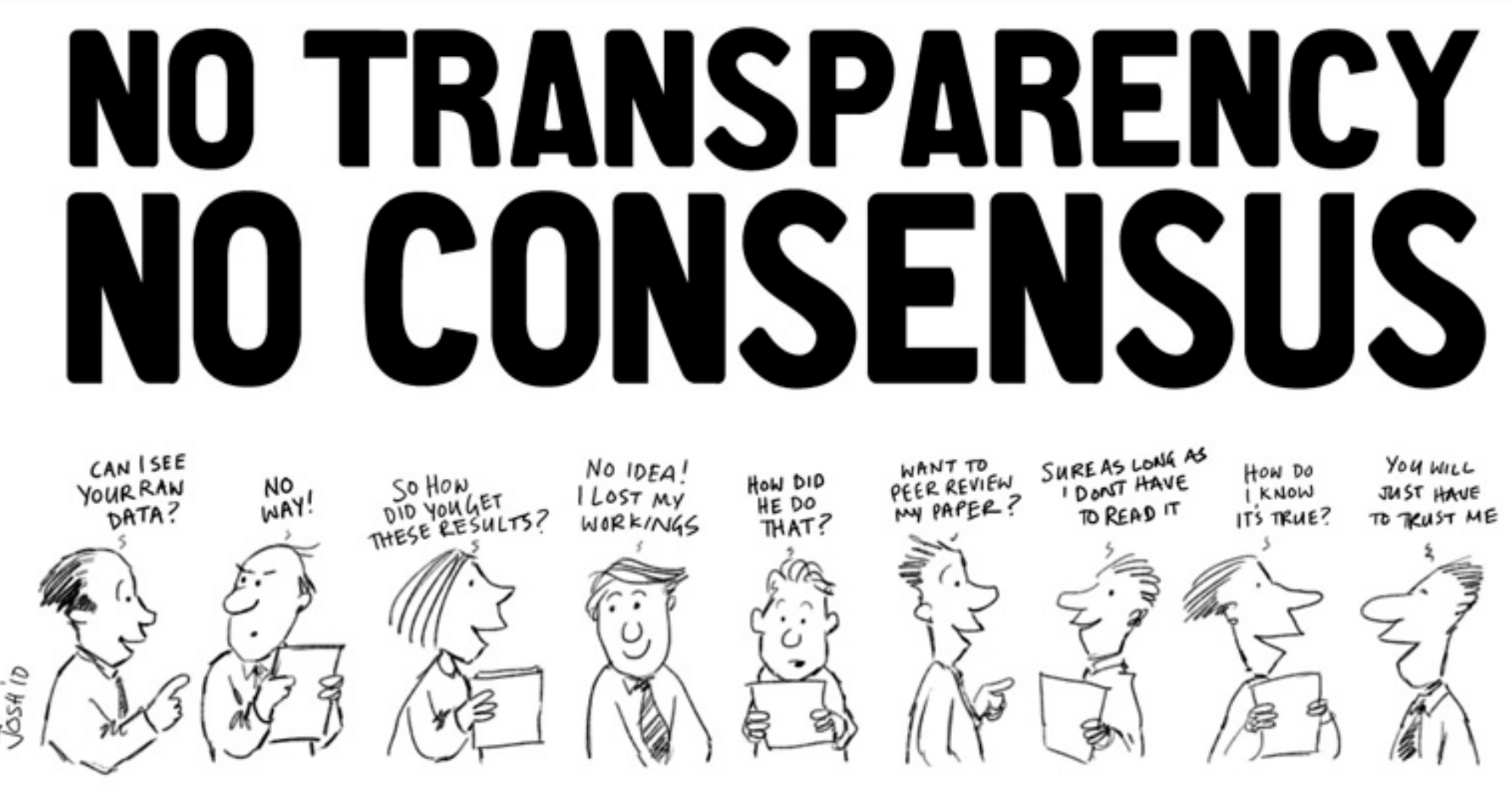
Reproducibilidad en Experimentos Computacionales

Lucas Mello Schnorr (with Arnaud Legrand)

ICI2ST Conference, November 23rd 2023

Scientific Consensus



Toward Open Science

Plan National pour la Science Ouverte (CoSO)

- France (CNRS, Inria, INRAE, . . .) but also Europe and US
- Many flavors: *Citizen Science* vs. *Ethics and Societal Responsibility*

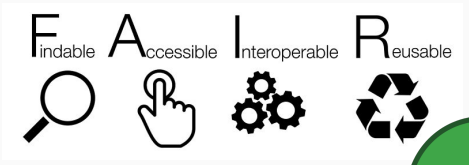
Toward Open Science

Plan National pour la Science Ouverte (CoSO)

- France (CNRS, Inria, INRAE, . . .) but also Europe and US
- Many flavors: *Citizen Science* vs. *Ethics and Societal Responsibility*

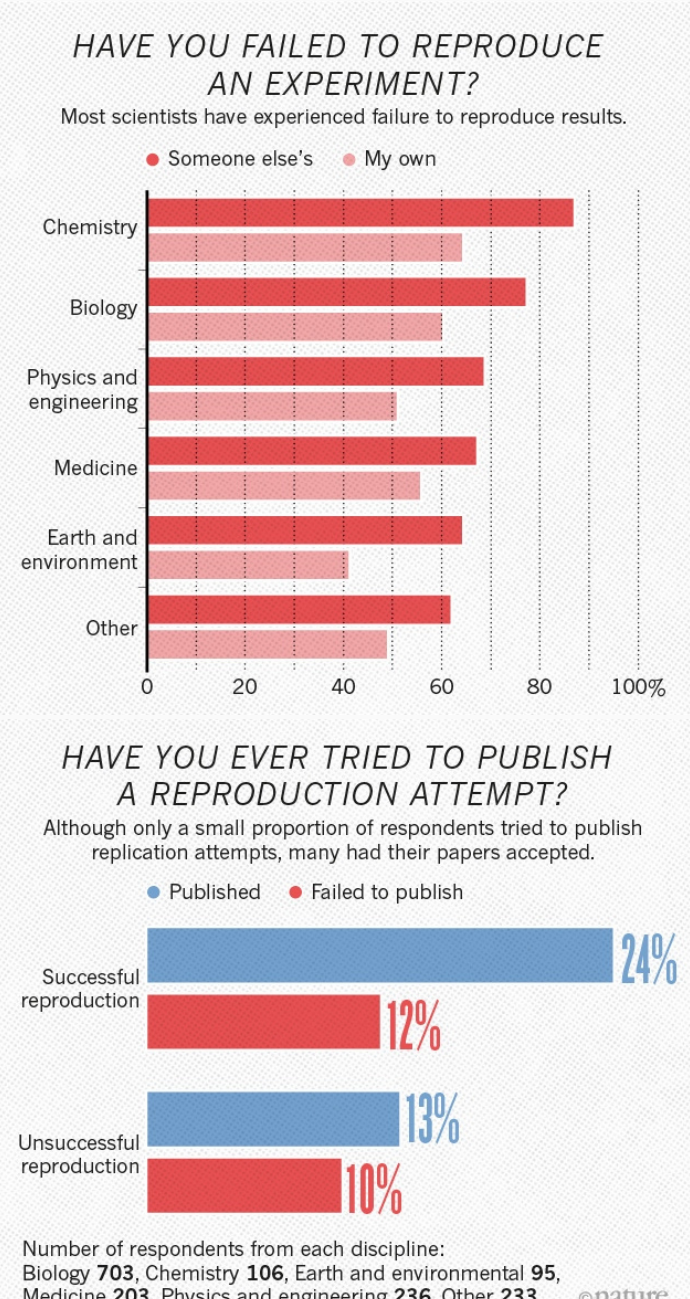
Main pillars:

1. Open access
2. Open data



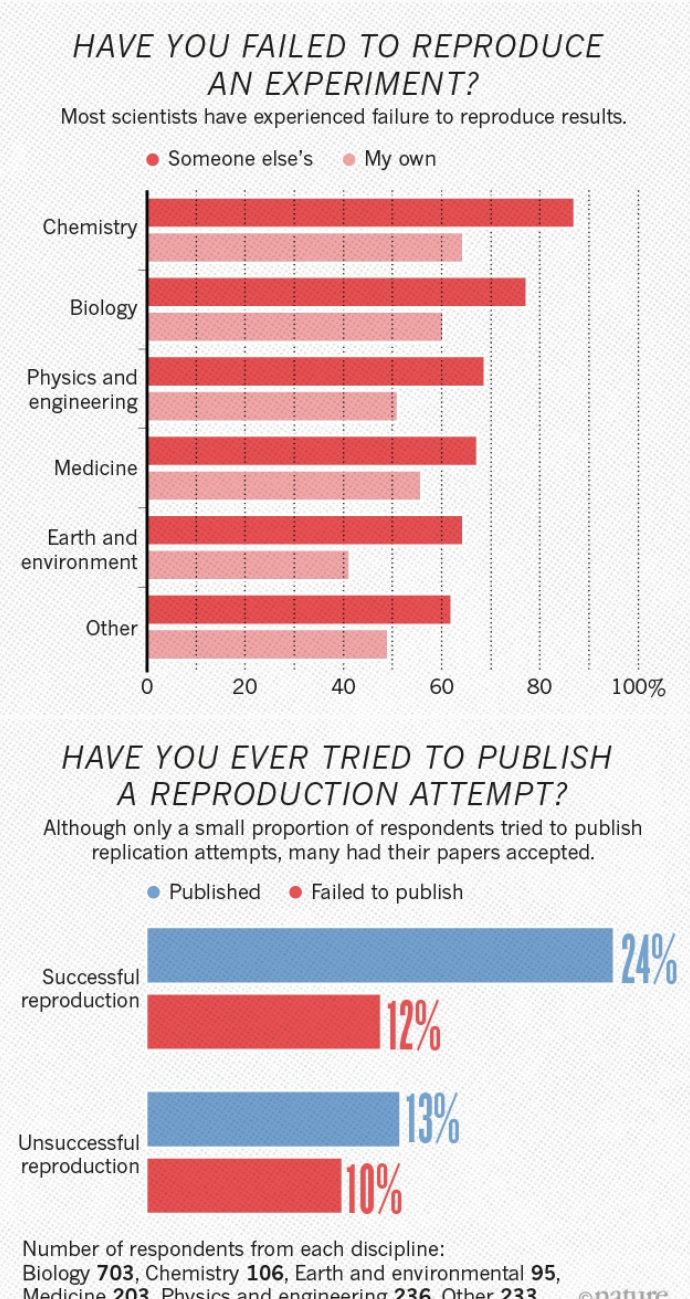
3. Open source
   - *Open hardware*
4. **Open methodology** (Reproducible Research)
   - *Open-notebook science*
   - *Open science infrastructures*
5. Open peer review
6. Open educational resources

Socio-technical Challenges



1,500 scientists lift the lid on reproducibility, Nature, May 2016

Socio-technical Challenges

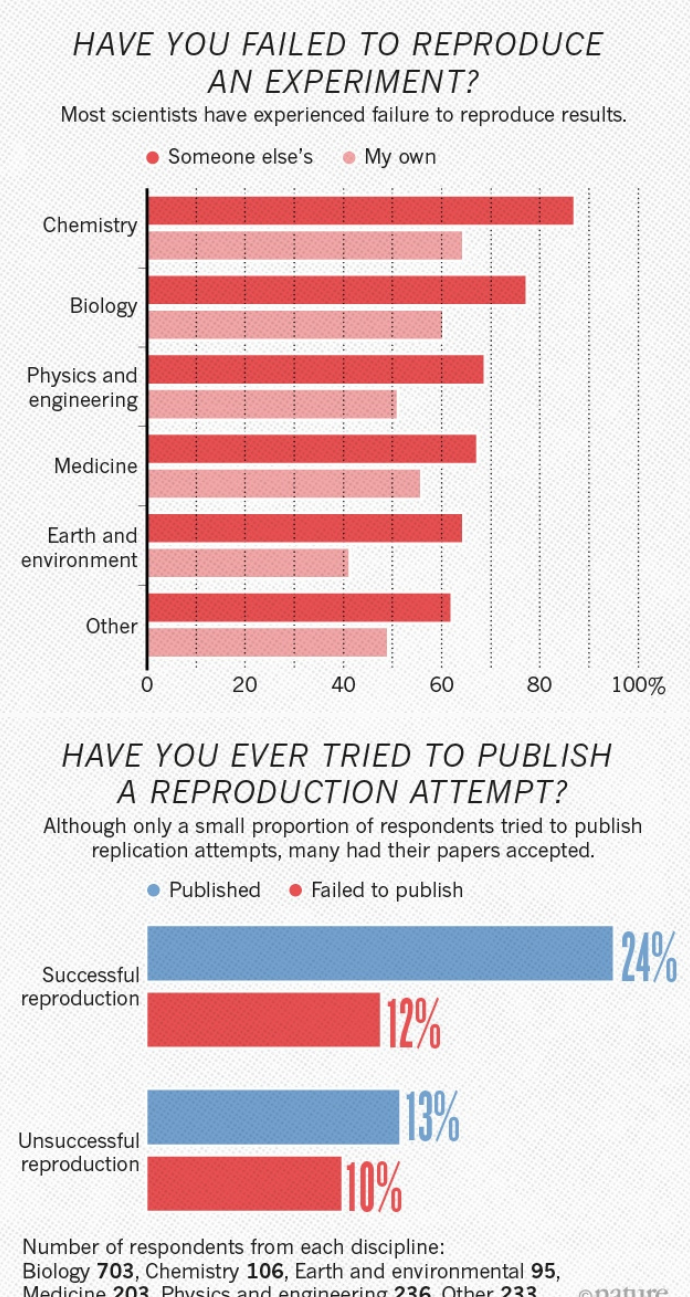


1,500 scientists lift the lid on reproducibility, Nature, May 2016

Social causes

- Fraud, conflict of interest (pharmaceutic, . . .)
- No incentive to reproduce/check our own work (afap), nor the work of others (big results!), nor to allow others to check (competition)
- Peer review does not scale: 1M+ articles per year!

