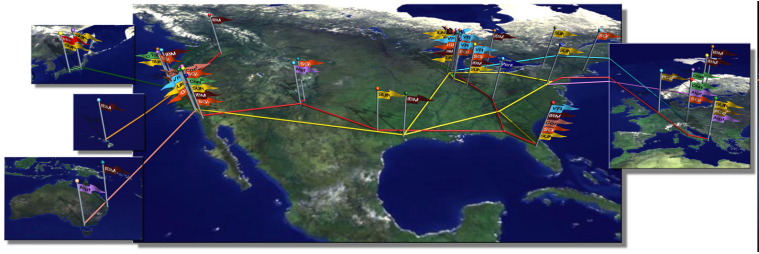
Reproducible Research: Where Do We Stand?

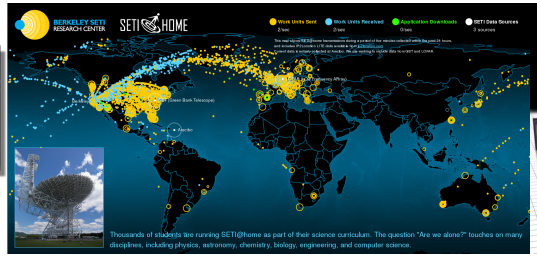
Arnaud Legrand
CNRS, Inria, University of Grenoble

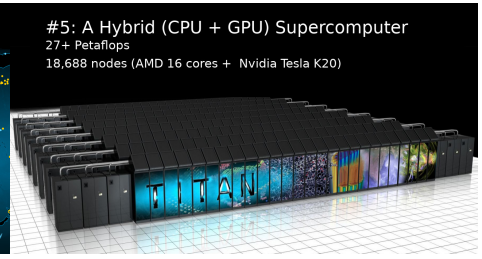
May 8th, 2018 – Luminy, Marseille

Who am I?

Application Domain Large scale computing infrastructures

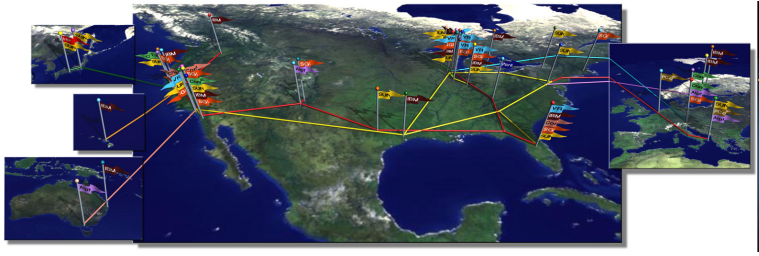


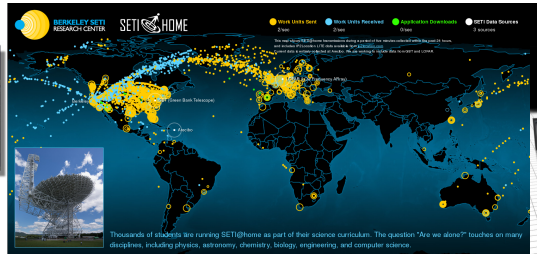


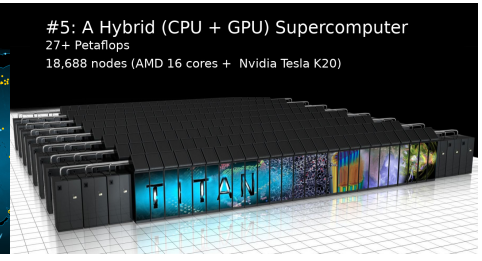


Who am I?

Application Domain Large scale computing infrastructures

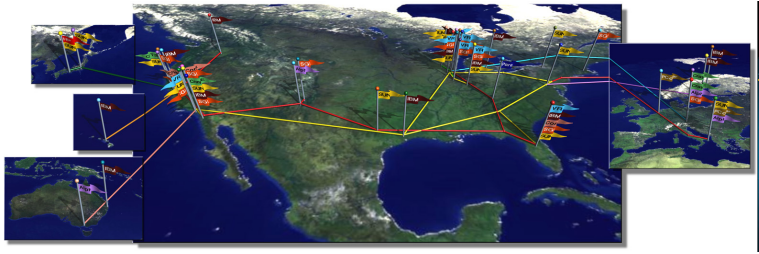


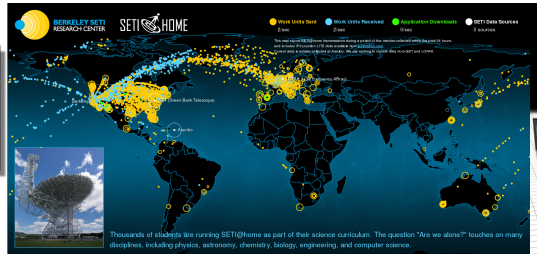


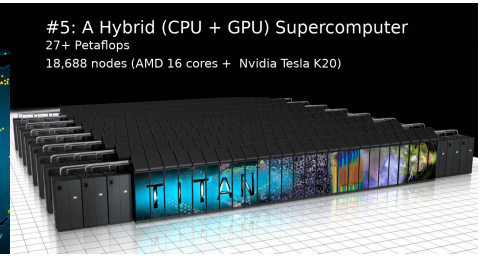


Who am I?

Application Domain Large scale computing infrastructures







Research Themes

- Optimization (scheduling, game theory)
- Performance Evaluation (modeling, simulation, analysis)

Proselytism in Scientific Methodology

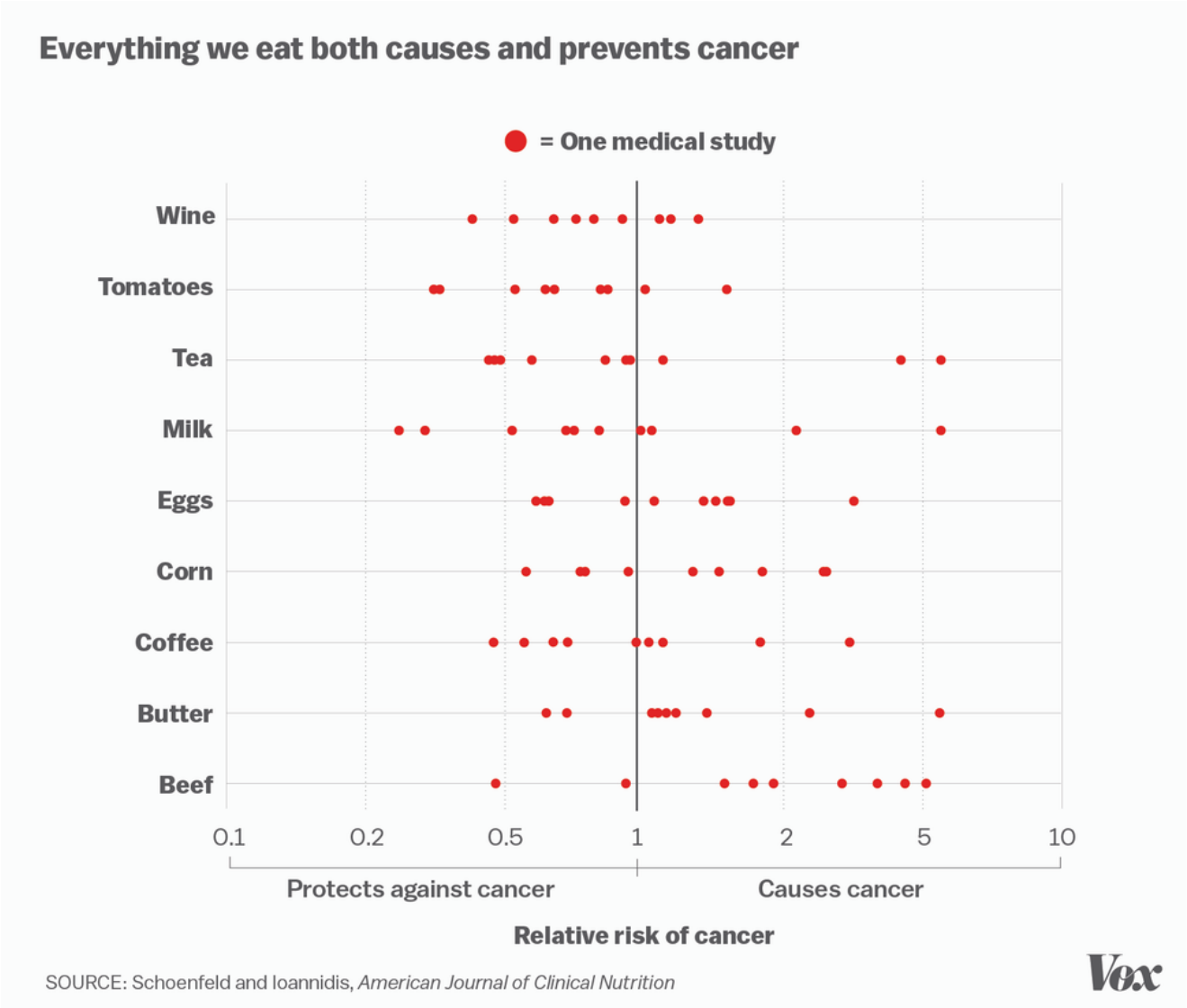
- Reproducible Research/Open Science
- Design of Experiments

Outline

Inconsistencies

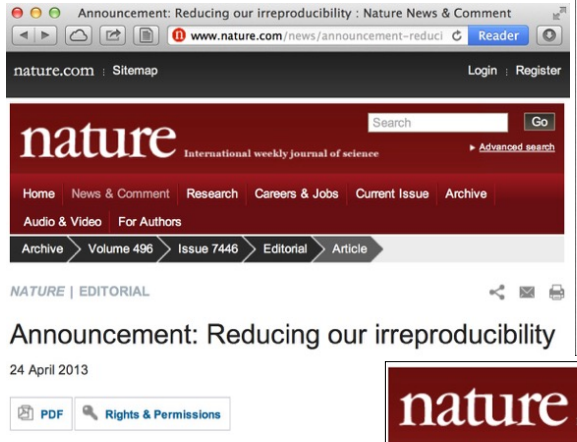
Is everything we eat associated with cancer? A systematic cookbook review, Schoenfeld and Ioannidis, *Amer. Jour. of Clinical Nutrition*, 2013.

Inconsistencies



Public evidence for a Lack of Reproducibility

- J.P. Ioannidis. *Why Most Published Research Findings Are False* PLoS Med. 2005.
- *Lies, Damned Lies, and Medical Science*, The Atlantic. Nov, 2010

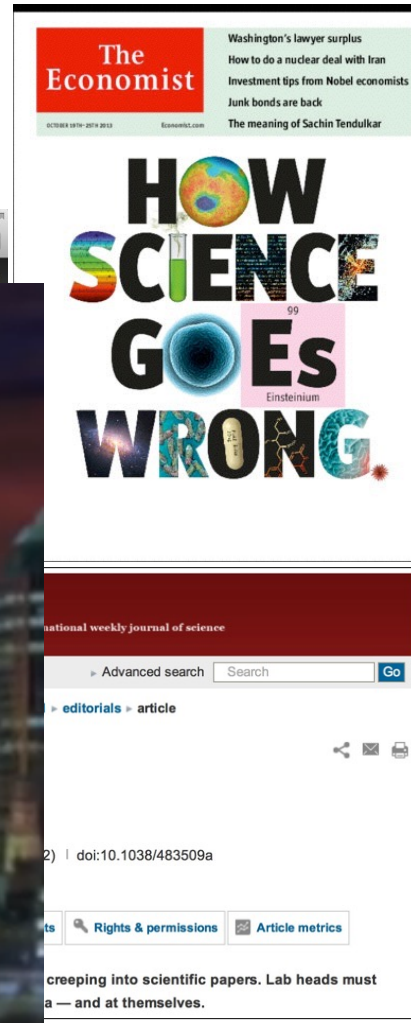


Courtesy V. Stodden, SC, 2015

Public evidence for a Lack of Reproducibility

- J.P. Ioannidis. *Why Most Published Research Findings Are False* PLoS Med. 2005.
- *Lies, Damned Lies, and Medical Science*, The Atlantic. Nov, 2010

Last Week Tonight with John Oliver: Scientific Studies (HBO), May 2016



Courtesy V. Stodden, SC, 2015

Austerity in Fiscal Policy

2010 *"gross debt [..] exceeding 90 percent of the economy has a significant negative effect on economic growth"*

– Reinhart et Rogoff: Growth in a Time of Debt

2013 *While using RR's working spreadsheet, we identified coding errors, selective exclusion of available data, and unconventional weighting of summary statistics.*

– Herndon, Ash and Pollin

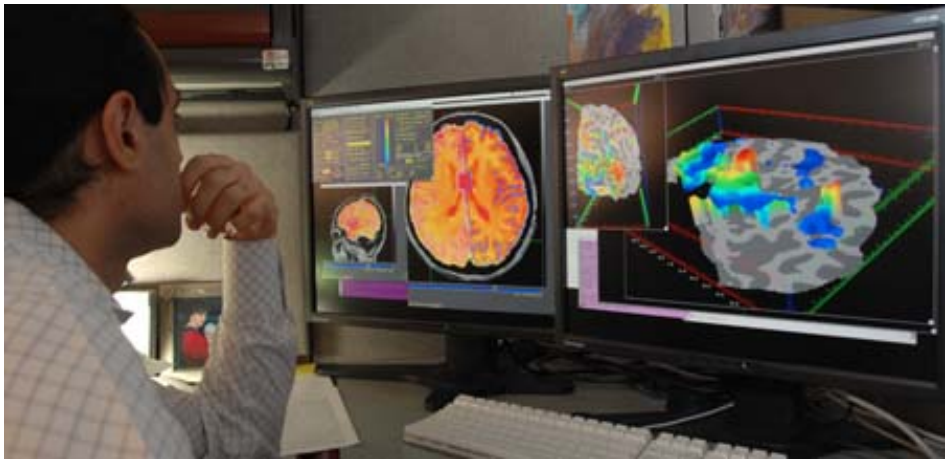
combining data across centuries, exchange rate regimes, public and private debt, and debt denominated in foreign currency as well as domestic currency

– Wray

For 3 years, austerity was not presented as an option but as a necessity.

Yet, a scientific debate has at least been possible.

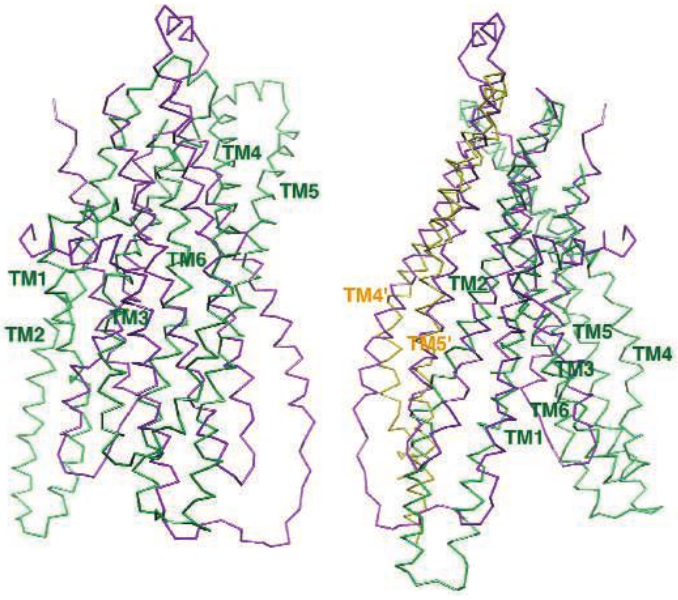
FRMI



- 2010: Bennett et al. and the dead salmon ☺
- 2016: Eklund, Nichols, and Knutsson. A bug in fmri software could invalidate 15 years of brain research (*40,000 articles*, although it is a bit more subtle than this).
- 2016: Nichols. $\approx$ *3 600 articles may have to be revisited for confirmation.*

These article do not necessarily invalidate everything but force the community to improve their practice.

Geoffray Chang's incorrect protein structures



Geoffrey Chang (Scripps, UCSD) works on crystalography and studies the structure of cell membrane proteins.
He specialized in structures of multidrug resistant transporter proteins in bacteria: MsbA de Escheria Choli (Science, 2001), Vibrio cholera (Mol. Biology, 2003), Salmonella typhimurium (Science, 2005)

2006: Inconsistencies reveal a programming mistake

> *a homemade data-analysis program had flipped two columns of data, inverting the electron-density map from which his team had derived the protein structure.*

5 retractations that motivate improved software engineering practices in computational biology

A Reproducibility Crisis? What are the Consequences ?

The Duke University scandal with scientific misconduct on lung cancer

- *Nature Medicine* - 12, 1294 - 1300 (2006) Genomic signatures to guide the use of chemotherapeutics, by Anil Potti and 16 other researchers from Duke University and University of South Florida
- Major commercial labs licensed it and were about to start using it before two statisticians discovered and publicized its faults

Dr. Baggerly and Dr. Coombes found errors almost immediately. Some seemed careless — moving a row or a column over by one in a giant spreadsheet — while others seemed inexplicable. The Duke team shrugged them off as "clerical errors."

The Duke researchers continued to publish papers on their genomic signatures in prestigious journals. Meanwhile, they started three trials using the work to decide which drugs to give patients.

- Retractions: January 2011. Ten papers that Potti coauthored in prestigious journals were retracted for varying reasons

Well... Stronger and Stronger Consequences

A recent scandal In 2013, *Dong-Pyou Han*, a former assistant professor of biomedical sciences at Iowa State University was disgraced:

- Falsified blood results to make it appear as though a vaccine he was working on had exhibited anti-HIV activity
- Han and his team received $\approx$ \$19 million from NIH
- Retraction and resignation of university

Han was sentenced in 2015 to 57 months imprisonment for fabricating and falsifying data in HIV vaccine trials. He was also fined US \$7.2 million!

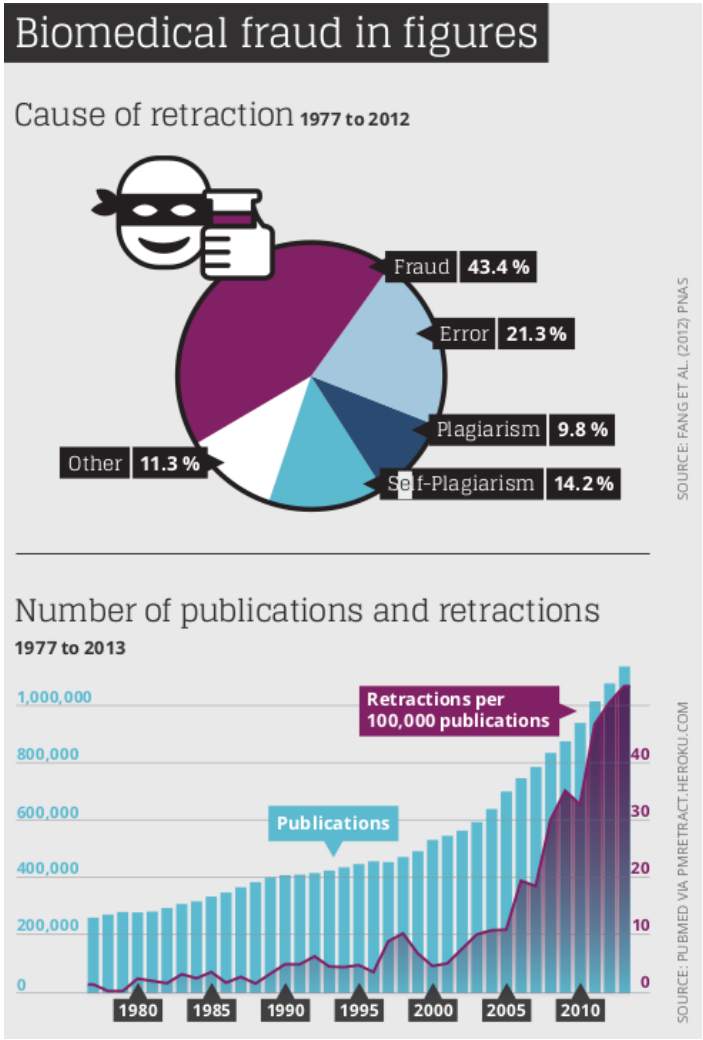
We should avoid witch-hunt

- August 5, 2014, Yoshiki Sasai (stem cell, considered for Nobel Prize) hanged in his laboratory at the RIKEN (Japan). Fraud suspicion...
- In 1986, a young postdoctoral fellow at MIT accused her director, Thereza Imanishi-Kari, of falsifying the results of a study published in Cell and co-signed by the Nobel laureate David Baltimore. [..] Declared guilty, Univ. presidency resignation, and finally cleared. This put the careers of two outstanding researchers on hold for ten years based on unfounded accusations.

