detected!

Reproducibility vs Replicability



Two main notions:

- Results of an experiment are regenerated using the same data and methods.
- Results of an experiment are regenerated using new data or alternative methods.

Reproducibility vs Replicability

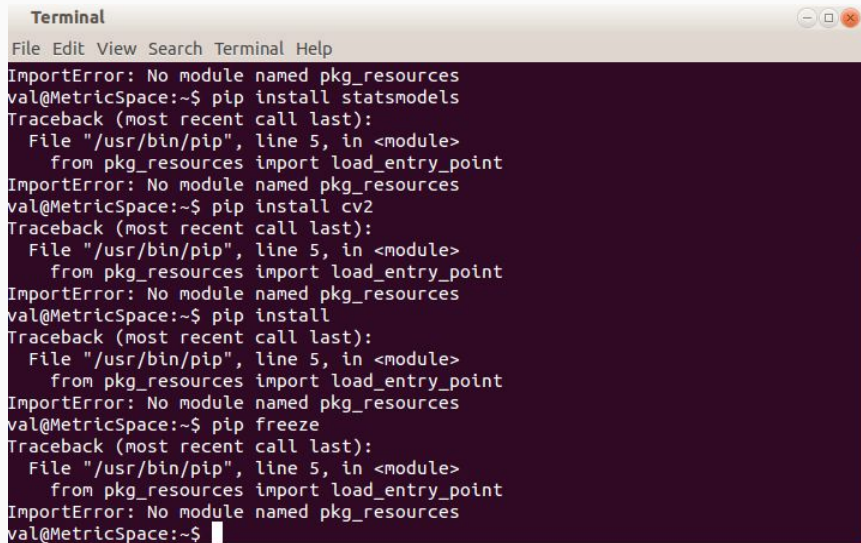


Two main notions:

- Results of an experiment are regenerated using the same data and methods.
- Results of an experiment are regenerated using new data or alternative methods.

[Reproducibility vs. Replicability: A Brief History of a Confused Terminology](#)

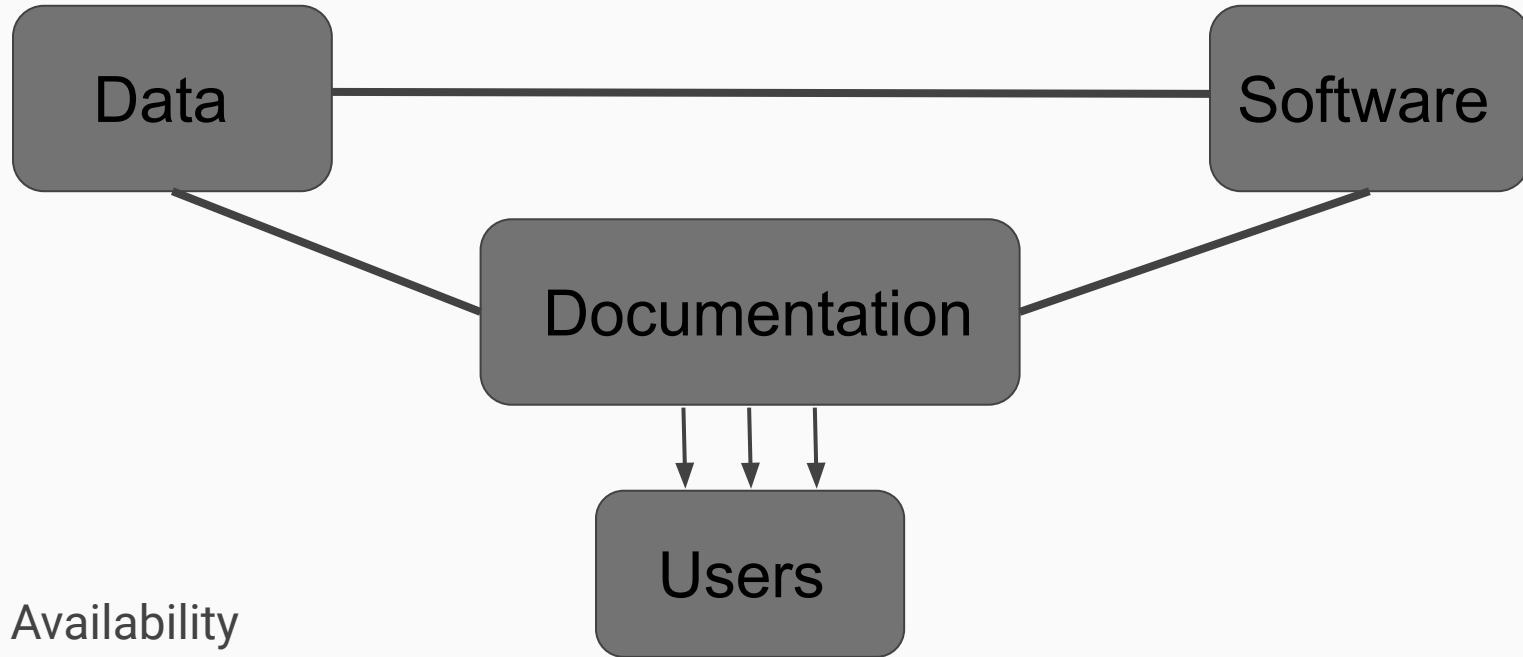
It is hard...