Socio-technical Challenges

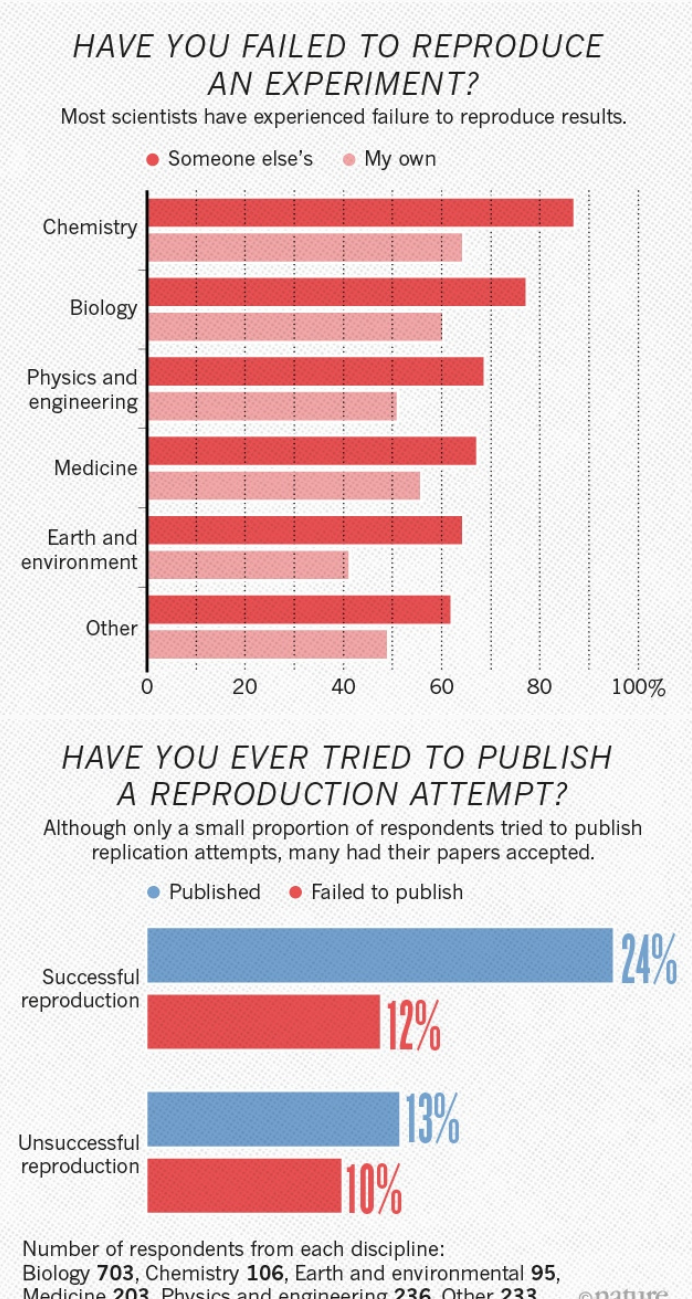


1,500 scientists lift the lid on reproducibility, Nature, May 2016

Social causes

- Fraud, conflict of interest (pharmaceutic, . . .)
- No incentive to reproduce/check our own work (afap), nor the work of others (big results!), nor to allow others to check (competition)
- Peer review does not scale: 1M+ articles per year!
- Emerging practices: DORA/Plan S/COARA, DMP and FAIR data, artefact evaluation, reproducibility badges, reproducibility challenges, open reviews, . . .

Socio-technical Challenges



1,500 scientists lift the lid on reproducibility, Nature, May 2016

Social causes

- Fraud, conflict of interest (pharmaceutic, . . .)
- No incentive to reproduce/check our own work (afap), nor the work of others (big results!), nor to allow others to check (competition)
- Peer review does not scale: 1M+ articles per year!
- Emerging practices: DORA/Plan S/COARA, DMP and FAIR data, artefact evaluation, reproducibility badges, reproducibility challenges, open reviews, . . .

Methodological/technical causes

- The many biases (apophenia, confirmation, hindsight, experimenter, . . .): bad designs
- Selective reporting, weak analysis (statistics, data manipulation mistakes, computational errors)
- Lack of information, code/raw data unavailable

Different Reproducibility Concerns in Modern Science

Biology, Oncology sample provenance, clinical trials $\rightsquigarrow$ standardized protocols

Psychology, Nutrition HARKING, p-hacking $\rightsquigarrow$ pre-registration

Different Reproducibility Concerns in Modern Science

Biology, Oncology sample provenance, clinical trials $\rightsquigarrow$ standardized protocols

Psychology, Nutrition HARKING, p-hacking $\rightsquigarrow$ pre-registration

Genomics software engineering, computational reproducibility, provenance

Computational fluid dynamics numerical chaos, parallel architectures

Different Reproducibility Concerns in Modern Science

Biology, Oncology sample provenance, clinical trials $\rightsquigarrow$ standardized protocols

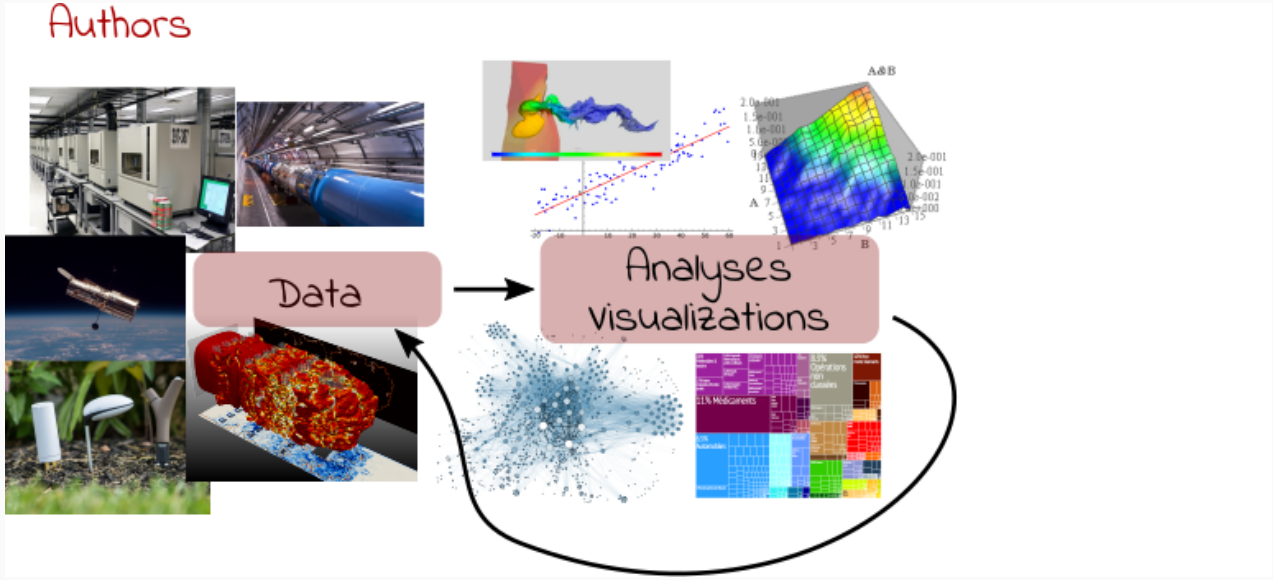
Psychology, Nutrition HARKING, p-hacking $\rightsquigarrow$ pre-registration

Genomics software engineering, computational reproducibility, provenance

Computational fluid dynamics numerical chaos, parallel architectures

Artificial Intelligence most of the above 😉

The processing steps between raw observations and findings have gotten increasingly numerous and complex



Different Reproducibility Concerns in Modern Science

Biology, Oncology sample provenance, clinical trials $\rightsquigarrow$ standardized protocols

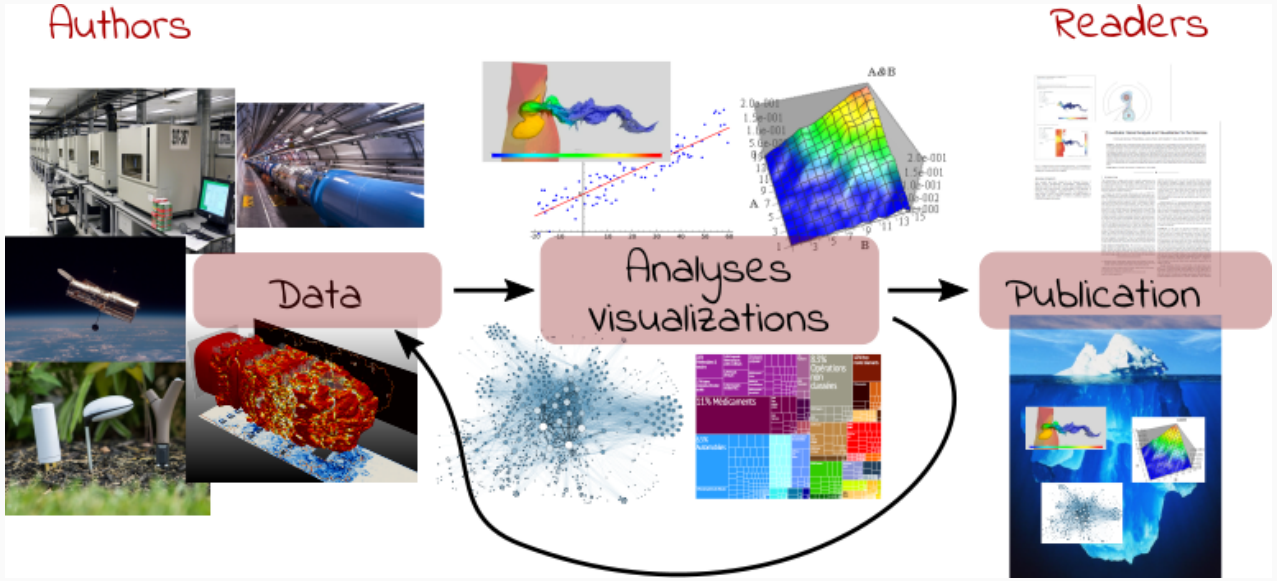
Psychology, Nutrition HARKING, p-hacking $\rightsquigarrow$ pre-registration

Genomics software engineering, computational reproducibility, provenance

Computational fluid dynamics numerical chaos, parallel architectures

Artificial Intelligence most of the above 😉

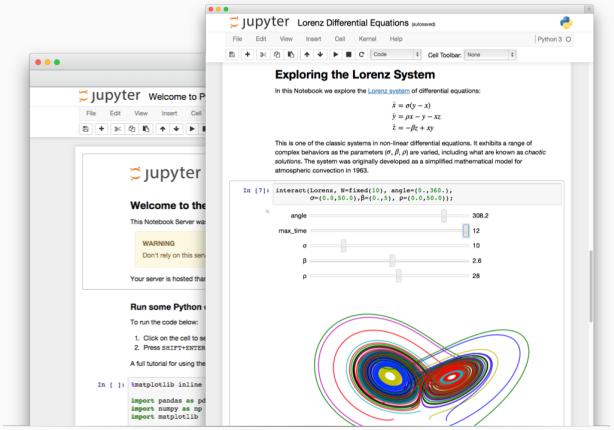
The processing steps between raw observations and findings have gotten increasingly numerous and complex



Reproducible Research = Bridging the Gap by working Transparently

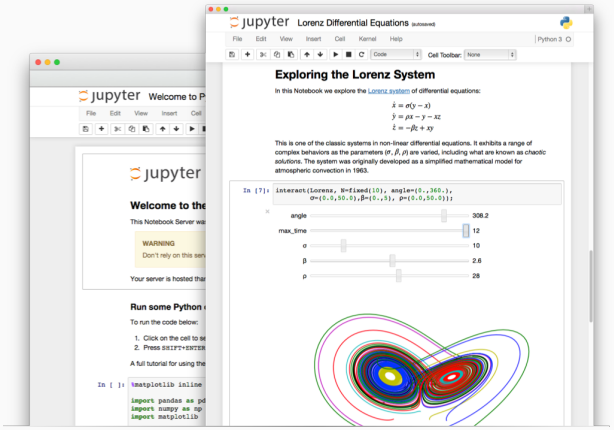
Reproducibility Issues Related to the use of Computers

Computation provenance: notebooks and workflows

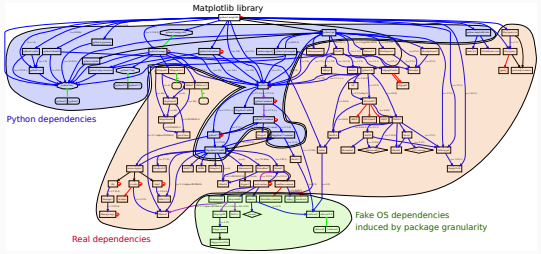


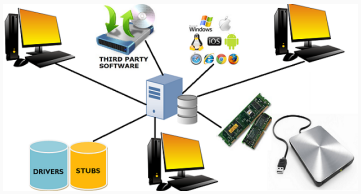
Reproducibility Issues Related to the use of Computers