Scientific fraud is bad but let's be careful Have a look at the wikipedia *list of academic scandals*.

Is Fraud a new phenomenon?

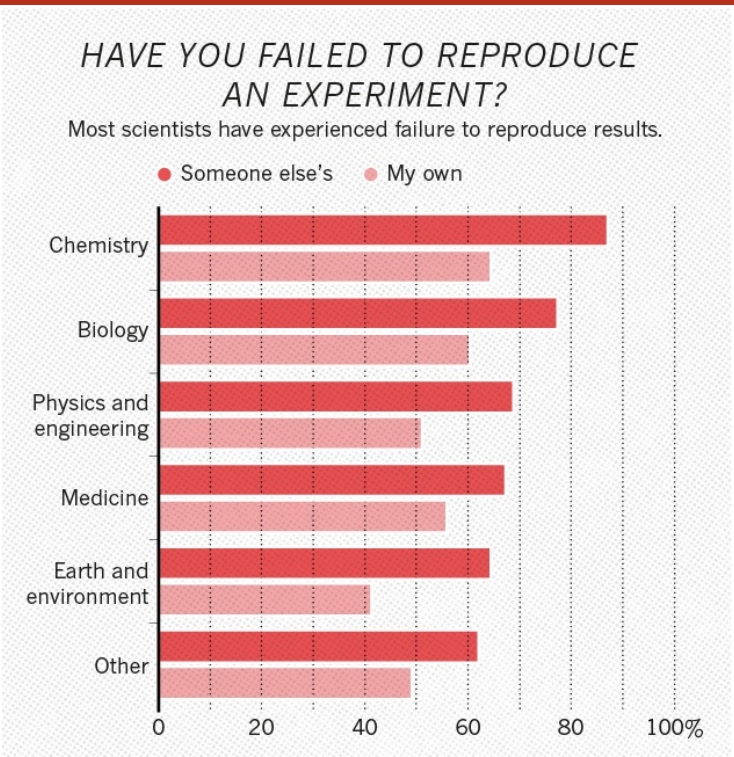
The Battle against Scientific Fraud in the CNRS International Magazine

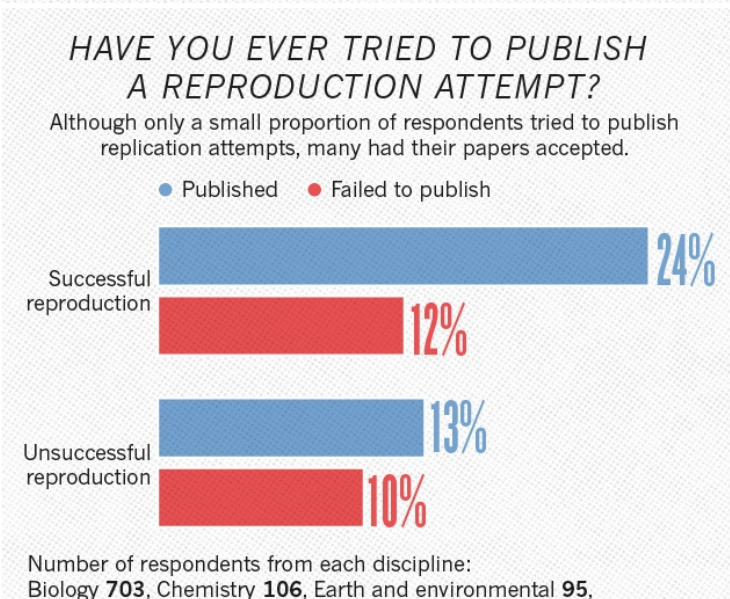


Galileo (data fabrication), Ptolemy (plagiarism), Mendel (data enhancement), Pasteur (rigorous but hided failures), ...

Is it only a matter of Fraud ?

Why are scientific studies so difficult to reproduce?





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: 1+ million articles per year!

Methodological or 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

Wrap-up

- Oncology : "*more than half studies published in prestigious journals cannot be reproduced in industrial labs*"
- Psychology : "*replicating a hundred of major articles: only one third of coherent results*"

Whistle blowers, sick institutions, broken system, ?..

Questionning previous work is part of the scientific process

Just like honesty, rigor and transparency. . .

Risks scientists credibility put into question. No more difference with crooks!

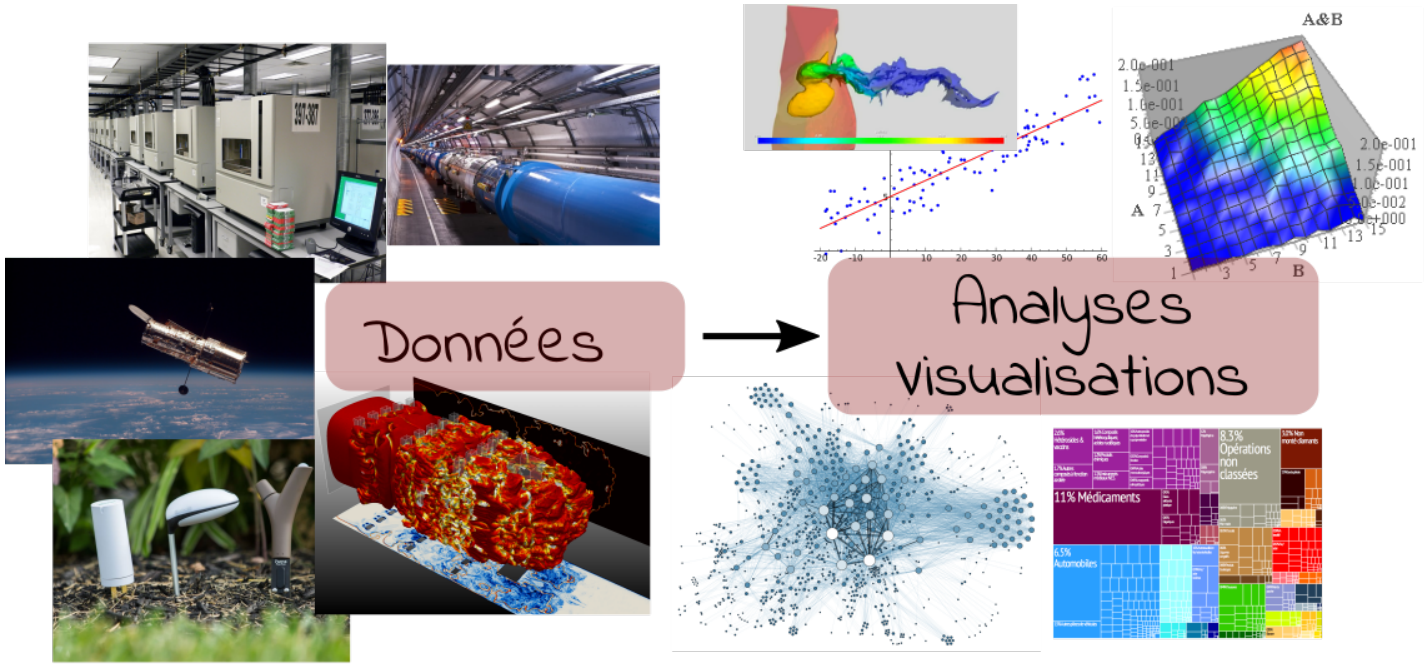
Outline

Computational science!

Today the computer is just as important a tool for chemists as the test tube. Simulations are so realistic that they predict the outcome of traditional experiments...

– Nobel Comity (Chemistry), 2013

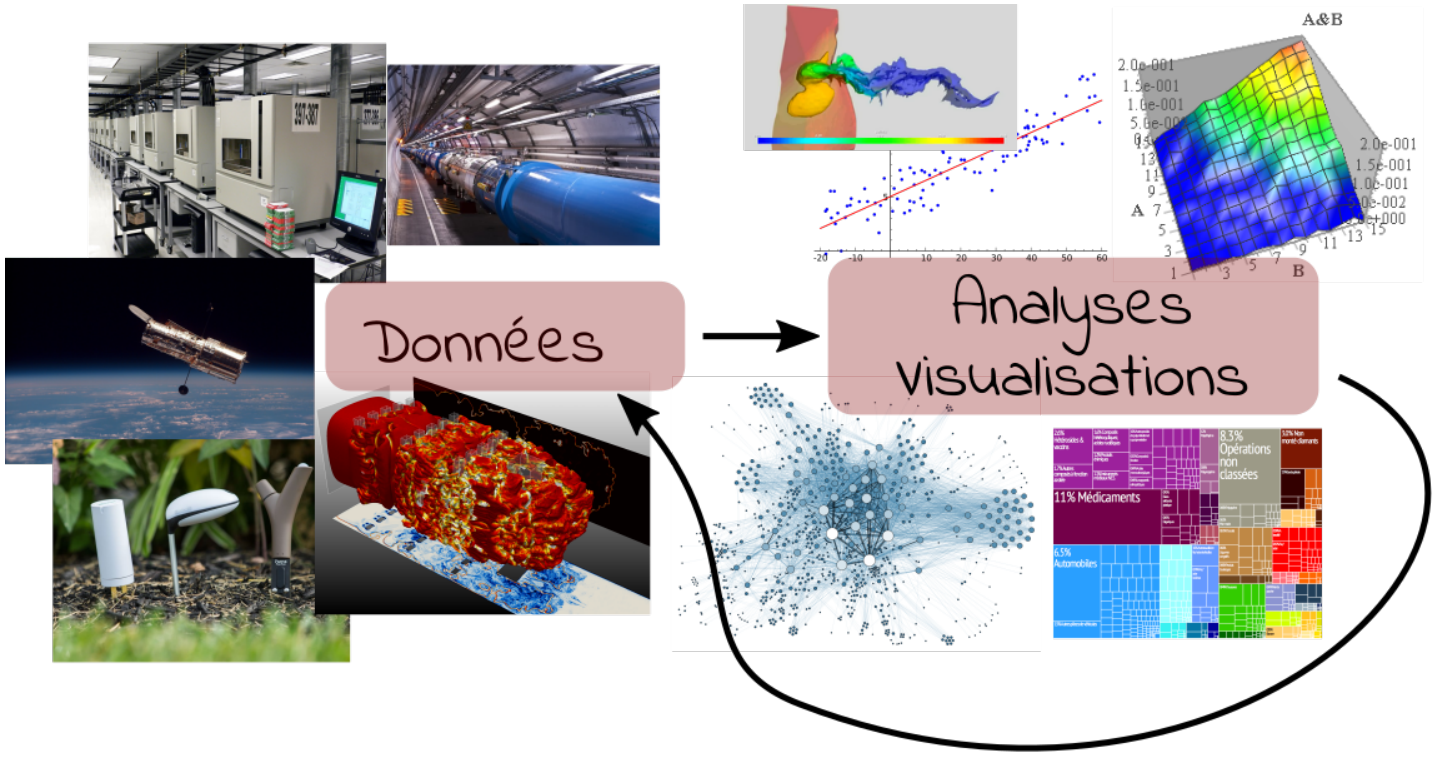
Computational science!



Today the computer is just as important a tool for chemists as the test tube. Simulations are so realistic that they predict the outcome of traditional experiments...

– Nobel Comity (Chemistry), 2013

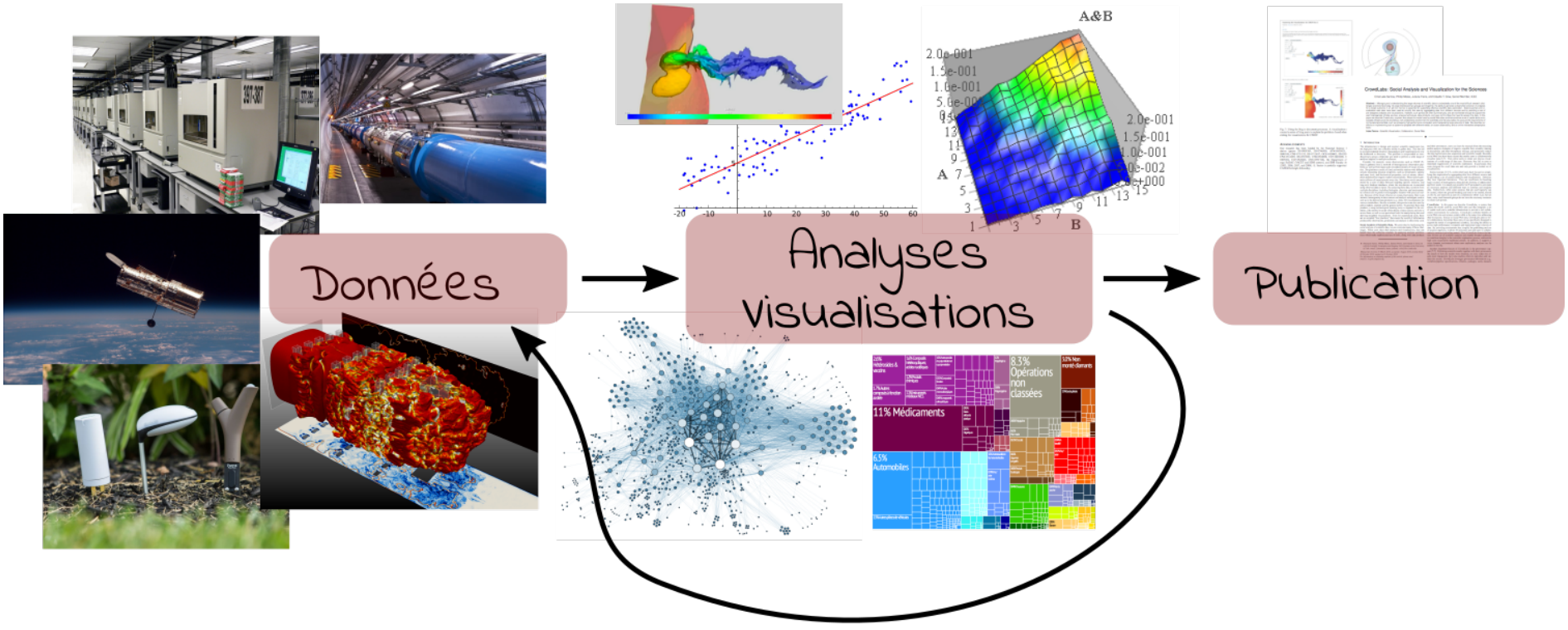
Computational science!



Today the computer is just as important a tool for chemists as the test tube. Simulations are so realistic that they predict the outcome of traditional experiments...

– Nobel Comity (Chemistry), 2013

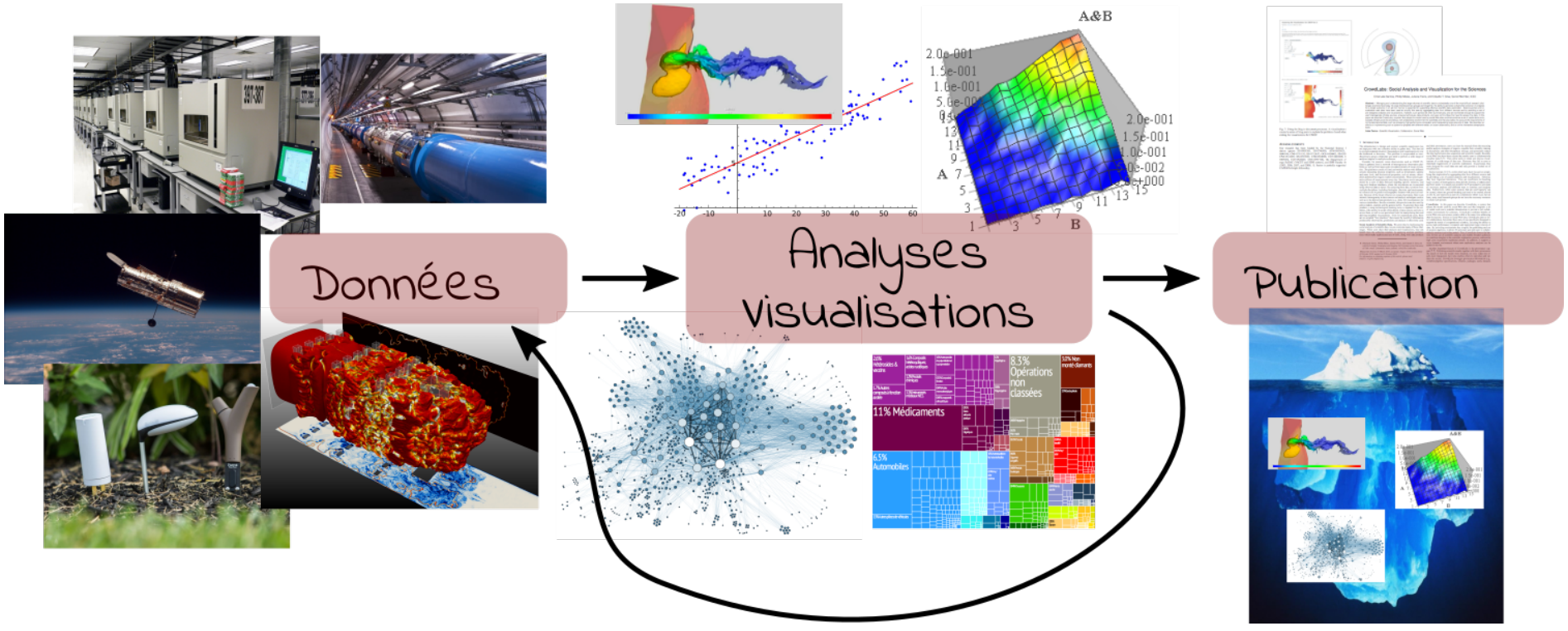
Computational science!



Today the computer is just as important a tool for chemists as the test tube. Simulations are so realistic that they predict the outcome of traditional experiments...

– Nobel Comity (Chemistry), 2013

Computational science!



Today the computer is just as important a tool for chemists as the test tube. Simulations are so realistic that they predict the outcome of traditional experiments. . .

– Nobel Comity (Chemistry), 2013

Aren't Computers Good for Science ?

How computers broke science – and what we can do to fix it.

- Point and click
- Spreadsheets : programming and data manipulation mistakes
  - `Membrane-Associated Ring Finger (C3HC4) 1, E3 Ubiquitin Pro Ligase` → `MARCH1` → 2016-03-01 → 1456786800
  - `2310009E13` → 2.31E+19
- Complex software stacks : avoid proprietary software as much as possible
- Bugs : *Programming is difficult !*

All this is about Natural Sciences. Should we care ?

Computer Science is young and inherits from Mathematics, Engineering, Nat. Sciences, Linguistic, ...

Purely theoretical scientists whose practice is close to mathematics *may* not be concerned (can't publish a math article without releasing the proofs).

- Have a look at talk by Vladimir Voevodsky in 2014 at Princeton 🙂

All this is about Natural Sciences. Should we care ?

Computer Science is young and inherits from Mathematics, Engineering, Nat. Sciences, Linguistic, ...

Purely theoretical scientists whose practice is close to mathematics *may* not be concerned (can't publish a math article without releasing the proofs).

- Have a look at talk by Vladimir Voevodsky in 2014 at Princeton 🙂

  Computer science is not more related to computers than Astronomy to telescopes

  – Dijkstra (mis-attributed)

Right, why should we care about computers? They are deterministic machines after all, right? 😉

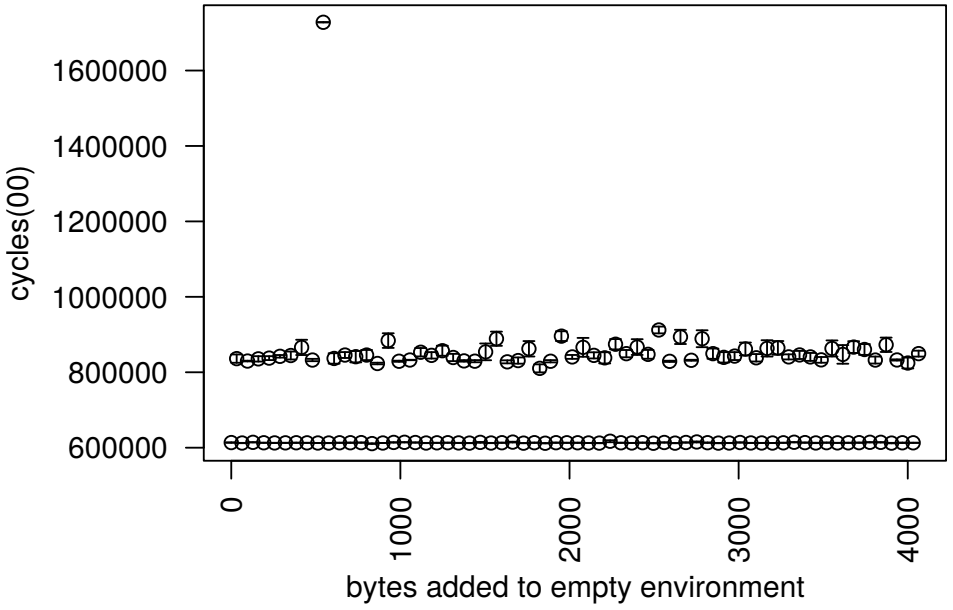
Model $\neq$ **Reality**. Although designed and built by human beings, computer systems are so complex that mistakes easily slip in...