A terminal window titled "Terminal" with a menu bar (File, Edit, View, Search, Terminal, Help) and standard window controls. The terminal shows a series of commands and errors. The user runs 'pip install statsmodels', 'pip install cv2', and 'pip install'. Each time, it fails with an 'ImportError: No module named pkg_resources' and a traceback pointing to line 5 of '/usr/bin/pip'. Finally, the user runs 'pip freeze', which also fails with the same error. The prompt is 'val@MetricSpace:~\$'.

It is not about reproducible or not reproducible.

It is about **more reproducible**.

Improving Reproducibility



- Availability
- Automation
- Sustainability

Tips for more reproducible data science.

So far you have learnt:

- Some programming/data analysis
- Code style and documentation
- Version control and collaborative programming

What we will discuss today:

- Project Organization
- Modular Programming
- Literate Programming
- Virtualization
- Testing
- Software Licensing
- Data Sharing

Project Repository Organization

- R Project Structure: <https://nicercode.github.io/blog/2013-04-05-projects/>
- R Project Template: http://projecttemplate.net/getting_started.html
- Data Science Project Structure: [Cookiecutter](#)
- Python Module Template: [Shablona](#)

Start simple:

```
.
+-- data
|   +-- raw
|   +-- processed
|
+-- src
|   +-- PythonModules
|   +-- tests
|
+-- notebooks
|   +-- exploratory
|   +-- expository
|
+-- references
|   +-- papers
|   +-- tutorials
|
+-- results
+-- README.md
+-- LICENSE.txt
```

Expand as needed:

```
.
├── AUTHORS.md
├── LICENSE
├── README.md
├── bin
├── config
├── data
│   ├── external
│   ├── interim
│   ├── processed
│   └── raw
├── docs
├── notebooks
├── reports
│   └── figures
├── src
│   ├── data
│   ├── external
│   ├── models
│   ├── tools
│   └── visualization
```

<- Your compiled model code can be stored here (not tracked by git)
<- Configuration files, e.g., for doxygen or for your model if needed
<- Data from third party sources.
<- Intermediate data that has been transformed.
<- The final, canonical data sets for modeling.
<- The original, immutable data dump.
<- Documentation, e.g., doxygen or scientific papers (not tracked by git)
<- Ipython or R notebooks
<- For a manuscript source, e.g., LaTeX, Markdown, etc., or any project reports
<- Figures for the manuscript or reports
<- Source code for this project
<- scripts and programs to process data
<- Any external source code, e.g., pull other git projects, or external libraries
<- Source code for your own model
<- Any helper scripts go here
<- Scripts for visualisation of your results, e.g., matplotlib, ggplot2 related.

Pick and adjust for your project!

Software License Selection

Code without a license is protected by the author's copyright law.

Choose a license: <http://choosealicense.com/>



I need to work in a community.



I want it simple and permissive.

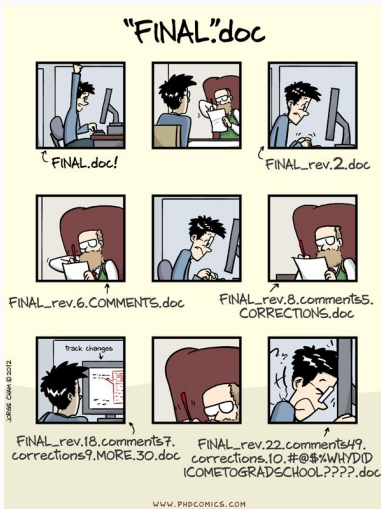


I care about sharing improvements.

