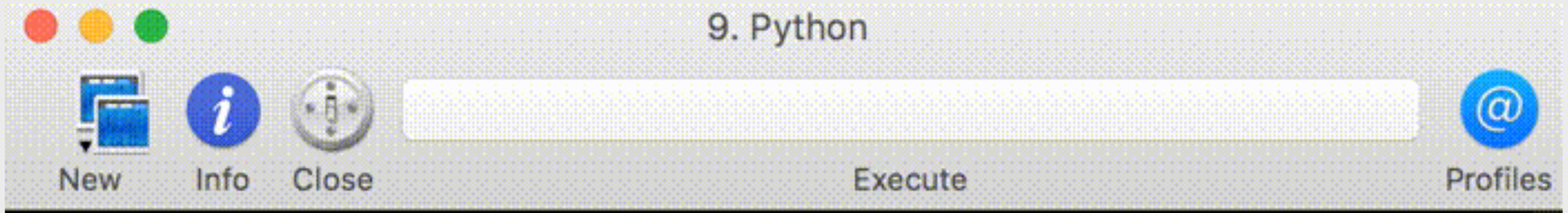
Reproducibility and Extensibility in Scientific Research

Jessica Forde
Project Jupyter
@projectjupyter
@mybinderteam

Overview

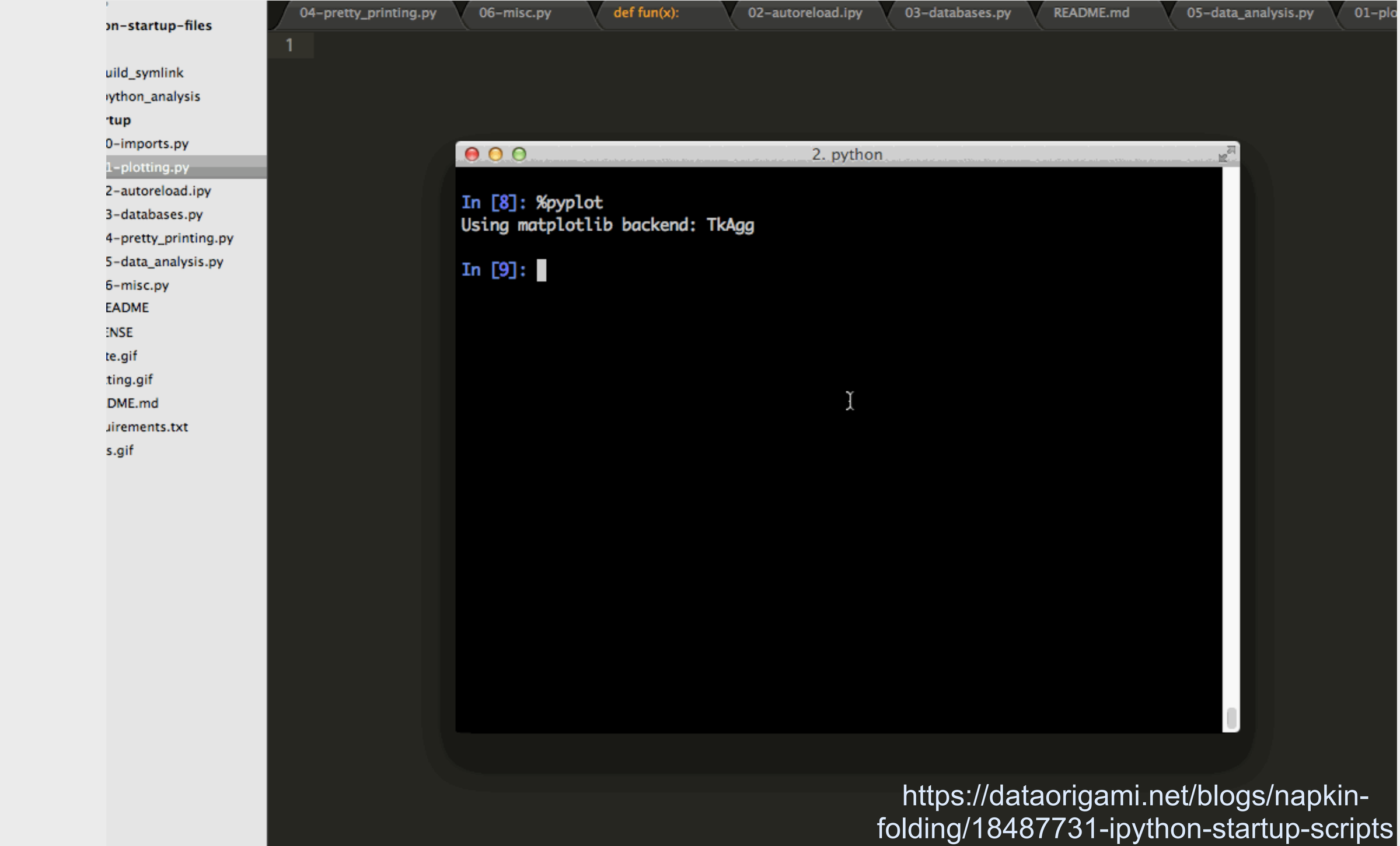
- Project Jupyter
- IPython
- Jupyter Notebook
- Architecture of JupyterHub
- The problem of reproducibility in science
- Repo2docker
- Binder
- Extending research via interactive computing

Who are we?




```
-bash-3.2$ ipython
WARNING: Attempting to work in a virtualenv. If you encounter problems, please
nstall IPython inside the virtualenv.
Python 2.7.10 (default, Sep 23 2015, 04:34:21)
Type "copyright", "credits" or "license" for more information.

IPython 4.0.2 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: from postal.expand import expand_address

In [2]:
```

https://machinelearnings.co/statistical-nlp-on-openstreetmap-b9d573e6cc86


on-startup-files
uild_symlink
ython_analysis
tup
0-imports.py
1-plotting.py
2-autoreload.ipy
3-databases.py
4-pretty_printing.py
5-data_analysis.py
6-misc.py
EADME
ENSE
te.gif
ting.gif
DME.md
uirements.txt
s.gif
04-pretty_printing.py
06-misc.py
def fun(x):
02-autoreload.ipy
03-databases.py
README.md
05-data_analysis.py
01-plo
1
2. python
In [8]: %pyplot
Using matplotlib backend: TkAgg
In [9]:
https://dataorigami.net/blogs/napkin-folding/18487731-ipython-startup-scripts

~2.1 Million Notebooks on GitHub

Repositories | Developers

Trending: **this month**

fivethirtyeight / **data**

★ Star

Data and code behind the articles and graphics at FiveThirtyEight

Jupyter Notebook ★ 9,553 3,929 Built by  ★ 829 stars this month

fastai / **fastai**

★ Star

The fast.ai deep learning library, lessons, and tutorials

Jupyter Notebook ★ 4,236 1,307 Built by  ★ 474 stars this month

tensorflow / **probability**

★ Star

Probabilistic reasoning and statistical analysis in TensorFlow

Jupyter Notebook ★ 500 62 Built by  ★ 326 stars this month

All languages

Unknown languages

C++

JavaScript

Jupyter Notebook

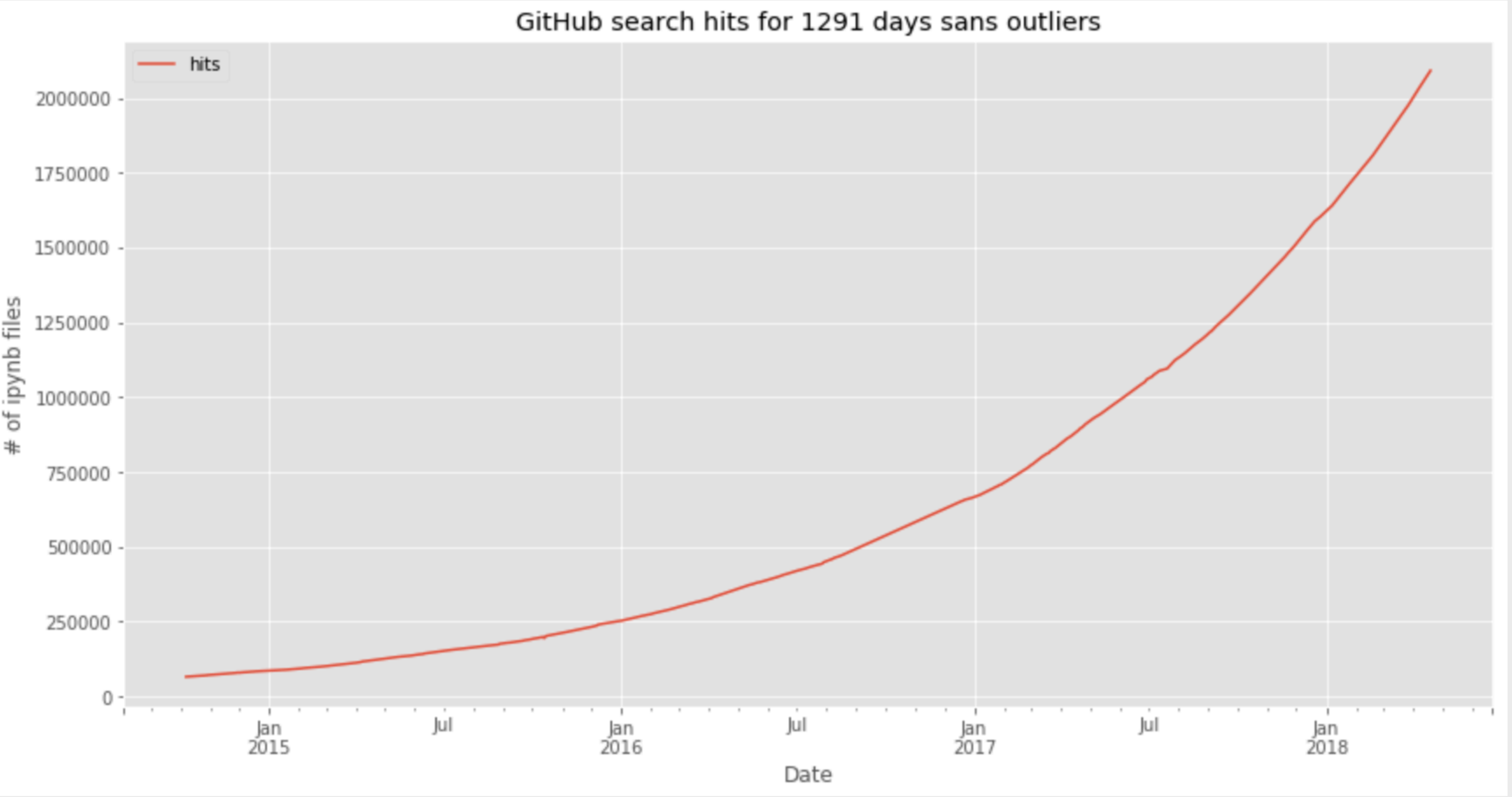
Matlab

Python

Other: **Languages**

ProTip! Looking for most forked Jupyter Notebook repositories? Try this search

https://github.com/trending/jupyter-notebook?since=monthly



https://mybinder.org/v2/gh/parente/nbestimate/master?filepath=estimate.src.ipynb

Project Jupyter exists to develop open-source software, open-standards, and services for interactive computing across dozens of programming languages.

Project Jupyter Mission

Project Jupyter
Project Jupyter exists to develop open-source software, open standards, and services for interactive and reproducible computing.
Jul 7, 2015 · 3 min read

GORDON AND BETTY MOORE FOUNDATION

ALFRED P. SLOAN FOUNDATION 1934

THE LEONA M. AND HARRY B. HELMSLEY CHARITABLE TRUST

New funding for Jupyter

We are pleased to announce that the Jupyter/IPython project has received $6M in funding from three organisations:

- The Leona M. and Harry B. Helmsley Charitable Trust
- The Gordon and Betty Moore Foundation
- The Alfred P. Sloan Foundation

The grant, which is being made both to the University of California, Berkeley and California Polytechnic State University, San Luis Obispo, will support the project for three years and includes new collaborations with the University of Southampton, and the Simula Research Lab in Norway.

Mission and Background

Project Jupyter's mission is to create open source tools for interactive scientific computing and data science in research, education and industry, with an emphasis on usability, collaboration and reproducibility.

This project is structured in a "3+1" format, with three main focus areas of research and development and one extra topic of ongoing work. The three focus areas are *Interactive Computing*, *Computational Narratives* and *Collaboration*. The problem of *Sustainability* will require ongoing attention but is conceptually distinct from the first three, as it doesn't focus on specific research questions or deliverables.

Newsjournal of the Society for Industrial and Applied Mathematics sinews.siam.org
siam news
Volume 51/ Issue 2
March 2018

Jupyter: Tools for the Life Cycle of a Computational Idea

By Min Ragan-Kelley, Carol Willing, and Jason Grout

Computation is increasingly becoming an integral part of science and education across disciplines. The life cycle of a computational idea typically involves interactive exploration and experimentation, as well as publication and communication of results. Reproducible computation demands open research tools, good software practices, and transparent documentation of research processes and results. Project Jupyter[1] is an open community that builds open-source software tools and protocols for the life cycle of a computational idea. Two core pieces of the project are an open protocol for interactive computation and an open document format with which to record and share computational ideas. The Jupyter Notebook application builds on these to provide a powerful, interactive, computational environment.

The Jupyter Notebook

What Is a Notebook? The Jupyter

The notebook document format is free, transparent, and understandable, in keeping with its aim to facilitate open and accessible science. It is stored as a single JSON-formatted text file, making it easy to manipulate and understand using standard programming tools, without the need for Jupyter software. The notebook file format is public,[2] and Jupyter software is open-source under the BSD license.

Many authors communicate using Jupyter Notebook. GitHub hosts 1.4 million notebooks, and some people have written entire books as collections of notebooks, such as Jake Vanderplas's *Python Data Science Handbook*.[3] Because notebook documents preserve their content structure and metadata, they are easily convertible to other formats, including plain scripts in the document's language of choice. This also makes them easy to integrate into publication pipelines via formats such as LaTeX, Markdown, and reStructuredText via Jupyter's conversion tool, nbconvert.[4]

Using Notebook Documents. The Jupyter Notebook server is a web-based application for interacting with notebook

cuting code cells in the same persistent kernel session, thus building and exploring a computational idea. The user can also create and interleave markdown cells to explain an idea using formatted text and mathematics (see Figure 1).

Users typically install the notebook application on their local computer, where it functions like a desktop application and works with local notebooks and data. Since it is a web application, the notebook server can also run on a remote computer and provide the notebook application via a browser, with no

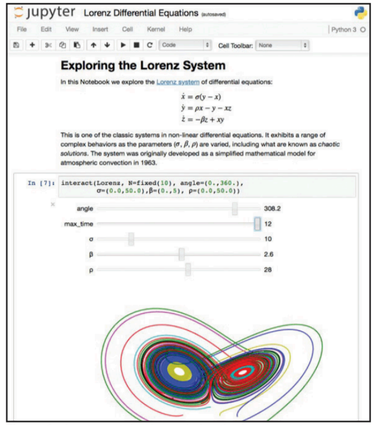


Figure 1. *Jupyter notebooks exploring the Lorenz system.*

output representations between front ends

We’re not-for-profit

README.md

Project Jupyter Governance

The purpose of this repository is to formalize the governance process that the IPython/Jupyter project has used informally since its inception in 2001. This document clarifies how decisions are made and how the various elements of our community interact, including the relationship between open source collaborative development and work that may be funded by for-profit or non-profit entities.

Table of Contents

License of Governance Documents

https://github.com/jupyter/governance

Jupyter Notebook

Narrative Text

Notebook title and introduction

Description of model parameters

Description of need to profile data

Code and Visualizations

Importing external packages

Implementation of parameters

Profile plotting code

Inline plot

Sampling from the generative model

In this notebook, we will use the generative model of the HDHP (Hierarchical Dirichlet-Hawkes Process) in order to sample events. We will start with a predifined number of users, say `10`, and we will attempt to model their behavior as they are posting questions in an online platform. For simplicity, our "vocabulary" will be dummy.

We start by importing all the libraries that will be required.

In [1]:
```
%matplotlib inline
import datetime
import string
import hdhp
import notebook_helpers
import seaborn as sns
```

Now, let us set some parameters for our model. These fall under two categories; the ones relevant to the content and then ones relevant to the time dynamics. Starting with the first set, we need to decide on:

- the vocabulary: a dummy set of `100` words, i.e. `word0`, `word1`, ... , `word99`.
- the minimum and maximum length of a question
- the number of words of each pattern

As far as the time dynamics is concerned, we need to set:

- α_0: the parameters of the Gamma prior for the time kernel of each pattern
- μ_0: the parameters of the Gamma prior for the user activity rate
- ω: the time decay parameter

Finally, in order to make the generative process more user-friendly, we can pre-set the number of patterns that our users can sample from.

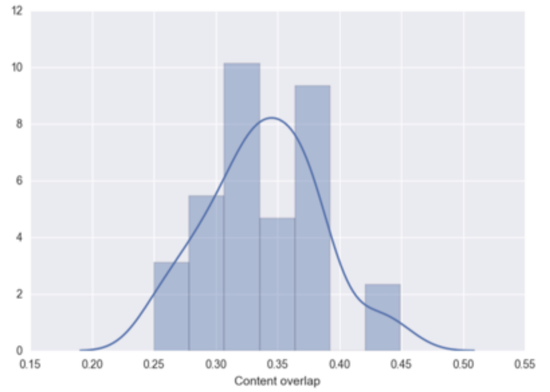
In [2]:
```
vocabulary = ['word' + str(i) for i in range(100)]  # the `words` of our documents
doc_min_length = 5
doc_length = 10
words_per_pattern = 50

alpha_0 = (2.5, 0.75)
mu_0 = (2, 0.5)
omega = 3.5

num_patterns = 10

process = hdhp.HDHProcess(num_patterns=num_patterns, alpha_0=alpha_0,
                          mu_0=mu_0, vocabulary=vocabulary,
                          omega=omega, words_per_pattern=words_per_pattern,
                          random_state=12)
```

Before generating any questions, we can take a look at the patterns that we initialized our process with, and look at the content distribution of each pattern. Although each pattern has a different word distribution, we can still plot the overlap (Jaccard similarity) between the words that have non-zero probability for each pattern. Since we used a limited number of patterns, the distribution of the overlap will not be smooth.