Experimenting with computers

Machines are real!

Brendan Gregg: Shouting in the data center

Machines are complicated



Mytkowicz et al. Producing wrong data without doing anything obviously wrong!
ACM SIGPLAN Not. 44(3), March 2009

Our reality evolves!!! The hardware keeps evolving so most results on old platforms quickly become obsolete (although, we keep building on such results 😉).

- We need to regularly revisit and allow others to build on our work!

Computer performance ? Well, I design algorithms!

- "Real" problems are all NP-hard, Log-APX, etc.
- Real workload = ~~NP-completeness proof widgets~~, regularities and properties (difficult to formally state but that should be exploited)

Algorithms are evaluated on particular workloads that impact
both their running time and the quality of the solutions

Computer performance ? Well, I design algorithms!

- "Real" problems are all NP-hard, Log-APX, etc.
- Real workload = ~~NP-completeness proof widgets~~, regularities and properties (difficult to formally state but that should be exploited)

Algorithms are evaluated on particular workloads that impact both their running time and the quality of the solutions
Machine Learning: Trouble at the lab, The Economist 2013

> *According to some estimates, three-quarters of published scientific papers in the field of machine learning are bunk because of this "overfitting".* – *Sandy Pentland (MIT)*

Computer performance ? Well, I design algorithms!

- "Real" problems are all NP-hard, Log-APX, etc.
- Real workload = ~~NP-completeness proof widgets~~, regularities and properties (difficult to formally state but that should be exploited)

Algorithms are evaluated on particular workloads that impact both their running time and the quality of the solutions

Machine Learning: Trouble at the lab, The Economist 2013

> *According to some estimates, three-quarters of published scientific papers in the field of machine learning are bunk because of this "overfitting".* – *Sandy Pentland (MIT)*

Image Processing: True horror stories, E. Meinhardt-Llopis, CANUM 2016

- *The proposed multigrid algorithm converges to the solution of the problem in O(N)* using biharmonic functions
- Surprisingly, our naive multi-scale Gauss-Seidel converges much faster

Computer performance ? Well, I design algorithms!

- "Real" problems are all NP-hard, Log-APX, etc.
- Real workload = ~~NP-completeness proof widgets~~, regularities and properties (difficult to formally state but that should be exploited)

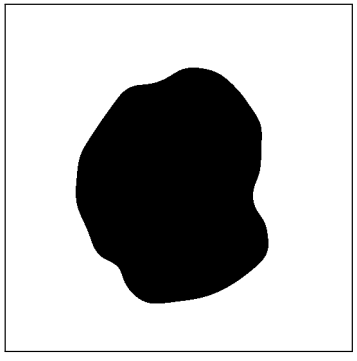
Algorithms are evaluated on particular workloads that impact both their running time and the quality of the solutions

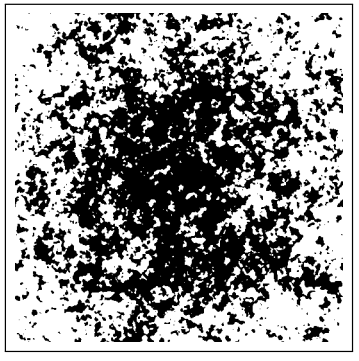
Machine Learning: Trouble at the lab, The Economist 2013

> *According to some estimates, three-quarters of published scientific papers in the field of machine learning are bunk because of this "overfitting".* – *Sandy Pentland (MIT)*

Image Processing: True horror stories, E. Meinhardt-Llopis, CANUM 2016

- *The proposed multigrid algorithm converges to the solution of the problem in O(N)* using biharmonic functions
- Surprisingly, our naive multi-scale Gauss-Seidel converges much faster





All I care about is the algorithm output

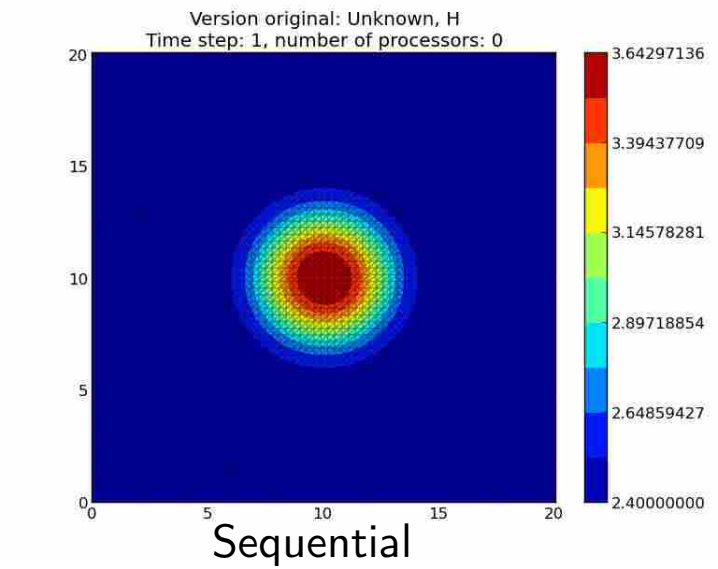
Did I mention we have parallel machines nowadays? 😉

Telemac2D: the simplest gouttedo simulation

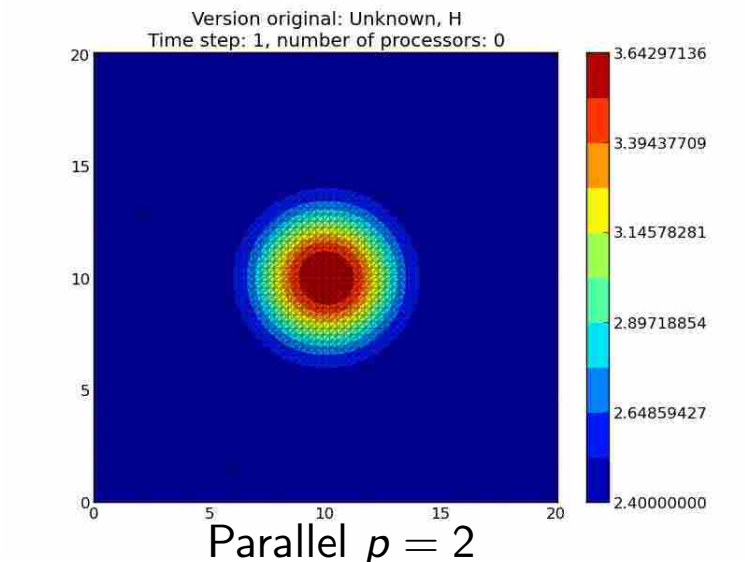
The gouttedo test case

- 2D-simulation of a water drop fall in a square bassin
- Unknown: water depth for a 0.2 sec time step
- Triangular mesh: 8978 elements and 4624 nodes

Expected numerical reproducibility (time step = 1, 2, . . .)



Sequential



Parallel $p = 2$

Courtesy of P. Langlois and R. Nheili

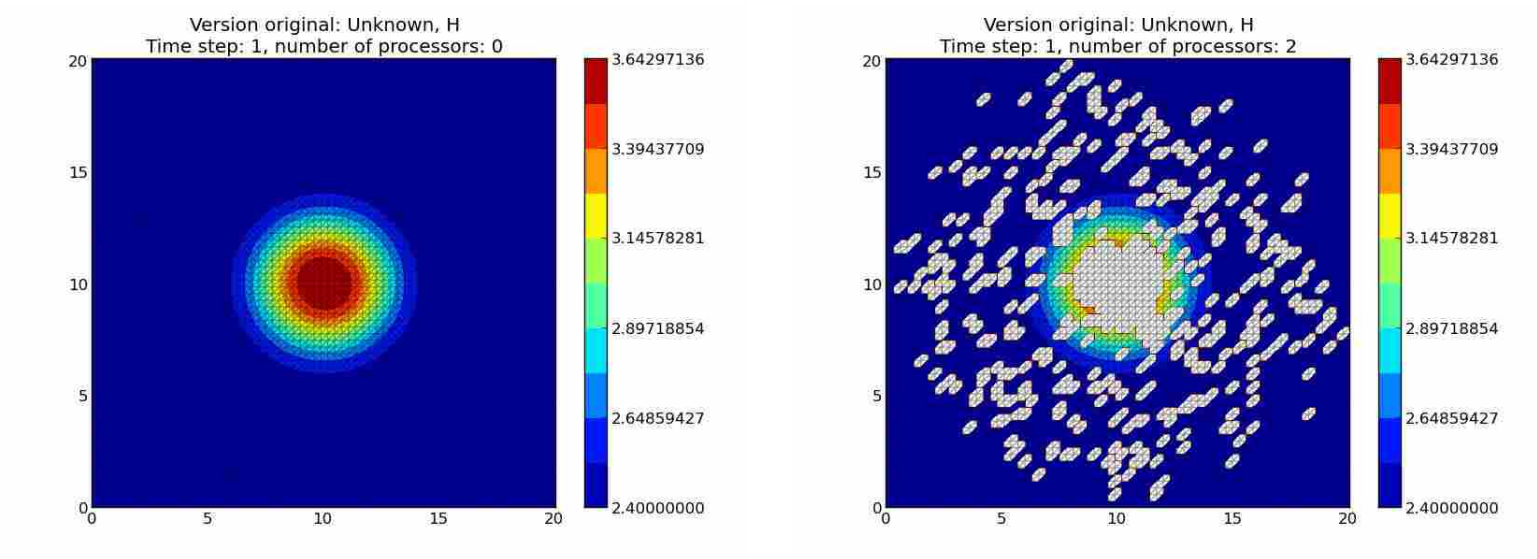
All I care about is the algorithm output

Did I mention we have parallel machines nowadays? 😉

A white plot displays a non-reproducible value

Numerical reproducibility?

time step $= 1$



Sequential

Parallel $p = 2$

Courtesy of P. Langlois and R. Nheili

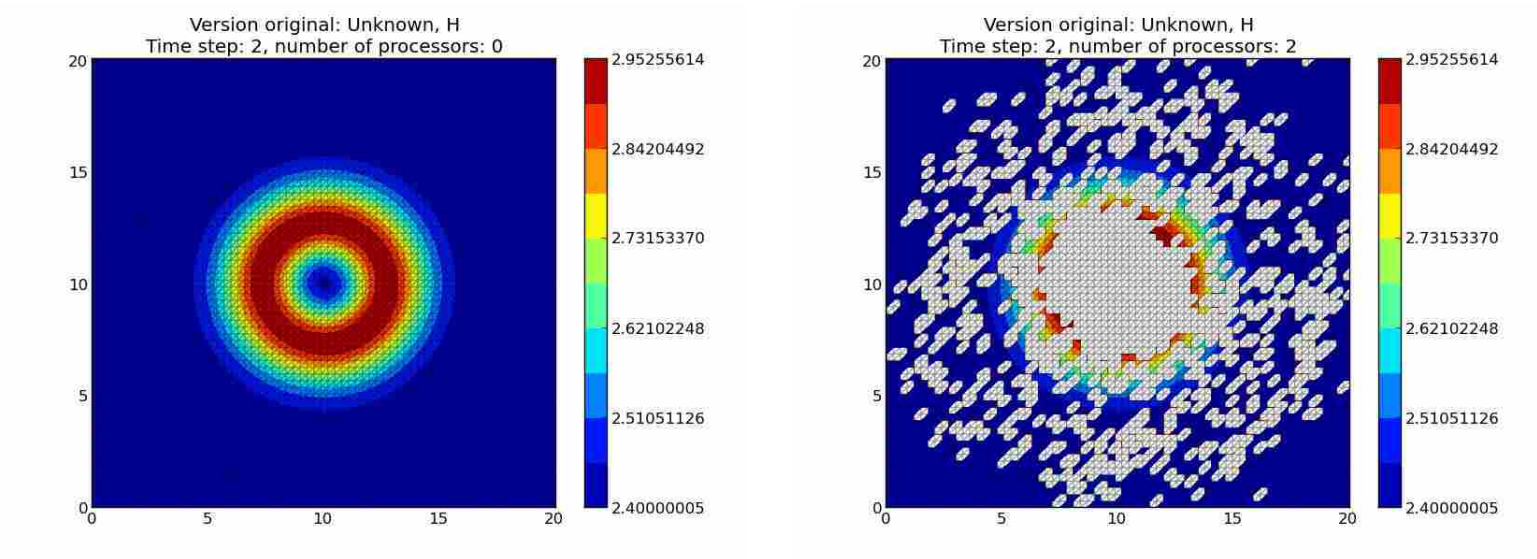
All I care about is the algorithm output

Did I mention we have parallel machines nowadays? 😉

A white plot displays a non-reproducible value

Numerical reproducibility?

time step = 2



Sequential

Parallel $p = 2$

Courtesy of P. Langlois and R. Nheili

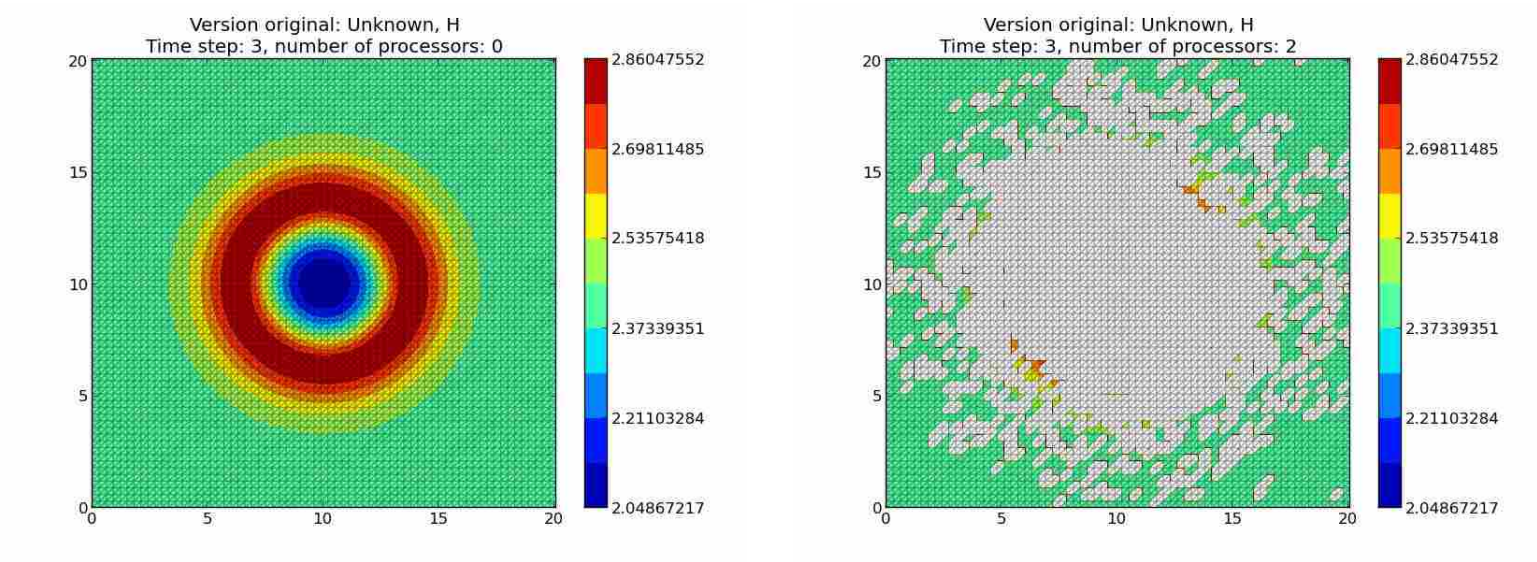
All I care about is the algorithm output

Did I mention we have parallel machines nowadays? 😉

A white plot displays a non-reproducible value

Numerical reproducibility?

time step = 3



Sequential

Parallel $p = 2$

Courtesy of P. Langlois and R. Nheili

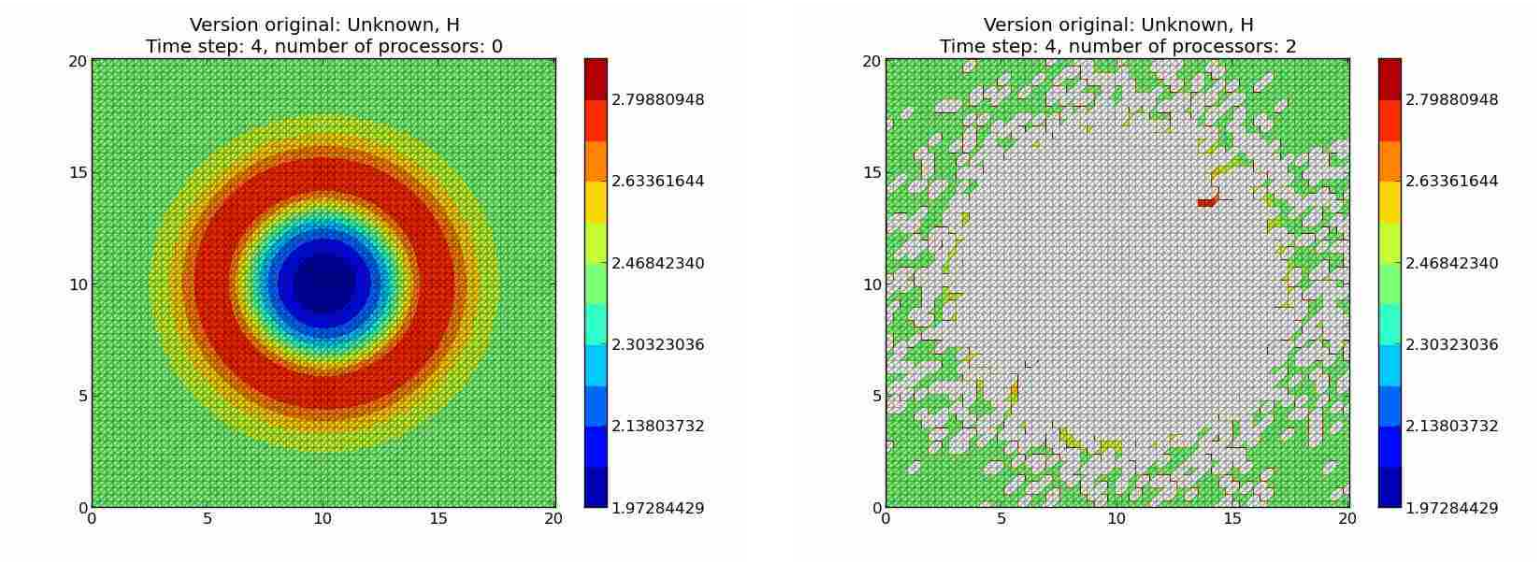
All I care about is the algorithm output

Did I mention we have parallel machines nowadays? 😉

A white plot displays a non-reproducible value

Numerical reproducibility?