- **Permissible licenses:** MIT, BDS, Apache
 - With/without attribution, with/without explicit modification explanation
- **Copyleft licenses:** GPL
 - Forces all derivatives to have the same license
 - Viral licensing: for example GPL code may be hard to integrate with MIT code.
- **Creative Commons** (good for documents or educational materials)
 - with/without attribution, with/without derivatives, with/without commercial use
 - <https://chooser-beta.creativecommons.org/>

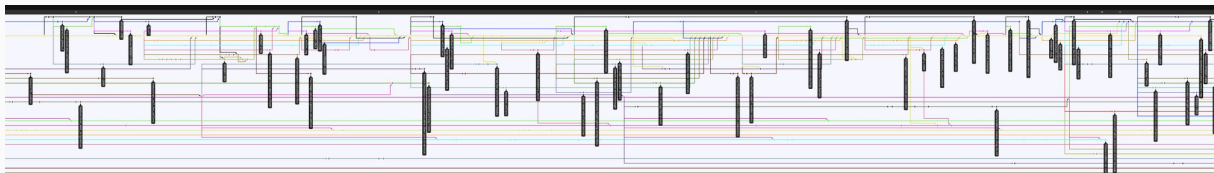
Code in a private repository can also have a license/sharing&use agreement.

Version Control Workflow



- Version control for code: git & Github
 - Software Carpentry Tutorials: <https://swcarpentry.github.io/git-novice/>
 - Atlassian Tutorials: <https://www.atlassian.com/git/tutorials/what-is-version-control>
 - Cheatsheets: <https://services.github.com/on-demand/downloads/github-git-cheat-sheet.pdf>
- Version control for data:
 - Git Large File Storage
 - Quilt: <https://quiltdata.com/>
 - Data Version Control: <https://dvc.org/> (for Machine Learning Projects)

Decide on a strategy with your team!



Documentation

Python - [Sphinx](#), [Read the Docs](#)



R - [Vignettes](#)

dplyr: A Grammar of Data Manipulation

A fast, consistent tool for working with data frame like objects, both in memory and out of memory.

Version: 0.7.4
Depends: R (≥ 3.1.2)
Imports: [assertthat](#), [bindrcpp](#) (≥ 0.2), [glue](#) (≥ 1.1.1), [magrittr](#), methods, [pkgconfig](#), [rlang](#) (≥ 0.1.2), [R6](#), [Rcpp](#) (≥ 0.12.7), [tibble](#) (≥ 1.3.1), utils
LinkingTo: [Rcpp](#) (≥ 0.12.0), [BH](#) (≥ 1.58.0-1), [bindrcpp](#), [plogr](#)
Suggests: [bit64](#), [covr](#), [dbplyr](#), [dplyr](#), [DBI](#), [ggplot2](#), [hms](#), [knitr](#), [Lahman](#) (≥ 3.0-1), [mgcv](#), [microbenchmark](#), [nycflights13](#), [rmarkdown](#), [RMySQL](#), [RPostgreSQL](#), [RSQLite](#), [testthat](#), [withr](#)
Published: 2017-09-28
Author: Hadley Wickham [aut, cre], Romain Francois [aut], Lionel Henry [aut], Kirill Müller [aut], RStudio [cph, fnd]
Maintainer: Hadley Wickham <hadley at rstudio.com>
BugReports: <https://github.com/tidyverse/dplyr/issues>
License: MIT + file LICENSE
URL: <http://dplyr.tidyverse.org>, <https://github.com/tidyverse/dplyr>
NeedsCompilation: yes
Materials: [README NEWS](#)
In views: [ModelDeployment](#)
CRAN checks: [dplyr results](#)

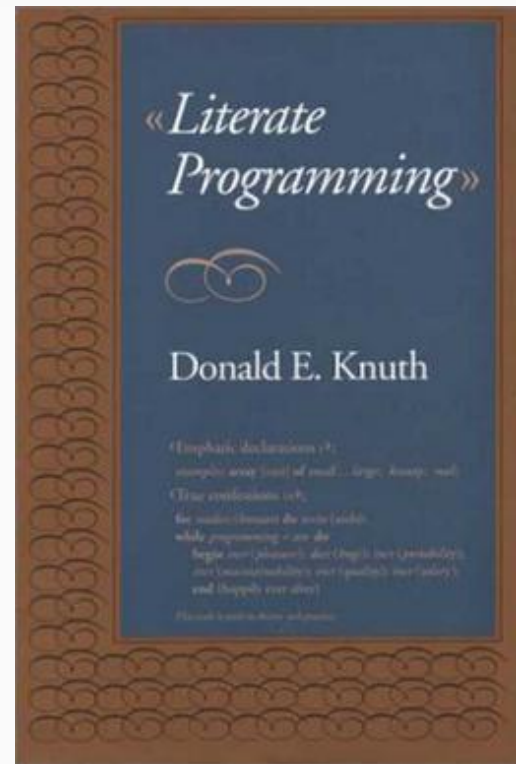
- [Journal of Open Source Software](#)
- [Journal of Statistical Software](#)