In [3]:
```
overlap = notebook_helpers.compute_pattern_overlap(process)
sns.distplot(overlap, kde=True, norm_hist=True, axlabel='Content overlap')
```

```
Average overlap: 0.338826769742
```

Out[3]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x10de8ca50>
```



Rule et al., 2018

JUPYTER NOTEBOOK

Browser

View and Enter Stuff Part

HTTP(S) Websocket

Notebook Server

ZeroMQ

Julia

R

Python

Language Expert Thing

Kernels

Carol Willing

More than Python

Interleave Python with C++: the %%cpp magic ¶

Thanks to ROOT, it is possibile to write cells in C++ within a Python notebook. This can be done using the %%cpp magic. Magics are a feature of Jupyter notebooks and when importing the ROOT module, the %%cpp magic was registered.

In [10]:
```
%%cpp
cout << "This is a C++ cell" << endl;
```

```
This is a C++ cell
```

Not bad. On the other hand, ROOT offers much more than this. Thanks to its interpreter and type system, entities such as functions, classes and variables, created in a C++ cell, can be accessed from within Python (and viceversa, partially).

In [11]:
```
%%cpp
class A{
    public:
    A(){cout << "Constructor of A!" << endl;}
};
```

In [12]:
```
a = ROOT.A()
```

```
Constructor of A!
```

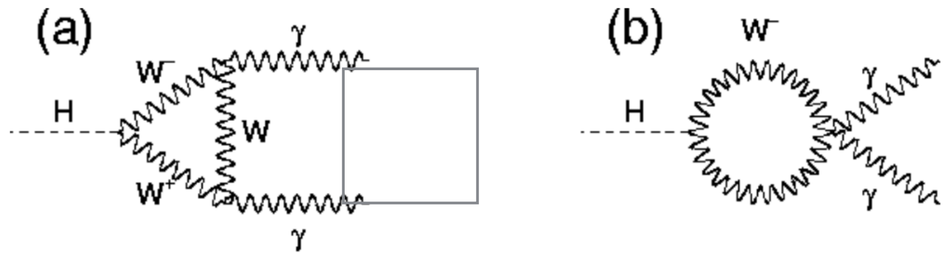
The Python and C++ worlds are so entangled that we can find back in C++ the entities created in Python. To illustrate this, from within a C++ cell, we are going to fit a function in the gauss histogram displayed above and then re-draw the canvas.

In [13]:
```
%%cpp
gauss->Fit("gaus", "S");
myCanvasName->Draw();
```

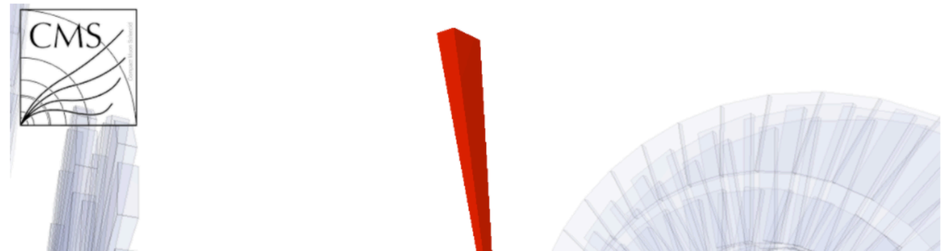
```
 FCN=47.4997 FROM MIGRAD    STATUS=CONVERGED      53 CALLS          54 TOTAL
                     EDM=8.44224e-09    STRATEGY= 1      ERROR MATRIX ACCURATE
```

Higgs decay to two photons

The Standard Model predicted the decay of the Higgs bosons into photons. The process is depicted by the diagrams below:

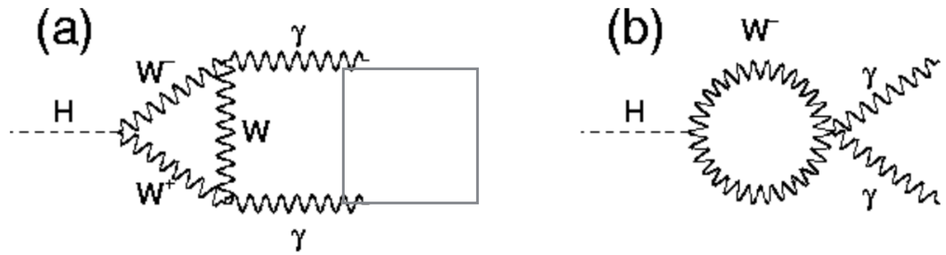


At the Large Hadron Collider, this process has been measured. This figure shows how an Higgs boson decay looks in the CMS detector:

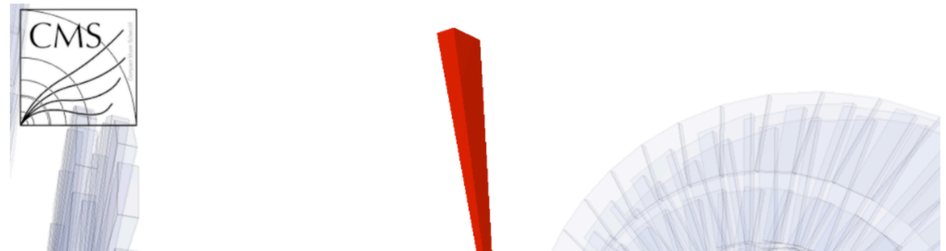


Higgs decay to two photons

The Standard Model predicted the decay of the Higgs bosons into photons. The process is depicted by the diagrams below:



At the Large Hadron Collider, this process has been measured. This figure shows how an Higgs boson decay looks in the CMS detector:



In [ ]:
```
import (
    "fmt"
    "time"
)
```

In [ ]:
```
// sum calculates the sum of two integers
func sum(x, y int) int {
    return x + y
}
```

In [ ]:
```
a, b := 3, 4
```

In [ ]:
```
fmt.sp("sum(%d, %d) = %d", a, b, sum(a, b))
```

In [ ]:
```
start := time.Now()
defer func() {
    end := time.Now()
    fmt.Println("Interrupted an infinite loop after",
                end.Sub(start))
}()
for {}
```

Jupyter kernels

Kernel Zero is IPython, which you can get through ipykernel, and is still a dependency of jupyter. The IPython kernel can be thought of as a reference implementation, as CPython is for Python.

Here is a list of available kernels. If you are writing your own kernel, feel free to add it to the table!

Name	Jupyter/IPython Version	Language(s) Version	3rd party dependencies	Exar Notel
Coarray-Fortran	Jupyter 4.0	Fortran 2008/2015	GFortran >= 7.1, OpenCoarrays, MPICH >= 3.2	Demo Binde demo
sparkmagic	Jupyter >=4.0	Pyspark (Python 2 & 3), Spark (Scala), SparkR (R)	Livy	Noteb Docke Image

https://github.com/jupyter/jupyter/wiki/Jupyter-kernels

Creating new Jupyter kernels

Making kernels for Jupyter in the documentation.

Simple example kernel

IHaskell creator blog post

Testing kernels against message specification (work in progress)

Tool to test a kernel against specification (work in progress)

https://github.com/jupyter/jupyter/wiki/Jupyter-kernels

More than Notebooks

jupyterhub

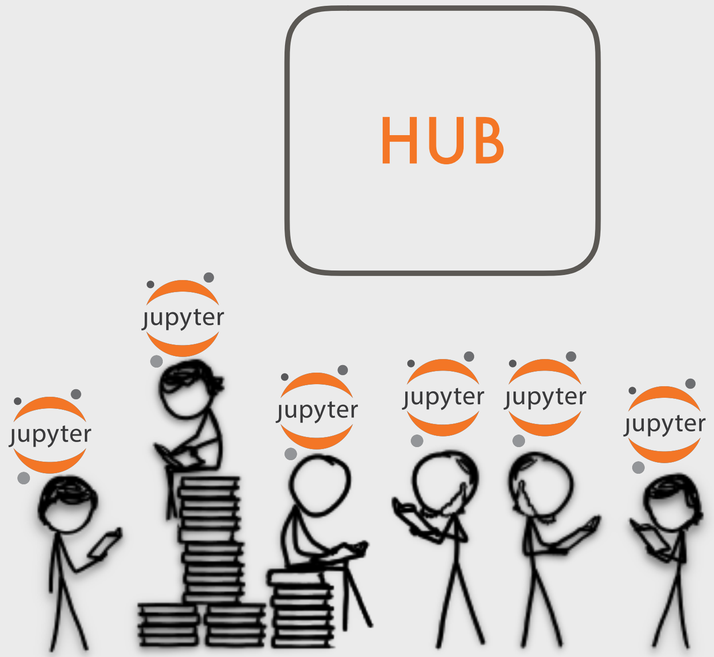


photo: xkcd, Jupyter, C. Willing

- JupyterHub provides a single user Jupyter Notebook server for each person in a group
- Similar to SWAN

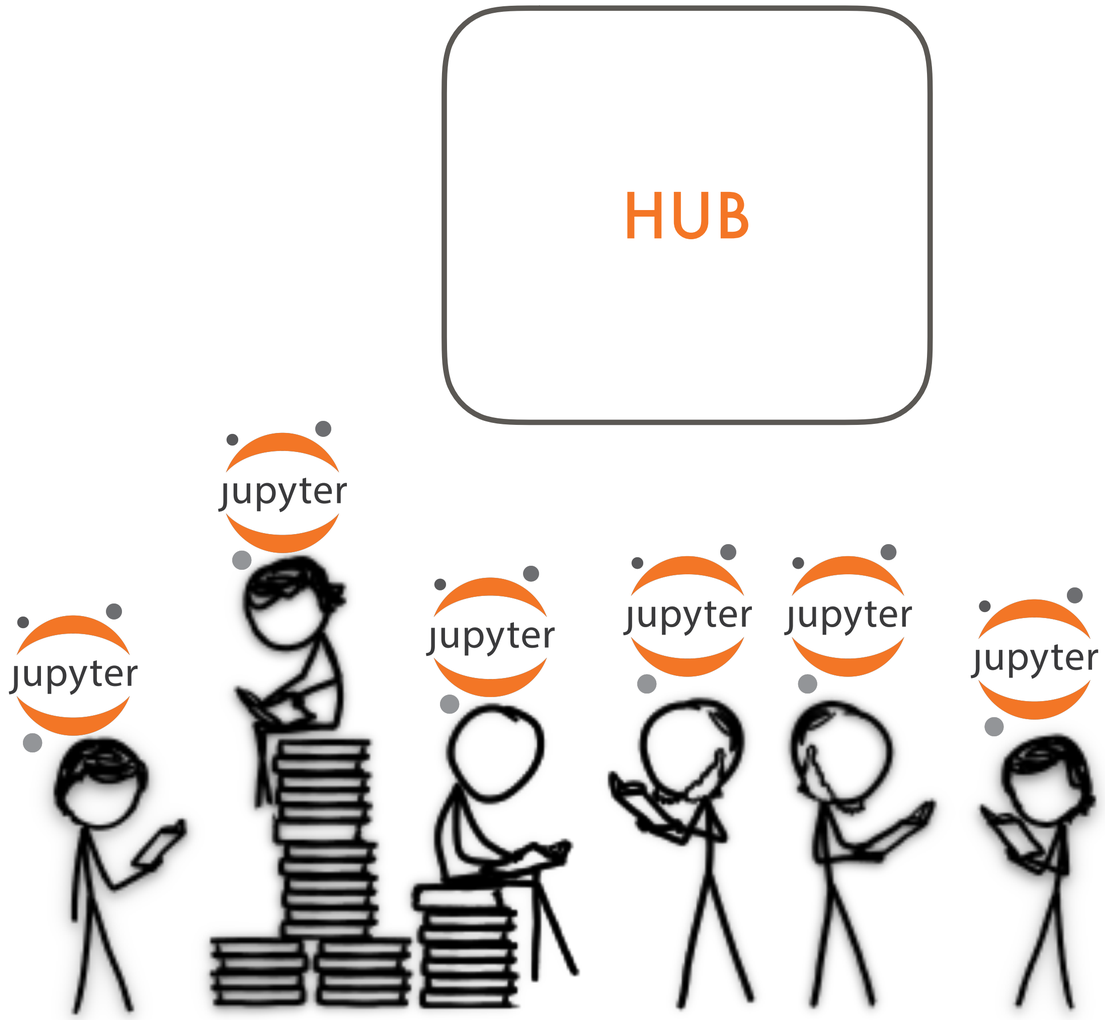
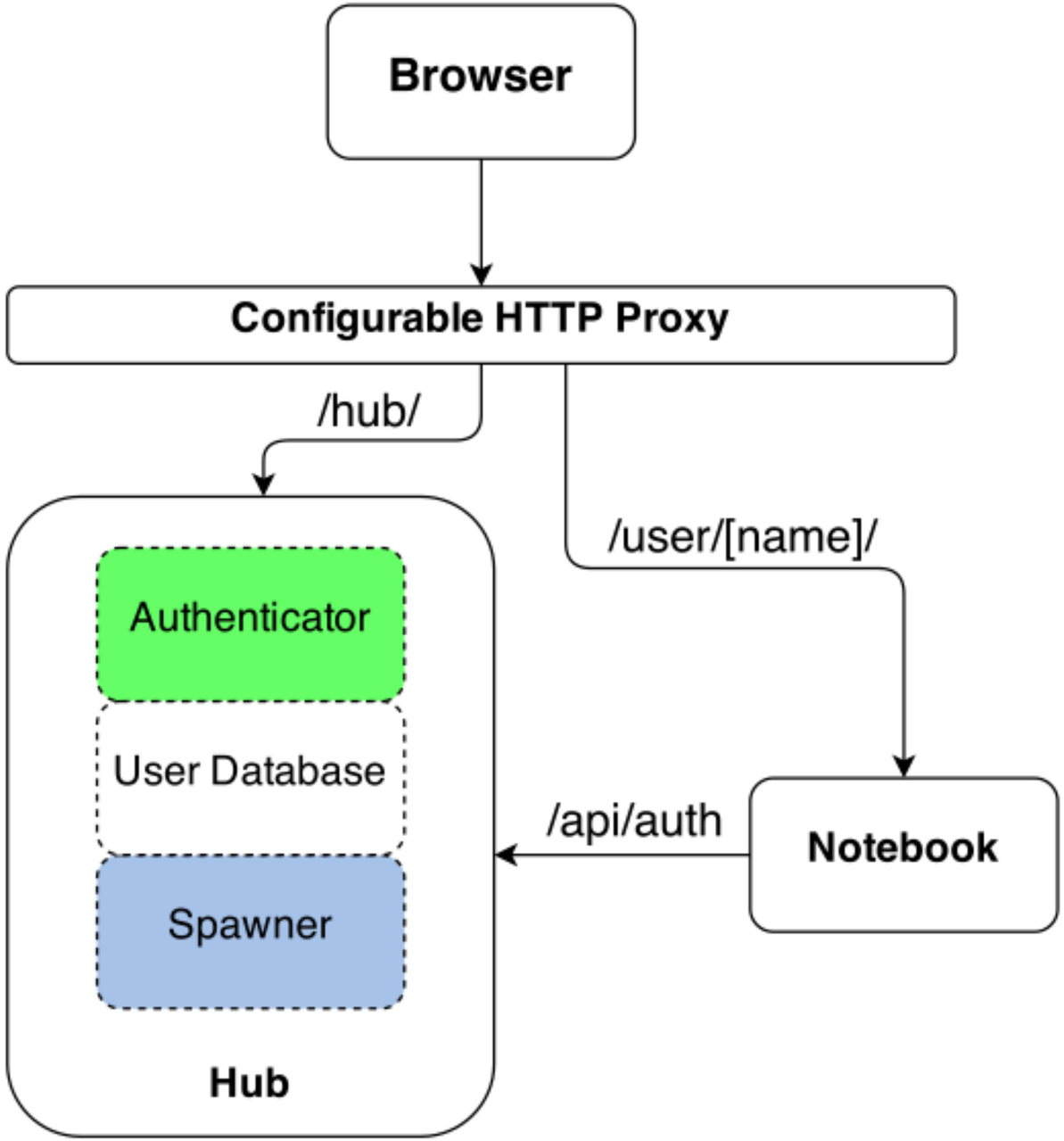


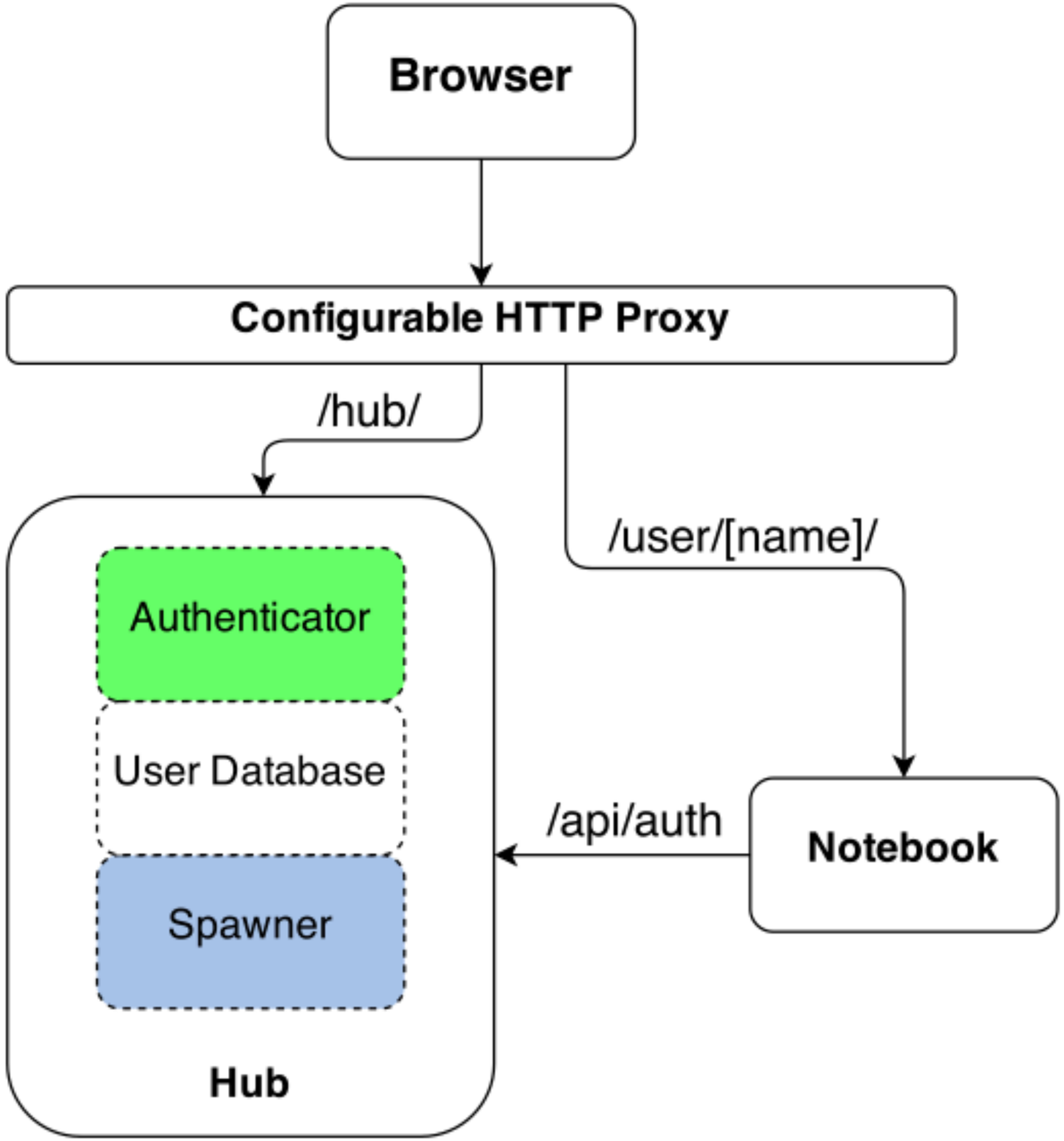
photo: xkcd, Jupyter, C. Willing

- **Hub**: manages user accounts, authentication, and coordinates Single User Notebook Servers using a Spawner
- **Proxy**: routes HTTP requests to the Hub and Single User Notebook Servers.
- **Spawner**: starts a **single-user notebook servers** when a user logs in



https://jupyterhub.readthedocs.io

- Hub launches a proxy
- Proxy forwards all requests to Hub by default
- Hub handles login, and spawns single-user servers on demand
- Hub configures proxy to forward url prefixes to the single-user notebook servers



https://jupyterhub.readthedocs.io

Kubernetes as Spawner

Zero to JupyterHub with Kubernetes

A tutorial to help install and manage JupyterHub with Kubernetes

Quick search

Zero to JupyterHub

JupyterHub is a tool that allows you to quickly utilize cloud computing infrastructure to manage a hub that enables users to interact remotely with a computing environment that you specify. JupyterHub offers a useful way to standardize the computing environment of a group of people (e.g., for a class of students or an analytics team), as well as allowing people to access the hub remotely.

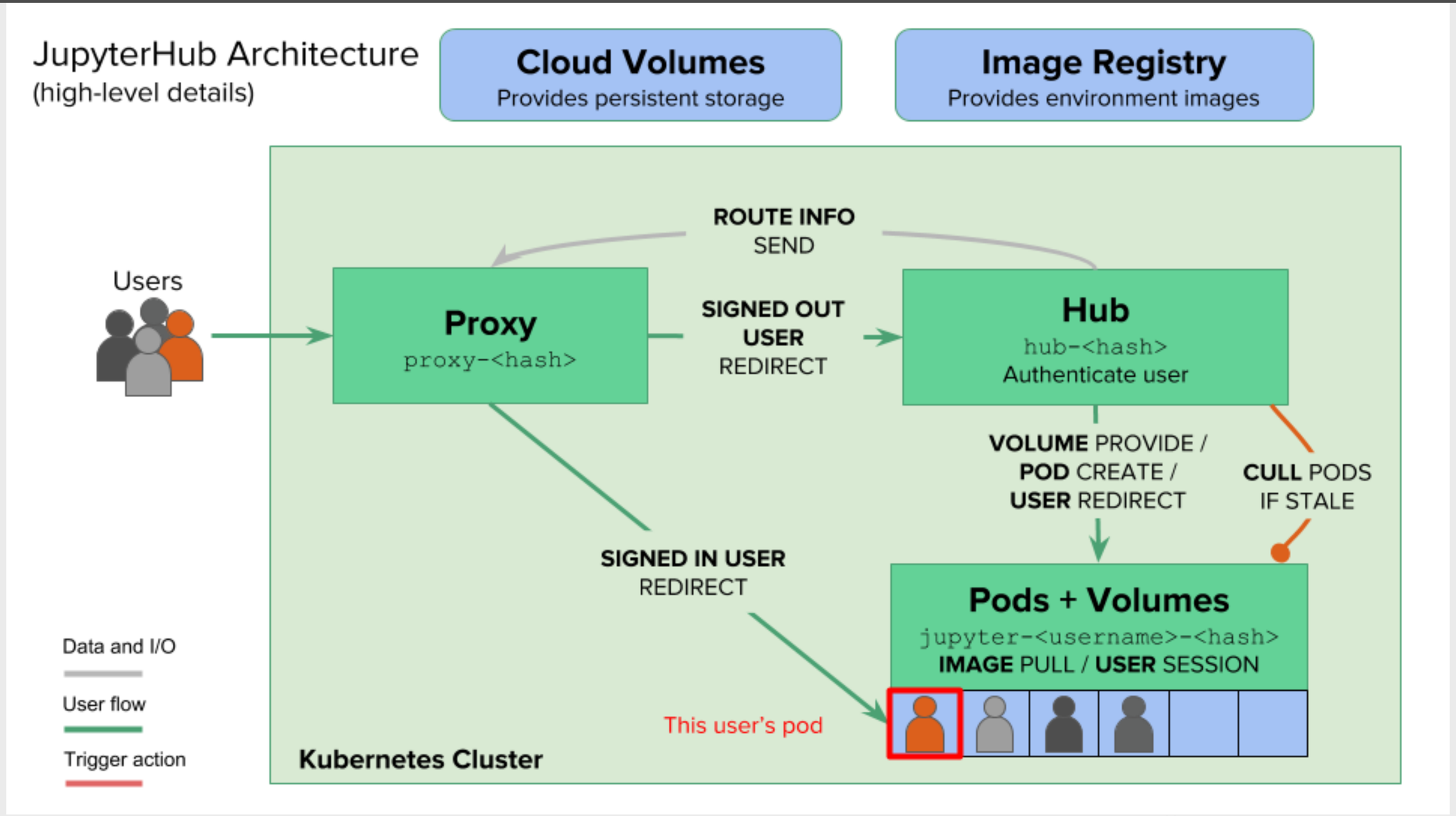
This growing collection of information will help you set up your own JupyterHub instance. It is in an early stage, so the information and tools may change quickly. If you see anything that is incorrect or have any questions, feel free to reach out at the issues page.

Creating your JupyterHub

This tutorial starts from "step zero" and walks through how to install and configure a complete JupyterHub deployment in the cloud. Using Kubernetes and the JupyterHub Helm chart provides sensible defaults for an initial deployment.