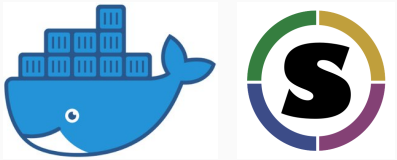
Computation provenance: notebooks and workflows



Software environments

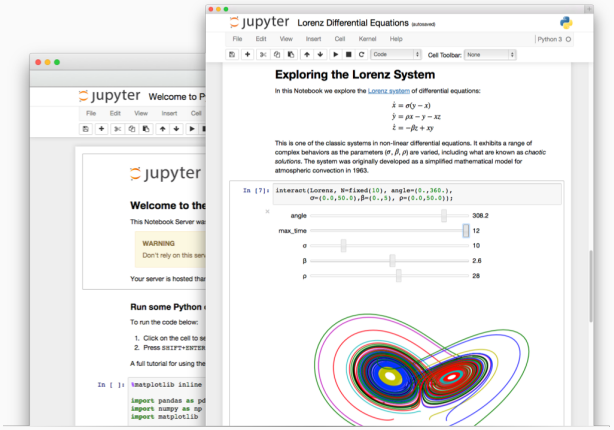




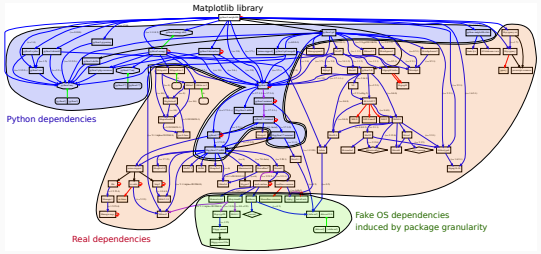


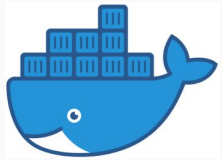
Reproducibility Issues Related to the use of Computers

Computation provenance: notebooks and workflows

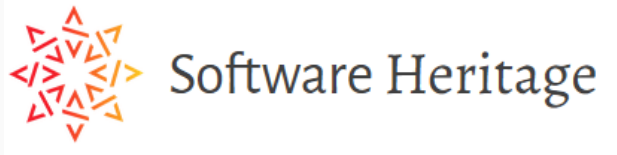


Software environments



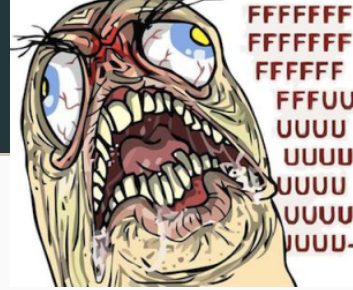


Sharing and Archiving



Good Practice #1
Taking Notes and Documenting

Frustration as an Author/Reviewer



Author

- I thought I used the same parameters but I'm getting different results!
- The new student wants to compare with the method I proposed last year
- My advisor asked me whether I took care of setting this or this but I can't remember
- The damned fourth reviewer asked for a major revision and wants me to change Figure 3. Which code and which data set did I use?
- It worked yesterday! 6 months later: Why did I do that?

Reviewer

- As usual, there is no confidence interval, I wonder about the variability and whether the difference is significant or not
- That can't be true, I'm sure they removed some points
- Why is this graph in logscale? How would it look like otherwise? I'm not even sure of what this value means. If only I could access the generation script

Tool 1: Computational Notebooks/Litterate Programming

Un document computationnel

Mon ordinateur m'indique que π vaut *approximativement*

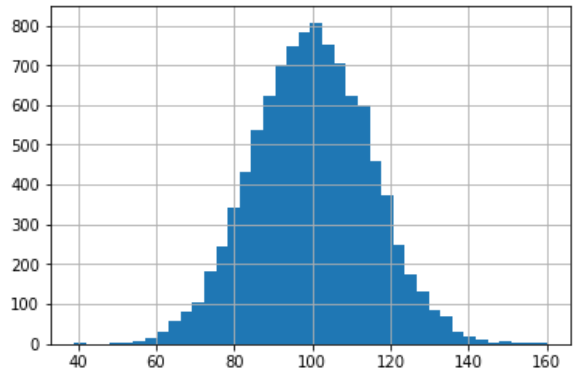
```
3.141592653589793
```

Mais calculé avec la **méthode** des aiguilles de Buffon, on obtiendrait comme **approximation** :

```
import numpy as np
N = 1000000
x = np.random.uniform(size=N, low=0, high=1)
theta = np.random.uniform(size=N, low=0, high=pi/2)
2/(sum((x+np.sin(theta))>1)/N)
```

```
3.1437198694098765
```

On peut inclure des formules mathématiques comme $\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$ et des *dessins qui n'ont rien à voir* avec π (si ce n'est une constante de normalisation... ☺).



Tool 1: Computational Notebooks/Litterate Programming

Document initial dans son environnement

jupyter example_pi (autosaved)

File Edit View Insert Cell Kernel Widgets Help Hide Code Python 3

Code CellToolbar

Hide Prompts Hide Code Hide Outputs

```
# Un document computationnel
```

Hide Prompts Hide Code Hide Outputs

```
Mon ordinateur m'indique que $\pi$ vaut *approximativement*
```

In [1]:

Hide Prompts Hide Code Hide Outputs

```
from math import *
print(pi)
```

```
3.141592653589793
```

Hide Prompts Hide Code Hide Outputs

```
Mais calculé avec la __méthode__ des [aiguilles de Buffon]
(https://fr.wikipedia.org/wiki/Aiguille_de_Buffon), on obtiendrait comme
__approximation__ :
```

In [2]:

Hide Prompts Hide Code Hide Outputs

```
import numpy as np
N = 1000000
x = np.random.uniform(size=N, low=0, high=1)
theta = np.random.uniform(size=N, low=0, high=pi/2)
2/(sum((x+np.sin(theta))>1)/N)
```

Out[2]: 3.1437198694098765

Hide Prompts Hide Code Hide Outputs

```
On peut inclure des formules mathématiques comme $\frac{1}{\sigma
\sqrt{2\pi}} \exp \left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$ et des
_dessins qui n'ont rien à voir_ avec $\pi$ (si ce n'est une constante de
normalisation... ☺).
```

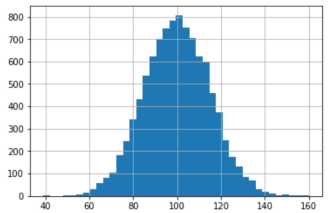
In [3]:

Hide Prompts Hide Code Hide Outputs

```
%matplotlib inline
import matplotlib.pyplot as plt

mu, sigma = 100, 15
x = mu + sigma*np.random.randn(10000)

plt.hist(x,40)
plt.grid(True)
plt.show()
```



Document final

Un document computationnel

Mon ordinateur m'indique que π vaut *approximativement*

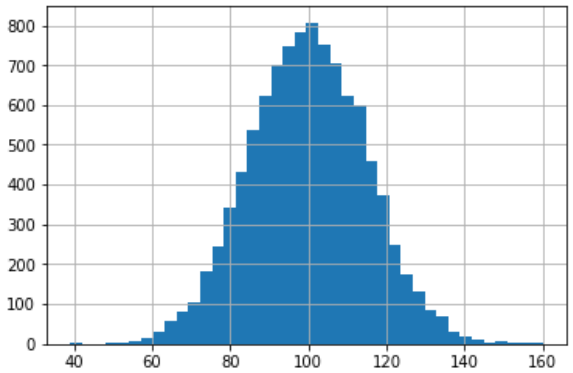
```
3.141592653589793
```

Mais calculé avec la **méthode** des aiguilles de Buffon, on obtiendrait comme **approximation** :

```
import numpy as np
N = 1000000
x = np.random.uniform(size=N, low=0, high=1)
theta = np.random.uniform(size=N, low=0, high=pi/2)
2/(sum((x+np.sin(theta))>1)/N)
```

```
3.1437198694098765
```

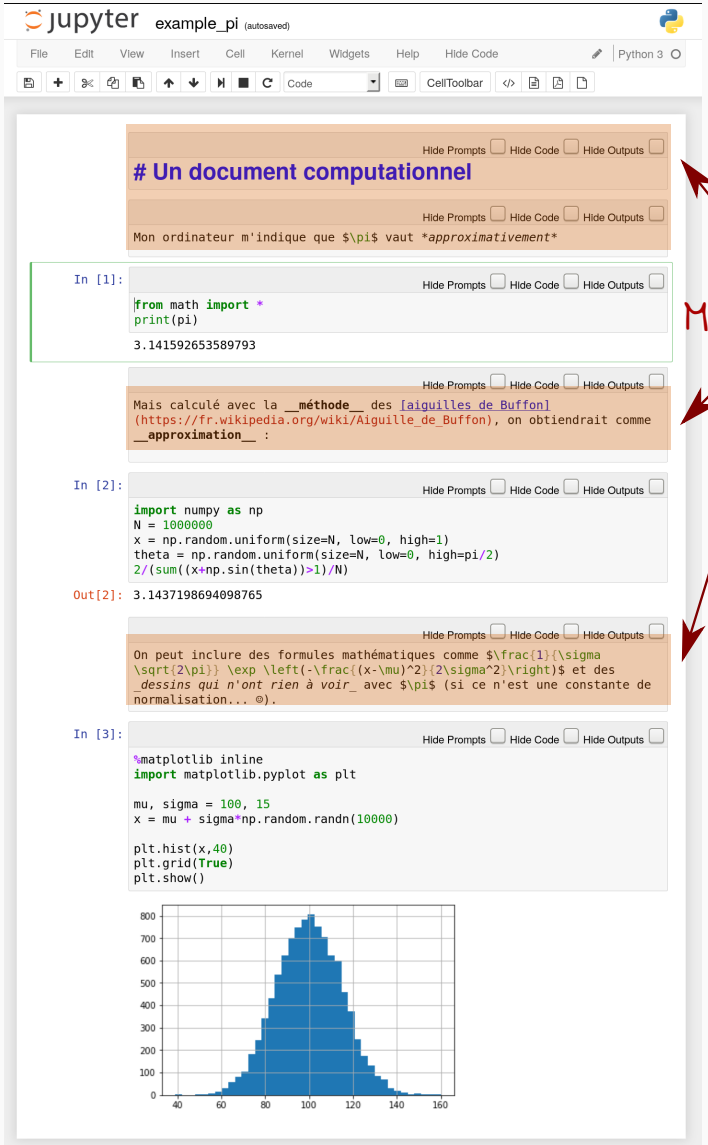
On peut inclure des formules mathématiques comme $\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$ et des *dessins qui n'ont rien à voir* avec π (si ce n'est une constante de normalisation... ☺).



Tool 1: Computational Notebooks/Litterate Programming

Document initial dans son environnement

Document final



MarkDown

Un document computationnel

Mon ordinateur m'indique que π vaut *approximativement*

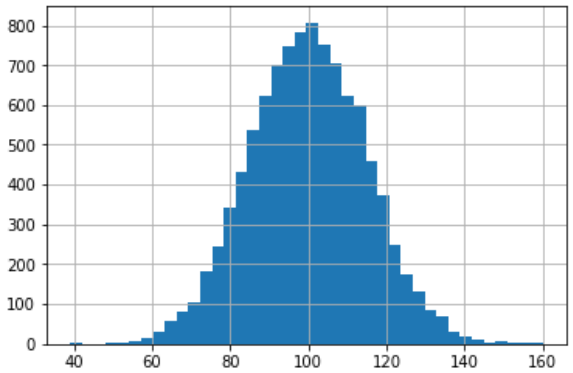
```
3.141592653589793
```

Mais calculé avec la **méthode** des aiguilles de Buffon, on obtiendrait comme **approximation** :

```
import numpy as np
N = 1000000
x = np.random.uniform(size=N, low=0, high=1)
theta = np.random.uniform(size=N, low=0, high=pi/2)
2/(sum((x+np.sin(theta))>1)/N)
```

```
3.1437198694098765
```

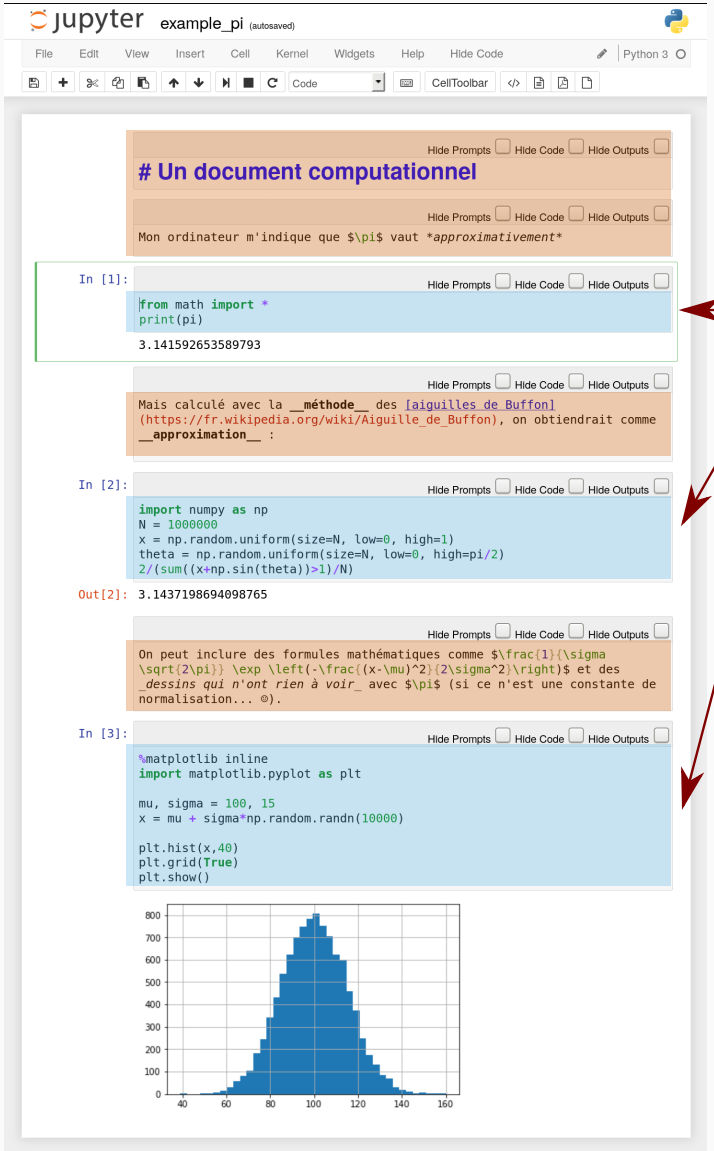
On peut inclure des formules mathématiques comme $\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$ et des *dessins qui n'ont rien à voir* avec π (si ce n'est une constante de normalisation... ☺).