Literate Programming

Combining documentation and code in a single program.

“Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do.”

R Reporting and sharing: [Knitr](#), [R Pubs](#)
Notebooks - [Jupyter](#), [R Notebooks](#), [Zeppelin](#), [CoCalc](#)
Notebook Environments: [Binder](#), [Colaboratory](#), [Kaggle](#),
[VSCode Notebooks](#), [AWS Sagemaker Notebooks](#)



[Image by Wikipedia](#)

Free Notebook Environments

Colaboratory Notebooks (Google)

- 13GB RAM
- 38 GB disk
- GPU support
- Notebooks and data on Google Drive
- Integration with Github
- Simultaneous Editing
- Python only so far

Cons:

- not real filesystem
- Github integration funky
- Can get blocklisted from GPU support

Kaggle Kernels (Google)

- 16GB RAM
- 5GB disk
- GPU support
- Upload/Edit/Download Notebooks
- Kaggle Datasets: public and private(20GB)
- Version Control Support
- R and Python

Cons: no Github integration

Binder

- 2GB RAM
- Works with Github Repos
- Python, R, Julia
- Good for demos

Cons: Ephemeral workspace

Non-free: Colab Pro, VS Code & Azure, [AWS Sagemaker](#) for ML

Label

Build

Train & Tune

Deploy & Manage

Combining Notebooks

- Binder (mybinder.org)
 - Binds and demos notebooks on a github repo
 - [xarray](#) example
 - Your [notebook example](#)
- Jupyterbook (<https://jupyterbook.org/intro.html>)
 - Combines notebooks into a static website
- Papermill (<https://papermill.readthedocs.io/en/latest/>)
 - Executes notebooks, parameterizes notebooks, generates reports

launch binder

jupyter {book}



papermill

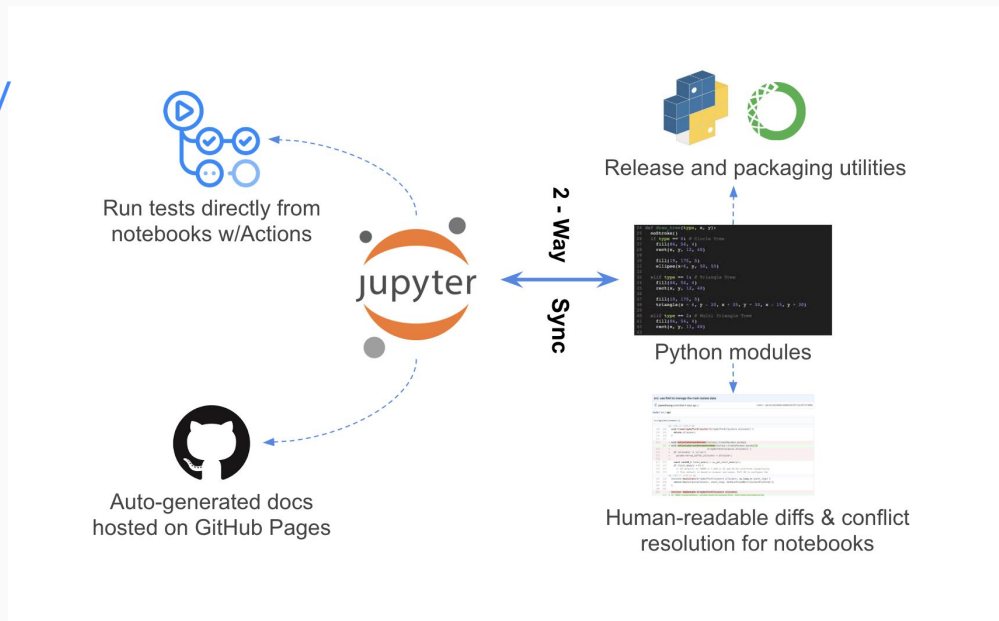
From Notebooks to Modules

Modular Programming:

- convert commands -> functions -> modules/libraries
- convert notebooks/scripts -> modules/libraries

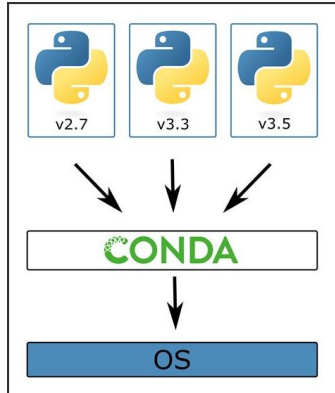
[nbdev](https://nbdev.fast.ai/)

<https://nbdev.fast.ai/>



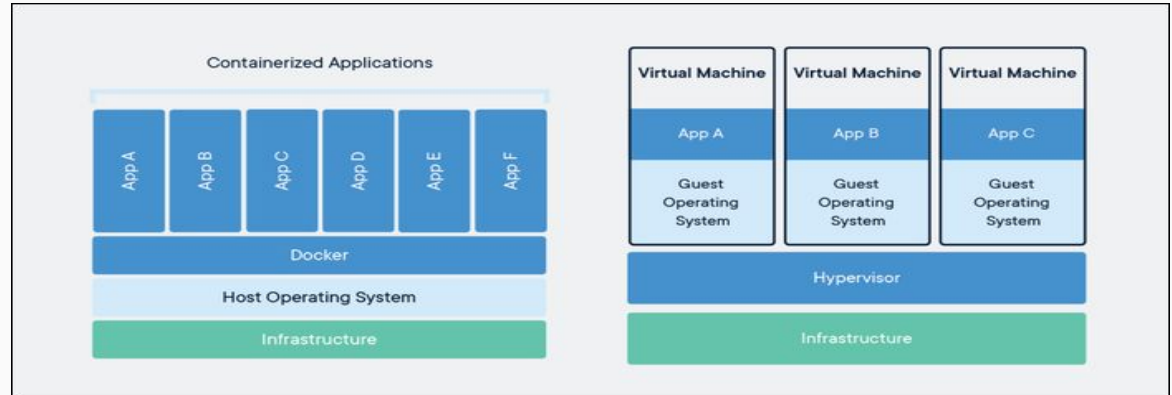
Virtualization

Virtual Environment (conda, pyenv, renv)

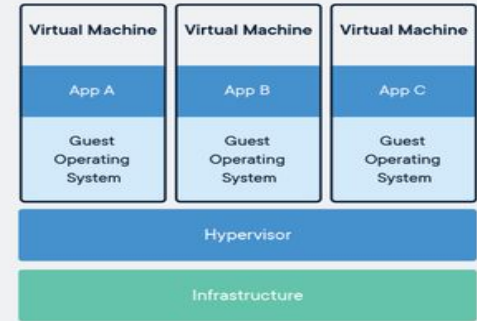


[Image Source](#)

Virtual Container (Docker)



Virtual Machine (Virtual Box, Vmware)



[Image Source](#)

Lighter, less isolated

heavier, more isolated

Virtualization - Virtual Environments

- Virtual Environments - handle package and distribution dependencies
 - [Conda](#) supports both for Python and R
 - Make your virtual environment now!
 - Store your dependencies in a `requirements.txt` file (or `.yml` file)
 - Document each installation while doing it not later!

Python

```
> conda create -n py38 python=3.8 jupyter numpy  
> conda activate py38  
> jupyter notebook  
#Do something, install extra packages  
> conda deactivate
```

R

```
> conda create -n r_env r-base r-essentials rstudio  
> conda activate r_env  
> rstudio  
#Do something, install extra packages  
> conda deactivate
```

Setting up different envs showing up in jupyter: [tutorial](#)

Setting up rstudio with anaconda: <https://docs.anaconda.com/anaconda/navigator/tutorials/create-r-environment/>