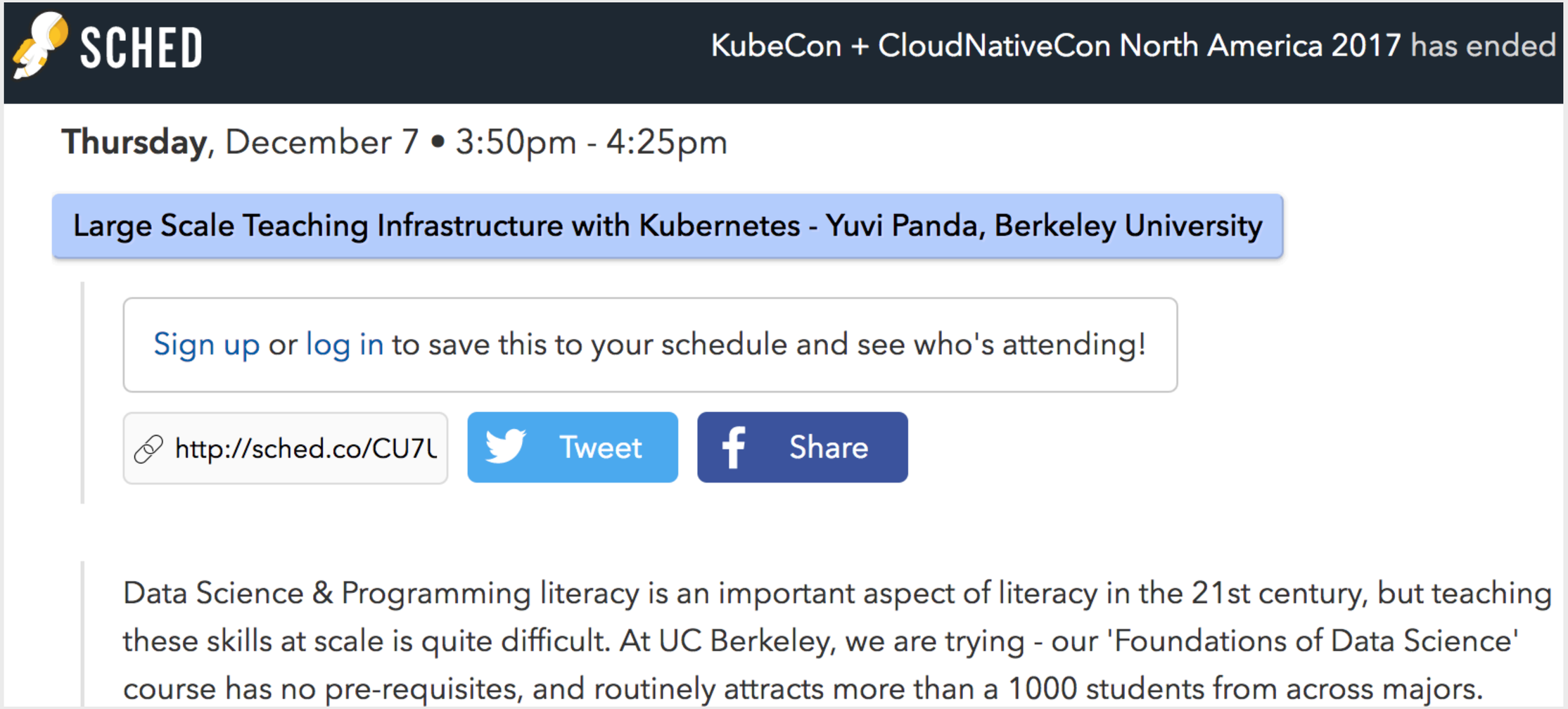
https://zero-to-jupyterhub.readthedocs.io

Kubernetes as Spawner



https://zero-to-jupyterhub.readthedocs.io

Scaling JupyterHub



https://kccncna17.sched.com/event/CU7U/large-scale-teaching-infrastructure-with-kubernetes-yuvi-panda-berkeley-university

Scaling JupyterHub

PAWS

Phabricator project: **#paws**

PAWS: A Web Shell (PAWS) is a Jupyter notebooks deployment that has been customized to make interacting with Wikimedia wikis easier. It allows users to create and share documents that contain live code, visualizations such as graphs, rich text, etc. The user created notebooks are a powerful tool that enables data analysis and scientific research, and also transforms the way in which programmers write code - by enabling an exploratory environment with a quick feedback loop, and a low barrier for entry through it's easy to use graphical interface.

Sign in with your wiki account and tada!

Contents [hide]

https://wikitech.wikimedia.org/wiki/PAWS

Scaling JupyterHub

Pangeo: JupyterHub, Dask, and XArray on the Cloud

This work is supported by Anaconda Inc, the NSF EarthCube program, and UC Berkeley BIDS

A few weeks ago a few of us stood up pangeo.pydata.org, an experimental deployment of JupyterHub, Dask, and XArray on Google Container Engine (GKE) to support atmospheric and oceanographic data analysis on large datasets. This follows on recent work to deploy Dask and XArray for the same workloads on super computers. This system is a proof of concept that has taught us a great deal about how to move forward. This blogpost briefly describes the problem, the system, then describes the collaboration, and finally discusses a number of challenges that we'll be working on in coming months.

The Problem

Atmospheric and oceanographic sciences collect (with satellites) and generate (with simulations) large datasets that they would like to analyze with distributed systems. Libraries like Dask and XArray already solve this problem computationally if scientists have their own clusters, but we seek to expand access by deploying on cloud-based systems. We build a system to which people can log in, get Jupyter Notebooks, and launch Dask clusters without much hassle. We hope that this increases access, and connects more scientists with more cloud-based datasets.

http://matthewrocklin.com/blog/work/2018/01/22/pangeo-2

Kubeflow

The Kubeflow project is dedicated to making **deployments** of machine learning (ML) workflows on Kubernetes simple, portable and scalable. Our goal is **not** to recreate other services, but to provide a straightforward way to deploy best-of-breed open-source systems **for ML** to diverse infrastructures. Anywhere you are running Kubernetes, you should be able to run Kubeflow.

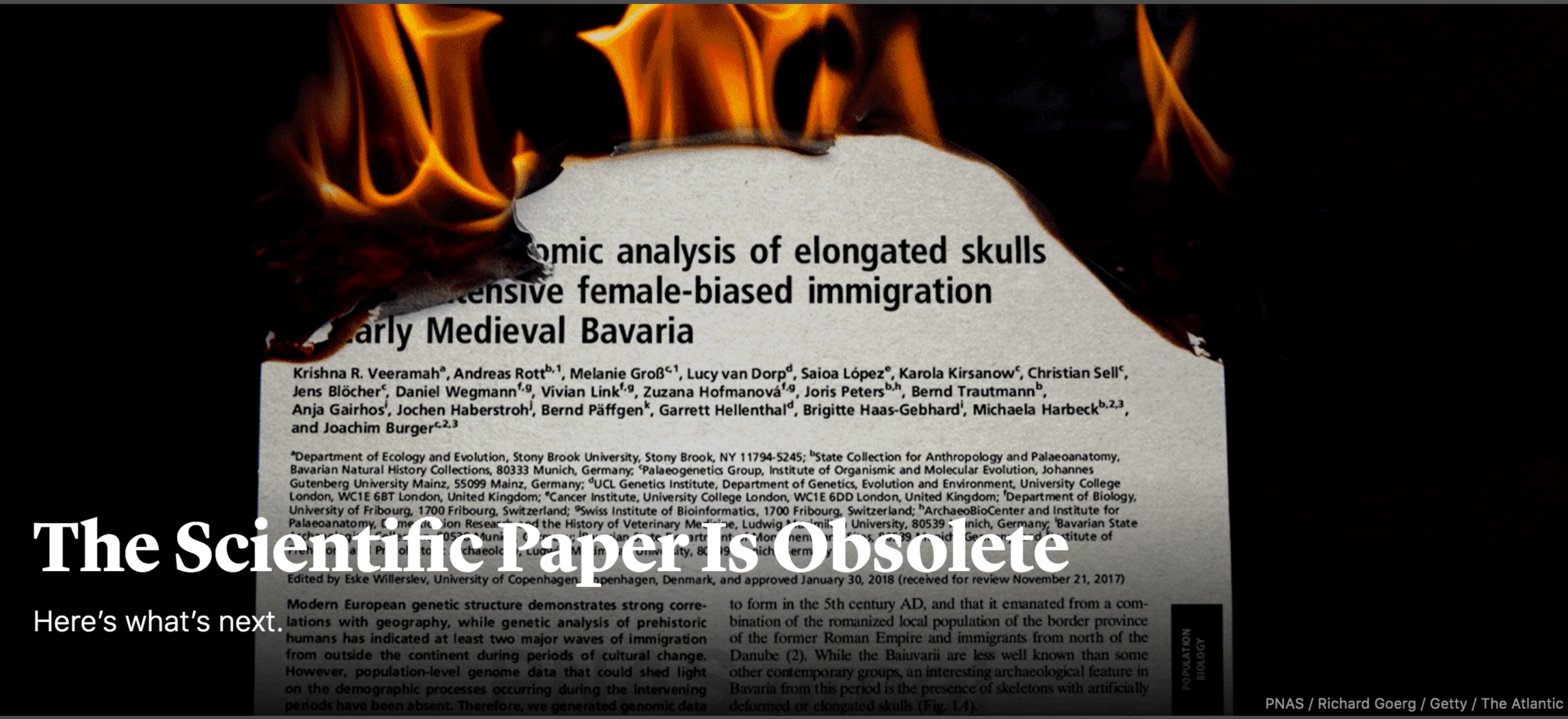
This repository contains the manifests for creating:

- A **JupyterHub** to create and manage interactive Jupyter notebooks. Project Jupyter is a non-profit, open-source project to support interactive data science and scientific computing across all programming languages.
- A **TensorFlow Training Controller** that can be configured to use either CPUs or GPUs and dynamically adjusted to the size of a cluster with a single setting
- A **TensorFlow Serving** container to export trained TensorFlow models to Kubernetes

This document details the steps needed to run the Kubeflow project in any environment in which Kubernetes runs.

https://github.com/kubeflow/kubeflow

Reproducibility



The Scientific Paper Is Obsolete

Here's what's next.

PNAS / Richard Goerg / Getty / The Atlantic

Atlantic Monthly, April 5 2018

The more sophisticated science becomes, the harder it is to communicate results. Papers today are longer than ever and full of jargon and symbols. They depend on chains of computer programs that generate data, and clean up data, and plot data, and run statistical models on data. These programs tend to be both so sloppily written and so central to the results that it's contributed to a replication crisis, or put another way, a failure of the paper to perform its most basic task: to report what you've actually discovered, clearly enough that someone else can discover it for themselves.

- James Somers

Reproducibility:

Producing similar results
with the same data

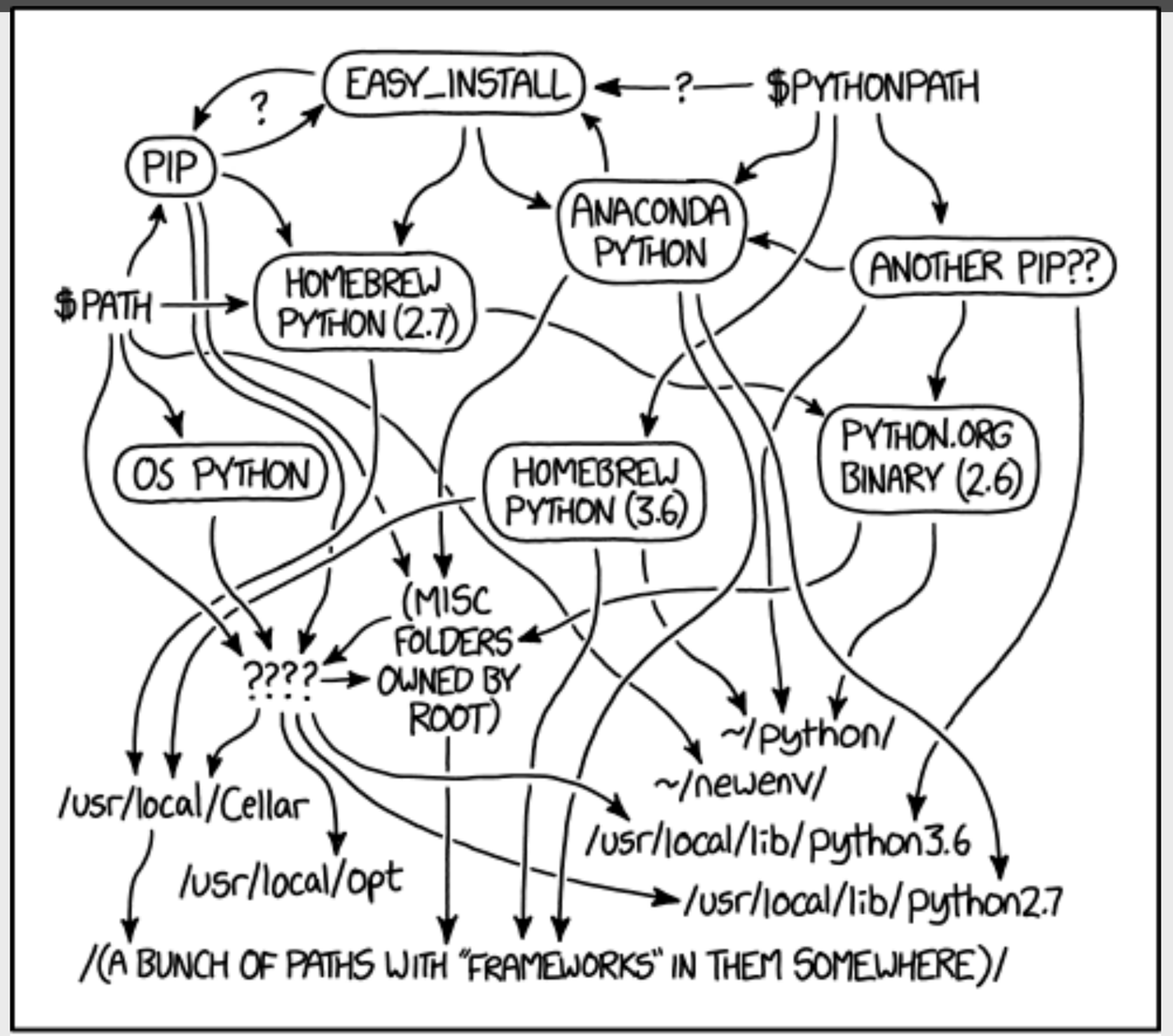
Reproducibility:

A software problem

Technical Solutions

- GitHub
- Open Science Framework
- CodaLab
- RunMyCode
- Research Journals

This week in xkcd



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED
THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

https://xkcd.com/1987/

Reproducible scientific software pipelines

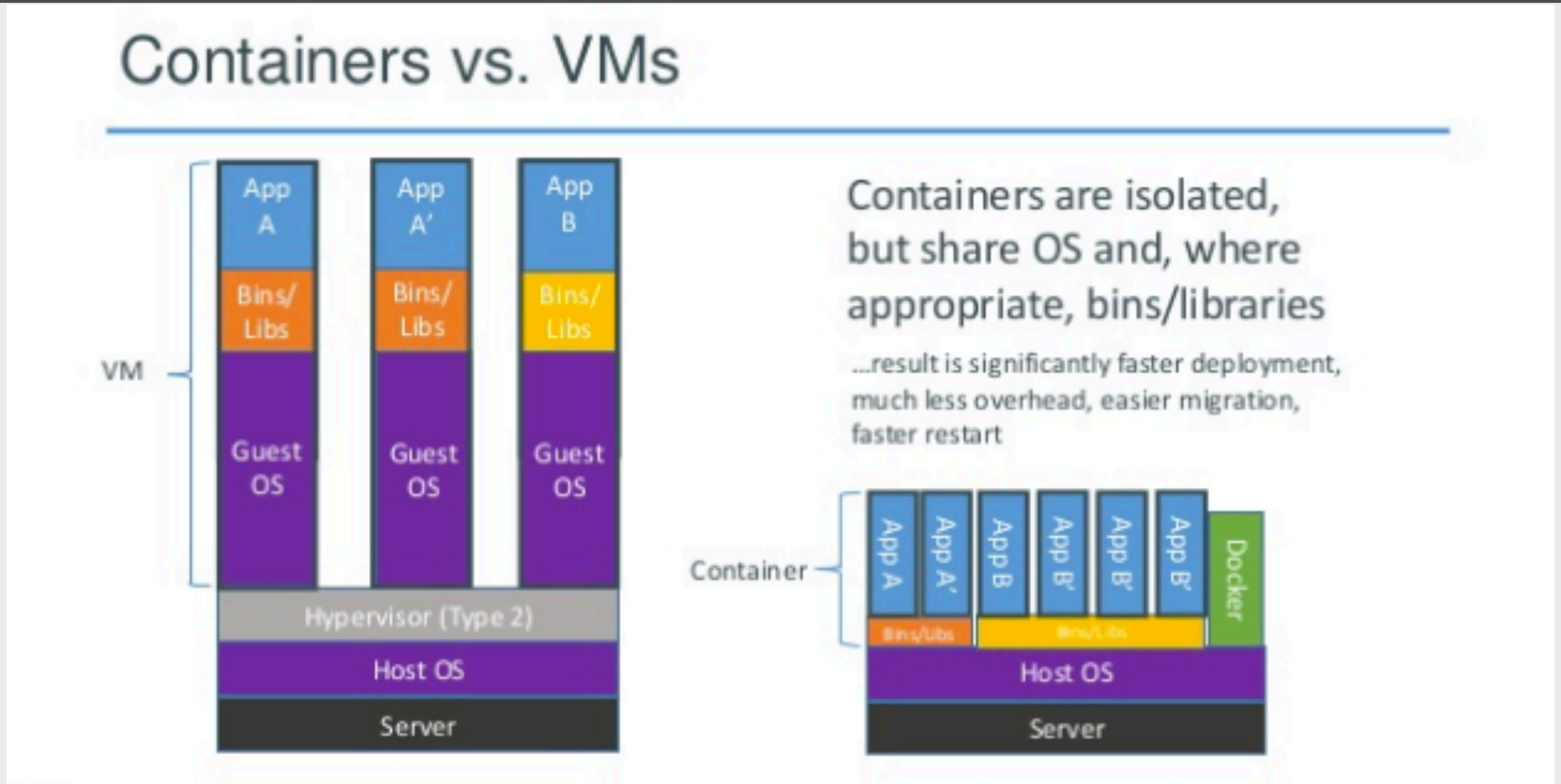
- Repository would contain:
  - Data
  - Dependencies
  - Hyperparameters
  - Scripts to run the jobs on similar hardware
  - Analysis code

Reproducible scientific software pipelines

- Repository would contain:
  - Data
  - Dependencies
  - Hyperparameters
  - Scripts to run the jobs on similar hardware
  - Analysis code

No mention of OS or lower-level software

Docker for reproducible software



source: Docker

Dockerfiles from GitHub Repos

jupyter-repo2docker

build passing docs passing

jupyter-repo2docker takes as input a repository source, such as a GitHub repo. It then builds, runs, and/or pushes Docker images built from that source.

See the repo2docker documentation for more information.

Pre-requisites

1. Docker to build & run the repositories. The community edition is recommended.
2. Python 3.4+.

Supported on Linux and macOS. See documentation note about Windows support.

https://github.com/jupyerhub/repo2docker

Using repo2docker

Note that Docker needs to be running on your machine for this to work.

Example:

```
jupyter-repo2docker https://github.com/norvig/pytudes
```

After building (it might take a while!), it should output in your terminal something like:

```
    Copy/paste this URL into your browser when you connect for the first time,
    to login with a token:
        http://0.0.0.0:36511/?token=f94f8fabb92e22f5bfab116c382b4707fc2cade56ad1ace0
```

https://github.com/jupyerhub/repo2docker

repo2docker as reproducibility unit-test

- repo2docker looks for configuration files to determine how to build the docker image
- Typically describe dependencies and other instructions to create the environment
- configs in either root or folder called `binder`

https://github.com/jupyerhub/repo2docker

Supported config files

- **Dockerfile**: full environment setup
- **environment.yml**: conda
- **requirements.txt**: pip
- **REQUIRE**: Julia
- **apt.txt**: Debian packages
- **postBuild**: custom install script
- **runtime.txt**: Python runtime

https://github.com/jupyerhub/repo2docker

Example repo

Branch: master ▾ **GAN_tutorial / environment.yml**

mickypaganini Update environment.yml

1 contributor

9 lines (8 sloc) | 139 Bytes

```
1  name: binder
2  dependencies:
3    - pytorch
4    - torchvision
5    - matplotlib==2.0.2
6    - numpy==1.14.1
7    - ipython==6.2.1
8    - scikit-learn==0.19.1
```

Branch: master ▾ **GAN_tutorial / runtime.txt**

mickypaganini Create runtime.txt

1 contributor

2 lines (1 sloc) | 11 Bytes

```
1  python-2.7
```

https://github.com/mickypaganini/GAN_tutorial

Installing LaTeX

Branch: master ▾ latex / apt.txt

Find file | Copy path

 choldgraf updating to latex symbol f152e6d on Dec 12, 2017

1 contributor

8 lines (7 sloc) | 126 Bytes

Raw | Blame | History

```
1  texlive-latex-base
2  texlive-latex-recommended
3  texlive-science
4  texlive-latex-extra
5  texlive-fonts-recommended
6  dvipng
7  ghostscript
```

https://github.com/binder-examples/latex/blob/master/apt.txt

Installing LaTeX

Branch: master ▾ latex / apt.txt

Find file | Copy path

 choldgraf updating to latex symbol f152e6d on Dec 12, 2017

1 contributor

8 lines (7 sloc) | 126 Bytes

Raw | Blame | History

```
1  texlive-latex-base
2  texlive-latex-recommended
3  texlive-science
4  texlive-latex-extra
5  texlive-fonts-recommended
6  dvipng
7  ghostscript
```

https://github.com/binder-examples/latex/blob/master/apt.txt

Using binder folder and postBuild

Branch: master ▾ **jupyterlab-demo** / binder / **postBuild** | Find file | Copy path

ian-r-rose Remove workspace. b0cd871 on Feb 16

1 contributor

Executable File | 23 lines (19 sloc) | 733 Bytes | Raw | Blame | History