Tool 1: Computational Notebooks/Litterate Programming

Document initial dans son environnement

Document final



Code

Un document computationnel

Mon ordinateur m'indique que π vaut *approximativement*

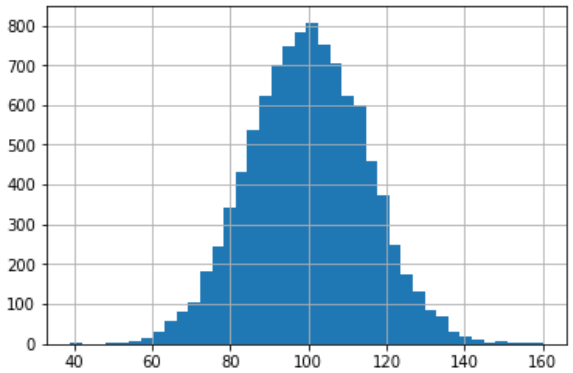
```
3.141592653589793
```

Mais calculé avec la **méthode** des aiguilles de Buffon, on obtiendrait comme **approximation** :

```
import numpy as np
N = 1000000
x = np.random.uniform(size=N, low=0, high=1)
theta = np.random.uniform(size=N, low=0, high=pi/2)
2/(sum((x+np.sin(theta))>1)/N)
```

```
3.1437198694098765
```

On peut inclure des formules mathématiques comme $\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$ et des *dessins qui n'ont rien à voir* avec π (si ce n'est une constante de normalisation... ☺).



Tool 1: Computational Notebooks/Litterate Programming

Document initial dans son environnement

Document final

jupyter example_pi (autosaved)

File Edit View Insert Cell Kernel Widgets Help Hide Code Python 3

Code CellToolbar

Hide Prompts Hide Code Hide Outputs

```
# Un document computationnel
```

Hide Prompts Hide Code Hide Outputs

```
Mon ordinateur m'indique que $\pi$ vaut *approximativement*
```

In [1]:

Hide Prompts Hide Code Hide Outputs

```
from math import *
print(pi)
```

```
3.141592653589793
```

Hide Prompts Hide Code Hide Outputs

```
Mais calculé avec la __méthode__ des [aiguilles de Buffon]
(https://fr.wikipedia.org/wiki/Aiguille_de_Buffon), on obtiendrait comme
__approximation__ :
```

In [2]:

Hide Prompts Hide Code Hide Outputs

```
import numpy as np
N = 1000000
x = np.random.uniform(size=N, low=0, high=1)
theta = np.random.uniform(size=N, low=0, high=pi/2)
2/(sum((x+np.sin(theta))>1)/N)
```

Out[2]: 3.1437198694098765

Hide Prompts Hide Code Hide Outputs

```
On peut inclure des formules mathématiques comme $\frac{1}{\sigma
\sqrt{2\pi}} \exp \left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$ et des
_dessins qui n'ont rien à voir_ avec $\pi$ (si ce n'est une constante de
normalisation... ☺).
```

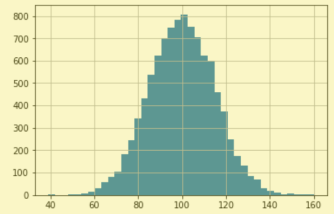
In [3]:

Hide Prompts Hide Code Hide Outputs

```
%matplotlib inline
import matplotlib.pyplot as plt

mu, sigma = 100, 15
x = mu + sigma*np.random.randn(10000)

plt.hist(x,40)
plt.grid(True)
plt.show()
```



Résultats

Un document computationnel

Mon ordinateur m'indique que π vaut *approximativement*

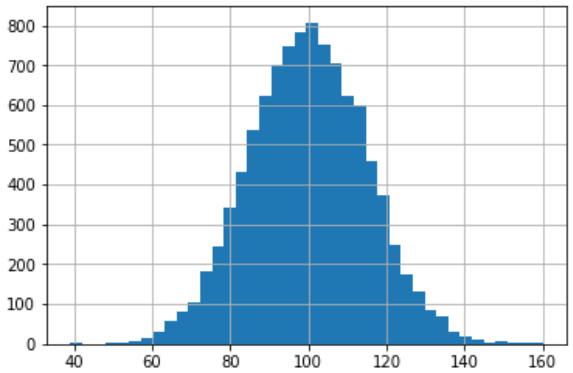
```
3.141592653589793
```

Mais calculé avec la **méthode** des aiguilles de Buffon, on obtiendrait comme **approximation** :

```
import numpy as np
N = 1000000
x = np.random.uniform(size=N, low=0, high=1)
theta = np.random.uniform(size=N, low=0, high=pi/2)
2/(sum((x+np.sin(theta))>1)/N)
```

```
3.1437198694098765
```

On peut inclure des formules mathématiques comme $\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$ et des *dessins qui n'ont rien à voir* avec π (si ce n'est une constante de normalisation... ☺).



Tool 1: Computational Notebooks/Litterate Programming

Document initial dans son environnement

jupyter example_pi (autosaved)

File Edit View Insert Cell Kernel Widgets Help Hide Code Python 3

Code CellToolbar

Hide Prompts Hide Code Hide Outputs

```
# Un document computationnel
```

Hide Prompts Hide Code Hide Outputs

```
Mon ordinateur m'indique que $\pi$ vaut *approximativement*
```

In [1]:

Hide Prompts Hide Code Hide Outputs

```
from math import *
print(pi)
```

```
3.141592653589793
```

Hide Prompts Hide Code Hide Outputs

```
Mais calculé avec la __méthode__ des [aiguilles de Buffon]
(https://fr.wikipedia.org/wiki/Aiguille_de_Buffon), on obtiendrait comme
__approximation__ :
```

In [2]:

Hide Prompts Hide Code Hide Outputs

```
import numpy as np
N = 1000000
x = np.random.uniform(size=N, low=0, high=1)
theta = np.random.uniform(size=N, low=0, high=pi/2)
2/(sum((x+np.sin(theta))>1)/N)
```

Out[2]: 3.1437198694098765

Hide Prompts Hide Code Hide Outputs

```
On peut inclure des formules mathématiques comme $\frac{1}{\sigma
\sqrt{2\pi}} \exp \left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$ et des
_dessins qui n'ont rien à voir_ avec $\pi$ (si ce n'est une constante de
normalisation... ☺).
```

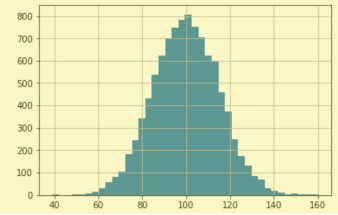
In [3]:

Hide Prompts Hide Code Hide Outputs

```
%matplotlib inline
import matplotlib.pyplot as plt

mu, sigma = 100, 15
x = mu + sigma*np.random.randn(10000)

plt.hist(x,40)
plt.grid(True)
plt.show()
```



Export

Document final

Un document computationnel

Mon ordinateur m'indique que π vaut *approximativement*

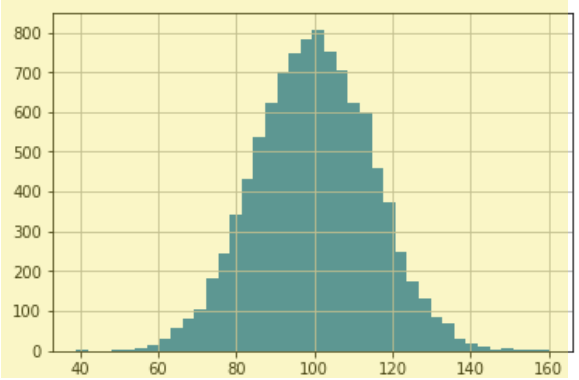
```
3.141592653589793
```

Mais calculé avec la **méthode** des aiguilles de Buffon, on obtiendrait comme **approximation** :

```
import numpy as np
N = 1000000
x = np.random.uniform(size=N, low=0, high=1)
theta = np.random.uniform(size=N, low=0, high=pi/2)
2/(sum((x+np.sin(theta))>1)/N)
```

```
3.1437198694098765
```

On peut inclure des formules mathématiques comme $\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$ et des *dessins qui n'ont rien à voir* avec π (si ce n'est une constante de normalisation... ☺).



Tool 1: Computational Notebooks/Litterate Programming

Document initial dans son environnement

jupyter example_pi (autosaved)

File Edit View Insert Cell Kernel Widgets Help Hide Code Python 3

Code CellToolbar

Hide Prompts Hide Code Hide Outputs

```
# Un document computationnel
```

Hide Prompts Hide Code Hide Outputs

```
Mon ordinateur m'indique que $\pi$ vaut *approximativement*
```

In [1]:

Hide Prompts Hide Code Hide Outputs

```
from math import *
print(pi)
```

```
3.141592653589793
```

Hide Prompts Hide Code Hide Outputs

```
Mais calculé avec la __méthode__ des [aiguilles de Buffon]
(https://fr.wikipedia.org/wiki/Aiguille_de_Buffon), on obtiendrait comme
__approximation__ :
```

In [2]:

Hide Prompts Hide Code Hide Outputs

```
import numpy as np
N = 1000000
x = np.random.uniform(size=N, low=0, high=1)
theta = np.random.uniform(size=N, low=0, high=pi/2)
2/(sum((x+np.sin(theta))>1)/N)
```

Out[2]: 3.1437198694098765

Hide Prompts Hide Code Hide Outputs

```
On peut inclure des formules mathématiques comme $\frac{1}{\sigma
\sqrt{2\pi}} \exp \left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$ et des
_dessins qui n'ont rien à voir_ avec $\pi$ (si ce n'est une constante de
normalisation... ☺).
```

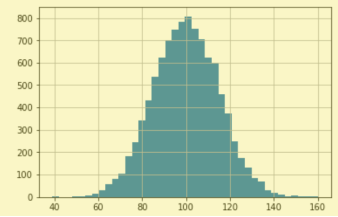
In [3]:

Hide Prompts Hide Code Hide Outputs

```
%matplotlib inline
import matplotlib.pyplot as plt

mu, sigma = 100, 15
x = mu + sigma*np.random.randn(10000)

plt.hist(x,40)
plt.grid(True)
plt.show()
```



Export

Document final

Un document computationnel

Mon ordinateur m'indique que π vaut *approximativement*

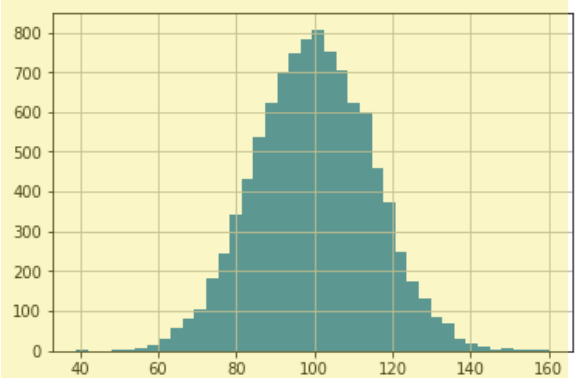
```
3.141592653589793
```

Mais calculé avec la **méthode** des aiguilles de Buffon, on obtiendrait comme **approximation** :