Virtualization - Virtual Environments

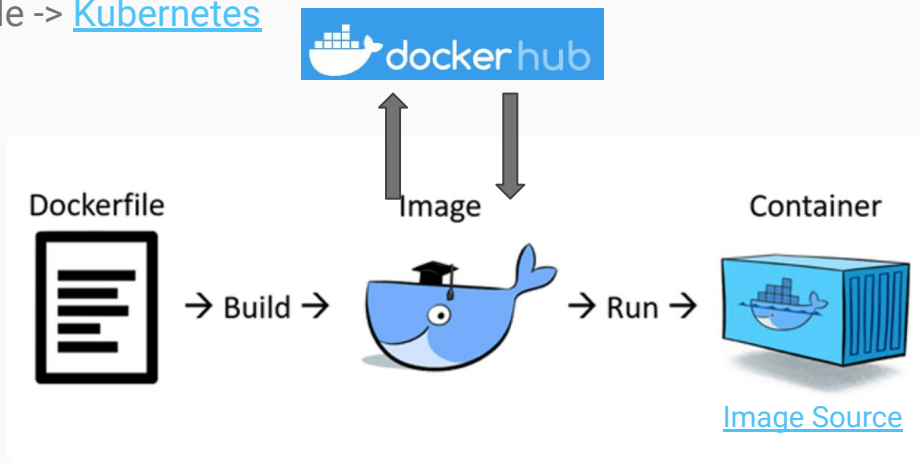
Virtual Environments - handle package and distribution dependencies

Deeper under the hood:

- Conda has its own version of language distributions
- Each library is a .whl wheel file which is precompiled for your OS
- Many packages included in anaconda, extra packages in community channels
 - <https://conda-forge.org/>
 - Contribute your own!
- Conda vs pip
 - [Pipy](#) is another package manager
 - Avoid mixing them if possible
- Other options:
 - [pyenv](#)
 - [renv](#)

Virtualization - Virtual Containers

- Docker Containers - Linux environment, works on all OS
 - Dockerfile: scriptable setup ([complex example](#), [simple example](#))
 - DockerHub: ready-to-go images
 - E.g. postgres database
 - E.g. [rocker](#) images
 - Resolve installation mess
 - Deploy, Scale -> [Kubernetes](#)




<https://carpentries-incubator.github.io/docker-introduction/>

Virtualization - Other

- [Vagrant](#) - virtual machine manager, can run both Docker containers and full VMs
- Virtual Machines - [VirtualBox](#), [VMWare](#)
- On Windows: [Subsystem for Linux](#)
- Cloud Images - AWS AMIs
- Cloud Container Services
- [Terraform](#): abstract the cloud provider (infrastructure as code)
-

Testing



The screenshot shows the top of the scikit-learn GitHub repository. It features a row of build status badges: 'build passing' (green), 'build pending' (grey), 'codecov 95%' (green), 'circleci passing' (green), 'python 2.7' (blue), 'python 3.5' (blue), and 'pypi package 0.19.1' (green). Below these is a DOI badge: 'DOI 10.5281/zenodo.1034765'. The repository name 'scikit-learn' is displayed in a large, bold font.

We are already writing tests, need to save them.

Types of testing: unit, integration, system, regression

- Locally
 - Python - nose, pytest, tox
 - R - testthat
- Remotely - Continuous Integration
 - Github Actions, CircleCI, AppVeyor

Start by testing the environment.

Digital Object Identifiers

- Articles :
 - Arxiv, [preprint server](#)
 - journals create it for you
 - read instructions about journal access
- Slides, Posters:
 - [F1000 Research](#)
- Software:
 - For any github repository using [Zenodo](#)
 - Register releases
- Datasets:
 - Persistent repositories provide DOI