time step = 4



Sequential

Parallel $p = 2$

Courtesy of P. Langlois and R. Nheili

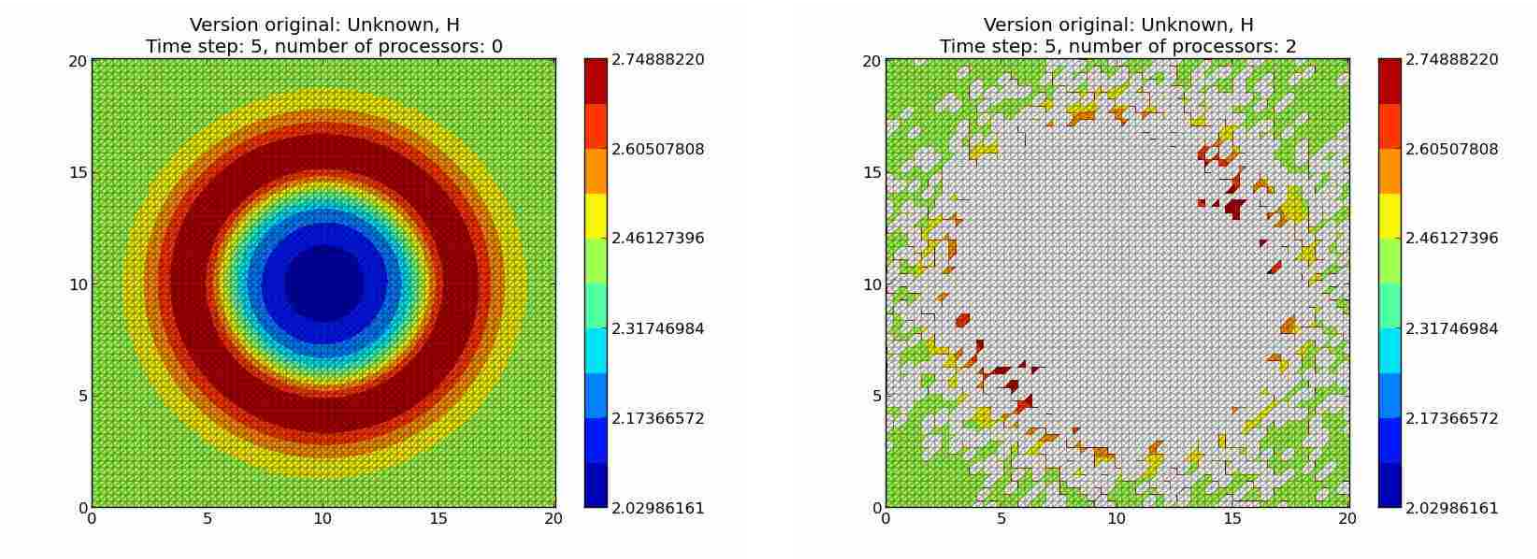
All I care about is the algorithm output

Did I mention we have parallel machines nowadays? 😉

A white plot displays a non-reproducible value

Numerical reproducibility?

time step = 5



Sequential

Parallel $p = 2$

Courtesy of P. Langlois and R. Nheili

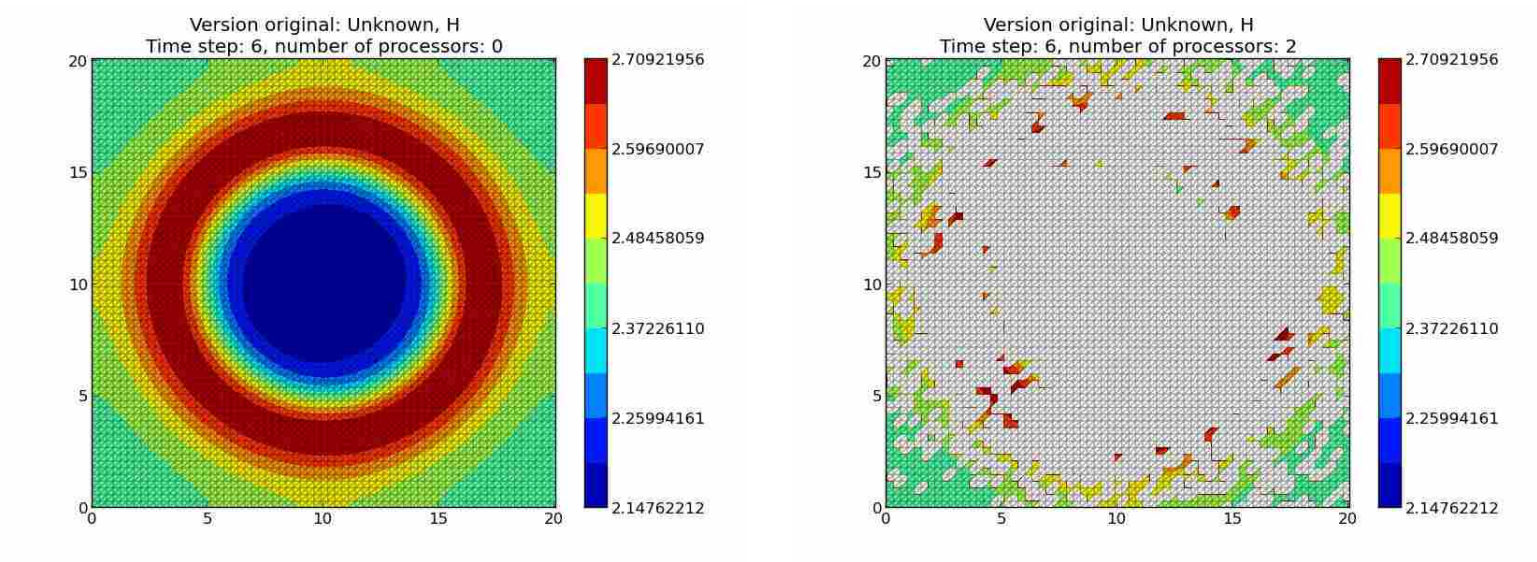
All I care about is the algorithm output

Did I mention we have parallel machines nowadays? 😉

A white plot displays a non-reproducible value

Numerical reproducibility?

time step = 6



Sequential

Parallel $p = 2$

Courtesy of P. Langlois and R. Nheili

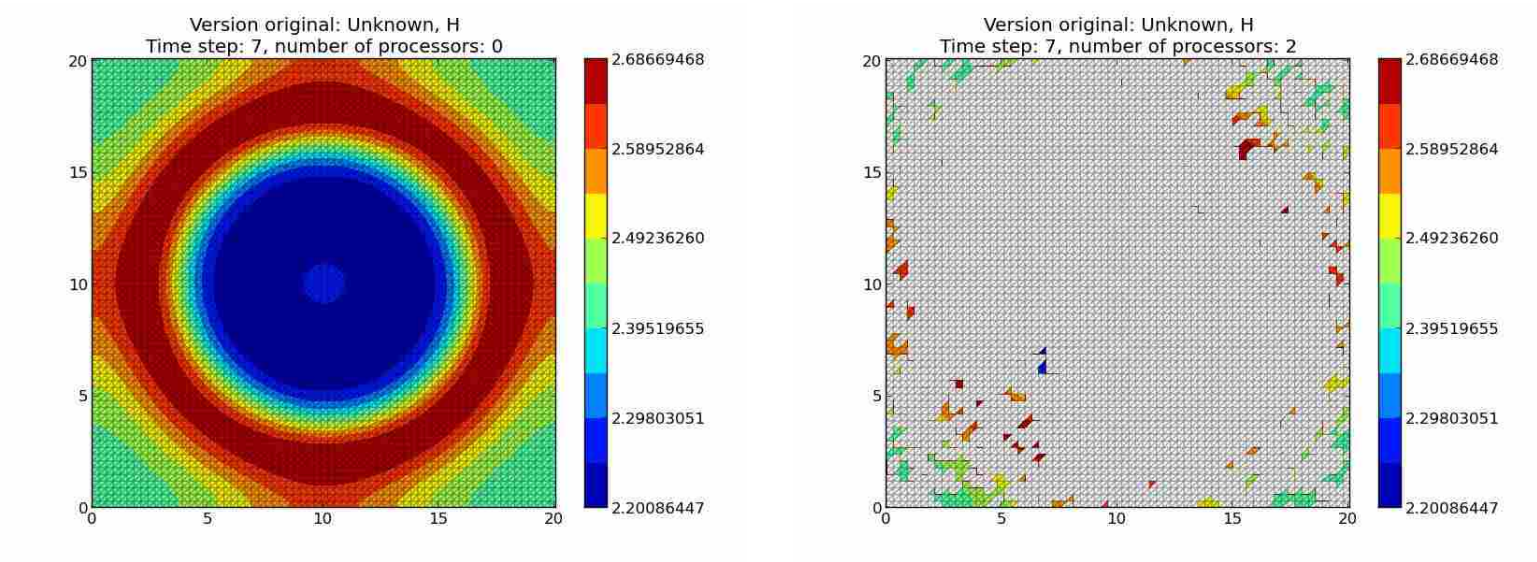
All I care about is the algorithm output

Did I mention we have parallel machines nowadays? 😉

A white plot displays a non-reproducible value

Numerical reproducibility?

time step = 7



Sequential

Parallel $p = 2$

Courtesy of P. Langlois and R. Nheili

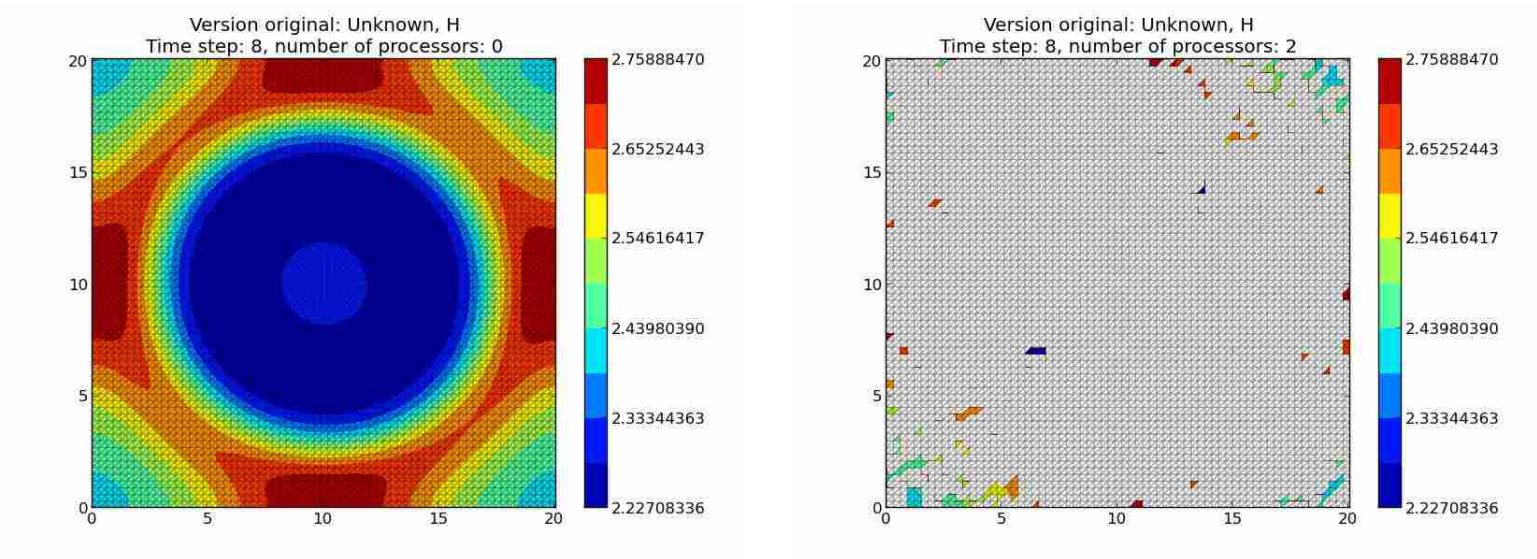
All I care about is the algorithm output

Did I mention we have parallel machines nowadays? 😉

A white plot displays a non-reproducible value

Numerical reproducibility?

time step = 8



Sequential

Parallel $p = 2$

Courtesy of P. Langlois and R. Nheili

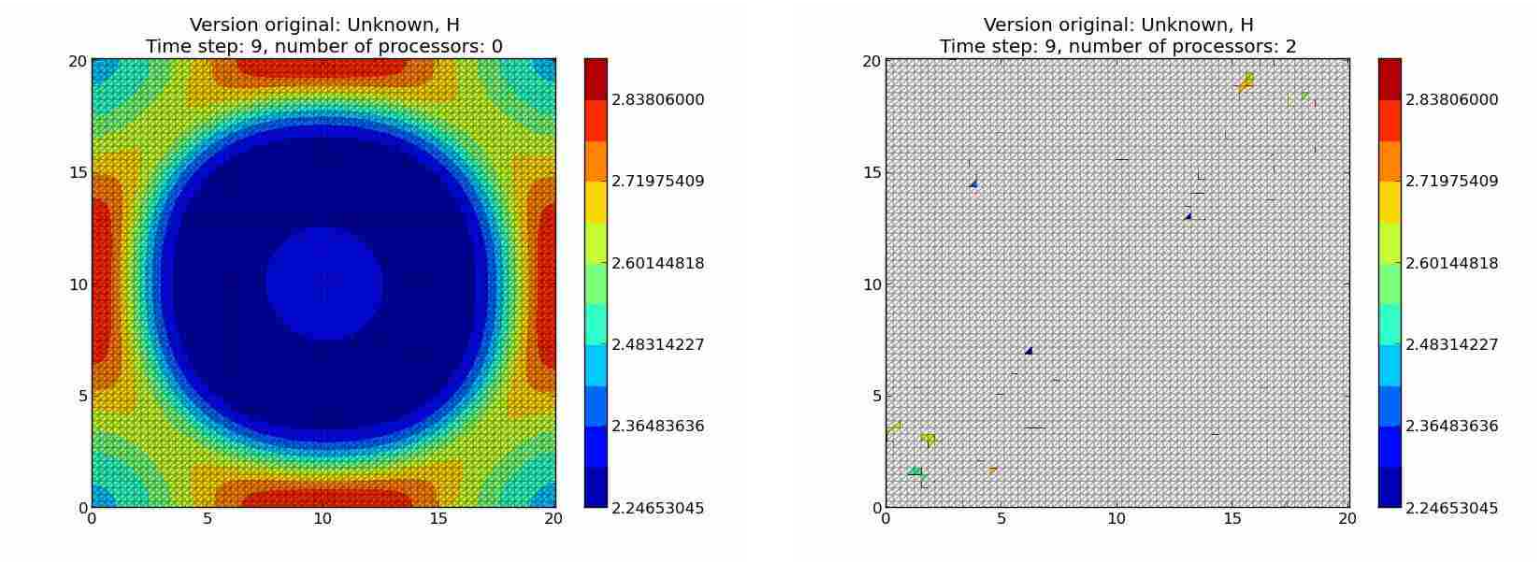
All I care about is the algorithm output

Did I mention we have parallel machines nowadays? 😉

A white plot displays a non-reproducible value

Numerical reproducibility?

time step = 9



Sequential

Parallel $p = 2$

Courtesy of P. Langlois and R. Nheili

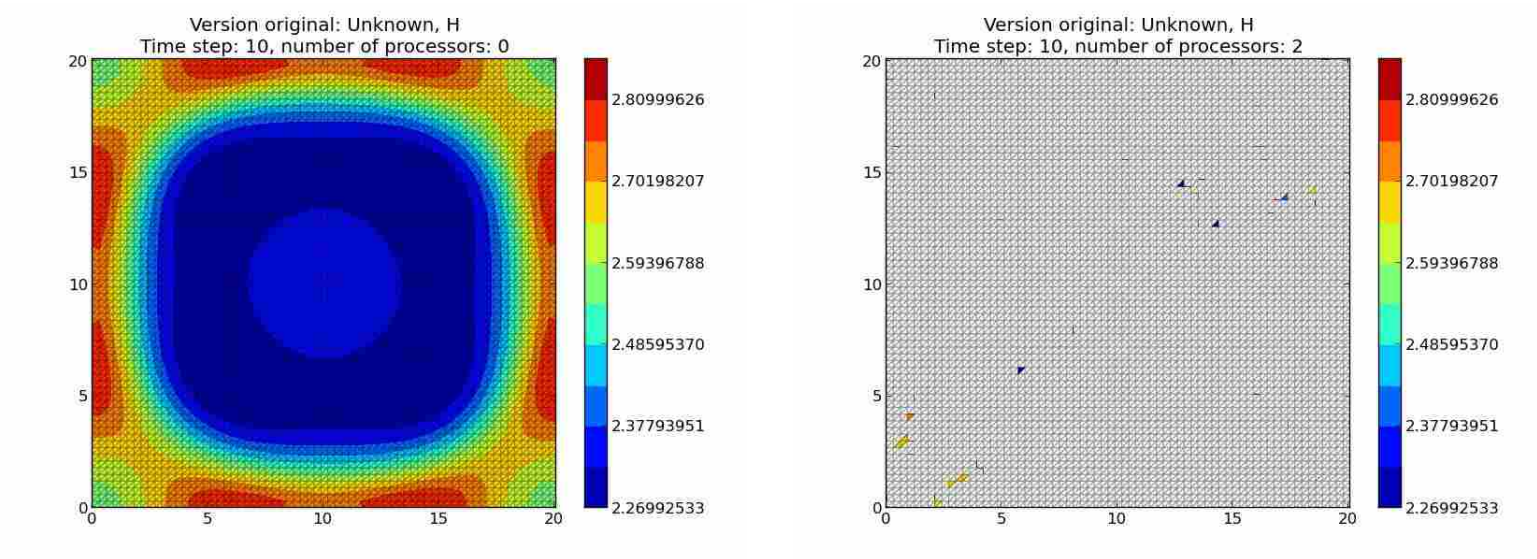
All I care about is the algorithm output

Did I mention we have parallel machines nowadays? 😉

A white plot displays a non-reproducible value

Numerical reproducibility?

time step = 10



Sequential

Parallel $p = 2$

Courtesy of P. Langlois and R. Nheili

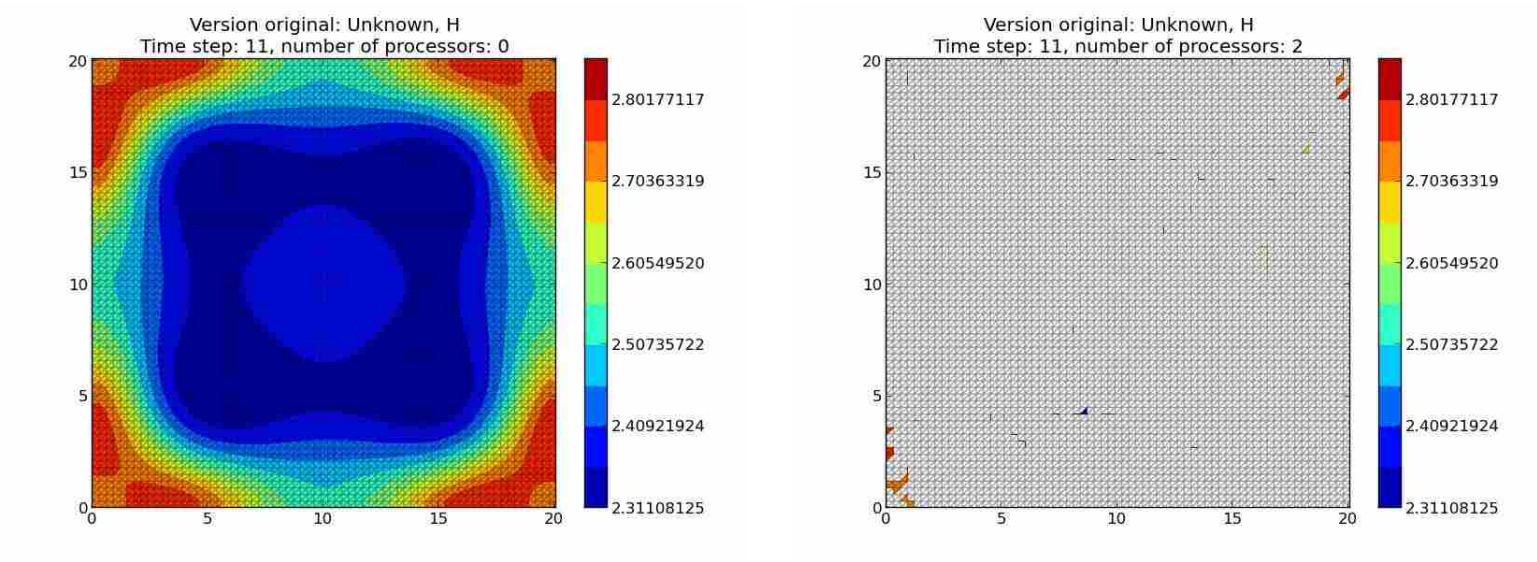
All I care about is the algorithm output

Did I mention we have parallel machines nowadays? 😉

A white plot displays a non-reproducible value

Numerical reproducibility?