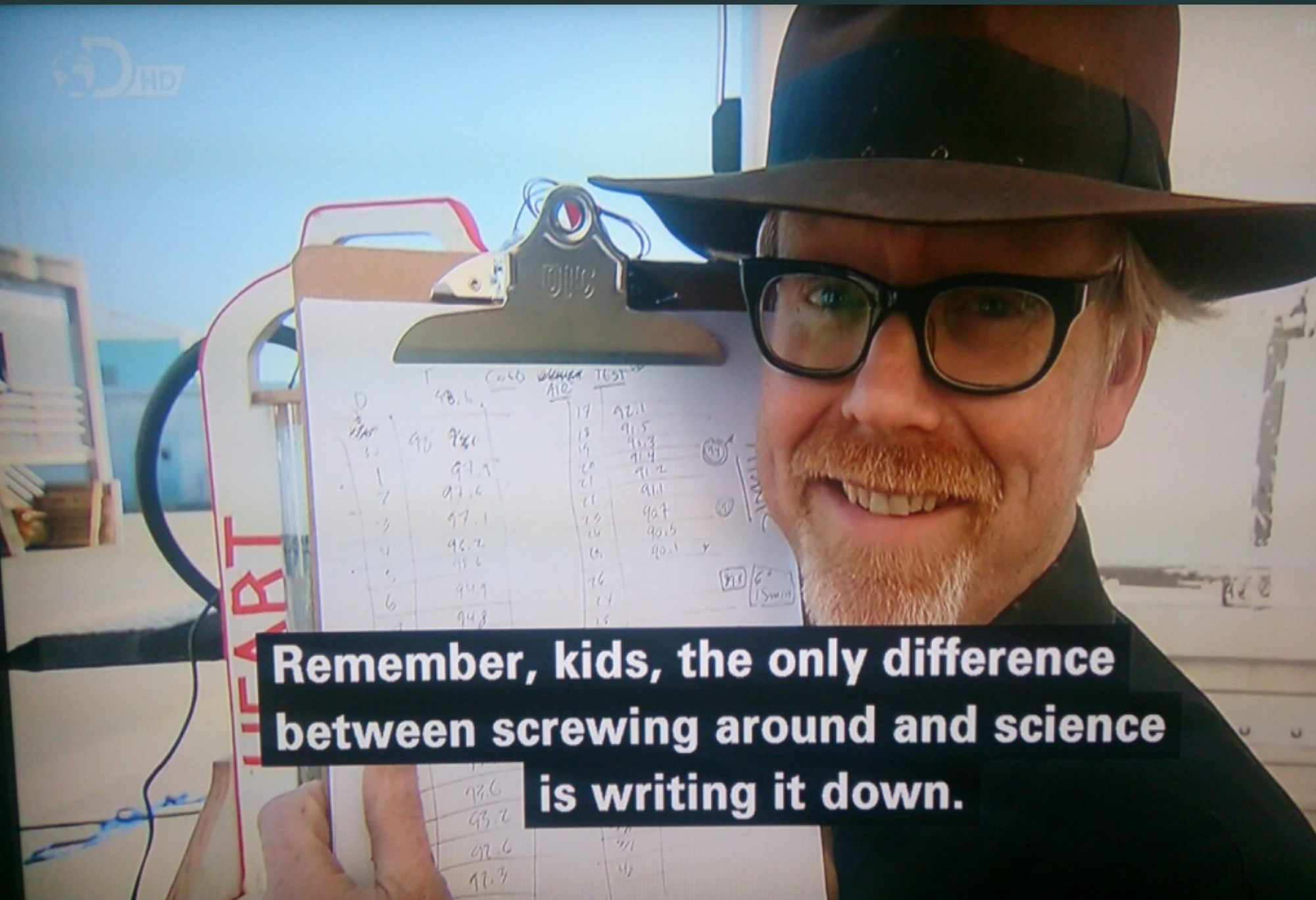
```
import numpy as np
N = 1000000
x = np.random.uniform(size=N, low=0, high=1)
theta = np.random.uniform(size=N, low=0, high=pi/2)
2/(sum((x+np.sin(theta))>1)/N)
```

```
3.1437198694098765
```

On peut inclure des formules mathématiques comme $\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$ et des *dessins qui n'ont rien à voir* avec π (si ce n'est une constante de normalisation... ☺).

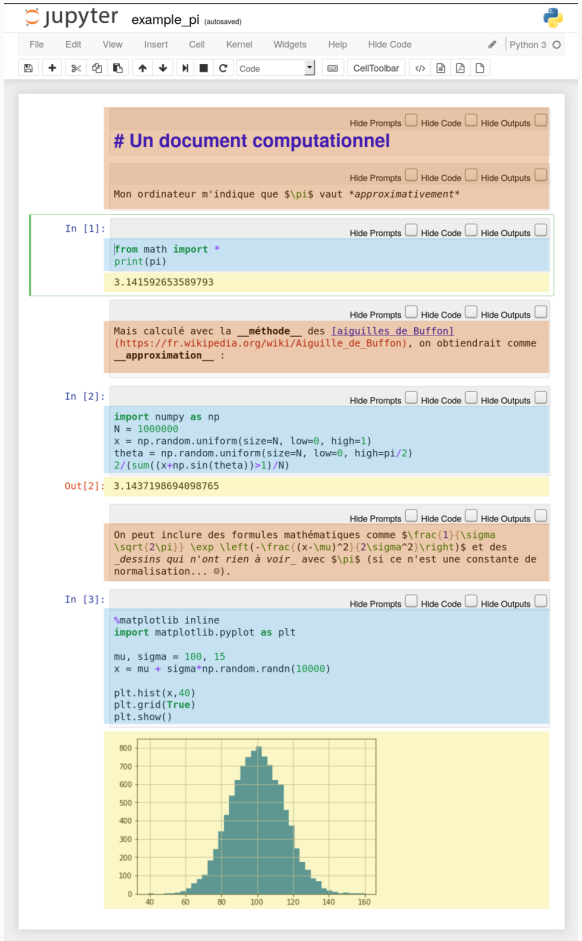


Tool 1bis: Lab. Notebooks, Computational Documents



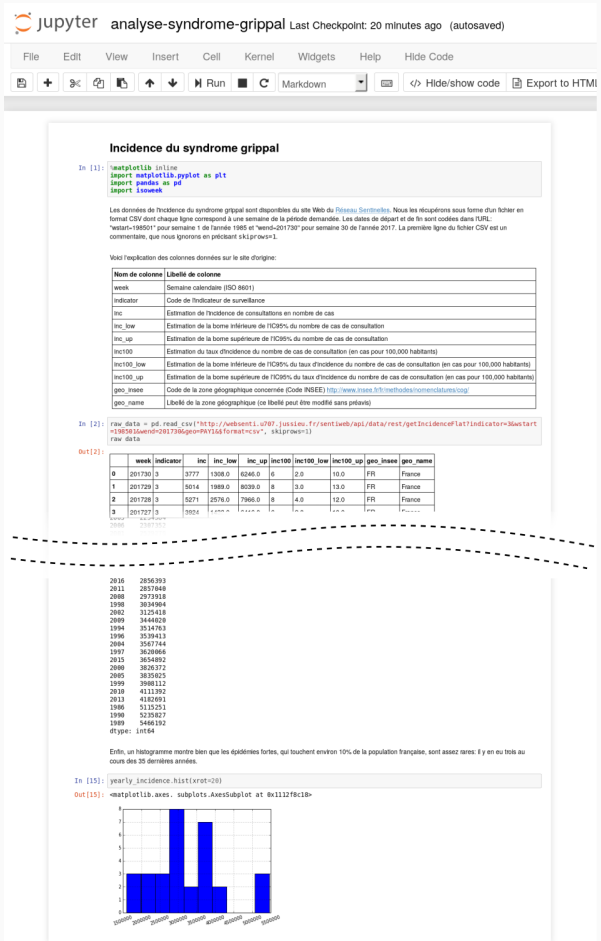
Tool 1ter: Workflows

Notebooks are no panacea and do not help developing clean code



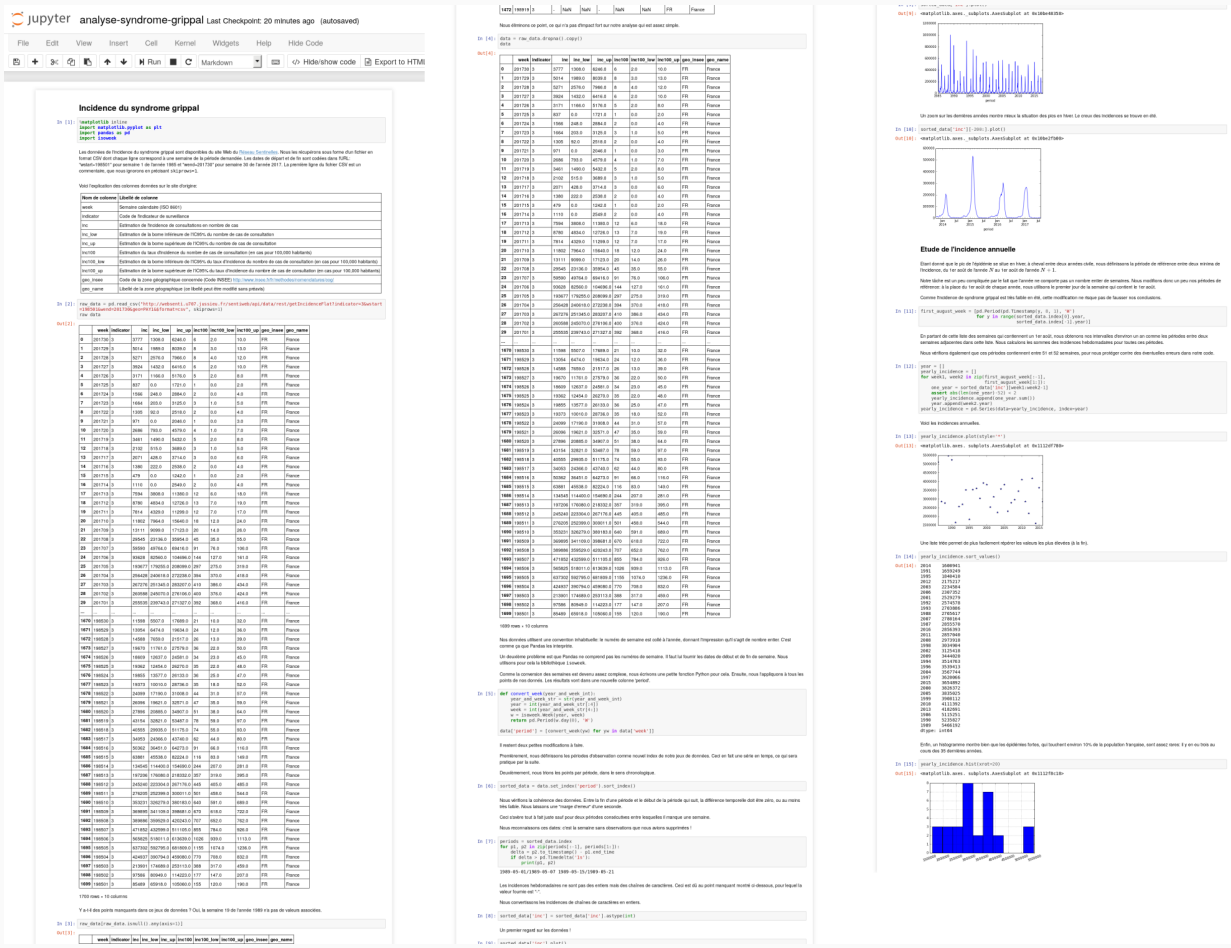
Tool 1ter: Workflows

Notebooks are no panacea and do not help developing clean code



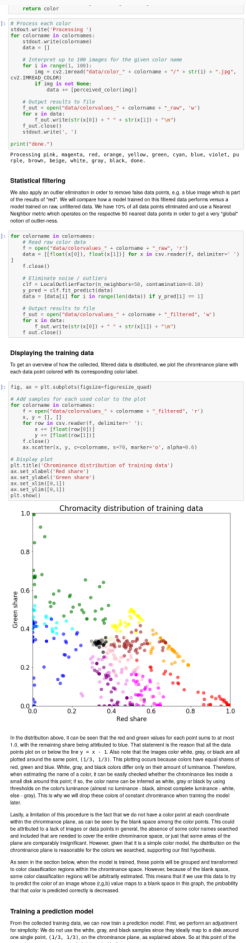
Tool 1ter: Workflows

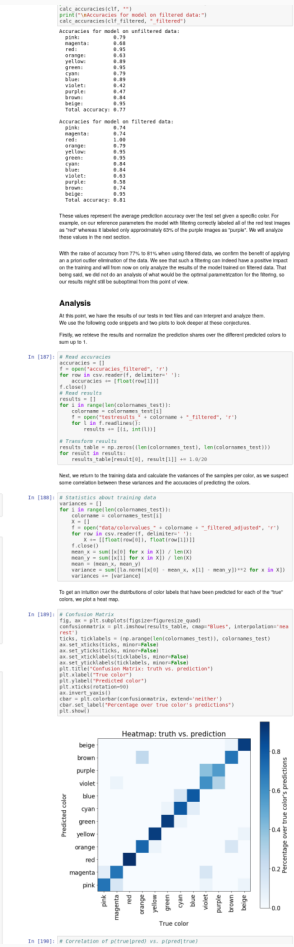
Notebooks are no panacea and do not help developing clean code