README.md

build
















passing

DOI

10.5281/zenodo.3906891

Experiment Tracking

Comparison of ML Experiment Tracking Tools

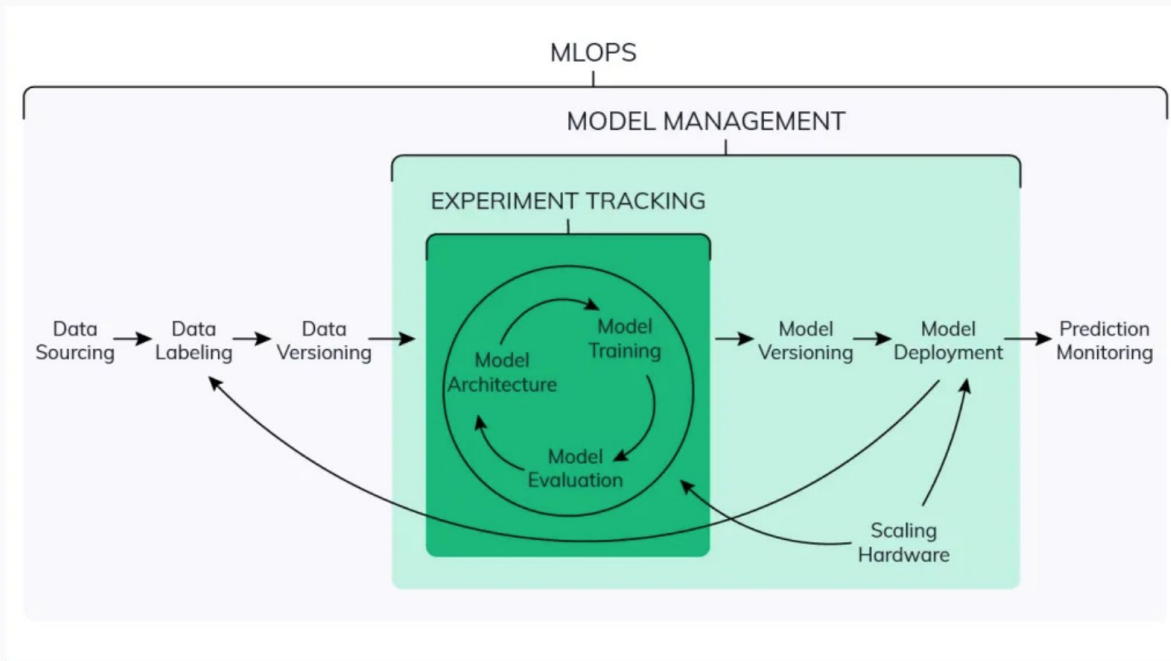
	 MLflow	 Tensorboard	 DVC	 ClearML	 Guild.ai	 Kubeflow	 Neptune.ai	 Weights & Biases	 Comet.ml	 SageMaker Experiments	 DAGsHub
Open source	✓ Apache	✓ Apache	✓ Apache	✓ SSPL	✓ Apache	✓ Apache	✗	✗	✗	✗	— Open-source formats
Platform & language agnostic	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓
Experiment data access <small>Local  / Cloud or API </small>	 + 				 + 						 + 
Easy to set up	✓	✓	✓	— Open-source server is hard to set up	✓	✗	✓	✓	✓	✓	✓
Custom visualizations	✓	✓	— Difficult to customize	✓	✓	✗	✓	✓	✓	✓	✗
Scalable for large number of experiments	✓	✗	??	✓	??	✓	✓	✓	✓	✓	✓

To read more, go to <https://DAGsHub.com/blog>



- Explosion of ML experiment tracking tools



Experiment Tracking



Source: neptune.ai

- Explosion of ML experiment tracking tools
- Some tools can be used for general experiment tracking (non ML analysis)

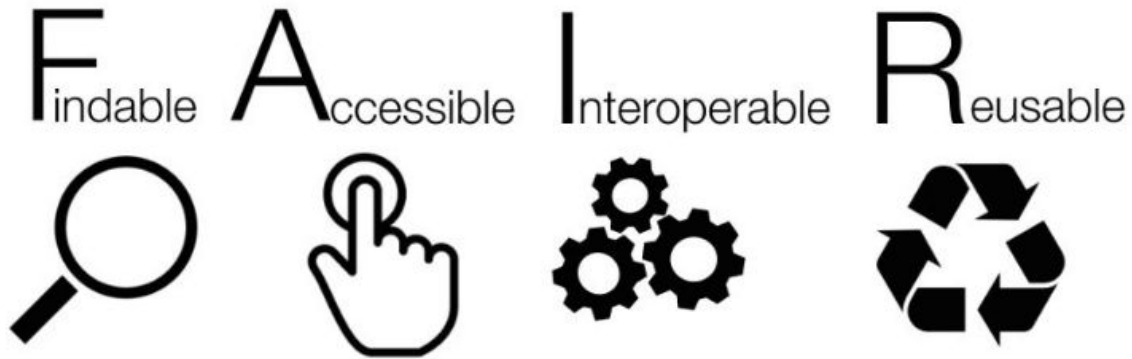
Data Repositories

		 figshare	
Up to 50GB free Not-for-profit - EU funded (contact if more)	Publishing Fee - \$120 Excess fees after 50GB Associated with articles Not-for-profit	20GB free per manuscript Institutional plans For-profit	Up to 2TB Subscription Based Free Promo Codes