time step = 11



Sequential

Parallel $p = 2$

Courtesy of P. Langlois and R. Nheili

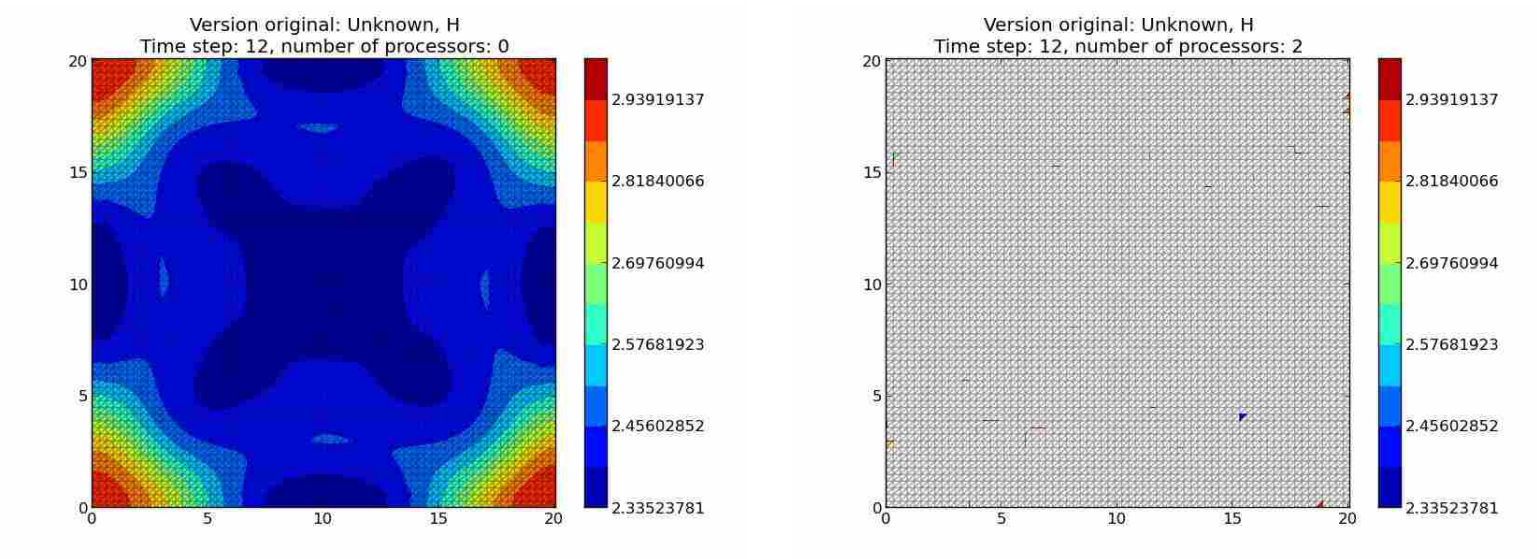
All I care about is the algorithm output

Did I mention we have parallel machines nowadays? 😉

A white plot displays a non-reproducible value

Numerical reproducibility?

time step = 12



Sequential

Parallel $p = 2$

Courtesy of P. Langlois and R. Nheili

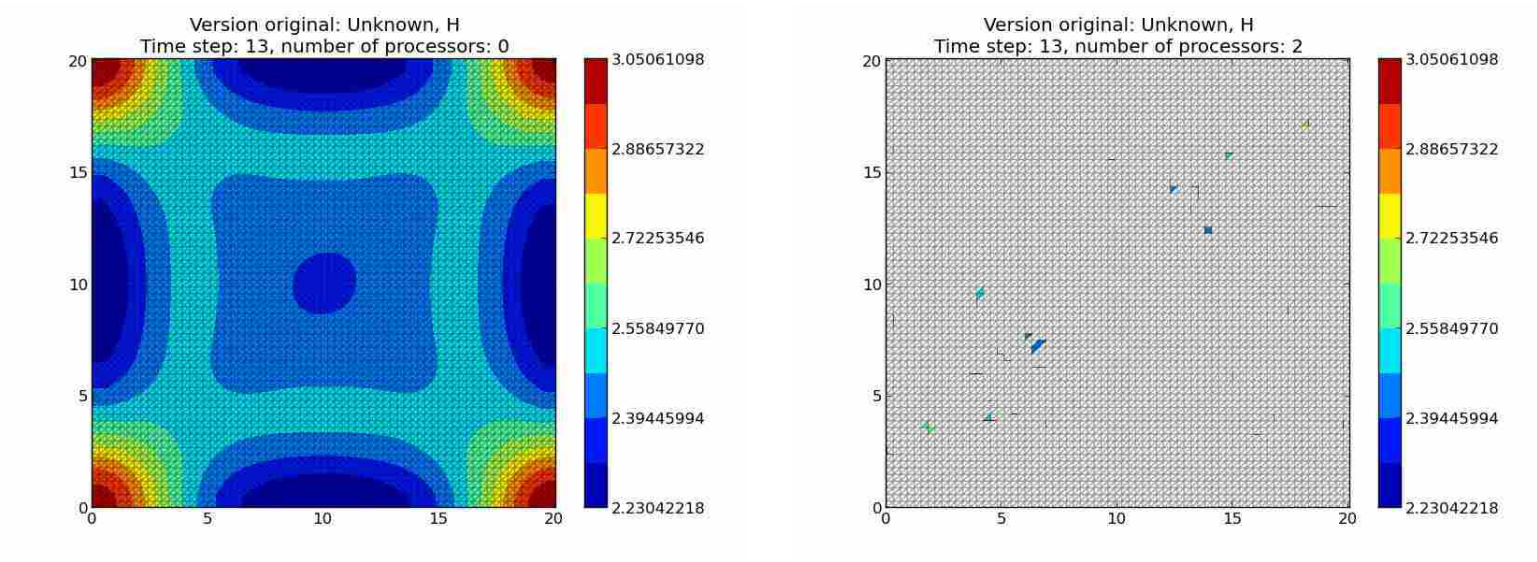
All I care about is the algorithm output

Did I mention we have parallel machines nowadays? 😉

A white plot displays a non-reproducible value

Numerical reproducibility?

time step = 13



Sequential

Parallel $p = 2$

Courtesy of P. Langlois and R. Nheili

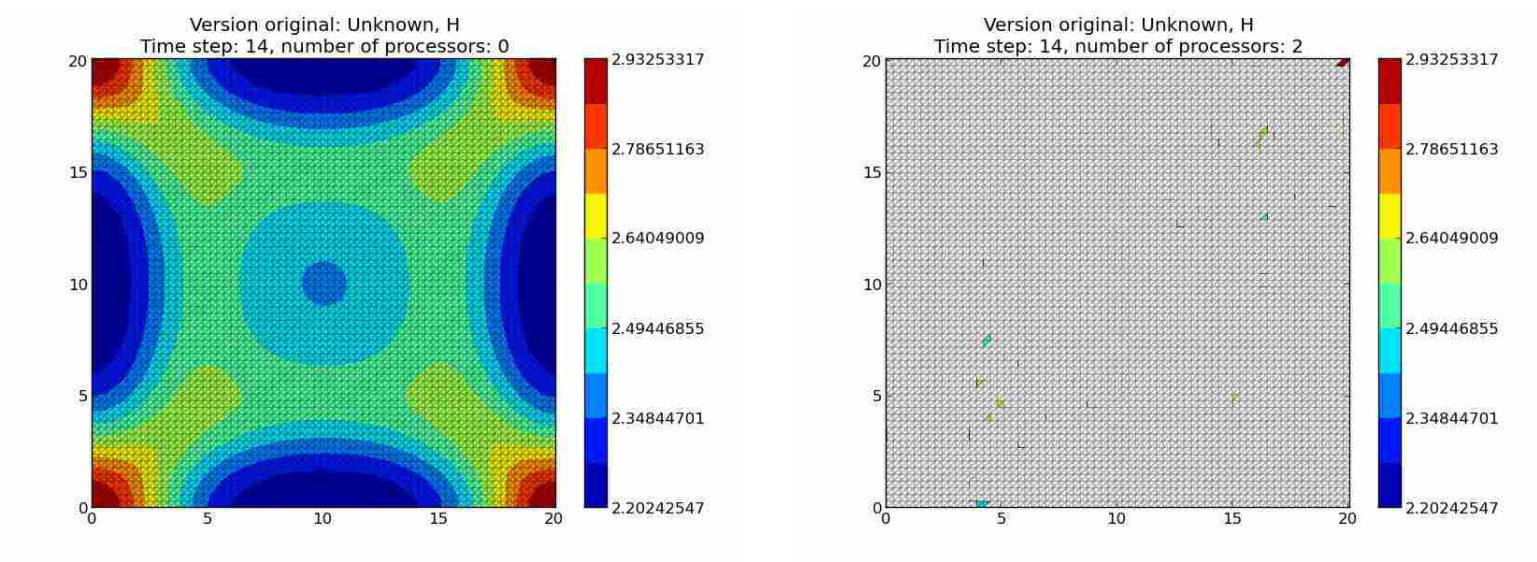
All I care about is the algorithm output

Did I mention we have parallel machines nowadays? 😉

A white plot displays a non-reproducible value

Numerical reproducibility?

time step = 14



Sequential

Parallel $p = 2$

Courtesy of P. Langlois and R. Nheili

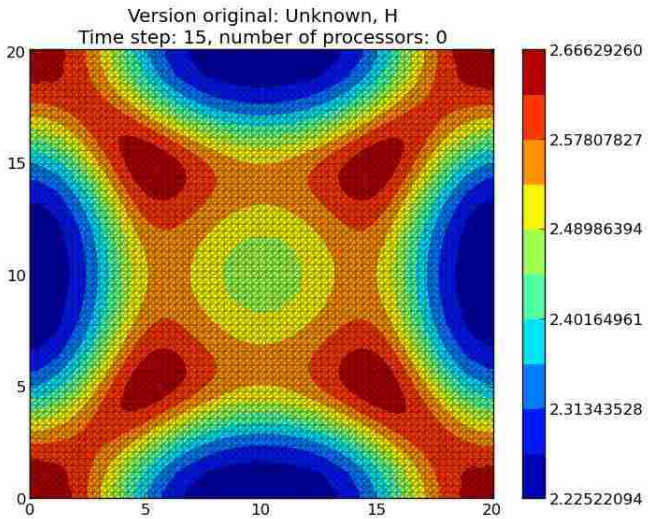
All I care about is the algorithm output

Did I mention we have parallel machines nowadays? 😉

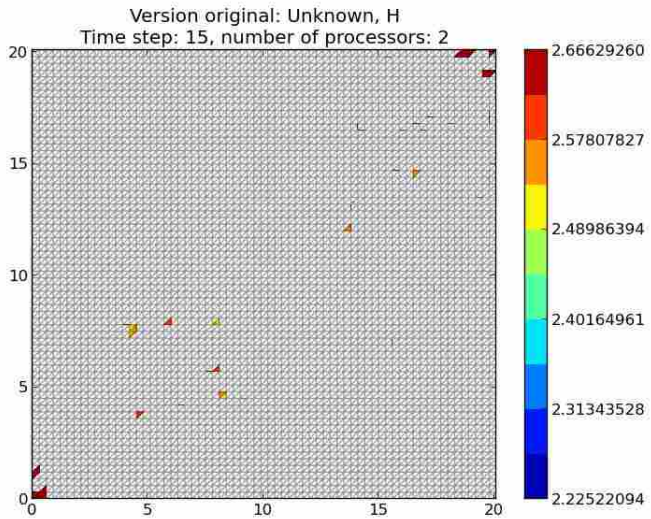
A white plot displays a non-reproducible value

NO numerical reproducibility!

time step = 15



Sequential



Parallel $p = 2$

Courtesy of P. Langlois and R. Nheili

All I care about is the algorithm output

Did I mention we have parallel machines nowadays? 😉

These numerical issues can become quite harmful in real use cases.

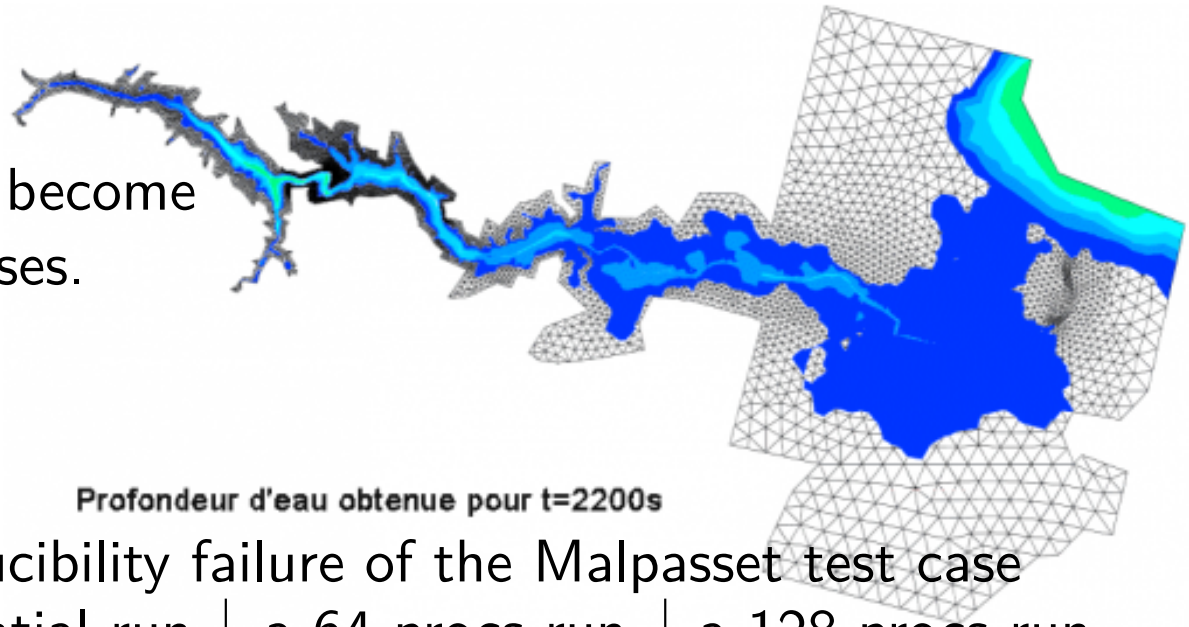


TABLE 1.1: Reproducibility failure of the Malpasset test case

	The sequential run	a 64 procs run	a 128 procs run
depth H	0.3500122E-01	0.2748817E-01	0.1327634E-01
velocity U	0.4029747E-02	0.4935279E-02	0.4512116E-02
velocity V	0.7570773E-02	0.3422730E-02	0.7545233E-02

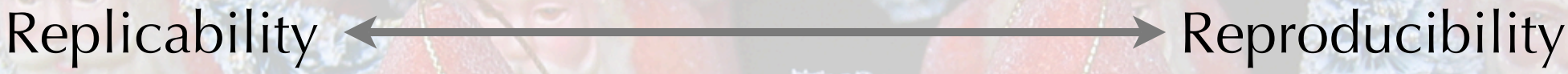
Numerical reproducibility?: Approximations in the model, in in the algorithm, in its implementation, in its execution.

The whole chain needs to be revisited.

Courtesy of P. Langlois and R. Nheili

Outline

Reproducibility: What Are We Talking About?



Reproduction of the original results using the same tools

by the original author on the same machine

by someone in the same lab/using a different machine

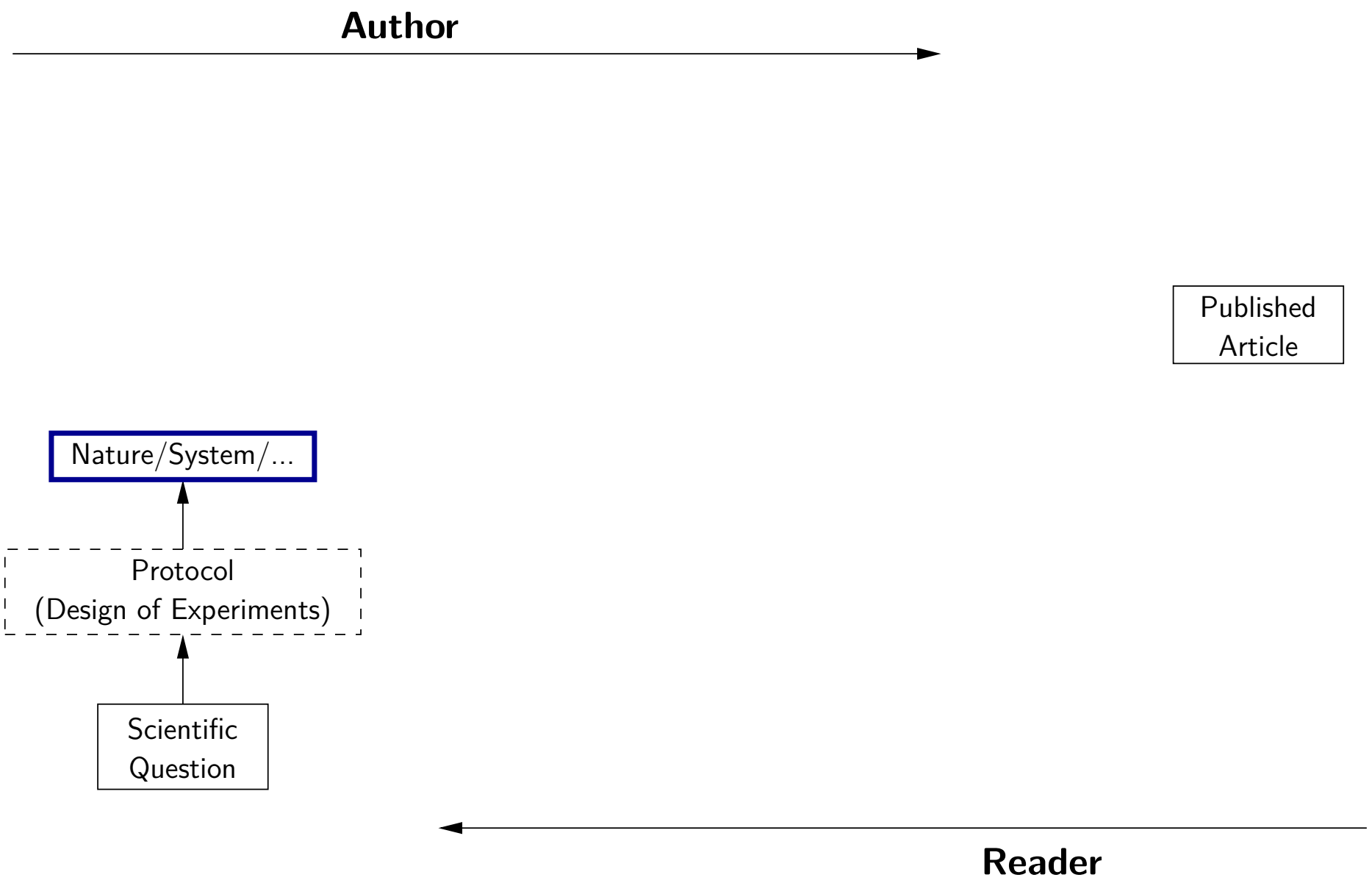
by someone in a different lab

Reproduction using different software, but with access to the original code

Completely independent reproduction based only on text description, without access to the original code

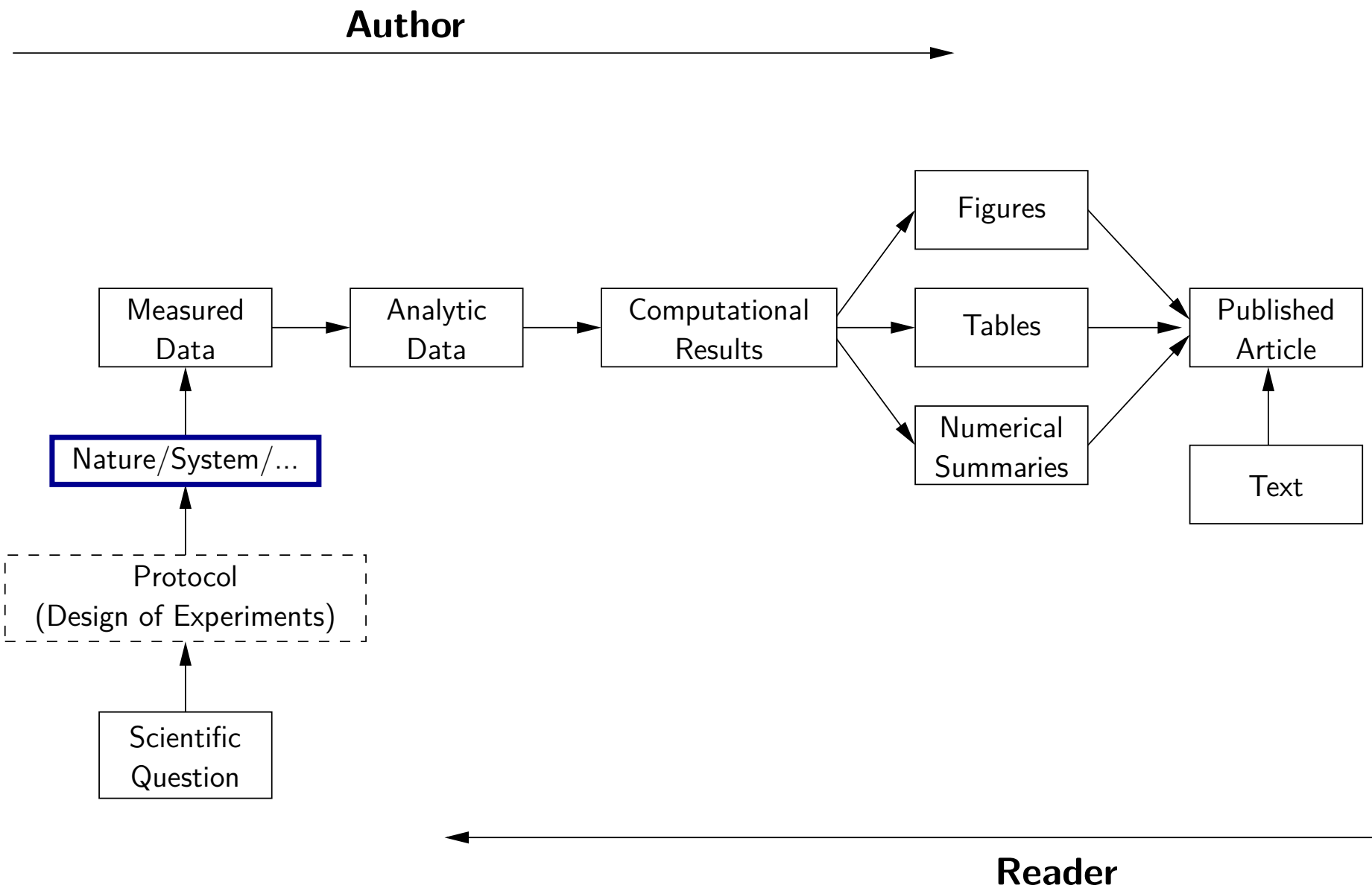
Courtesy of Andrew Davison (AMP Workshop on Reproducible research)