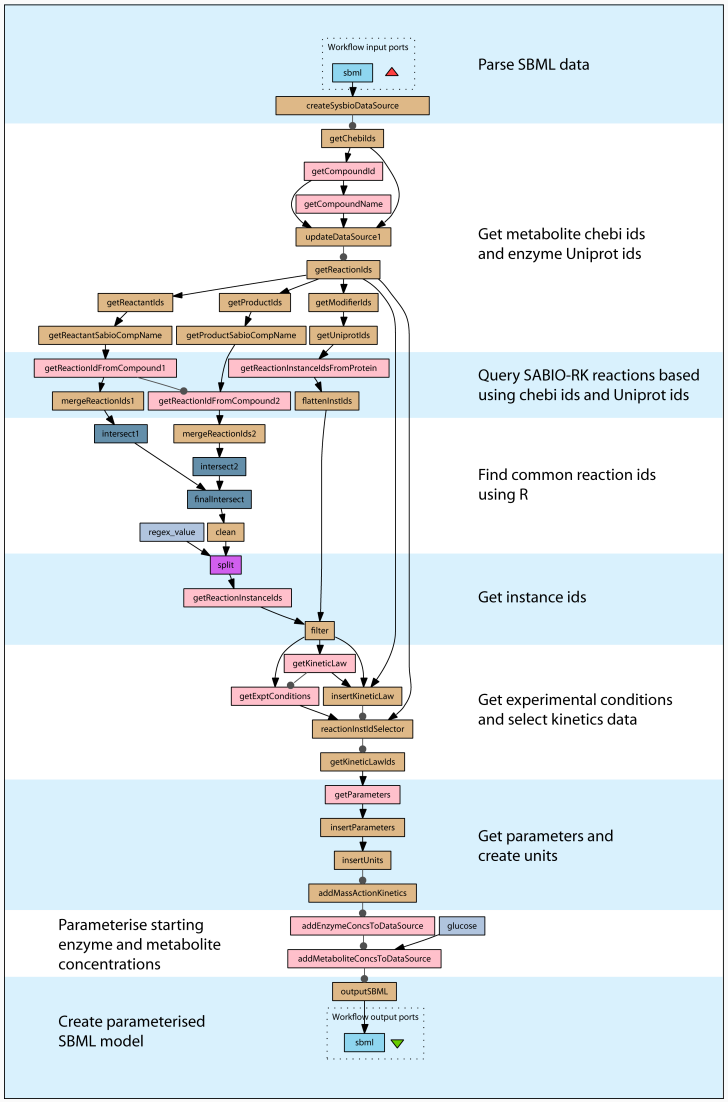
Tool 1ter: Workflows

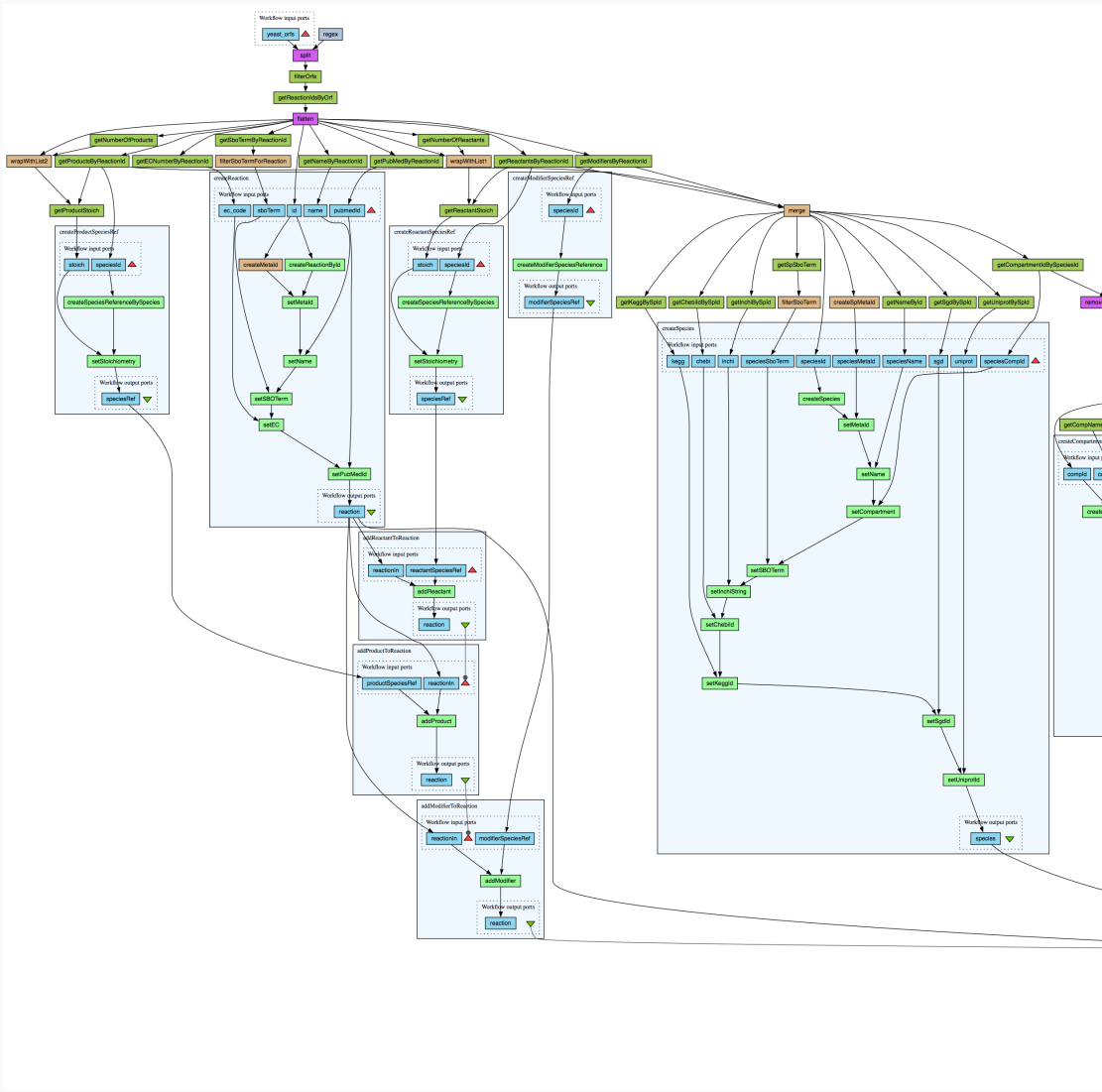
Notebooks are no panacea and do not help developing clean code





Tool 1ter: Workflows





Tool 1ter: Workflows

Workflows:

- Clearer high-level view
- Explicit composition of codes and data movement
- Safer sharing, reusing, and execution
- Notebooks are a variant that is both impoverished and richer
  - No simple/mature path from a notebook to a workflow

Examples:

- Galaxy, Kepler, Taverna, Pegasus, Collective Knowledge, VisTrails
- Light-weight: `make`, dask, drake, swift, `snakemake`, ...
- Hybrids: SOS-notebook, ...

Good Practice #2
Controling Software Environment

Argh... damned computers

- **Alice**: I got 3.123123 **Bob**: I got segfault
- Damned! It used to work!!! Whenever I upgrade my computer, things break so I try to stay away from this ☹

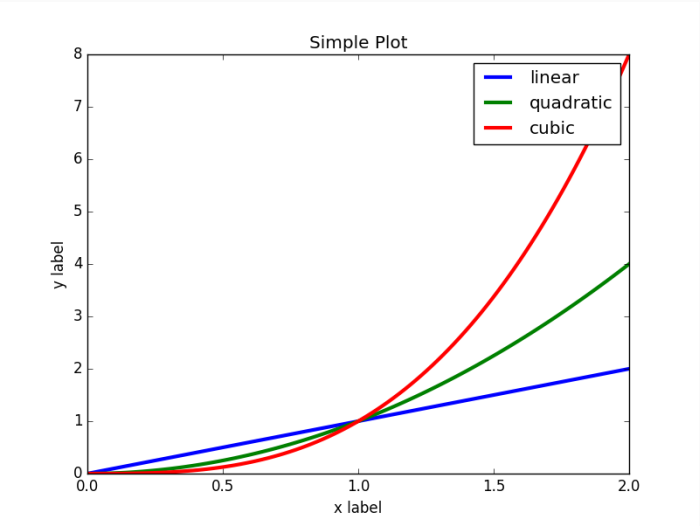
- Whenever trying the code of my colleague, I had to install `libFoo-1.5c` but I broke everything and now neither his code nor mine works! ☹

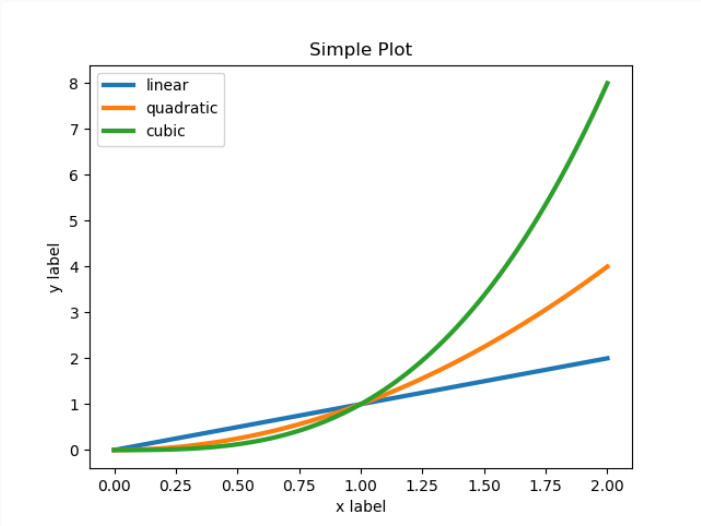
- But hey! Here is my code. It's on GitHub so feel free to play with it! I'm doing open science 😉
  1. No one will ever run/use your code if it isn't easy to install
  2. No one will ever manage to run your code if you don't document how to run it
  3. Others (even you) are unlikely to get the same results unless you control and share your software environment

Software dependencies: horror stories

Software dependencies: horror stories

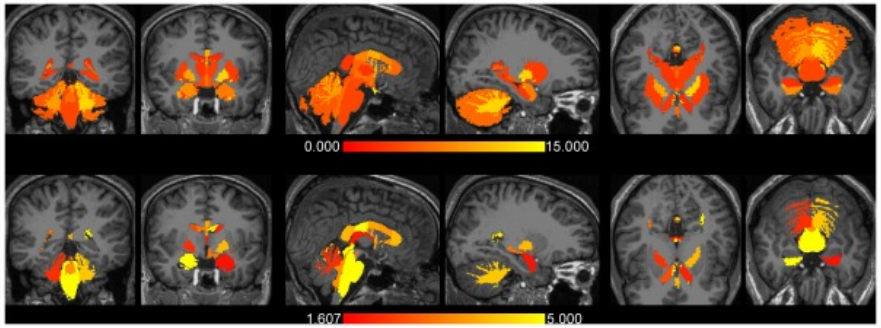
- Software environment evolution





Software dependencies: horror stories

- Software environment evolution
- OS heterogeneity

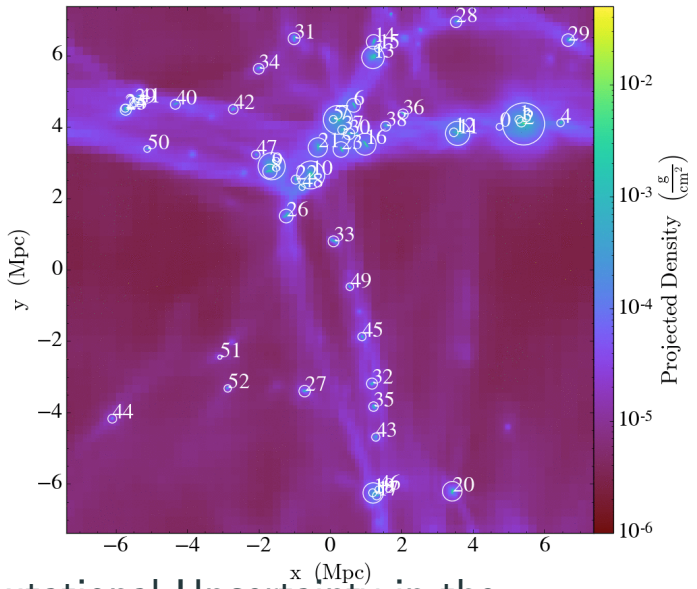


The Effects of FreeSurfer Version, Workstation Type, and Macintosh Operating System Version on Anatomical Volume and Cortical Thickness Measurements (PLOS ONE, 2012)

> *Significant differences in volume and cortical thickness were revealed across FreeSurfer versions:*
>
> - *volume:* $8.8 \pm 6.6\%$ *(range 1.3-64.0%)*
> - *cortical thickness:* $2.8 \pm 1.3\%$ *(range 1.1-7.7%)*
>
> *About a factor two smaller differences were found between the Mac and HP workstations and between Mac OSX 10.5 and OSX 10.6.*
>
> *In the context of an ongoing study, users are discouraged to update to a new major release of either FreeSurfer or operating system.*
>
> *Formal assessment of the accuracy of FreeSurfer is desirable.*