```
1  set -ex
2
3  # Set up r kernel
4  export R_USER_LIB=$HOME/.R/library
5  mkdir -p $R_USER_LIB
6  echo "R_LIBS_USER='$R_USER_LIB'" > $HOME/.Renviron
7  cat $HOME/.Renviron
8  R -e "install.packages(c('repr', 'IRdisplay', 'evaluate', 'crayon', 'pbdZMQ', 'devtools', 'uuid', 'digest'), lib='$R_USER_LIB',
9  R -e "devtools::install_github('IRkernel/IRkernel', lib='$R_USER_LIB')"
10 R -e "IRkernel::installspec()"
11
12 jupyter nbconvert --to notebook --execute  --ExecutePreprocessor.timeout=60 --stdout notebooks/R.ipynb > /dev/null;
13
14 invoke build --env-name=root --no-kernel
15 invoke demofiles
16 invoke talk -t demo
17 rm -rf demofiles
18 rm -rf notebooks
19 rm -rf narrative
20 rm -rf slides
21 rm demo/notebooks/Julia.ipynb
22 jupyter lab clean
```

https://github.com/jupyterlab/jupyterlab-demo

Future: repo2docker with cloud services

Easy way to run a single repo on your own cloud instance #37

Open **yuvipanda** opened this issue on Jul 4, 2017 · 1 comment

yuvipanda commented on Jul 4, 2017 • edited ▾ **Member**

We want to provide really easy ways for users to run a github repo on their own AWS or gcloud or OpenStack whatever instance. It should ideally allow them to do persistent long term work on it, including pushing stuff back to GitHub. We can use cloud-init + packer to make this happen across clouds without too much work on our part.

We also have a responsibility to do this as securely as possible, rather than leaving random botnet invitations across the internet.

yuvipanda commented on Jul 4, 2017 **Member**

Note that this will be only for single-users - we should also have this for binderhub itself (which is multiuser).

willingc added **enhancement** **needs: discussion** labels on Dec 13, 2017

repo2docker as reproducibility unit-test

- making an image from a repository requires:
  1. version of the base docker image
  2. version of repo2docker itself
  3. versions of the libraries installed by the repository

https://github.com/jupyerhub/repo2docker

repo2docker as reproducibility unit-test

- making an image from a repository requires:
  1. version of the base docker image
  2. version of repo2docker itself
  3. versions of the libraries installed by the repository
- repo2docker deterministically controls 1 & 2
- user controls 3

https://github.com/jupyerhub/repo2docker

repo2docker as reproducibility unit-test

- Because repo2docker is deterministic, we can determine whether researchers have fully described the software environment used to load, analyze, and visualize their data
- Version numbers especially important
- If the configuration files don't create an environment that can run the code provided, the repository is **not reproducible** as currently configured

https://github.com/jupyerhub/repo2docker

Reproducibility:

Producing similar results
with the same data

Author

photo credit: xkcd

Author

Other scientist

photo credit: xkcd

Reproducibility:

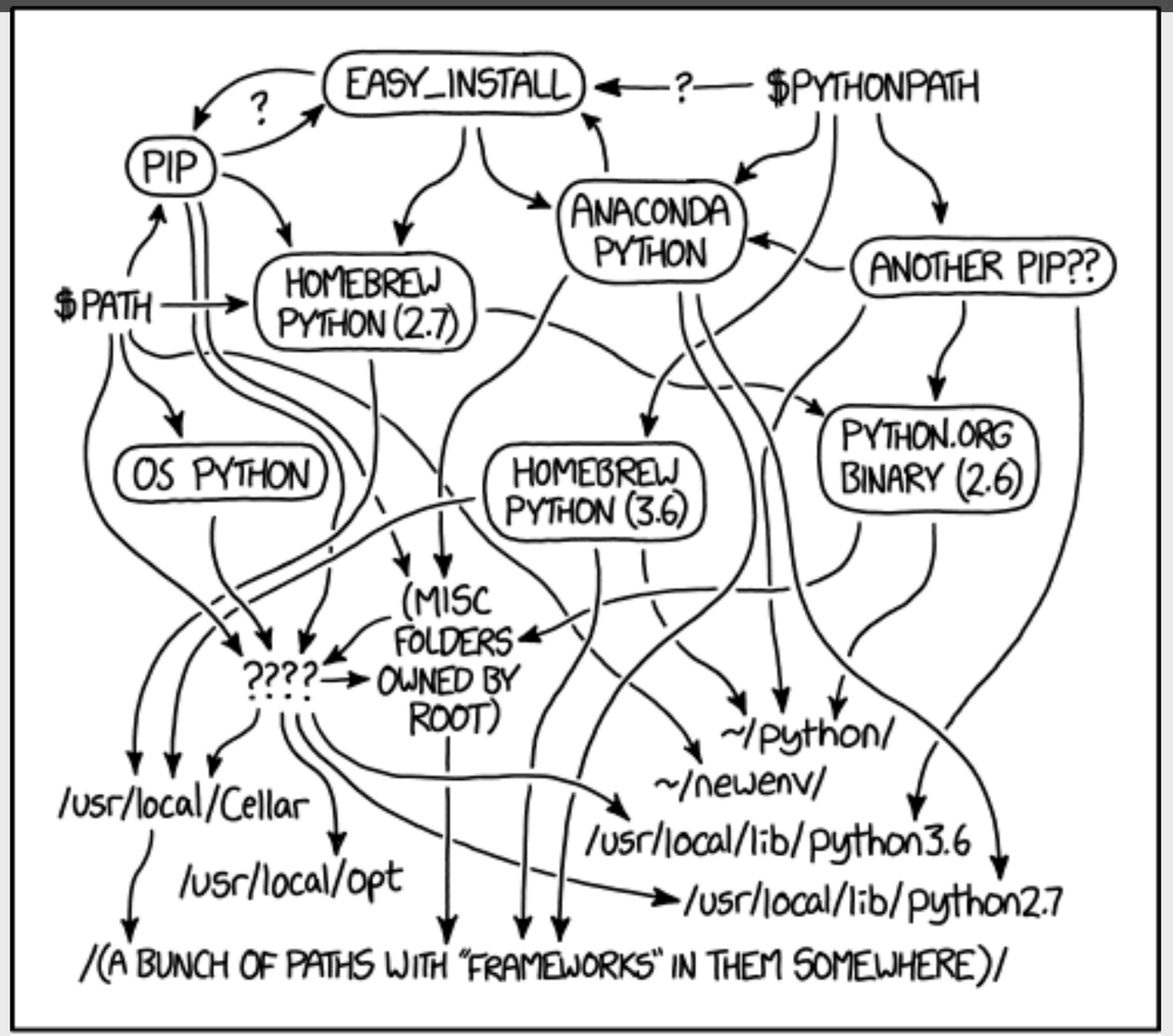
Producing similar results
with the same data
independently

repo2docker:

Produces the same
environment as the scientist

docker opens the
doors to the
laboratory itself

At a minimum, we should specify our environment



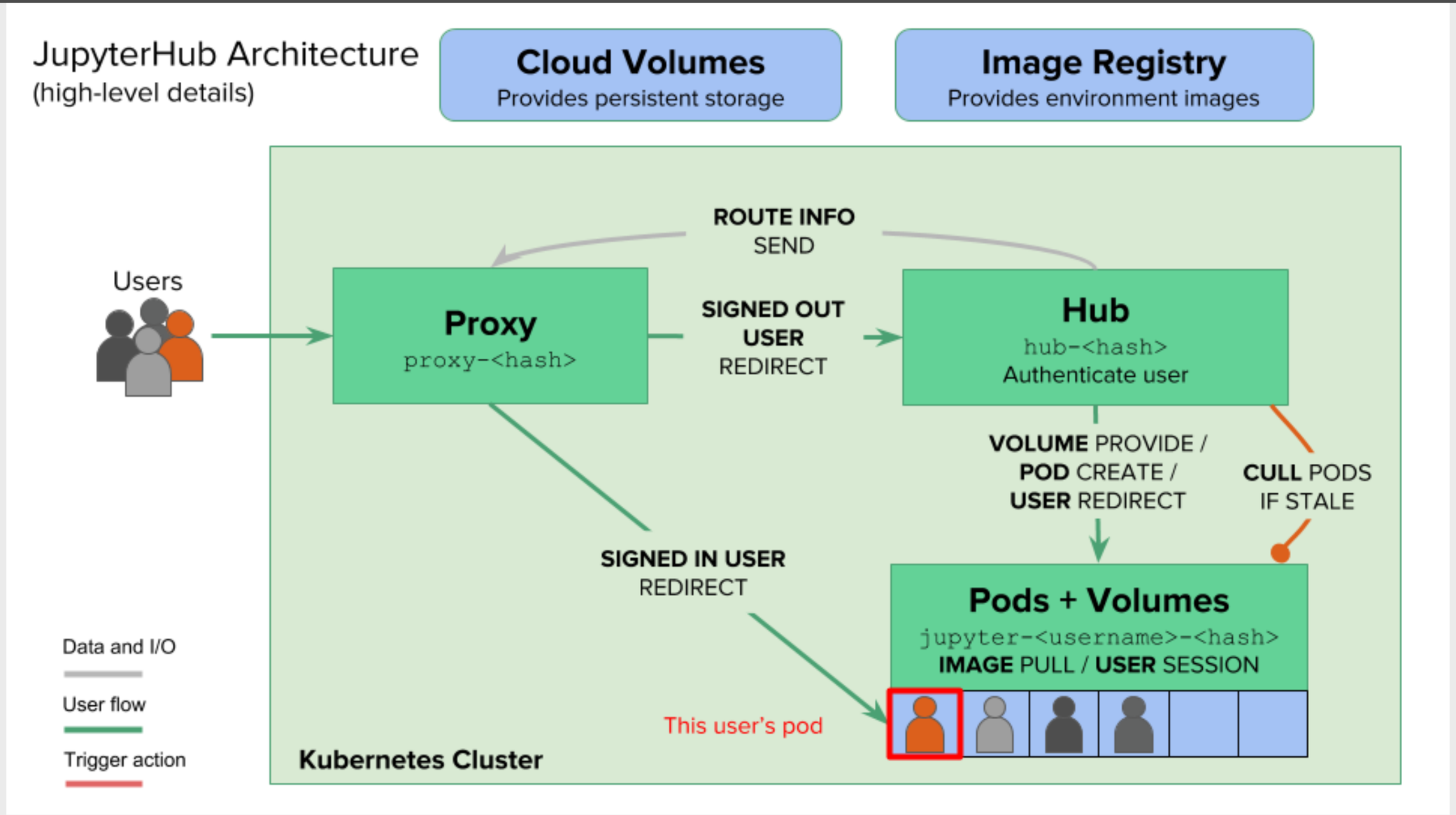
MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED
THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

https://xkcd.com/1987/

```
conda env export > environment.yml
```

```
pip freeze > requirements.txt
```

Docker for pods



https://zero-to-jupyterhub.readthedocs.io

JupyterHub, repo2docker as a service

Binder:

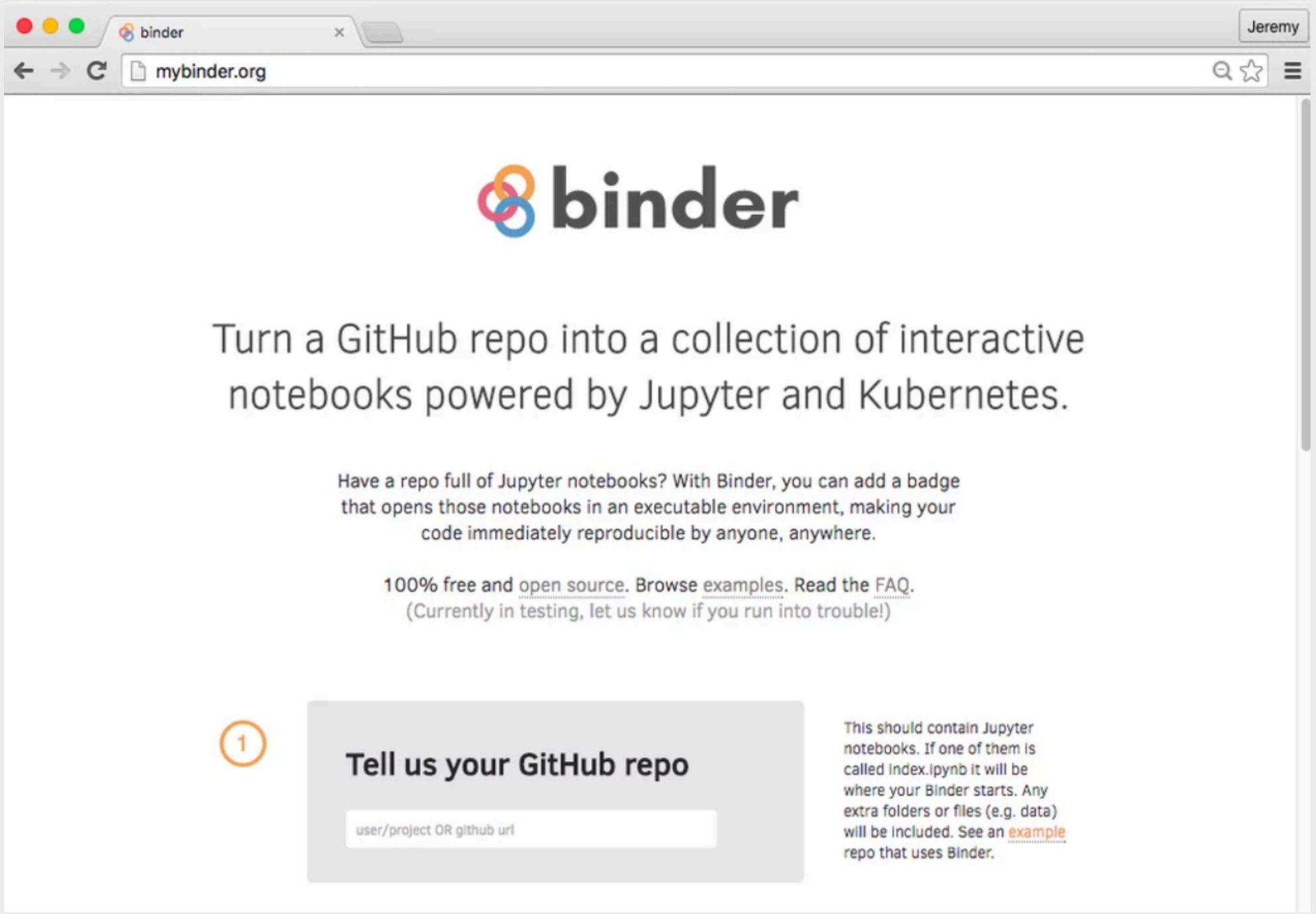
Public repositories using JupyterHub and repo2docker

Toward publishing reproducible computation with Binder

Binder makes it easy to include an interactive version of your analysis, with the supporting data and code, alongside a published paper.

LABS May 13, 2016

https://elifesciences.org/labs/a7d53a88/toward-publishing-reproducible-computation-with-binder



https://elifesciences.org/labs/a7d53a88/toward-publishing-reproducible-computation-with-binder

Introducing Binder 2.0 – share your interactive research environment

The Project Jupyter team shares its reboot of Binder, the tool that allows researchers to make their GitHub repositories executable by others.

LABS Nov 30, 2017

https://elifesciences.org/labs/8653a61d/introducing-binder-2-0-share-your-interactive-research-environment

Turn a GitHub repo into a collection of interactive notebooks

Have a repository full of Jupyter notebooks? With Binder, open those notebooks in an executable environment, making your code immediately reproducible by anyone, anywhere.

Build and launch a repository

GitHub repo or URL

GitHub repository name or link

Git branch, tag, or commit

master

Path to a notebook file (optional)

Path to a notebook file (optional)

File

launch

Copy the URL below and share your Binder with others:

```
Fill in the fields to see a URL for sharing your Binder.
```

Copy the text below, then paste into your README to show a binder badge: launch binder

mybinder.org

Build and launch a repository

GitHub repo or URL

https://github.com/jzf2101/paper-learning-to-pivot

Git branch, tag, or commit

master

Path to a notebook file (optional)

Path to a notebook file (optional)

File

launch

Copy the URL below and share your Binder with others:

```
https://mybinder.org/v2/gh/jzf2101/paper-learning-to-pivot/master
```

Copy the text below, then paste into your README to show a binder badge: launch binder

Already built! Launching

mybinder.org

https://mybinder.org/gh/
username/repo/branch

GitHub (GitLab and others also available)

https://mybinder.org/gh/username/repo/branch

https://mybinder.org/gh/
username/repo/branch?
urlpath=lab

JupyterLab

Secure https://hub.mybinder.org/user/jzf2101-paper-learning-to-pivot-xx6cna24/lab

File Edit View Run Kernel Tabs Settings Help

Files Running Commands Cell Tools Tabs

code

Name	Last Modified
Jets.ipynb	seconds ago
Toy.ipynb	29 minutes ago
jets.py	29 minutes ago

Jets.ipynb

Code Python 3

In [1]:
```
%matplotlib inline
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = (6,6)

import numpy as np
np.random.seed = 777
```

Prepare data

In [2]:
```
import h5py

# Get data from http://www.igb.uci.edu/~pfbaldi/p

f = h5py.File("hepjets/test_no_pile_5000000.h5",
X_no_pile = f["features"].value
y_no_pile = f["targets"].value.ravel()

f = h5py.File("hepjets/test_pile_5000000.h5", "r"
X_pile = f["features"].value
y_pile = f["targets"].value.ravel()
```

In [3]:
```
from sklearn.cross_validation import train_test_

X = np.vstack((X_no_pile, X_pile))
y = np.concatenate((y_no_pile, y_pile)).ravel()
z = np.zeros(len(X))
z[len(X_no_pile):] = 1

strates = np.zeros(len(X))
strates[(y==0)&(z==0)]=0
strates[(y==0)&(z==1)]=1
strates[(y==1)&(z==0)]=2
```

paper.pdf

Learning to Pivot with Adversarial Networks

Gilles Louppe
New York University
g.louppe@nyu.edu

Michael Kagan
SLAC National Accelerator Laboratory
makagan@slac.stanford.edu

Kyle Cranmer
New York University
kyle.cranmer@nyu.edu

Abstract

Several techniques for domain adaptation have been proposed to account for differences in the distribution of the data used for training and testing. The majority of this work focuses on a binary domain label. Similar problems occur in a scientific context where there may be a continuous family of plausible data generation processes associated to the presence of systematic uncertainties. Robust inference is possible if it is based on a pivot – a quantity whose distribution does not depend on the unknown values of the nuisance parameters that parametrize this family of data generation processes. In this work, we introduce and derive theoretical results for a training procedure based on adversarial networks for enforcing the pivotal property (or, equivalently, fairness with respect to continuous attributes) on a predictive model. The method includes a hyperparameter to control the trade-off between accuracy and robustness. We demonstrate the effectiveness of this approach with a toy example and examples from particle physics.

1 Introduction

Machine learning techniques have been used to enhance a number of scientific disciplines, and they have the potential to transform even more of the scientific process. One of the challenges of applying machine learning to scientific problems is the need to incorporate systematic uncertainties, which affect both the robustness of inference and the metrics used to evaluate a particular analysis strategy.

In this work, we focus on supervised learning techniques where systematic uncertainties can be associated to a data generation process that is not uniquely specified. In other words, the lack of systematic uncertainties corresponds to the (rare) case that the process that generates training data is unique, fully specified, and an accurate representative of the real world data. By contrast, a common situation when systematic uncertainty is present is when the training data are not representative of the real data. Several techniques for domain adaptation have been developed to create models that are more robust to this binary type of uncertainty. A more generic situation is that there are several plausible data generation processes, specified as a family parametrized by continuous nuisance parameters, as is typically found in scientific domains. In this broader context, statisticians have for long been working on robust inference techniques based on the concept of a pivot – a quantity whose distribution is invariant with the nuisance parameters (see e.g., (Degroot and Schervish, [1975])).

Assuming a probability model $p(X, Y, Z)$, where X are the data, Y are the target labels, and Z are the nuisance parameters, we consider the problem of learning a predictive model $f(X)$ for Y conditional on the observed values of X that is robust to uncertainty in the unknown value of Z. We introduce a flexible learning procedure based on adversarial networks (Goodfellow et al., 2014) for enforcing that $f(X)$ is a pivot with respect to Z. We derive theoretical results proving that the procedure converges towards a model that is both optimal and statistically independent of the nuisance parameters (if that model exists) or for which one can tune a trade-off between accuracy and robustness (e.g., as driven by a higher level objective). In particular, and to the best of our knowledge, our contribution is the first solution for imposing pivotal constraints on a predictive model, working regardless of the

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

pure-python fitting/limit-setting/interval estimation HistFactory-style

The HistFactory p.d.f. template [CERN-OPEN-2012-016] is per-se independent of its implementation in ROOT and sometimes, it's useful to be able to run statistical analysis outside of ROOT, RooFit, RooStats framework.

This repo is a pure-python implementation of that statistical model for multi-bin histogram-based analysis and its interv estimation is based on the asymptotic formulas of "Asymptotic formulae for likelihood-based tests of new physics" [arxiv:1007.1727]. The aim is also to support modern computational graph libraries such as PyTorch and Tensorflow in order to make use of features such as autodifferentiation and GPU acceleration.

What does it support

Implemented variations:

- [x] HistoSys
- [x] OverallSys
- [x] ShapeSys
- [x] NormFactor
- [x] Multiple Channels
- [x] Import from XML + ROOT via `uproot`
- [x] ShapeFactor
- [x] StatError

https://github.com/diana-hep/pyhf

pytorch_tests_onoff

Secure https://hub.mybinder.org/user/diana-hep-pyhf-1djdehjf/notebooks/examples/notebooks/pytorch_tests_onoff.ipynb

jupyter pytorch_tests_onoff Last Checkpoint: 8 minutes ago (unsaved changes)

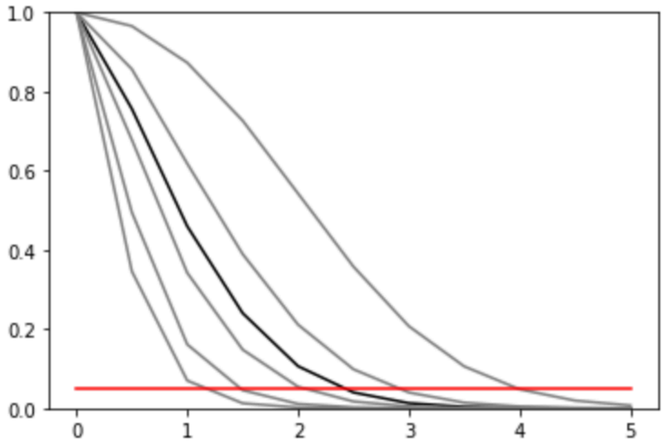
Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

In [5]:
```
%pylab inline