Reproducible Research: Trying to Bridge the Gap



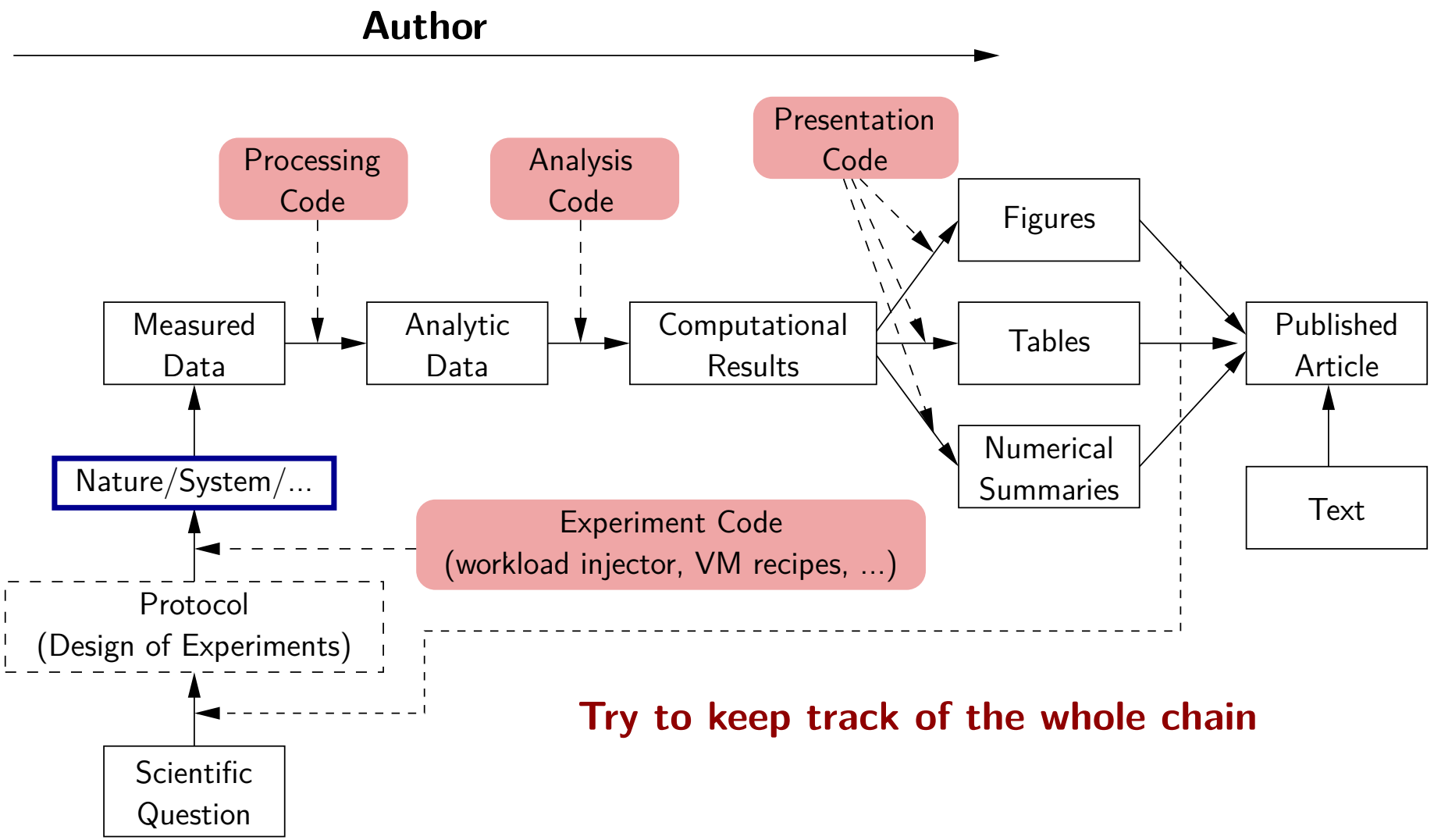
Inspired by Roger D. Peng's lecture on reproducible research, May 2014

Reproducible Research: Trying to Bridge the Gap



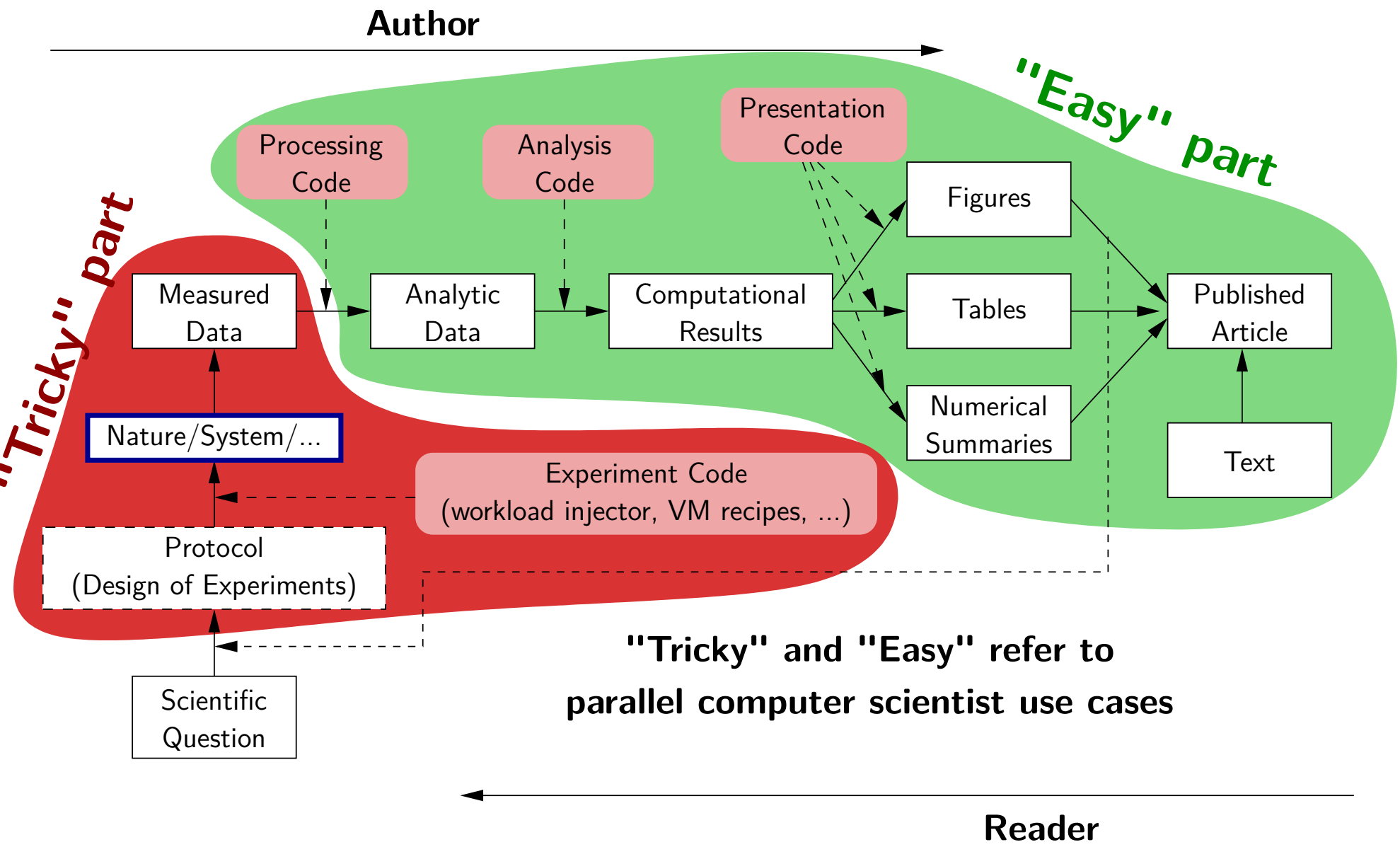
Inspired by Roger D. Peng's lecture on reproducible research, May 2014

Reproducible Research: Trying to Bridge the Gap



Inspired by Roger D. Peng's lecture on reproducible research, May 2014

Reproducible Research: Trying to Bridge the Gap



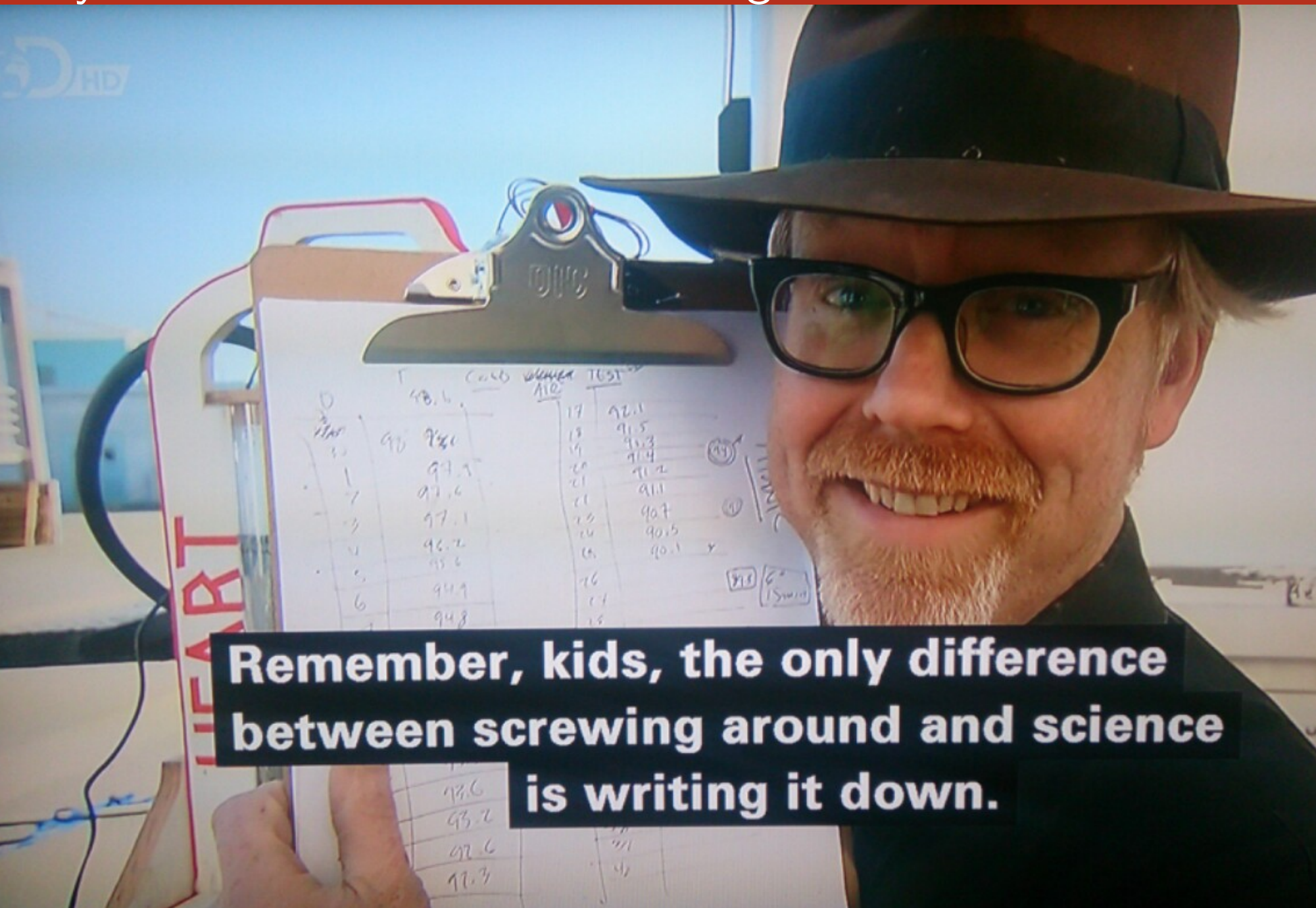
Inspired by Roger D. Peng's lecture on reproducible research, May 2014

Paradigm Shift

1. Lack of information, data access
2. Computation/programming/statistics mistakes
3. Lack of technical and scientific rigor

Transparency increases the chances of finding mistakes and getting rid of them

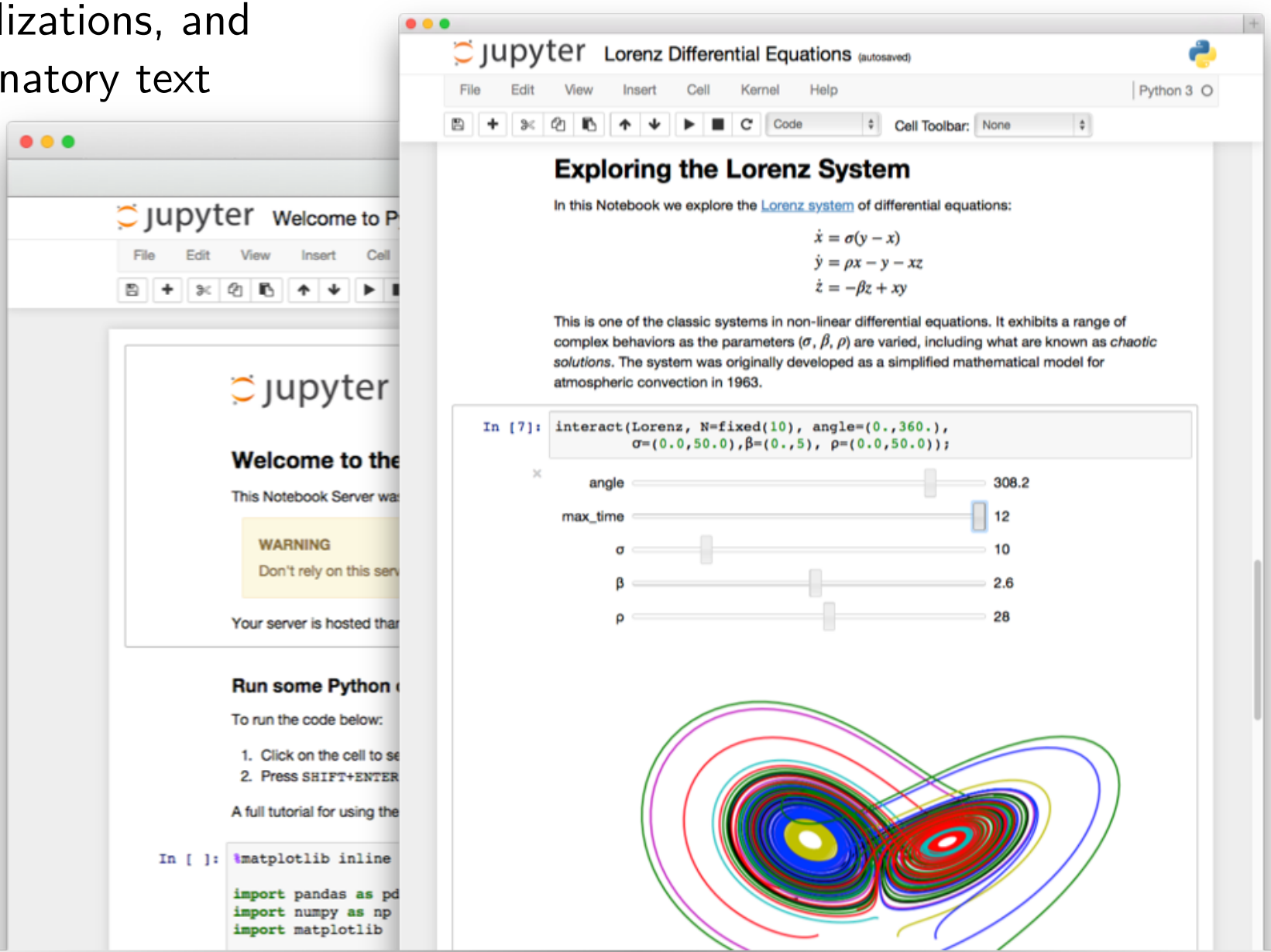
Mythbusters: Science vs. Screwing Around



Outline

Ipython/Jupyter Notebook

Web app: create and share documents that contain live code, equations, visualizations, and explanatory text



Computational Document (Jupyter Notebook)

Un document computationnel

Mon ordinateur m'indique que π vaut *approximativement*

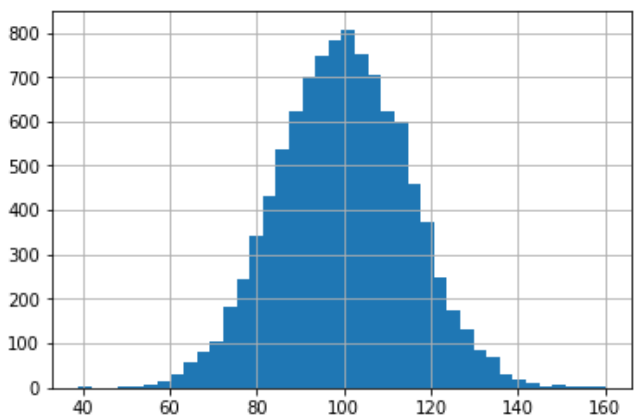
```
3.141592653589793
```

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... ☺).