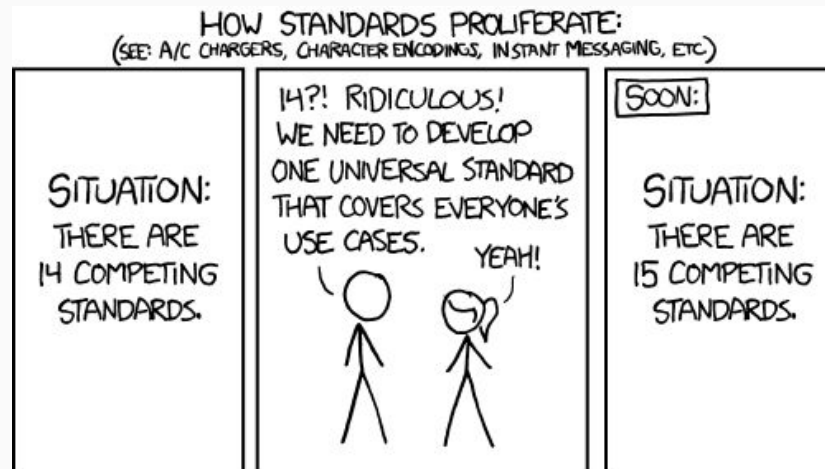
- Datasets receive Digital Object Identifier (DOI)
- Cloud Storage: free to upload, fees for storage, higher fees to download
 - Some public datasets can be stored for free:
<https://registry.opendata.aws/>
- Nature Journal Scientific Data: <https://www.nature.com/sdata/>

Data and Metadata

['FAIR Guiding Principles for scientific data management and stewardship', Wilkinson et.al, Nature Scientific Data, 2016](#)



Standardization



- Try using standard formats whenever possible!
- If format does not fit your case discuss with the community! ISO, NIST
- Often standards are more permissive than data formats!
- Interoperability
- Persistency

Standardization: examples

Example 1: you have pulled some election data and you want to organize it so that it is easy for other researchers to analyse it

- Store it in excel sheets
- Store it in csv files
- Check how the data for other states is stored and store it the same format
- Check out if there are standards for election data
 - <https://www.nist.gov/publications/election-results-common-data-format-specification-revision-20>

Example 2: you want to save a Deep Learning model so you can apply to future data:

- Python pickle file (Python specific, sometimes version dependent)
- HDF format (Python and domain independent, local database: but fields are not standardized)
- Tensorflow HDF (libraries come and go)
- [ONNX](#) (Open Neural Network Exchange) format (library/language independent, stores the 'math', i.e. the computational graph operations, hardware)

Beyond exact reproducibility

- Design your study
- Register Hypothesis
- Create a Baseline
- Perform Nested Analyses
- Test simple scenarios first
- Test different methods
- Understand your errors:
 - Precision Recall Curves:
 - Baseline changes for different class distributions
 - Cross-validation with dependence in time series and groups
 - https://scikit-learn.org/stable/modules/cross_validation.html
 - [Multiple Testing Problem](#)
 - The more hypotheses we are testing, the more likely one of them will be falsely true

What about your projects?

Reproducibility Checklist:

Assessing Work Reproducibility

Data

- Are the data publically available? If not all, can a summary of them be made publically available?
- Are they in a format easily accessible by open source software libraries?
- Do they have a license that permits broad use?
- Are they permanent, or do they have versions?
- For how long can they be stored at their current location?

Software

- Is your software publicly available?
- Is your software under version control?
- Can your software run on different operating systems?
- Is it easy to install all the dependencies for your software?
- If not can you provide the users with a pre-built environment?
- Does your software have a license?
- Does your software use other softwares: are their licenses compatible with yours?
- Do you have a way to test whether adding new code features or library updates preserve the software's functionality?

Documentation & Results

- Do you provide instructions on how to install the software? Are the versions of the dependencies provided?
- Do you provide examples how to use the software?
- Can a user run the examples?
- Do you describe how the data was collected?
- Do you have a document providing information for obtaining both the software and the data to generate the results?
- If so, does that document have associated copyright?
- Is it going to be available in 1 year?
- Can a user regenerate the results? If not all of them, maybe a subset?
- Is the procedure for generating all of the results automated?
- Are the results stochastic? Is it indicated somewhere?
- Are some of the steps requiring manual input? Is there a description of how it was done?

Summarize:

- What are the major challenges of making your entire work reproducible?
- What tools/approaches have you already used to make some of your work more reproducible?
- What simple steps can you make to improve the reproducibility of your work?

Built upon:

Assessing Reproducibility

<https://tinyurl.com/2022ReproducibleScience>