testmus = np.linspace(0,5,11)
results = []
for mu in testmus:
    results.append(runOnePoint(mu, data, pdf, init_pars, par_bounds))

obs = [1./r[-2:][0] for r in results ]
exp = [[1./r[-2:][1][i] for r in results] for i in range(5) ]

def plot_results(testmus,cls_obs, cls_exp, test_size = 0.05):
    plt.plot(testmus,cls_obs, c = 'k')
    for i,c in zip(range(5),['grey','grey','grey','grey','grey']):
        plt.plot(testmus,cls_exp[i], c = c)
    plt.plot(testmus,[test_size]*len(testmus), c = 'r')
    plt.ylim(0,1)


plot_results(testmus,obs,exp)
```

```
Populating the interactive namespace from numpy and matplotlib
```



https://mybinder.org/v2/gh/diana-hep/pyhfl/master?urlpath=tree/examples/notebooks/pytorch_tests_onoff.ipynb

StatisticalAnalysis

Secure https://hub.mybinder.org/user/diana-hep-pyhf-omzijddo/notebooks/examples/notebooks/binderexample/StatisticalAnalysis.ipynb

jupyter StatisticalAnalysis (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run Code

pyhf

`pyhf` is a work-in-progress standalone implementation of the HistFactory p.d.f. template and an implementation of the test statistics and asymptotic formulae described in the paper by Cowan, Cranmer, Gross, Vitells: *Asymptotic formulae for likelihood-based tests of new physics* [arxiv:1007.1727].

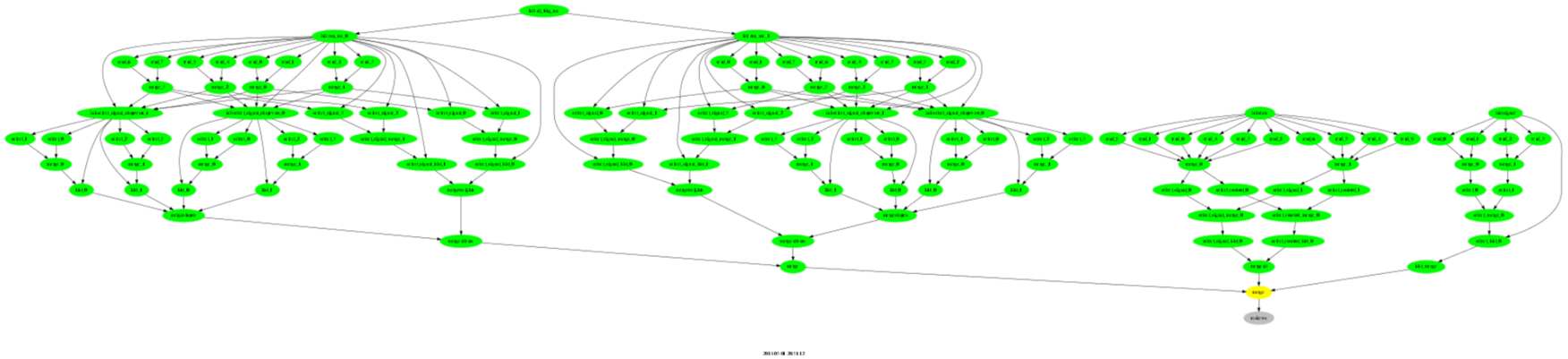
Models can be defined using JSON specification, but existing models based on the XML + ROOT file scheme are readable as well.

The Demo

The input data for the statistical analysis was build generated using the containerized workflow engine yadage (see demo from KubeCon 2018 [youtube]). Similarly to Binder this utilizes modern container technology for reproducible science. Below you see the execution graph leading up to the model input data at the bottom.

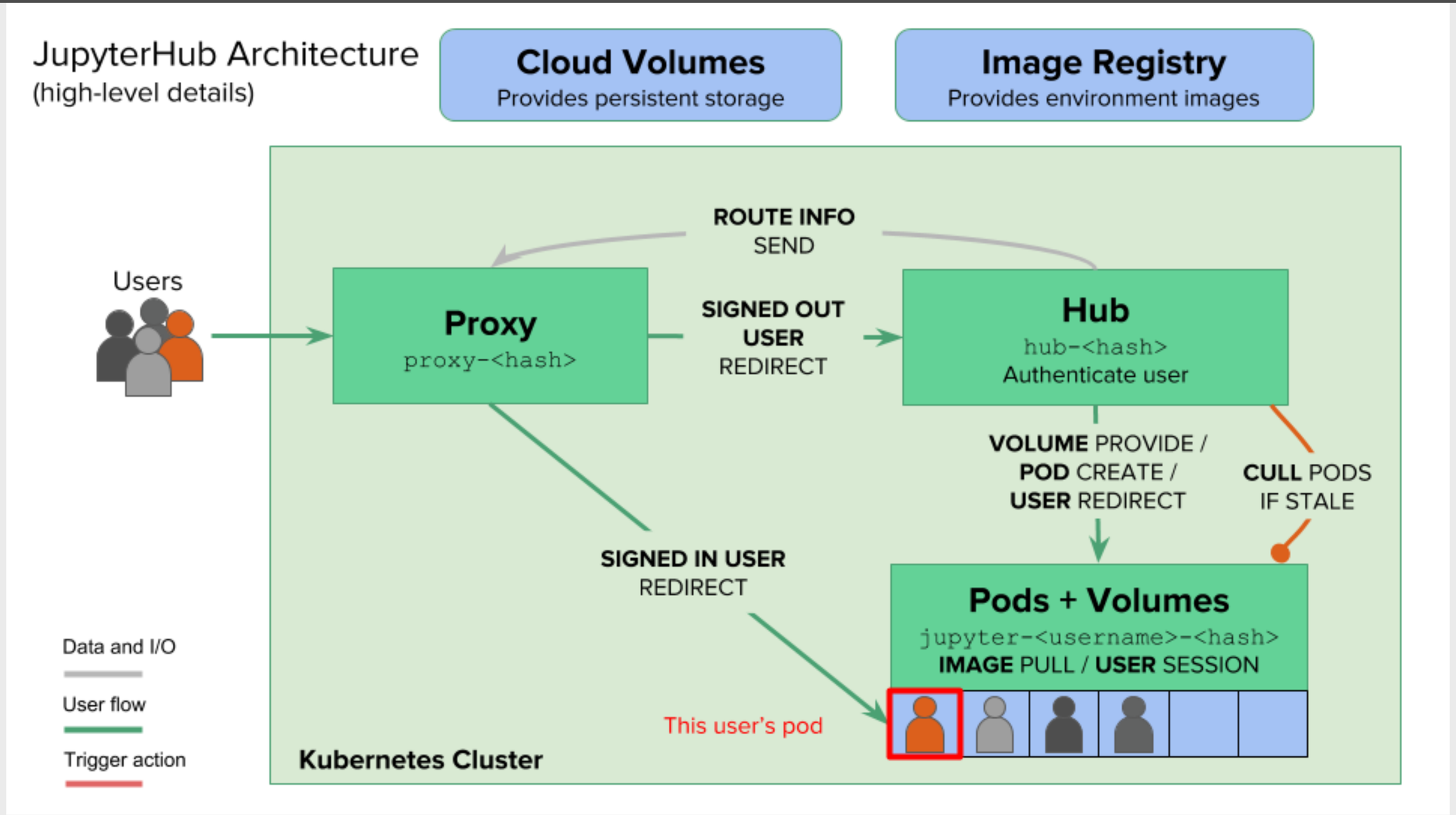
In [3]:
```
import base64
from IPython.core.display import display, HTML
anim = base64.b64encode(open('workflow.gif','rb').read()).decode('ascii')
HTML('<img src="{}">'.format(anim))
```

Out[3]:



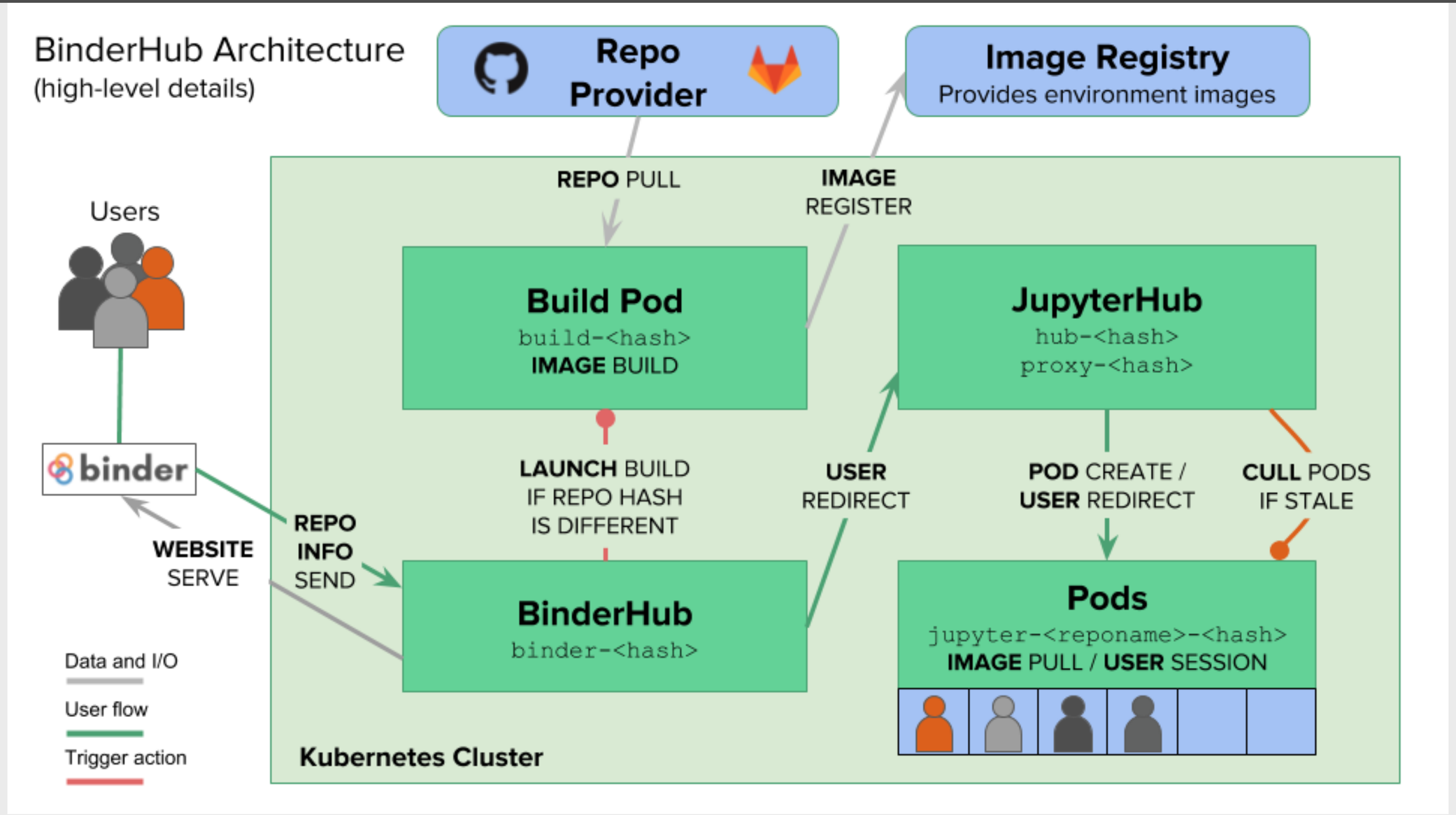
https://mybinder.org/v2/gh/diana-hep/pyhfl/binderdemo?urlpath=tree/examples/notebooks/binderexample/StatisticalAnalysis.ipynb

JupyterHub



https://zero-to-jupyterhub.readthedocs.io

JupyterHub + repo2docker = binder



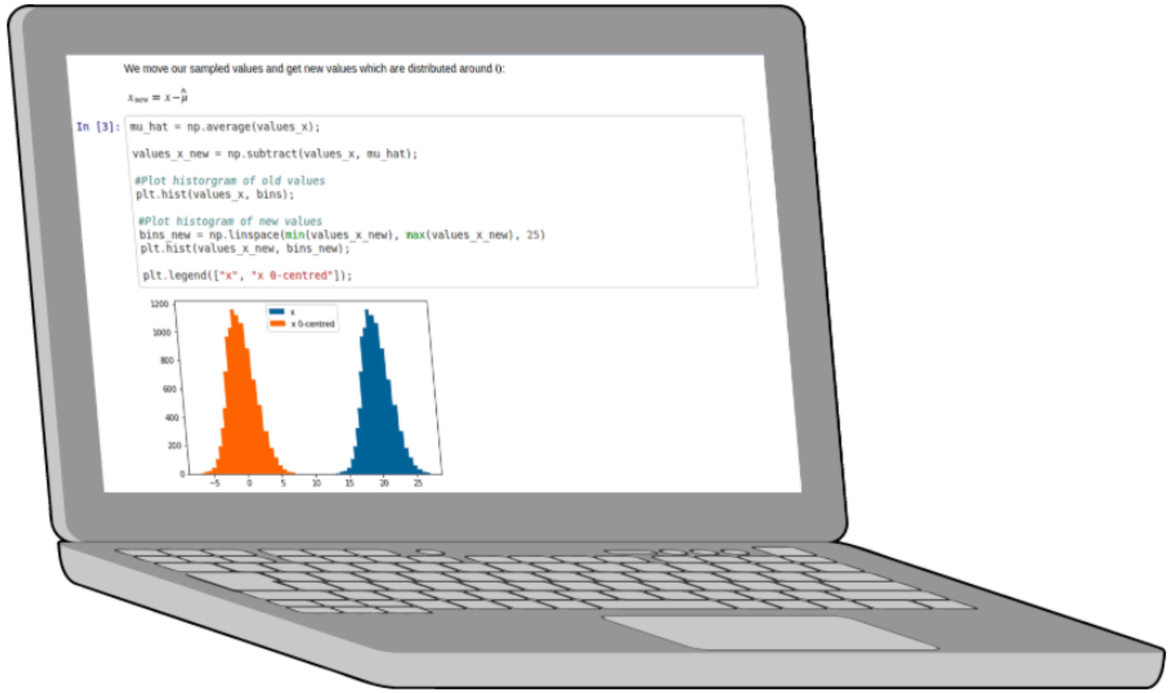
photocredit: binder team

gesis Leibniz Institute for the Social Sciences

Contact About

GESIS Notebooks (beta)

Home Your Server Binder



Open Research Computing for the Social Sciences

Your Server

https://notebooks.gesis.org/

Create your own JupyterHub

Zero to JupyterHub with Kubernetes

A tutorial to help install and manage JupyterHub with Kubernetes

Quick search

Zero to JupyterHub

JupyterHub is a tool that allows you to quickly utilize cloud computing infrastructure to manage a hub that enables users to interact remotely with a computing environment that you specify. JupyterHub offers a useful way to standardize the computing environment of a group of people (e.g., for a class of students or an analytics team), as well as allowing people to access the hub remotely.

This growing collection of information will help you set up your own JupyterHub instance. It is in an early stage, so the information and tools may change quickly. If you see anything that is incorrect or have any questions, feel free to reach out at the issues page.

Creating your JupyterHub

This tutorial starts from "step zero" and walks through how to install and configure a complete JupyterHub deployment in the cloud. Using Kubernetes and the JupyterHub Helm chart provides sensible defaults for an initial deployment.

https://zero-to-jupyterhub.readthedocs.io

Create your own BinderHub

Zero to BinderHub

A guide to help you create your own BinderHub from scratch.

https://binderhub.readthedocs.io/

BinderHub

A project to build and serve Binders

Watch 41

Table of Contents

5. BinderHub Deployments

BinderHub is open-source technology that can be deployed anywhere that Kubernetes is deployed. The Binder community hopes that it will be used for many applications in research and education. As new organizations adopt BinderHub, we'll update this page in order to provide inspiration to others who wish to do so.

If you or your organization has set up a BinderHub that isn't listed here, please open an issue on our GitHub repository to discuss adding it!

5.1. GESIS Institute for Social Sciences

Deployed on bare-metal using `kubeadm`.

- Deployment repository
- BinderHub / JupyterHub links

https://binderhub.readthedocs.io/

The scope of the problem

PNAS PNAS PNAS

An empirical analysis of journal policy effectiveness for computational reproducibility

Victoria Stodden[a,1], Jennifer Seiler[b], and Zhaokun Ma[b]

[a]School of Information Sciences, University of Illinois at Urbana–Champaign, Champaign, IL 61820; and [b]Department of Statistics, Columbia University, New York, NY 10027

Edited by David B. Allison, Indiana University Bloomington, Bloomington, IN, and accepted by Editorial Board Member Susan T. Fiske January 9, 2018 (received for review July 11, 2017)

A key component of scientific communication is sufficient information for other researchers in the field to reproduce published findings. For computational and data-enabled research, this has often been interpreted to mean making available the raw data from which results were generated, the computer code that generated the findings, and any additional information needed such as workflows and input parameters. Many journals are revising author guidelines to include data and code availability. This work evaluates the effectiveness of journal policy that requires the data and code necessary for reproducibility be made available postpublication by the authors upon request. We assess the effectiveness of such a policy by (*i*) requesting data and code from authors and (*ii*) attempting replication of the published findings. We chose a random sample of 204 scientific papers published in the journal *Science* after the implementation of their policy in February 2011. We found that we were able to obtain artifacts from 44% of our sample and were able to reproduce the findings for 26%. We find this policy—author remission of data and code postpublication upon request—an improvement over no policy, but currently insufficient for reproducibility.

reproducible research | data access | code access | reproducibility policy | open science

putational reproducibility of published results. We use a survey instrument to test the availability of data and code for articles published in *Science* in 2011–2012. We then use the scientific communication standards from the 2012 Institute for Computational and Experimental Research in Mathematics (ICERM) workshop report to evaluate the reproducibility of articles for which artifacts were made available (11). We then assess the impact of the policy change directly, by examining articles published in *Science* in 2009–2010 and comparing artifact ability to our postpolicy sample from 2011–2012. Finally, we discuss possible improvements to journal policies for enabling reproducible computational research in light of our results.

Results

We emailed corresponding authors in our sample to request the data and code associated with their articles and attempted to replicate the findings from a randomly chosen subset of the articles for which we received artifacts. We estimate the artifact recovery rate to be 44% with a 95% bootstrap confidence interval of the proportion $[0.36, 0.50]$, and we estimate the replication rate to be 26% with a 95% bootstrap confidence interval $[0.20, 0.32]$.

COMPUTER SCIENCES

SOCIAL SCIENCES

The scope of the problem

An empirical analysis of journal policy effectiveness for computational reproducibility

Victoria Stodden[a,1], **Jennifer Seiler**[b], **and Zhaokun Ma**[b]

[a]School of Information Sciences, University of Illinois at Urbana–Champaign, Champaign, IL 61820; and [b]Department of Statistics, Columbia University, New York, NY 10027

Edited by David B. Allison, Indiana University Bloomington, Bloomington, IN, and accepted by Editorial Board Member Susan T. Fiske January 9, 2018 (received for review July 11, 2017)