Computational Document (Jupyter Notebook)

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... ☺).
```

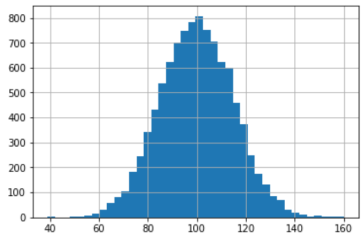
In [3]:

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*

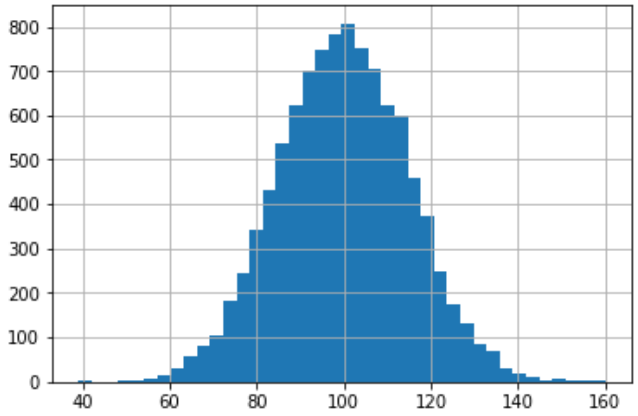
```
3.141592653589793
```

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... ☺).



Computational Document (Jupyter Notebook)

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... ☺).
```

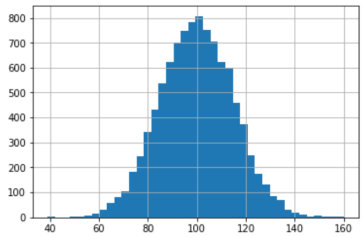
In [3]:

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()
```



Mark Down

Document final

Un document computationnel

Mon ordinateur m'indique que π vaut *approximativement*

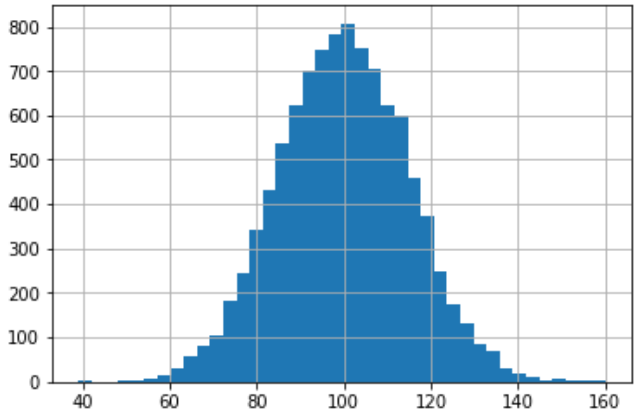
```
3.141592653589793
```

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... ☺).



Computational Document (Jupyter Notebook)

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... ☺).
```

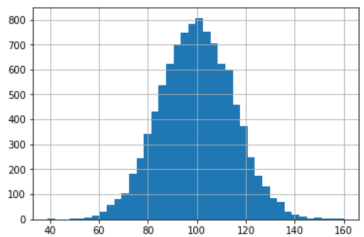
In [3]:

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()
```



Code

Document final

Un document computationnel

Mon ordinateur m'indique que π vaut *approximativement*

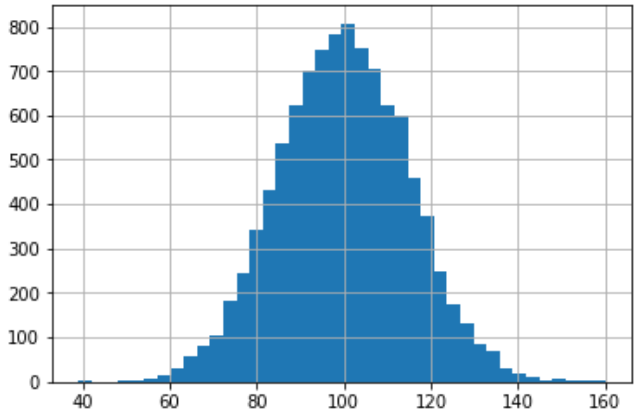
```
3.141592653589793
```

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... ☺).



Computational Document (Jupyter Notebook)

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... ☺).
```

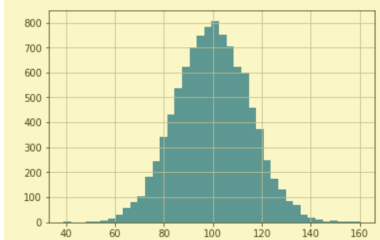
In [3]:

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()
```



Résultats

Document final

Un document computationnel

Mon ordinateur m'indique que π vaut *approximativement*

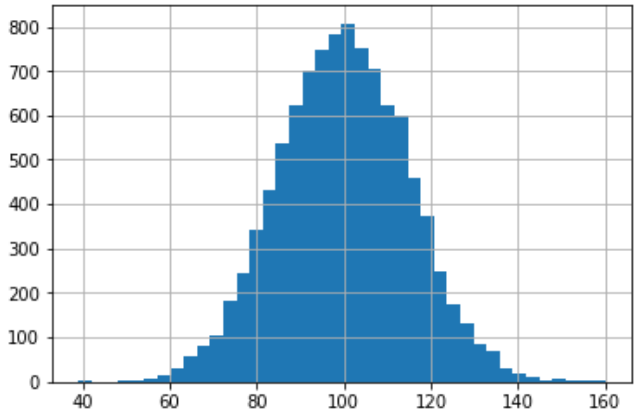
```
3.141592653589793
```

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... ☺).



Computational Document (Jupyter Notebook)

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... ☺).
```

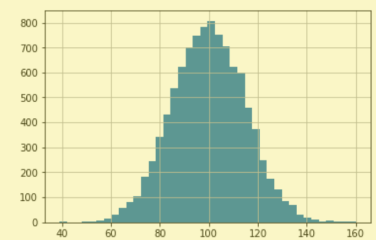
In [3]:

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*

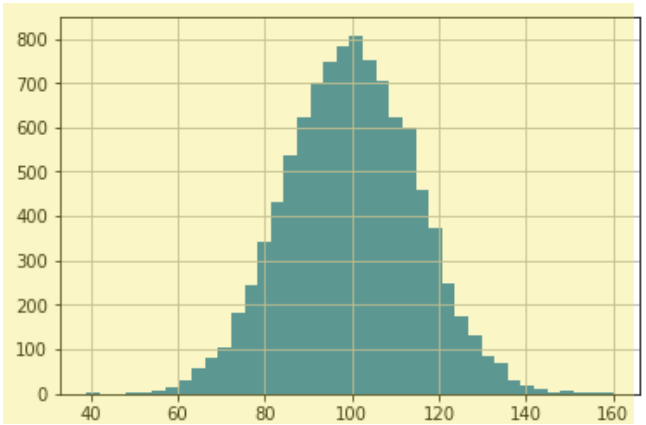
```
3.141592653589793
```

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... ☺).



Vistrails: a Workflow Engine for Provenance Tracking

Our Approach: An Infrastructure to Support Provenance-Rich Papers [Koop et al., ICCS 2011]

- Tools for *authors* to create reproducible papers
  - Specifications that encode the computational processes
  - Package the results
  - Link from publications

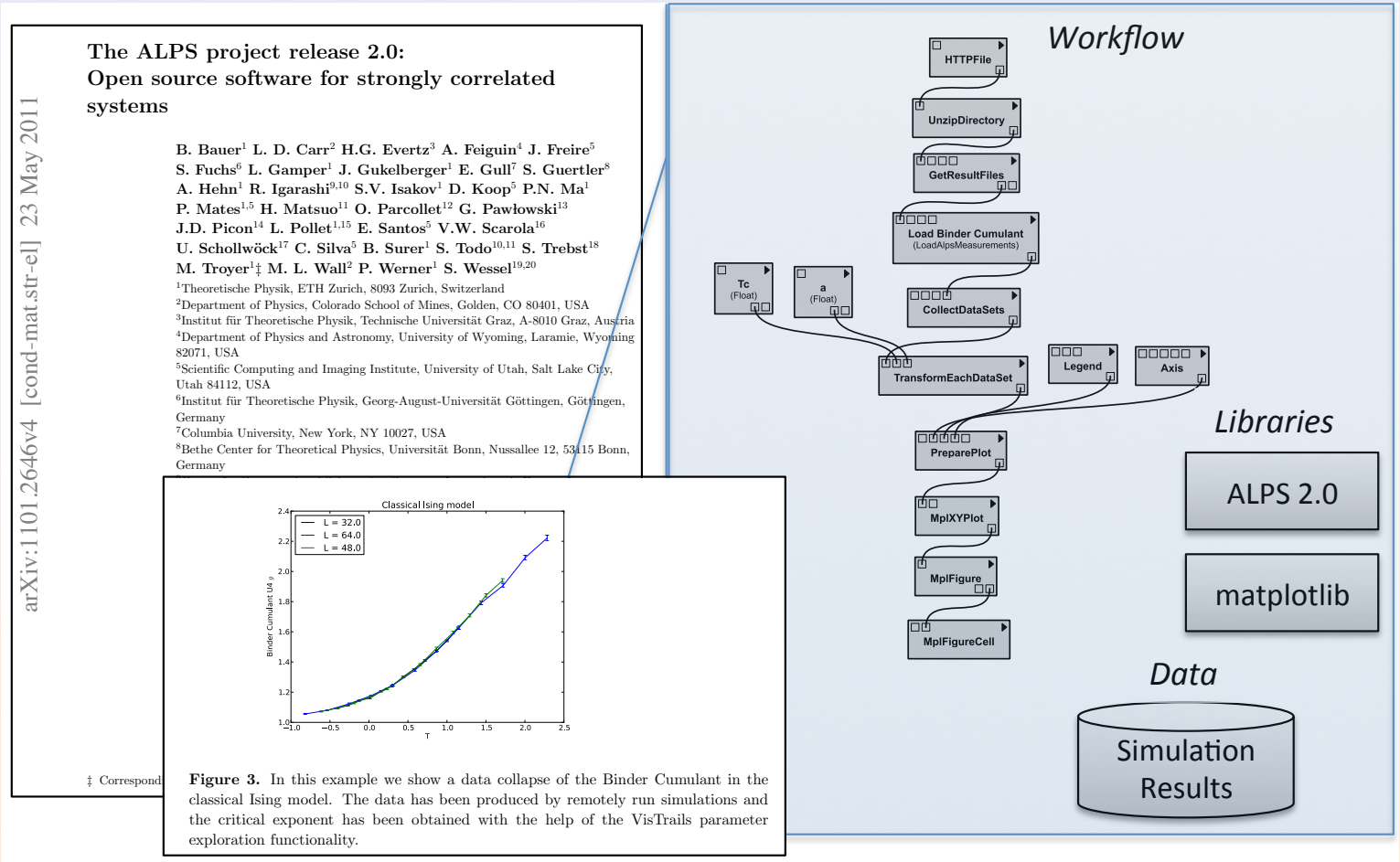
Support different approaches

- Tools for testers to repeat and validate results
  - Explore different parameters, data sets, algorithms
- Interfaces for searching, comparing and analyzing experiments and results
  - Can we discover better approaches to a given problem?
  - Or discover relationships among workflows and the problems?
  - How to describe experiments?

Courtesy of Juliana Freire (AMP Workshop on Reproducible research)

Vistrails: a Workflow Engine for Provenance Tracking

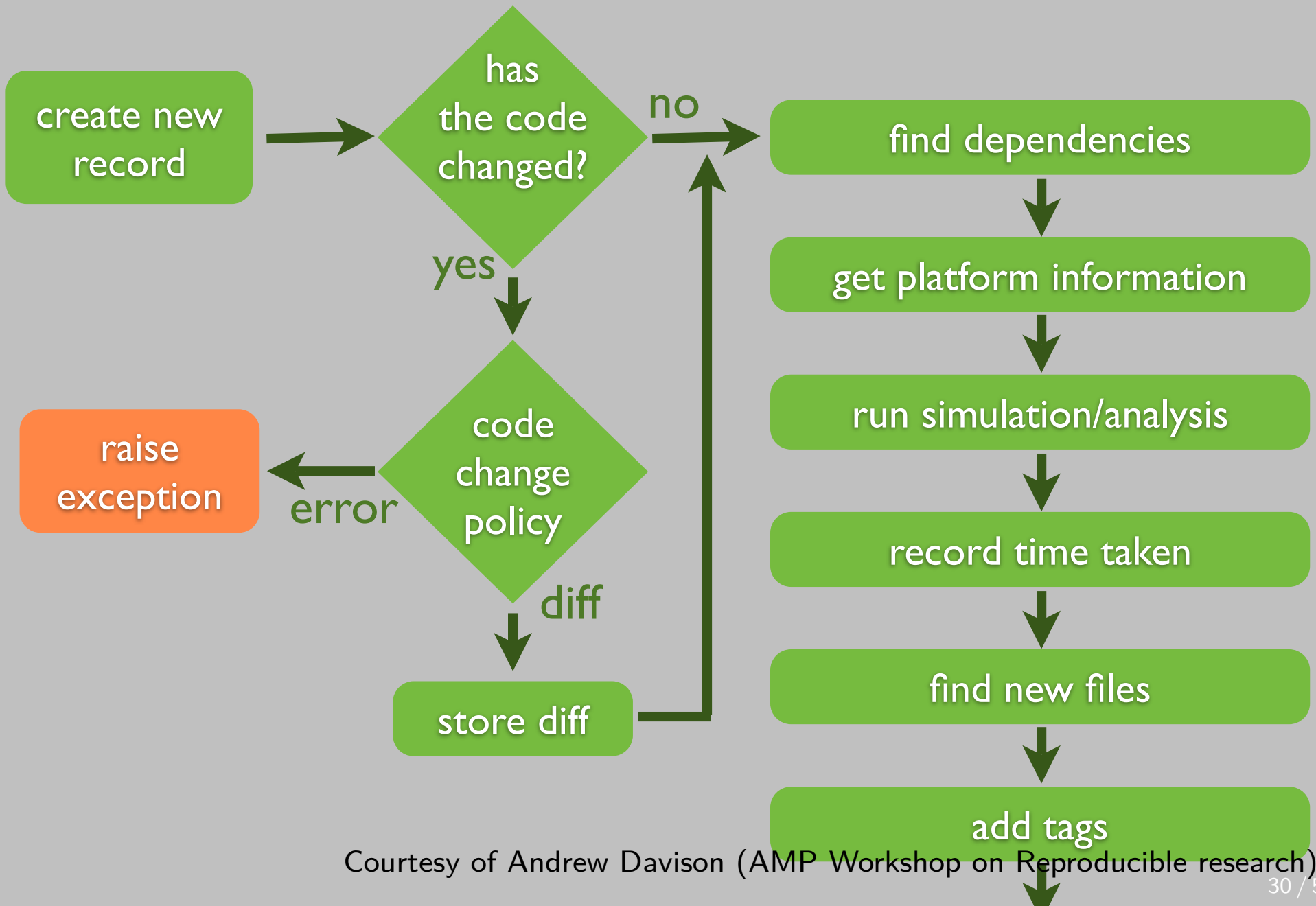
An *Provenance-Rich* Paper: ALPS2.0



[Bauer et al., JSTAT 2011]

Courtesy of Juliana Freire (AMP Workshop on Reproducible research)

Sumatra: an "experiment engine" that helps taking notes



Courtesy of Andrew Davison (AMP Workshop on Reproducible research)

Sumatra: an "experiment engine" that helps taking notes

```
$ smt comment 20110713-174949 "Eureka! Nobel prize
here we come."
```

Courtesy of Andrew Davison (AMP Workshop on Reproducible research)

Sumatra: an "experiment engine" that helps taking notes

```
$ smt tag “Figure 6”
```

Courtesy of Andrew Davison (AMP Workshop on Reproducible research)

Sumatra: an "experiment engine" that helps taking notes

Sumatra: TestProject: List of records

http://127.0.0.1:8002/

Google

TestProject: List of records

Delete / include data	Label	Reason	Outcome	Duration	Processes	Simulator		Script			Date	Time	Tags
						Name	Version	Repository	Main file	Version			
☐	20100709-154255		'Eureka! Nobel prize here we come.'	0.59 s		Python	2.5.2	/Users/andrew/tmp/SumatraTest	main.py	396c2020ca50	09/07/2010	15:42:55	
☐	20100709-154309			0.59 s		Python	2.5.2	/Users/andrew/tmp/SumatraTest	main.py	396c2020ca50	09/07/2010	15:43:09	
☐	haggling	'determine whether the gourd is worth 3 or 4 shekels'	'apparently, it is worth NaN shekels.'	0.59 s		Python	2.5.2	/Users/andrew/tmp/SumatraTest	main.py	396c2020ca50	09/07/2010	15:43:20	foobar
☐	20100709-154338	'test effect of a smaller time constant'		0.59 s		Python	2.5.2	/Users/andrew/tmp/SumatraTest	main.py	396c2020ca50	09/07/2010	15:43:38	
☐	haggling_repeat	Repeat experiment haggling	The new record exactly matches the original.	0.58 s		Python	2.5.2	/Users/andrew/tmp/SumatraTest	main.py	396c2020ca50	09/07/2010	15:43:47	

Courtesy of Andrew Davison (AMP Workshop on Reproducible research)

Complex Environments

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

```
1 2.1.1
```

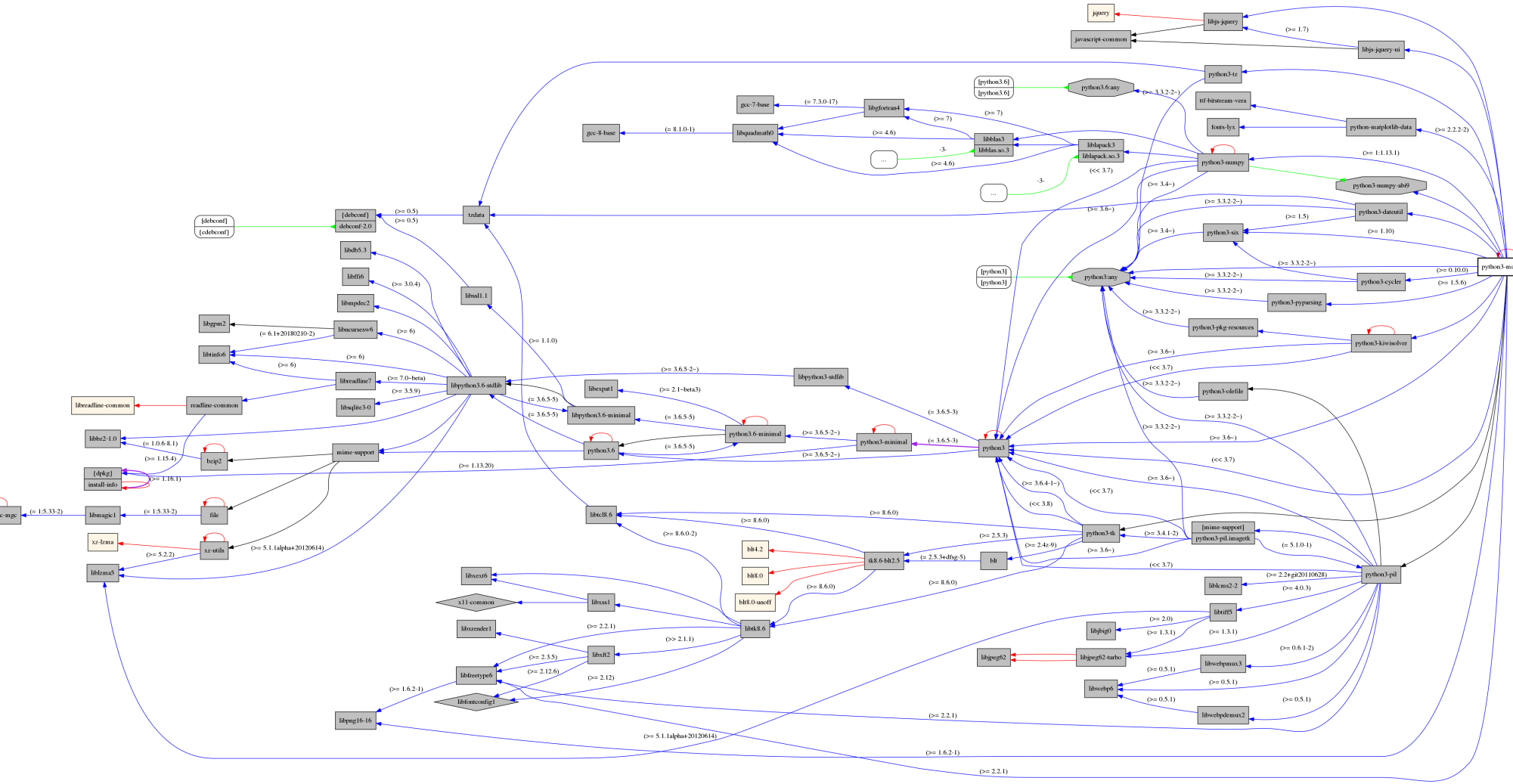
Complex Environments

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

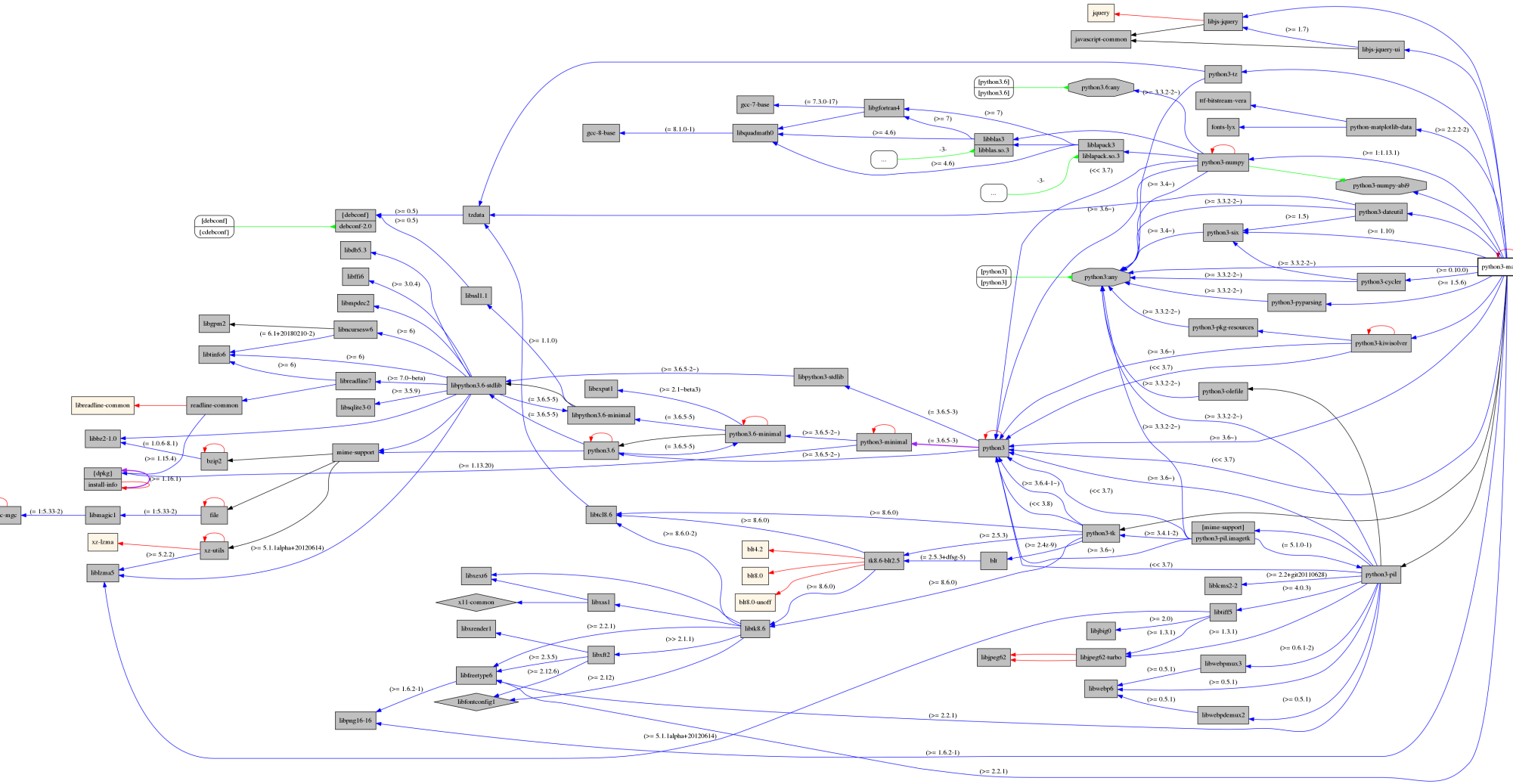
```
1 2.1.1
```

```
1 Version: 2.1.1-2
2 Depends: python3-dateutil, python-matplotlib-data (>= 2.1.1-2),
3 python3-pyparsing (>= 1.5.6), python3-six (>= 1.10), python3-tz,
4 libjs-jquery, libjs-jquery-ui, python3-numpy (>= 1:1.13.1),
5 python3-numpy-abi9, python3 (<< 3.7), python3 (>= 3.6~),
6 python3-cycler (>= 0.10.0), python3:any (>= 3.3.2-2~), libc6 (>=
7 2.14), libfreetype6 (>= 2.2.1), libgcc1 (>= 1:3.0), libpng16-16 (>=
8 1.6.2-1), libstdc++6 (>= 5.2), zlib1g (>= 1:1.1.4)
```