Software dependencies: horror stories

- Software environment evolution
- OS heterogeneity
- Impact of the compiler

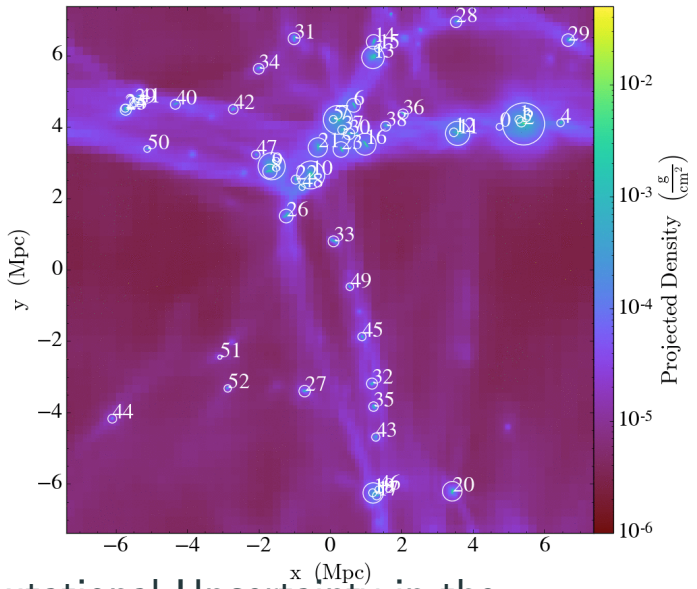


Assessing Reproducibility: An Astrophysical Example of Computational Uncertainty in the HPC Context (ResCuE-HPC, 2018)

Compiler	Optim.	Largest Halo Avg Mass.	Std. Err	Walltime
gcc@6.2.0	None	2.273E 46	1.069E 44	22h

Software dependencies: horror stories

- Software environment evolution
- OS heterogeneity
- Impact of the compiler

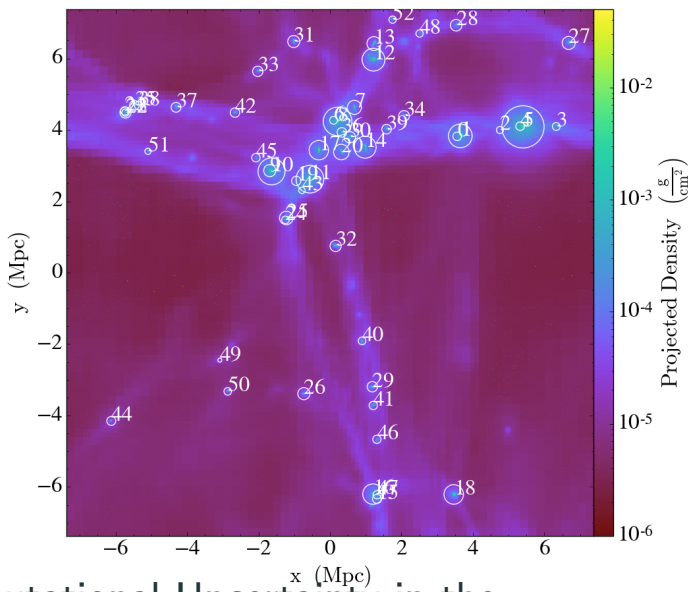


Assessing Reproducibility: An Astrophysical Example of Computational Uncertainty in the HPC Context (ResCuE-HPC, 2018)

Compiler	Optim.	Largest Halo Avg Mass.	Std. Err	Walltime
gcc@6.2.0	None	2.273E 46	1.069E 44	22h

Software dependencies: horror stories

- Software environment evolution
- OS heterogeneity
- Impact of the compiler

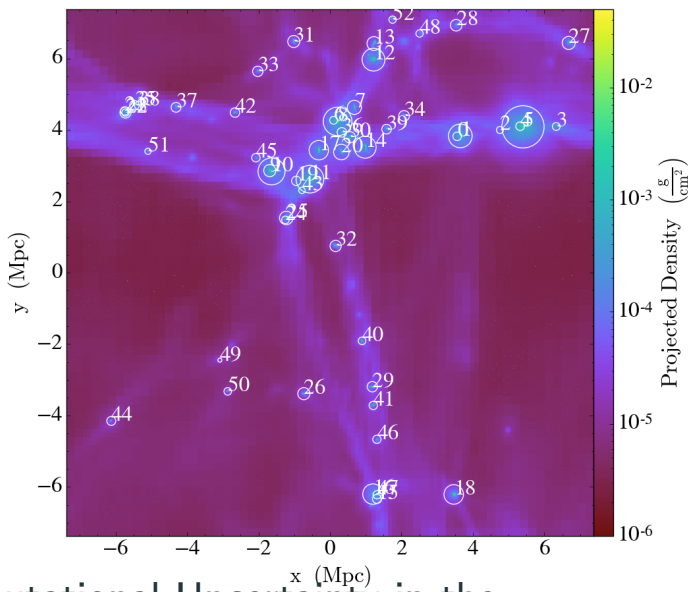


Assessing Reproducibility: An Astrophysical Example of Computational Uncertainty in the HPC Context (ResCuE-HPC, 2018)

Compiler	Optim.	Largest Halo Avg Mass.	Std. Err	Walltime
gcc@6.2.0	None	2.273E 46	1.069E 44	22h
gcc@6.2.0	Normal	2.266E 46	1.218E 44	10h
gcc@6.2.0	High	2.275E 46	1.199E 44	9h

Software dependencies: horror stories

- Software environment evolution
- OS heterogeneity
- Impact of the compiler



Assessing Reproducibility: An Astrophysical Example of Computational Uncertainty in the HPC Context (ResCuE-HPC, 2018)

Compiler	Optim.	Largest Halo Avg Mass.	Std. Err	Walltime
gcc@6.2.0	None	2.273E 46	1.069E 44	22h
gcc@6.2.0	Normal	2.266E 46	1.218E 44	10h
gcc@6.2.0	High	2.275E 46	1.199E 44	9h
intel@16.0.3	None	22.71 E 46	1.587E 44	39h
intel@16.0.3	Normal	43.30 E 46	1.248E 44	7h
intel@16.0.3	High	2.268E 46	1.414E 44	6h
cce@8.5.5	Low	43.11 E 46	1.353E 44	16h
cce@8.5.5	Normal	2.271E 46	1.261E 44	6h
cce@8.5.5	High	2.272E 46	1.341E 44	5h

Complex ecosystems

```
1 import matplotlib
2 print(matplotlib.__version__)
```

```
3.5.1
```

Complex ecosystems

```
1 import matplotlib
2 print(matplotlib.__version__)
```

```
3.5.1
```

```
1 apt show python3-matplotlib
```

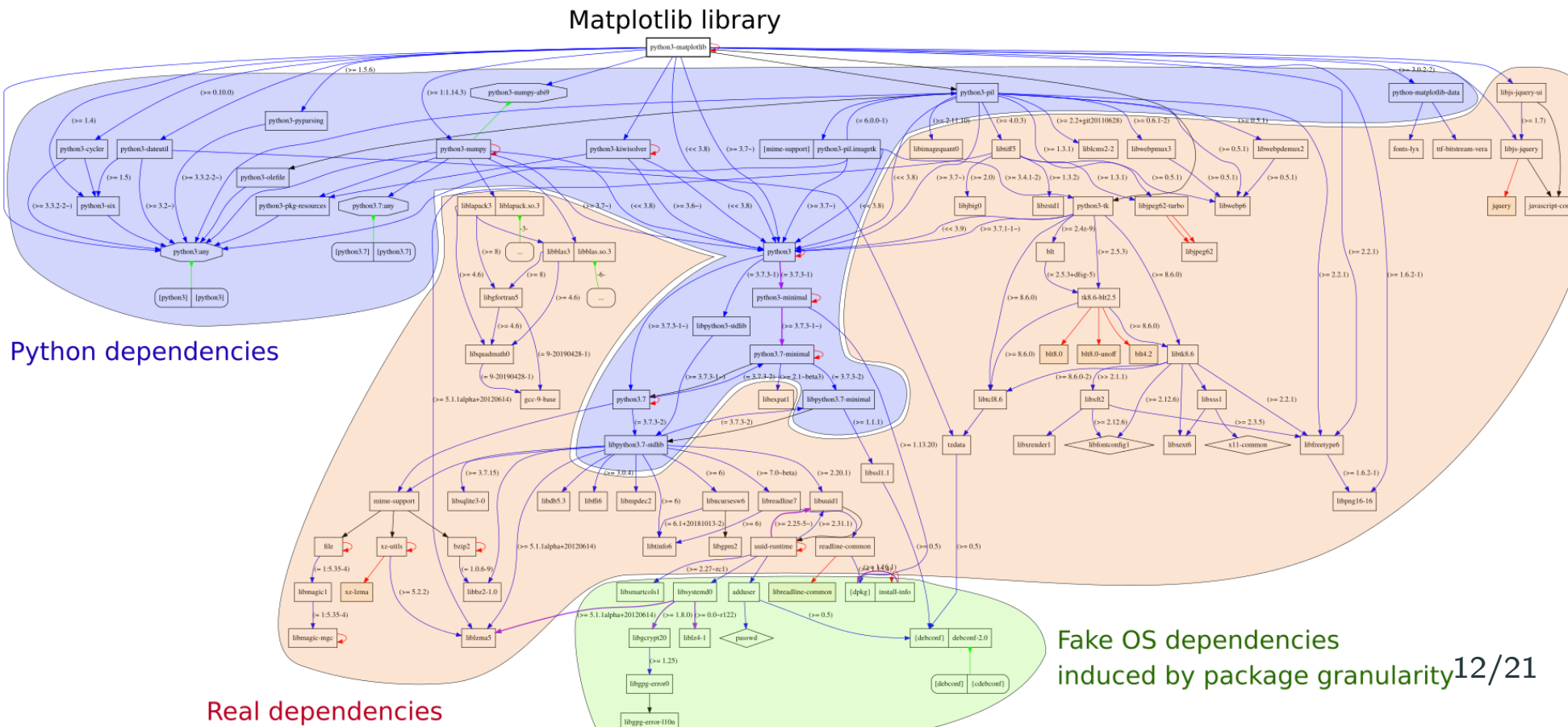
```
Package: python3-matplotlib
Version: 3.5.1-2+b1
Source: matplotlib (3.5.1-2)
Maintainer: Sandro Tosi <morph@debian.org>
Installed-Size: 27.6 MB
Depends: libjs-jquery, libjs-jquery-ui, python-matplotlib-data (>= 3.5.1),
         python3-dateutil, python3-pil.imagetk, python3-pyparsing (>= 1.5.6),
         python3-six (>= 1.4), python3-numpy (>= 1:1.20.0), python3-numpy-abi9,
         python3 (<< 3.11), python3 (>= 3.9~), python3-cycler (>= 0.10.0),
         python3-fonttools, python3-kiwisolver, python3-packaging, python3-pil,
         python3:any, libc6 (>= 2.29), libfreetype6 (>= 2.2.1),
         libgcc-s1 (>= 3.3.1), libqhull-r8.0 (>= 2020.1), libstdc++6 (>= 11)
Recommends: python3-tk
Suggests: dvipng, ffmpeg, fonts-staypuft, ghostscript, gir1.2-gtk-3.0, inkscape,
          ipython3, librsvg2-common, python-matplotlib-doc, python3-cairocffi,
          python3-gi, python3-gi-cairo, python3-gobject, python3-pyqt5,
```

Complex ecosystems

```
1  import matplotlib
2  print(matplotlib.__version__)
```

```
3.5.1
```

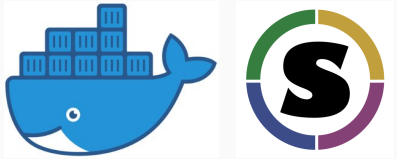
```
1  apt show python3-matplotlib
```



Tool 2: Containers and Package Managers

The good

The bad

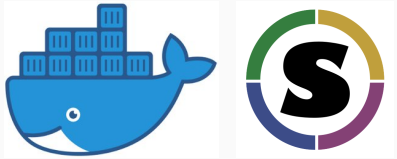


The uggly

Automatic tracking

Tool 2: Containers and Package Managers

The good | The bad | The uggly



Automatic tracking

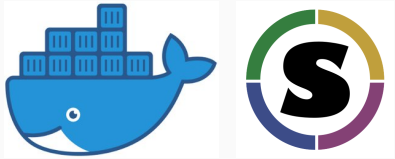
Containers

- Pros: Lightweight, Good isolation, Easy to use
  - Running as easy as `docker run <img> <cmd>`
  - Building images: `docker build -f <Dockerfile>`
  - Sharing through the Docker Hub: `docker pull/push <img>`

Tool 2: Containers and Package Managers

The good

The bad



The uggly

Automatic tracking

Containers

- Pros: Lightweight, Good isolation, Easy to use
- Cons: Opaque, Container build is generally not reproducible
  - Recipes rarely follow *reproducible good practices*

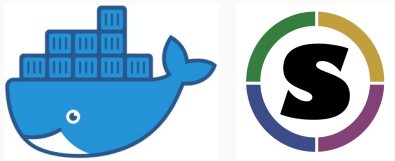
```
1    FROM ubuntu:20.04
2    RUN apt-get update
3         && apt-get upgrade -y
4         && apt-get install -y ...
```

- Choose a stable image (and the smallest possible)
- Include only the necessary libraries (e.g. no graphics libs)
- Avoid system updates (instead freeze sources)