A key component of scientific communication is sufficient information for other researchers in the field to reproduce published findings. For computational and data-enabled research, this has often been interpreted to mean making available the raw data from which results were generated, the computer code that generated the findings, and any additional information needed such as workflows and input parameters. Many journals are revising author guidelines to include data and code availability. This work evaluates the effectiveness of journal policy that requires the data and code necessary for reproducibility be made available postpublication by the authors upon request. We assess the effectiveness of such a policy by (*i*) requesting data and code from authors and (*ii*) attempting replication of the published findings. We chose a random sample of 204 scientific papers published in the journal *Science* after the implementation of their policy in February 2011. We found that we were able to obtain artifacts from 44% of our sample and were able to reproduce the findings for 26%. We find this policy—author remission of data and code postpublication upon request—an improvement over no policy, but currently insufficient for reproducibility.

reproducible research | data access | code access | reproducibility policy | open science

putational reproducibility of published results. We use a survey instrument to test the availability of data and code for articles published in *Science* in 2011–2012. We then use the scientific communication standards from the 2012 Institute for Computational and Experimental Research in Mathematics (ICERM) workshop report to evaluate the reproducibility of articles for which artifacts were made available (11). We then assess the impact of the policy change directly, by examining articles published in *Science* in 2009–2010 and comparing artifact ability to our postpolicy sample from 2011–2012. Finally, we discuss possible improvements to journal policies for enabling reproducible computational research in light of our results.

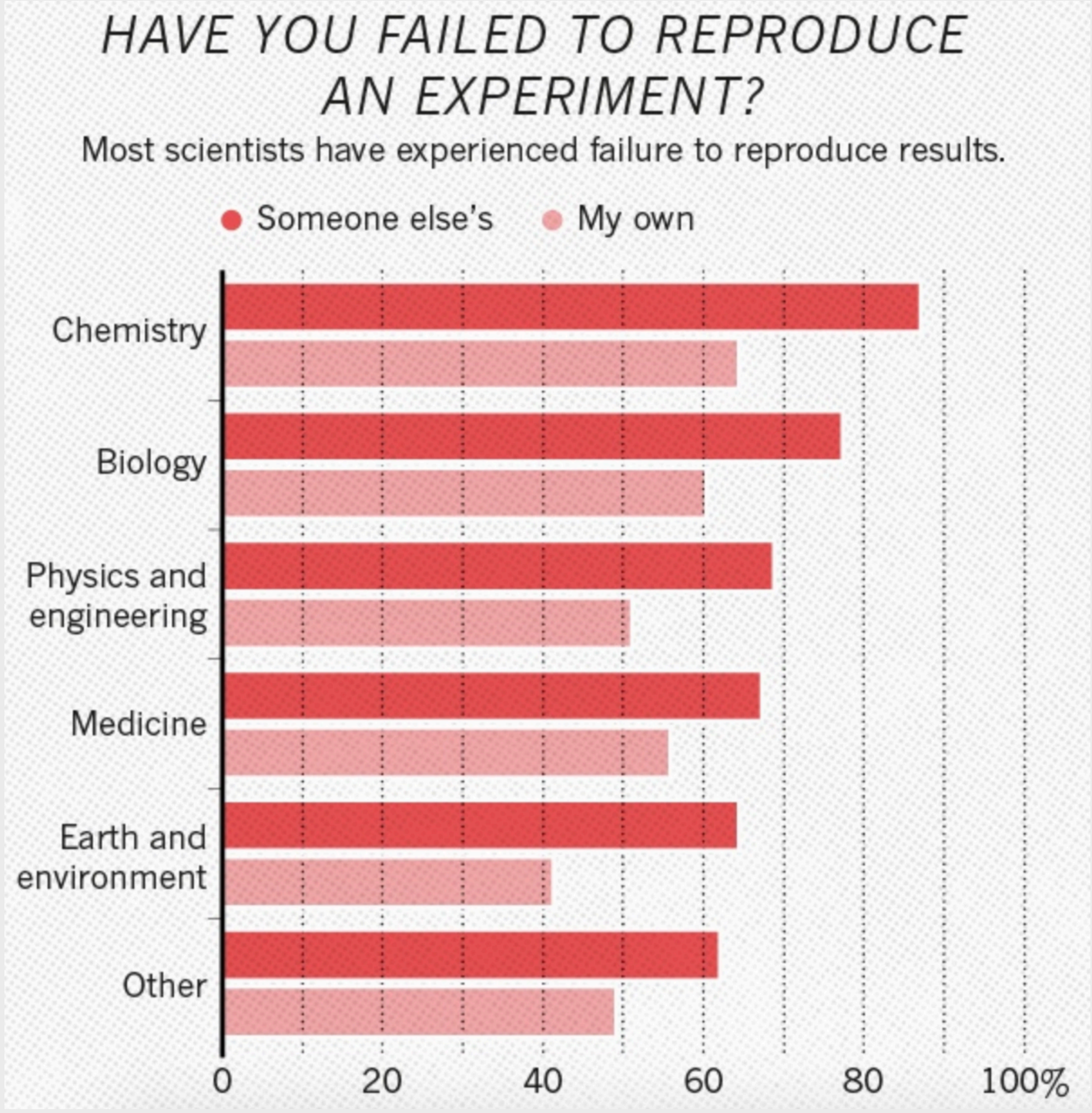
Results

We emailed corresponding authors in our sample to request the data and code associated with their articles and attempted to replicate the findings from a randomly chosen subset of the articles for which we received artifacts. We estimate the artifact recovery rate to be 44% with a 95% bootstrap confidence interval of the proportion $[0.36, 0.50]$, and we estimate the replication rate to be 26% with a 95% bootstrap confidence interval $[0.20, 0.32]$.

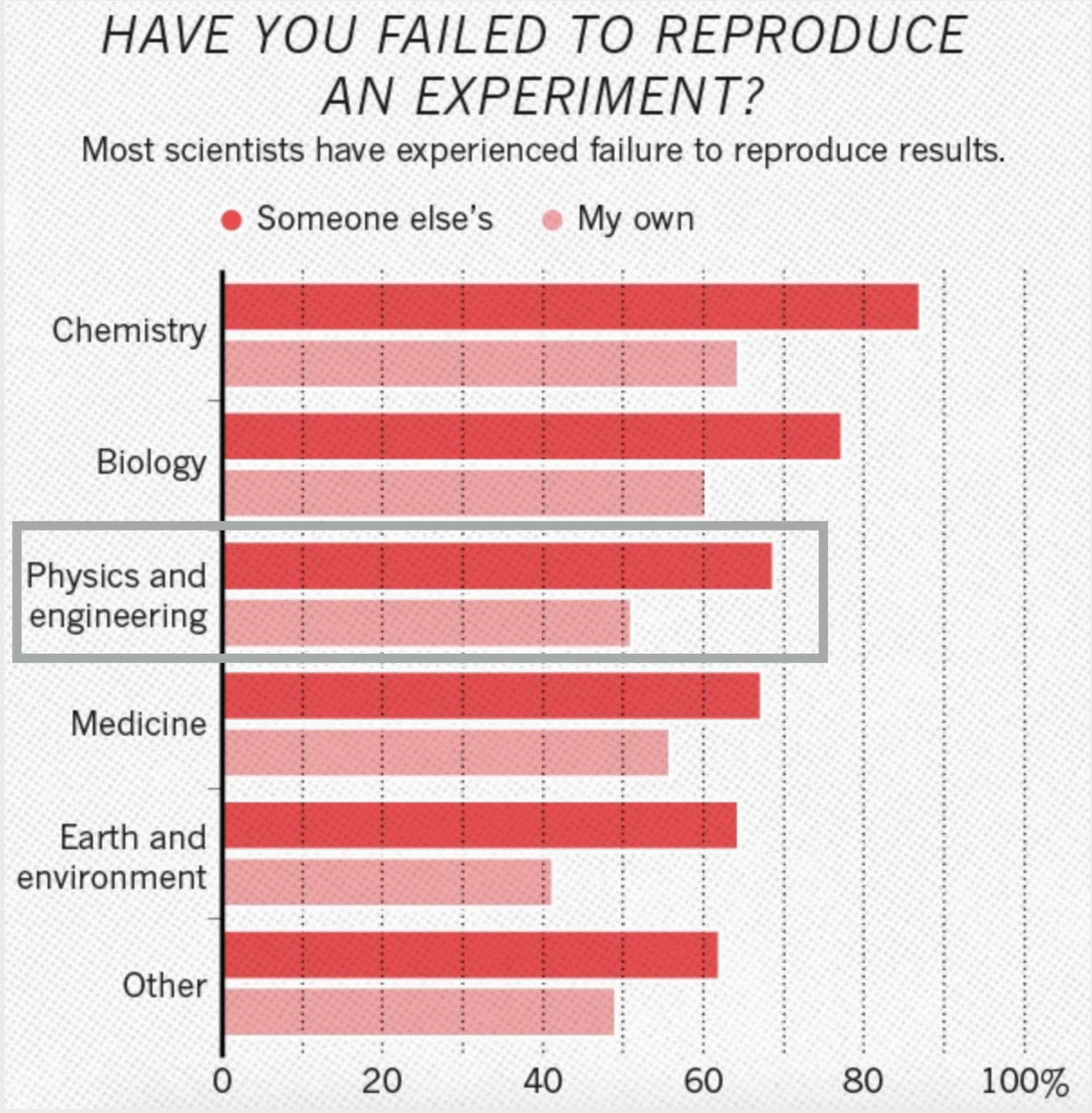
We emailed corresponding authors in our sample to request the data and code associated with their articles and attempted to replicate the findings from a randomly chosen subset of the articles for which we received artifacts. We estimate the artifact recovery rate to be 44% with a 95% bootstrap confidence interval of the proportion [0.36, 0.50], and we estimate the replication rate to be 26% with a 95% bootstrap confidence interval [0.20, 0.32].

- Stodden et al.

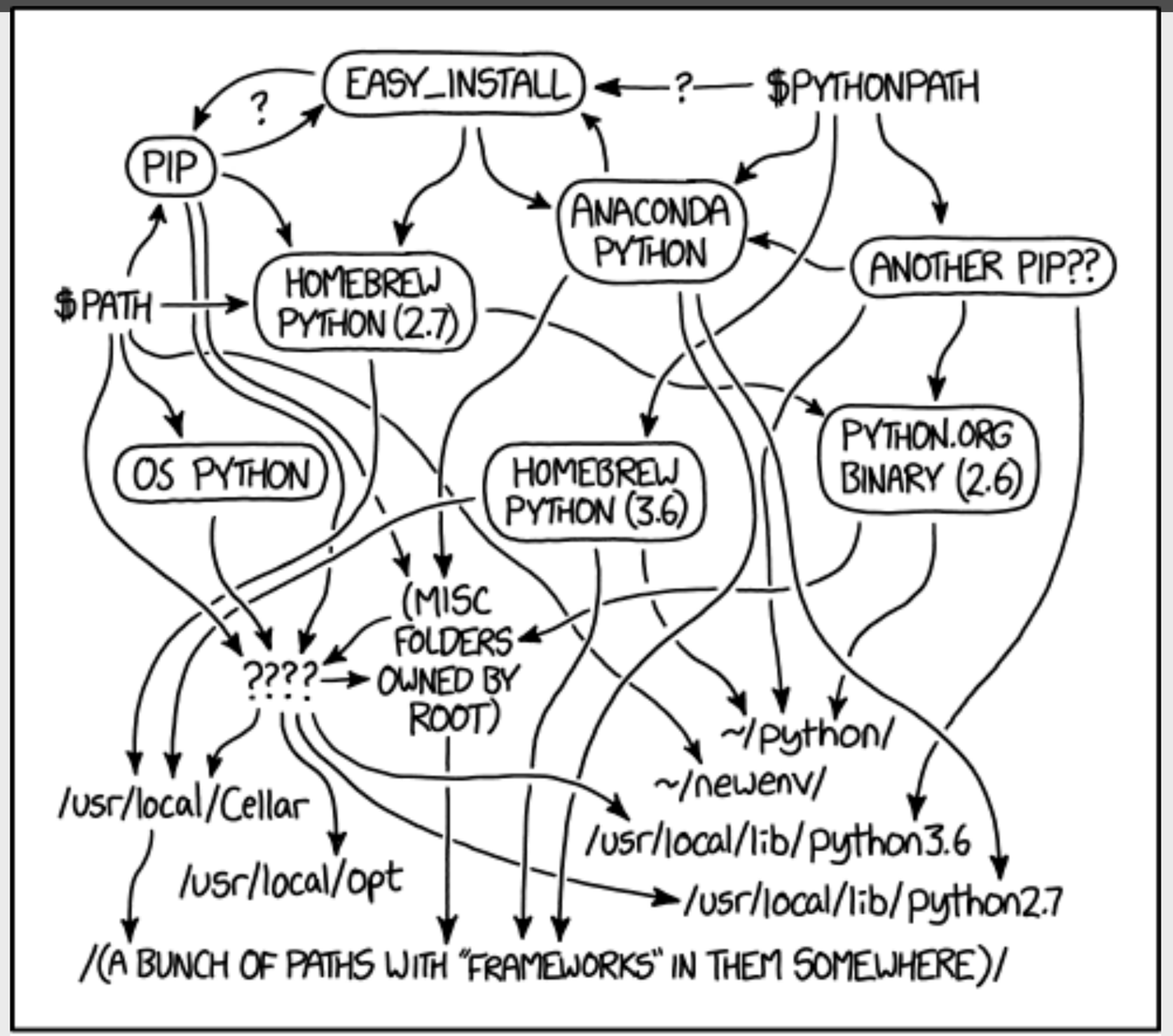
Moyna Baker, 2016 Nature (n=1,500)



Moyna Baker, 2016 Nature



GitHub repo's aren't enough



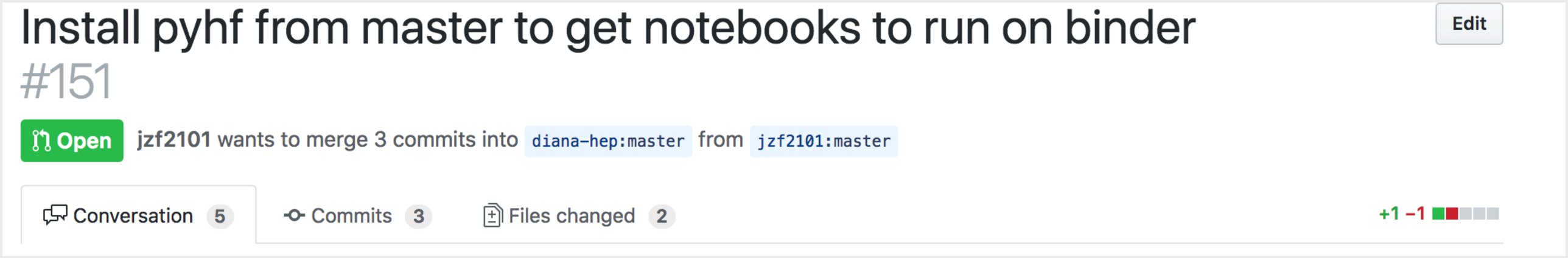
MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED
THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

https://xkcd.com/1987/

Report what you've actually discovered, clearly enough that someone else can discover it for themselves.

- James Somers

The maintainer’s mind



The maintainer’s mind

matthewfeickert commented 18 hours ago • edited ▾ Collaborator +😀

Hi **@jzf2101**, wow I can't tell you how happy this makes me to see someone from the Jupyter team fixing up the Binder env I wrote! 😄

I had also noticed this behavior (had to open up a terminal in Binder and run `python setup.py install` first) but I hadn't gotten around to fixing it yet. Thanks so much for doing this for me!

LGTM, so unless **@lukasheinrich** has any comments I think we can squash and merge this in.

lukasheinrich commented 18 hours ago Collaborator +😀

Thanks **@jzf2101** for the PR. I agree, the binder from the repo, should take the code from the repo, not from PyPI. I think perhaps we should install in "editable" mode e.g. `pip install -e .`

That way it should be possible to edit the pyhf code in-place, reload the python modules and use the edited version. Other than that this LGTM.

The scope of the problem

An empirical analysis of journal policy effectiveness for computational reproducibility

Victoria Stodden[a,1], **Jennifer Seiler**[b], **and Zhaokun Ma**[b]

[a]School of Information Sciences, University of Illinois at Urbana–Champaign, Champaign, IL 61820; and [b]Department of Statistics, Columbia University, New York, NY 10027

Edited by David B. Allison, Indiana University Bloomington, Bloomington, IN, and accepted by Editorial Board Member Susan T. Fiske January 9, 2018 (received for review July 11, 2017)

A key component of scientific communication is sufficient information for other researchers in the field to reproduce published findings. For computational and data-enabled research, this has often been interpreted to mean making available the raw data from which results were generated, the computer code that generated the findings, and any additional information needed such as workflows and input parameters. Many journals are revising author guidelines to include data and code availability. This work evaluates the effectiveness of journal policy that requires the data and code necessary for reproducibility be made available postpublication by the authors upon request. We assess the effectiveness of such a policy by (*i*) requesting data and code from authors and (*ii*) attempting replication of the published findings. We chose a random sample of 204 scientific papers published in the journal *Science* after the implementation of their policy in February 2011. We found that we were able to obtain artifacts from 44% of our sample and were able to reproduce the findings for 26%. We find this policy—author remission of data and code postpublication upon request—an improvement over no policy, but currently insufficient for reproducibility.

reproducible research | data access | code access | reproducibility policy | open science

putational reproducibility of published results. We use a survey instrument to test the availability of data and code for articles published in *Science* in 2011–2012. We then use the scientific communication standards from the 2012 Institute for Computational and Experimental Research in Mathematics (ICERM) workshop report to evaluate the reproducibility of articles for which artifacts were made available (11). We then assess the impact of the policy change directly, by examining articles published in *Science* in 2009–2010 and comparing artifact ability to our postpolicy sample from 2011–2012. Finally, we discuss possible improvements to journal policies for enabling reproducible computational research in light of our results.

Results

We emailed corresponding authors in our sample to request the data and code associated with their articles and attempted to replicate the findings from a randomly chosen subset of the articles for which we received artifacts. We estimate the artifact recovery rate to be 44% with a 95% bootstrap confidence interval of the proportion $[0.36, 0.50]$, and we estimate the replication rate to be 26% with a 95% bootstrap confidence interval $[0.20, 0.32]$.

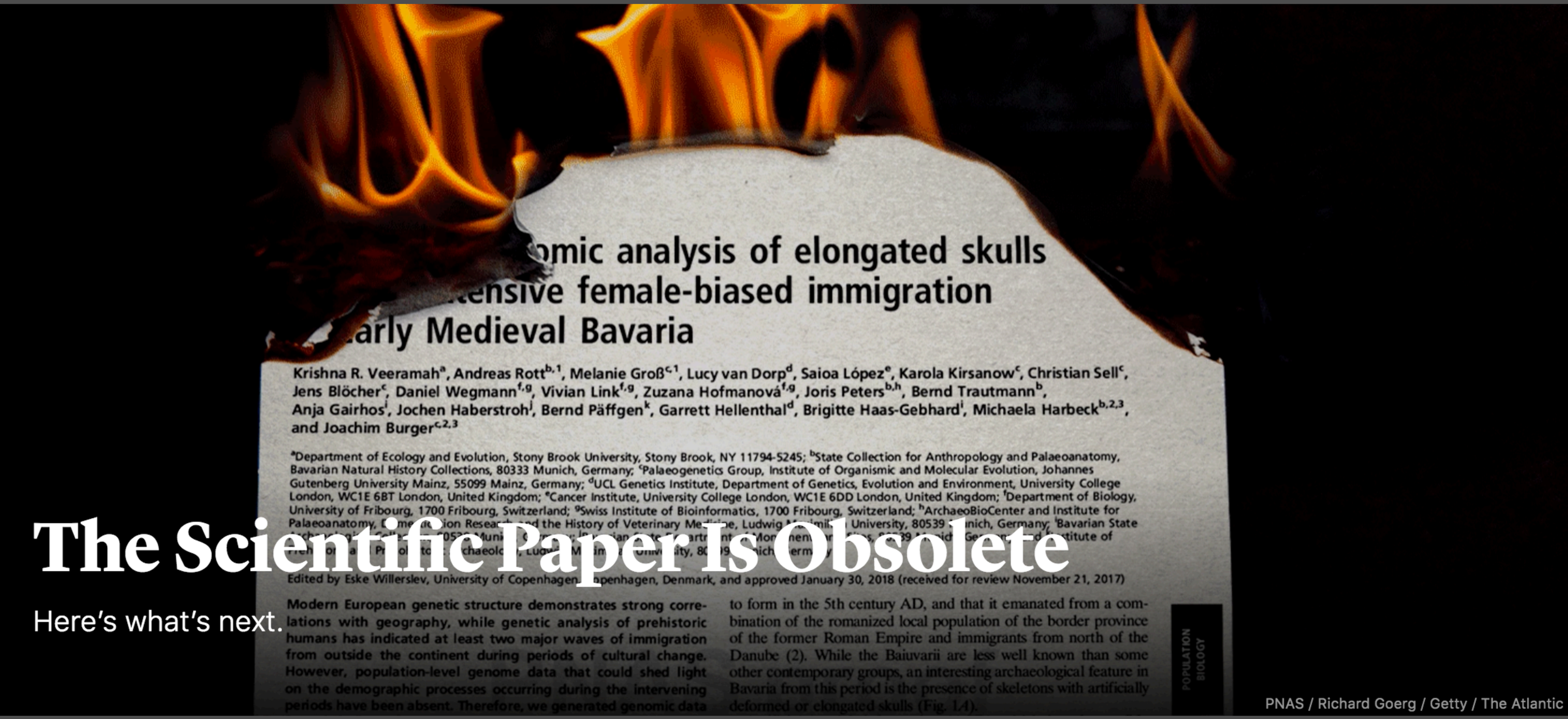
Scientific communication has become software engineering

Always code and comment in such a way that if someone a few notches junior picks up the code, they will take pleasure in reading and learning from it.

- Code for the Maintainer

As scientists, we are now all becoming open source maintainers

As scientists, we are now all becoming open source maintainers
Welcome to the club :D



The Scientific Paper Is Obsolete

Here's what's next.

PNAS / Richard Goerg / Getty / The Atlantic

Atlantic Monthly, April 5 2018

Here’s what’s next

BinderHub

A project to build and serve Binders

Watch 41

Table of Contents

5. BinderHub Deployments

BinderHub is open-source technology that can be deployed anywhere that Kubernetes is deployed. The Binder community hopes that it will be used for many applications in research and education. As new organizations adopt BinderHub, we'll update this page in order to provide inspiration to others who wish to do so.

If you or your organization has set up a BinderHub that isn't listed here, please open an issue on our GitHub repository to discuss adding it!

5.1. GESIS Institute for Social Sciences

Deployed on bare-metal using `kubeadm`.

- Deployment repository
- BinderHub / JupyterHub links

https://binderhub.readthedocs.io/

Create your own BinderHub

Zero to BinderHub

A guide to help you create your own BinderHub from scratch.

https://binderhub.readthedocs.io/

jupyterhub / **binderhub**

Unwatch 41 | Star 432 | Fork 65

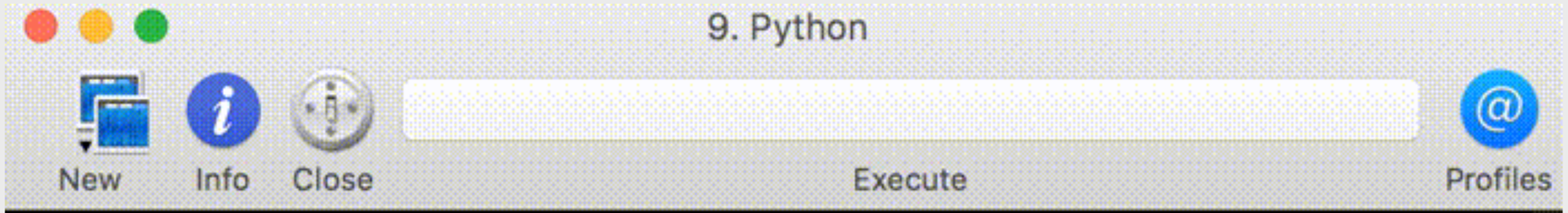
Code | Issues 112 | Pull requests 12 | Projects 1 | Wiki | Insights | Settings

Federation support #63

Edit | New issue

Open **yuvipanda** opened this issue on Jun 24, 2017 · 10 comments

Show, don’t tell