Complex Environments



Complex Environments



Virtual images or *Docker containers* allow to freeze environements

Two approaches: *preserve the mess* vs. *encourage cleanliness*

Reprozip

Automagically pack your experiment to fight dependency hell

ON THE ORIGINAL MACHINE

```
$ pip install reprozip
$ reprozip trace ./myexperiment -my --options inputs/somefile.csv other_file_here.bin
experiment: 0%... 25%... 50%... 75%... 100%
result: 42.137
Configuration file written in .reprozip/config.yml
Edit that file then run the packer -- use 'reprozip pack -h' for help
$ reprozip pack my_experiment.rpz
[REPROZIP] 17:26:42.588 INFO: Creating pack my_experiment.rpz...
[REPROZIP] 17:26:42.589 INFO: Adding files from package coreutils...
[REPROZIP] 17:26:42.601 INFO: Adding files from package libc6...
[REPROZIP] 17:26:42.906 INFO: Adding other files...
[REPROZIP] 17:26:43.450 INFO: Adding metadata...
```


ON ANOTHER MACHINE

```
$ pip install reprounzip[all]
$ reprounzip vagrant setup my_experiment.rpz mydirectory
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Importing base box 'remram/debian-7-amd64'...
==> default: Booting VM...
==> default: Machine booted and ready!
==> default: Running provisioner: shell...
$ reprounzip vagrant run mydirectory
experiment: 0%... 25%... 50%... 75%... 100%
result: 42.137
$ reprounzip vagrant upload /tmp/new_config:global-config
$ reprounzip vagrant run mydirectory --cmdline ./myexperiment --other --options
inputs/somefile.csv
experiment: 0%... 25%... 50%... 75%... 100%
result: -17.814
```

New Tools for Computational Reproducibility

- Dissemination Platforms:
  - ResearchCompendia.org
  - IPOL
  - Madagascar
  - MLOSS.org
  - thedatahub.org
  - nanoHUB.org
  - Open Science Framework
  - The DataVerse Network
  - RunMyCode.org
- Workflow Tracking and Research Environments:
  - VisTrails
  - Kepler
  - CDE
  - Galaxy
  - GenePattern
  - Synapse
  - Sumatra
  - Taverna
  - Pegasus
- Embedded Publishing:
  - Verifiable Computational Research
  - Sweave
  - knitR
  - Collage Authoring Environment
  - SHARE

Courtesy of Victoria Stodden (UC Davis, Feb 13, 2014)

And also: **Org-Mode** ☺, **Figshare**, **Zenodo**, **ActivePapers** ☺, **GUIX/Nix**, **Elsevier executable paper** ☹, ...

Outline

A Difficult Trade-off

Many different tools/approaches developed in various communities

But mainly two approaches: Automatic vs. Explicit

- **Automatically keeping track of everything**
  - the code that was run (source code, libraries, compilation procedure)
  - processor architecture, OS, machine, date, ...
- **Ensuring others can understand/adapt what was done**
  - Why did I run this? Does it still work when I change this piece of code for this one?

A Difficult Trade-off

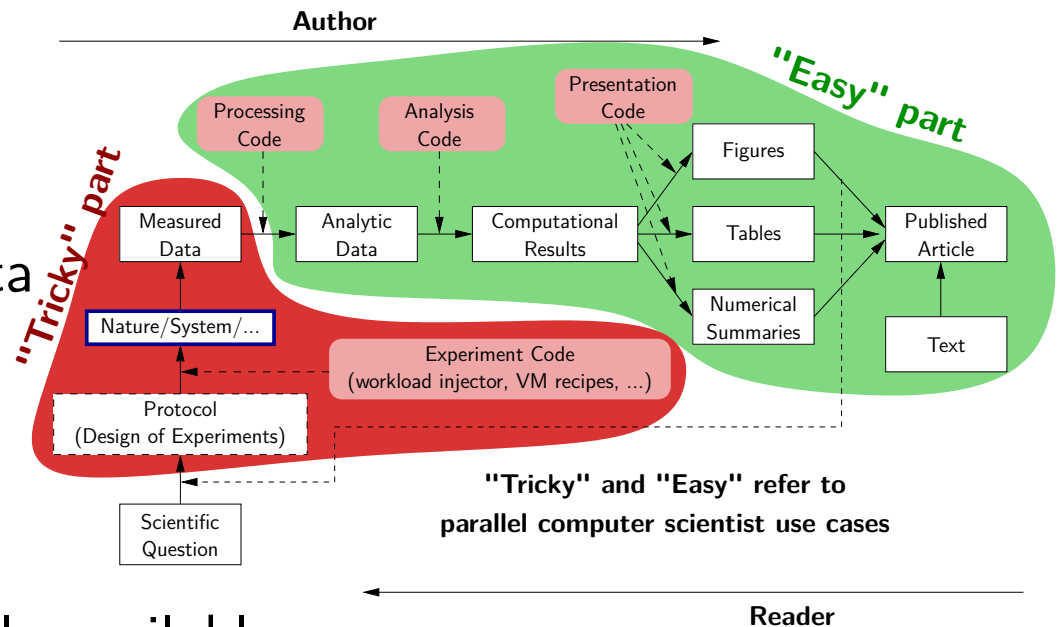
Many different tools/approaches developed in various communities

But mainly two approaches: Automatic vs. Explicit

- **Automatically keeping track of everything**
  - the code that was run (source code, libraries, compilation procedure)
  - processor architecture, OS, machine, date, ...
- **Ensuring others can understand/adapt what was done**
  - Why did I run this? Does it still work when I change this piece of code for this one?

And the following key points:

1. Replicable article
2. Logging your activity
3. Logging and backup your data
4. Organizing your data
5. Mastering your environment
6. Controlling your experiments
7. Making your data/code/article available



3. Logging and backup your data

What are the options?

- Nothing ☹ (remember the funny examples from the beginning. . . 😉)
- Incremental backup mechanisms (e.g., time machine)
- The cloud! (e.g., Dropbox and Google Drive ☹ . . .)
- Flexible version control systems (e.g. git 🙂) where you're in control of what's happening
  - Use a crontab if you really do not want to think about it
  - We have come up with a specific git branching workflow for managing experimental results

4. Organizing and managing your data

- Use the machine readable CSV format
- Provide raw data and meta data, not just statistical outputs
- Organization
  - Explain your conventions (e.g., `src/`, `data/`, `script/`, `journal.org`)
  - Git submodules
- Never do data manipulation and statistical tests by hand or with a spreadsheet 
- Use R, Python or another free software to read and process raw data.
  - Use a workflow that documents both data and process
  - The org-mode tangling mechanism may help

5. Mastering your environment

What are the options?

- Nothing 😉 No, it's not, you have to do something...
- Restrict your tools/dependencies to the bare minimum (e.g., python)
  - List them all manually in a README
  - Use custom shell scripts or `sosreport` that log all the dependencies you are aware. Ask your friends to check whether this is sufficient...
  - Combine everything in *activepapers*, i.e., an HDFS5 file combining datasets and programs working on these datasets in a single package, along with meta data, history, ...
- Create and distribute your own virtual image (VM, docker, Singularity)
- Have tools that automatically keep track of dependencies/files and packages up the Code, Data, and Environment
  - CDE (Guo et al., 2011) ReproZip (Freire et al., 2013), CARE (Janin et al., 2014),
  - See Preserve the Mess or Encourage Cleanliness? (Thain et al., 2015)
- Use a specific tool to generate customized appliances (kvm, LXC, Virtualbox, iso, ...): recipes with steps and aliases, execution in contexts, checkpoints, ... (*Kameleon*)

6. Controlling your experiments

- Naive way: sh + ssh + ...
- Better way: use a workflow management system (taverna, galaxy, kepler, vistrails, ...)
- Parallel/distributed experiments require specific experiment engines
  - Expo (2007-, G5K)
  - XPflow (2012-, G5K)
  - Execo (2013-, G5K)

  } although nothing specific to G5K

  - Plush (2006-, Planetlab)
  - OMF (2009-, Wireless)
  - Splay (2008), ...

  They differ in the underlying paradigms and the platforms for which they have been designed

  > A survey of general-purpose experiment management tools for distributed systems, T. Buchert, C. Ruiz, L. Nussbaum, O. Richard, FGCS, 2014

- Control your **numerical results** (random generators, libraries, rounding and non-determinism, ...)

7. Making your data/code/article available

- Your webpage ☹
- Figshare, Zenodo ☺, ...
- Companion websites (elsevier executable paper ☹, runmycode, exec&share ☺, ...)
- Github (damn, they're good! ☺), ...

This may seem easy but is more tricky than it looks like:

- Arbitrary limits can make your life painful
- Perennity (Roberto Di Cosmo's talk at R^4)
  - CodeSpaces murdered on Amazon, Google Code termination, Gitorious shutdown, ...
  - Disruption of the web of reference: URLs decay (half-life of 4 years), DOIs have little guarantee, ...

Many legal aspects about data/code/idea sharing

- I am a civil servant and I strongly believe research is a team sport
- Intellectual property is an important topic we do not want to leave to bureaucrats and lawyers...

Remember the general picture

The article is only the top of the iceberg, we need a way to dive and unveil what's behind every graphics and number...

1. Replicable article (Literate programming)

Donald Knuth: explanation of the program logic in a natural language interspersed with snippets of macros and traditional source code.

I'm way too 3133t to program this way 🙂 but that's
exactly what we need for writing a reproducible article/analysis!

KnitR (a.k.a. Sweave)

For R and ~~emacs~~ users. Easy replicable articles with a modern IDE (e.g., Rstudio)

Ipython/Jupyter notebook

Python user $\rightsquigarrow$ go for Jupyter. Web app, easy to use/setup... Writing replicable article may be tricky though

Org-mode (my favorite! requires emacs though)

- Org-mode is plain text, very smooth, works both for html, pdf, ...
- Allows to combine all my favorite languages

Note that this generation depends on a computational environment whose preservation is not addressed here (see for example *activepapers*).

A replicable article with Org-Mode

See for example our recent article on the simulation of Multithreaded Sparse Linear Algebra Solvers at ICPADS 2015.

Here are the following important features to exploit:

Structure highly hierarchical

- Sectioning, itemize, enumerate, fonts
- Tags to control what will be exported

Export in several output formats

- Fine control with `#+BEGIN_EXPORT latex`
- Unfortunate need for verbose headers (because of LaTeX☹) and black magic in the end of the file (for emacs portability ☹)

Babel (the literate programming part of org-mode). Many possible usage:

- Run babel on export
- Or not... and make sure intermediate results are stored (this is how I proceed)
- Dependencies can be expressed
- Caching mechanism
- Side effects are the enemy of reproducibility

2. Logging your activity (Laboratory Notebook)

Do not tie your hands with non-free software like Evernote or OneNote

- Org-mode again!
  - Capture mechanism (notes, todo, . . .)
  - Babel favors code reuse, ssh connections in sessions, meta-programming
  - Tagging mechanism to structure the journal
  - Link mechanism, Todo, Calendar views, Tables, . . .

I have a very intense usage and so do all my master/PhD students (e.g., here)

- Spending more than an hour without at least writing what you're working on is not right. . . Take a 5 minutes break and ask yourself what you're doing, what is keeping you busy and where all this is leading you
- While working on something, you will often notice/think about something you should fix/improve but you just don't want to do it now. Take 20 seconds to write a TODO entry
- There are moments where you have to wait for something (compiling, deployment, . . .). It is generally the perfect time for improving your notes (e.g., detail the steps to accomplish a TODO entry)
- By the end of the day: daily (and weekly) review!
  - Update your lists, decide the next steps, summarize what you did/learnt,. . .

Pros and Cons of these three tools

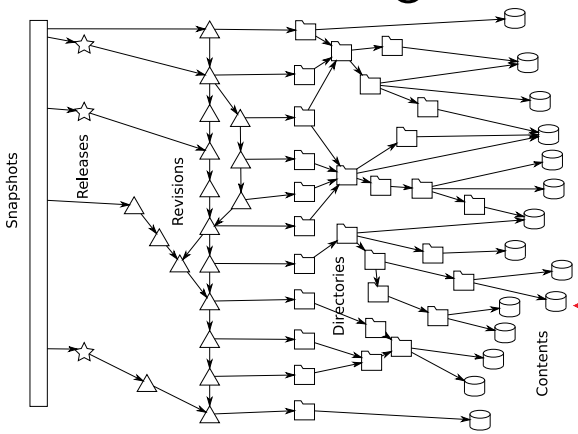
- Ipython notebook:
  - ☺ Easy to set up, user-friendly, machine readable format (JSON), easy sharing on the cloud
  - ☹ Writing an article, JSON, not fully polyglot
- knitR/Rstudio:
  - ☺ Easy to set up, user-friendly, writing articles, easy publishing on rpubs
  - ☹ not fully polyglot
- Emacs/Org-mode:
  - ☹ Emacs, steep learning curve
  - ☺ Powerful and versatile, yields control to power users, works both for writing articles and a notebook, good integration on github

The ultimate tool would combine an engine in an editor that allows collaborative interactive edition

Outline

On the "technical" side (1/2)

- Better documenting what we do: Laboratory notebooks
  - Literate programming is great for analysis, and reproducible articles but does not go well yet with conducting experiments and workflows
  - A real adoption of such practice requires more storage and the ability to navigate in such information
- Better software engineering practice: Public releases, devops approach, reproducible builds, numerical aspects
  - Moving/evolving technology. Preservation ? Adoption ?
  - Should not slow down research
- Fighting against software/data degradation: Software Heritage, zenodo
  - Challenges: multiple! curation, access/privacy, exploitation, navigation, storage, ...

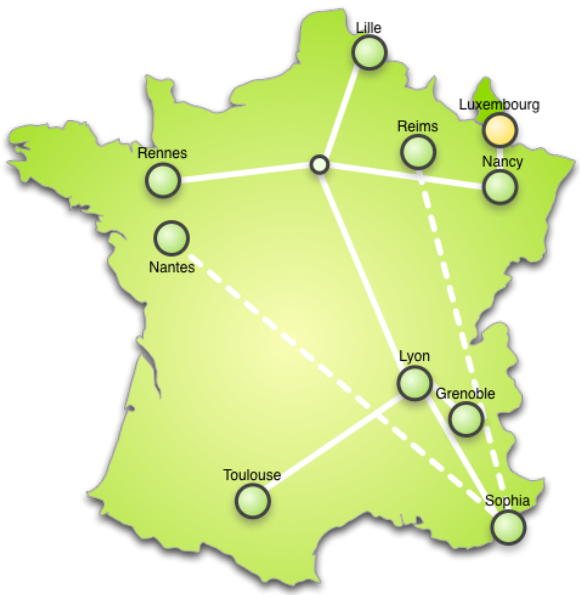


Software Heritage

THE GREAT LIBRARY OF SOURCE CODE

On the "technical" side (2/2)

- Better experimental practice and platforms: FIT IoT-lab, G5K are world leading experimental infrastructures; rely on standard simulators (SimGrid, NS3)
  - Maintenance cost, keeping in pace with technology, practices for prototype platforms, control, sharing of experimental conditions with others, experimental engines



- Need for convergence in term of software infrastructure and practice (e.g., security, account management, access, isolation, experiment management, etc.) ?

On the "social" side

Slight cultural changes in our relation to publication and daily practice

- Changing our social model to favor adoption of better practice
  - Artifact evaluation, open reviews, ... (e.g., IPOL, ReScience)
  - Promote a different model
- Learning is the essence of our work. $\rightsquigarrow$ Train our researchers and students
  - Better teaching/understanding of statistics, experimental practice, design of experiments

It's up to us. **We** should **care** and **take the lead**

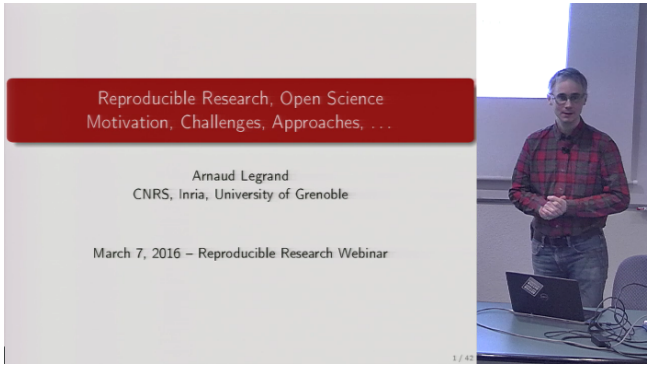
Main benefit:

- Higher confidence in our work $\rightsquigarrow$ definite competitive advantage
- Our research becomes sound, deeper, auditable, more visible, reusable, ...

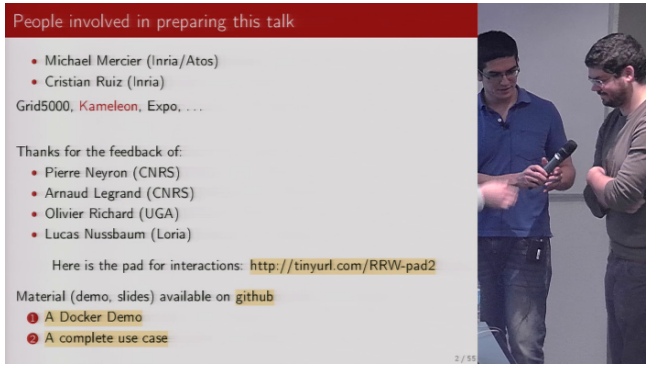
Webinars: Learning by Doing

Many different tools/approaches developed in various communities

1. Replicable article
2. Logging your activity
3. Logging and backing up your data
4. Organizing your data
5. Mastering your environment
6. Controlling your experiments
7. Making your data/code/article available
8. Publication modes
9. Artifact Evaluation



Litterate programming



Controling your environment

Numerical reproducibility

Logging and backing up

https://github.com/alegrand/RR_webinars