Tool 2: Containers and Package Managers

The good

The bad



The uggly

Automatic tracking

Containers

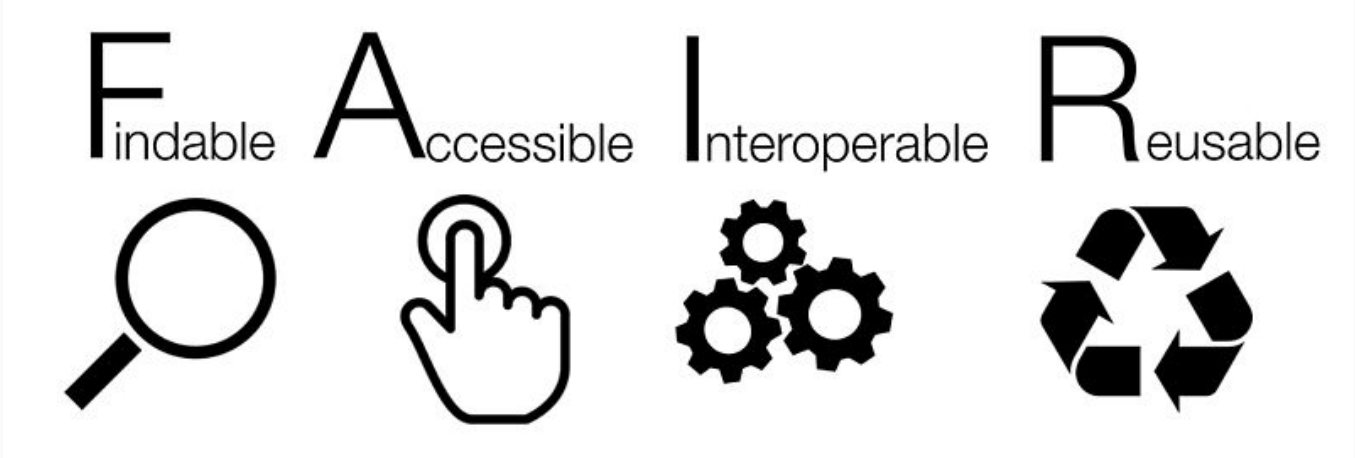
- Pros: Lightweight, Good isolation, Easy to use
- Cons: Opaque, Container build is generally not reproducible

Package managers (the uggly and the good)

- Language specific: `pip/pipenv/virtualenv`, `conda`, `CRAN/Bioconducto`
  - Limits: version management, durability, permeable, language centric
- GUIX/NiX = Full-fledged functional package manager
  - Native support for environment (*à la git*)
  - Isolation through `--pure`
  - Recompile from source (cache recommended)

Good Practice #3
Version Control and Archiving

FAIR principles



https://www.go-fair.org/fair-principles/

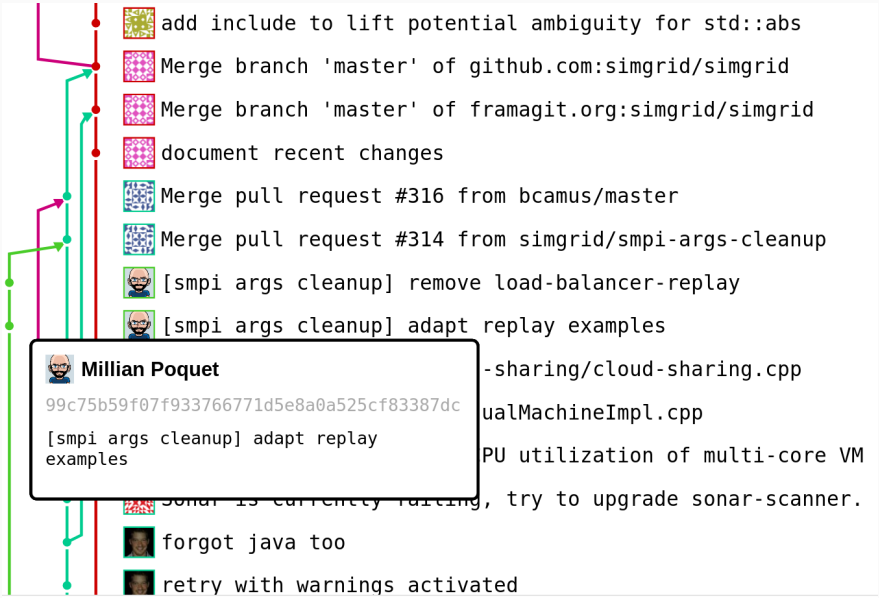
- "*Open as much as possible and close as much as necessary*"
- Management, publication, annotation (metadata), archiving
- Source code = specific data with specific consideration

Let's go beyond general principles!

Tool 3: Version Control and Forge

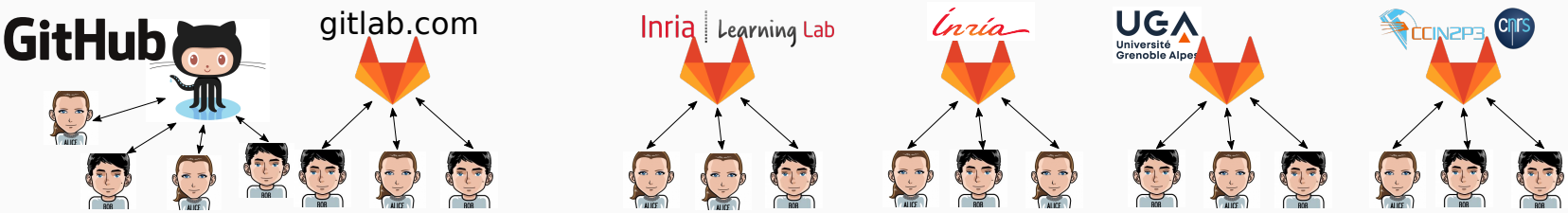
Git = version control

- Developed in 2005 by Linus Torvalds for the kernel development
- Local and efficient rollbacks
- Distributed: everyone has a full copy of the history



GitHub, GitLab, and Co

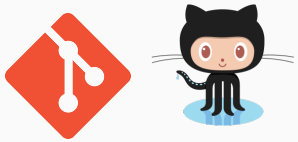
- Free hosting of public projects, social network



Limitation

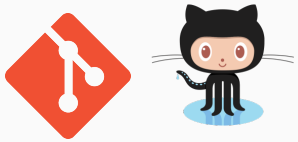
- Managing large data: ~~Git LFS~~ `Git Annex` (or DataLad)

Tool 3bis: Fighting Information Loss with Archives

 or  = awesome collaborations ($\neq$ archive)

- D. Spinellis. *The Decay and Failures of URL References*. CACM, 46(1), 2003
  The half-life of a referenced URL is approximately 4 years from its publication date.
- P. Habibzadeh. *Decay of References to Web sites in Articles Published in General Medical Journals: Mainstream vs Small Journals*. Applied Clinical Informatics. 4 (4), 2013
  half life ranged from 2.2 years in EMHJ to 5.3 years in BMJ
- Discontinuated forges: Code Space, Gitorious, Google code, Inria Gforge

Tool 3bis: Fighting Information Loss with Archives



or

= awesome collaborations ($\neq$ archive)

- D. Spinellis. *The Decay and Failures of URL References*. CACM, 46(1), 2003
  The half-life of a referenced URL is approximately 4 years from its publication date.
- P. Habibzadeh. *Decay of References to Web sites in Articles Published in General Medical Journals: Mainstream vs Small Journals*. Applied Clinical Informatics. 4 (4), 2013
  half life ranged from 2.2 years in EMHJ to 5.3 years in BMJ
- Discontinuated forges: Code Space, Gitorious, Google code, Inria Gforge

Article archives arXiv.org HAL archives-ouvertes.fr

INTERNET ARCHIVE

Data archives figshare zenodo

Software Archive Software Heritage

Collect/Preserve/Share

What Will it Take ?

Changing Research Practices

Soft. Engineering, Statistics, and Reproducible Research in the curricula

Manifesto: "*I solemnly pledge*" (WSSSPE, Lorena Barba, FAIR)

1. I will teach my graduate students about reproducibility
2. All our research code (and writing) is under version control
3. We will always carry out verification and validation
4. We will share data, plotting script & figure under CC-BY
5. We will upload the preprint to arXiv at the time of submission of a paper
6. We will release code at the time of submission of a paper
7. We will add a "Reproducibility" declaration at the end of each paper
8. I will keep an up-to-date web presence

Learn and Teach using online resources like

- Software Carpentry, The Turing Way, ...

Changing Publishing Practices

Artifact evaluation and ACM badges

Major conferences

- Supercomputing: Artifact Description (AD) mandatory, Artifact Evaluation (AE) still optional, Double blind vs. RR
- NeurIPS, ICLR: open reviews, reproducibility challenge

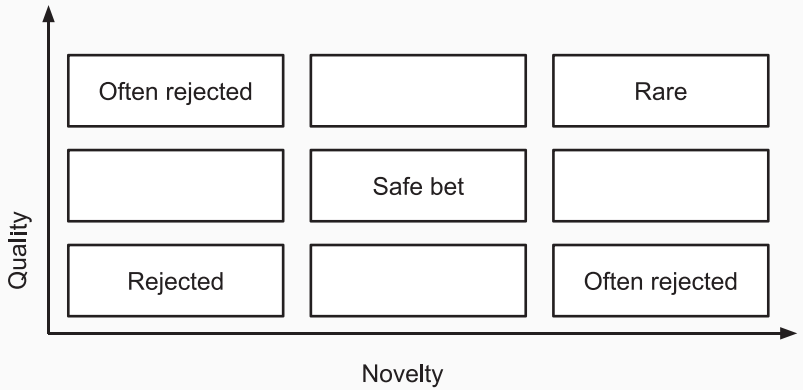
Joelle Pineau @ NeurIPS'18

- ACM SIGMOD 2015-2019, Most Reproducible Paper Award...

Mentalitie are evolving people care, make stuff available, errors are found and fixed

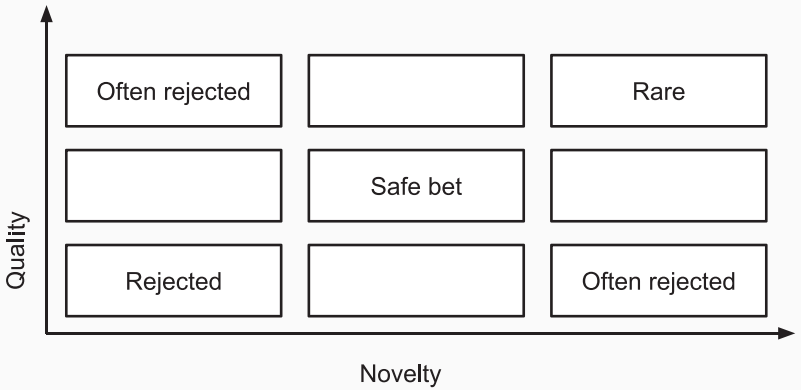
Changing Academic Practices (~~Publish or Perish~~)

- Goodhart's Law: Are Academic Metrics Being Gamed?, M. Fire 2019
  - AI: over 1,000 ranked journals ($\times$10 in 15 years)
  - Shorter papers with increasing self references
  - More and more papers without any citation
  - Sharp increase in the number of new authors publishing at a much faster rate given their career age
- The Truth, The Whole Truth, and Nothing But the Truth: A Pragmatic Guide to Assessing Empirical Evaluations, *TOPLAS* 2016



Changing Academic Practices (~~Publish or Perish~~)

- Goodhart's Law: Are Academic Metrics Being Gamed?, M. Fire 2019
  - AI: over 1,000 ranked journals ($\times$10 in 15 years)
  - Shorter papers with increasing self references
  - More and more papers without any citation
  - Sharp increase in the number of new authors publishing at a much faster rate given their career age
- The Truth, The Whole Truth, and Nothing But the Truth: A Pragmatic, Guide to Assessing Empirical Evaluations, *TOPLAS* 2016



- Impact factor abandoned by Dutch university in hiring and promotion, decisions. Nature, June 2021. *Faculty and staff members at Utrecht University will be evaluated by their commitment to open science*

Reproducible Research = Rigor and Transparency

Good research requires time and resources

1. **Train yourself and your students**: RR, statistics, experiments
   - Beware of checklists and norms Understand what's at stake
2. **Change the norm:** make publication practices evolve
3. **Incentive**: consider RR/open science when hiring/promoting

Reproducible Research = Rigor and Transparency

Good research requires time and resources

1. **Train yourself and your students**: RR, statistics, experiments
   - Beware of checklists and norms Understand what's at stake
2. **Change the norm:** make publication practices evolve
3. **Incentive**: consider RR/open science when hiring/promoting
4. **Prepare the Future:** Toward literate experimentation?
   - Reuse, reuse, reuse!
   - Shared and controled testbeds
   - How to share Experiments ?

Resources and Acknowledgments

Slides are available here

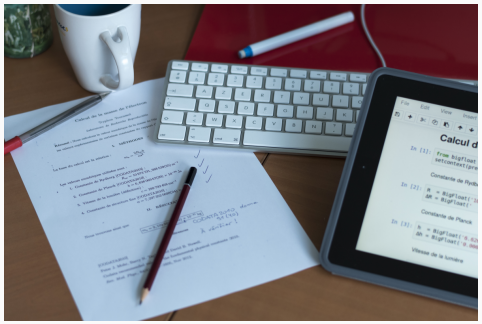
https://github.com/schnorr/SMPE/raw/master/lectures/talk_23_11_23_ICI2

Thanks to

- Arnaud Legrand for sharing slides and knowledge about this topic
- My students that have undertaken many of these methods

MOOC "Advanced RR" planned for Nov. 2023

- Managing data (`FITS/HDF5`, `git annex`)
- Software environment control (`docker`, `singularity`, `guix`)
- Scientific workflow (`make`, `snakemake`)



More information available

- MOOC Reproducible Research: Methodological principles for a transparent science, Inria Learning Lab