```
-bash-3.2$ ipython
WARNING: Attempting to work in a virtualenv. If you encounter problems, please
nstall IPython inside the virtualenv.
Python 2.7.10 (default, Sep 23 2015, 04:34:21)
Type "copyright", "credits" or "license" for more information.

IPython 4.0.2 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: from postal.expand import expand_address

In [2]:
```

https://machinelearnings.co/statistical-nlp-on-openstreetmap-b9d573e6cc86

dtak / rrr

Unwatch 5 | Star 17 | Fork 4

Code | Issues 2 | Pull requests 0 | Projects 0 | Wiki | Insights

Code/figures in Right for the Right Reasons https://www.ijcai.org/proceedings/201...

34 commits | 2 branches | 0 releases | 2 contributors | MIT

JupyterLab Alpha Preview

Secure | https://hub.mybinder.org/user/dtak-rrr-q7b8xg6s/lab

File Notebook Editor Terminal Console Help

Files Running Commands Cell Tools Tabs

Toy Colors.ipyn

Code Python 3

color_dataset_generator for more details.

We will train a multilayer perceptron to classify these images, explore which rule(s) it implicitly learns, and constrain it to use only one rule (or neither).

Let's first load our dataset:

In [2]:
```
X, Xt, y, yt = generate_dataset(cachefile='../data/toy-col
E1 = np.array([ignore_rule2 for _ in range(len(y))])
E2 = np.array([ignore_rule1 for _ in range(len(y))])
```

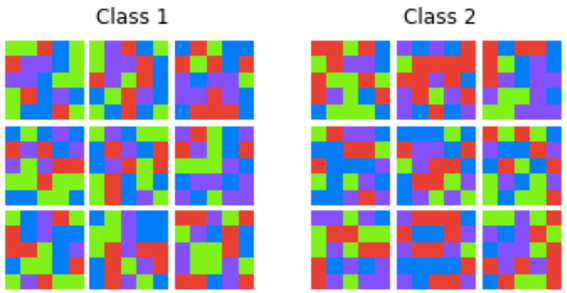
In [3]:
```
print(X.shape, Xt.shape, y.shape, yt.shape, E1.shape, E2.s
```

```
(20000, 75) (20000, 75) (20000,) (20000,) (20000, 75) (200
00, 75)
```

Understanding the Dataset

Let's just examine images from each class quickly and verify that in class 1, the corners are all the same color and the top-middle three pixels are all different (none of which should hold true in class 2):

In [4]:
```
plt.subplot(121)
plt.title('Class 1')
image_grid(X[np.argwhere(y == 0)[:9]], (5,5,3), 3)
plt.subplot(122)
plt.title('Class 2')
image_grid(X[np.argwhere(y == 1)[:9]], (5,5,3), 3)
plt.show()
```

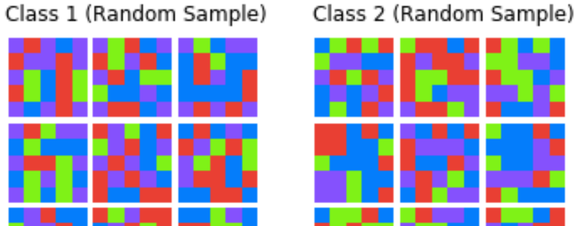


toy_colors.py

```
1  import numpy as np
2  import os
3
4  colors = {
5    'r': np.array([255,   0,   0], dtype=np.uint8),
6    'o': np.array([255, 128,   0], dtype=np.uint8),
7    'y': np.array([255, 255,   0], dtype=np.uint8),
8    'g': np.array([0,   255,   0], dtype=np.uint8),
9    'b': np.array([0,   128, 255], dtype=np.uint8),
10   'i': np.array([0,     0, 255], dtype=np.uint8),
11   'v': np.array([128,   0, 255], dtype=np.uint8)
12 }
13
14 imglen = 5
15 imgshape = (imglen, imglen, 3)
16 topleft = [0,[0]]
17 topright = [0,[imglen-1]]
18 botleft = [imglen-1,[0]]
19 botright = [imglen-1,[imglen-1]]
20
21 ignore_rule1 = np.zeros(imgshape)
```

Cell: Toy Colors

In [19]:
```
import numpy as np
plt.subplot(121)
plt.title('Class 1 (Random Sample)')
image_grid(X[np.argwhere(y == 0)\
  [np.random.choice(X[np.argwhere(y == 0)].shape[0], 9,
                    replace=True)]], (5,5,3), 3)
plt.subplot(122)
plt.title('Class 2 (Random Sample)')
image_grid(X[np.argwhere(y == 1)\
  [np.random.choice(X[np.argwhere(y == 1)].shape[0], 9,
                    replace=True)]], (5,5,3), 3)
```



What if?

JupyterLab Alpha Preview

Secure | https://hub.mybinder.org/user/dtak-rrr-zhr2fa2t/lab

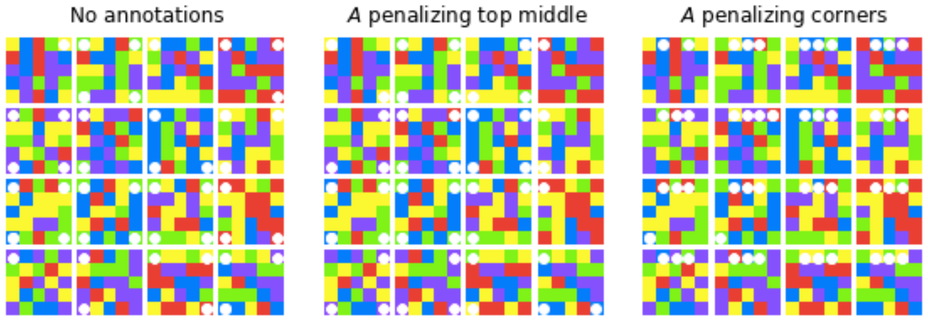
File Notebook Editor Terminal Console Help

Files Running Commands Cell Tools Tabs

Launcher | Toy Colors.ipyn

Code | Python 3

```
Out[7]: 0.9987000000000003

In [8]: # Train a model constrained to use the second rule
        mlp_rule2 = MultilayerPerceptron(l2_grads=1000)
        mlp_rule2.fit(X, y, E2)
        mlp_rule2.score(Xt, yt)

Out[8]: 0.98804999999999998

In [19]: # Visualize largest weights
         with figure_grid(1,3, rowwidth=8) as g:
           g.next()
           explain(mlp_plain, 'No annotations')
           g.next()
           explain(mlp_rule1, '$A$ penalizing top middle')
           g.next()
           explain(mlp_rule2, '$A$ penalizing corners')
```



Notice that when we explicitly penalize corners or the top middle, the model appears to learn the *other* rule perfectly. We haven't identified the pixels it does treat as significant in any way, but they are significant, so the fact that they show up in the explanations means that the explanation is probably an accurate reflection of the model's implicit logic.

When we don't have any annotations, the model does identify the top-middle pixels occasionally, suggesting it defaults to learning a heavily but not completely corner-weighted combination of the rules.

What happens when we forbid it from using either rule?

toy_colors.py

```
  'y': np.array([255, 255,   0], dtype=np.uint8),
  'g': np.array([0,   255,   0], dtype=np.uint8),
  'b': np.array([0,   128, 255], dtype=np.uint8),
  'i': np.array([0,     0, 255], dtype=np.uint8),
  'v': np.array([128,   0, 255], dtype=np.uint8)
}

imglen = 5
imgshape = (imglen, imglen, 3)
topleft = [0,[0]]
topright = [0,[imglen-1]]
botleft = [imglen-1,[0]]
botright = [imglen-1,[imglen-1]]

ignore_rule1 = np.zeros(imgshape)
for corner in [topleft, topright, botleft, botright]:
ignore_rule1[corner] = 1
ignore_rule1 = ignore_rule1.ravel().astype(bool)

ignore_rule2 = np.zeros(imgshape)
ignore_rule2[[0,[1,2,3]]] = 1
ignore_rule2 = ignore_rule2.ravel().astype(bool)

def random_color():
  return colors[np.random.choice(['r','g','b','v', 'y'])]

def Bern(p):
  return np.random.rand() < p

def any_repeats(row):
  n_unique = len(set(tuple(c) for c in row))
  return n_unique < len(row)

def ensure_class_0_rules_apply(img):
  # Rule 1
  img[topleft] = img[botright]
  img[topright] = img[botright]
  img[botleft] = img[botright]

  # Rule 2
  toprow = img[0]
  while any_repeats(toprow[1:-1]):
    toprow[1+np.random.choice(imglen-2)] = random_color()

def ensure_class_1_rules_apply(img):
```

Contribute, get involved

- jupyter.org
- Twitter:
  - @mybinderteam
  - @projectjupyter

Contribute, get involved

- GitHub:
  - github.com/jupyterhub
  - github.com/jzf2101
- We read issues!
- We mark issues for new contributors as ‘good first issue’ or ‘help wanted’
- github.com/jupyter/governance for code of conduct

Contribute, get involved

- Gitter
  - gitter.im/juptyerhub/jupyterhub
  - gitter.im/juptyerhub/binder
  - gitter.im/juptyer/jupyter

Thanks to all the maintainers

And the contributors too!

Acknowledgements

GORDON AND BETTY MOORE FOUNDATION

Alfred P. Sloan FOUNDATION

THE LEONA M. AND HARRY B. HELMSLEY CHARITABLE TRUST