Reproducible Research and Computer Science

Arnaud Legrand

RSD Winter School, Le Pleynet
February 2024

Quick Poll

- Statistical language:

Quick Poll

- Statistical language:

- Laboratory Notebook

Quick Poll

- Statistical language:
- Laboratory Notebook
- Computationl Notebook

Quick Poll

- Statistical language:
- Laboratory Notebook
- Computationl Notebook
- Version Tracking





Quick Poll

- Statistical language:
- Laboratory Notebook
- Computationl Notebook
- Version Tracking
- Data Archiving








Quick Poll

- Statistical language:
- Laboratory Notebook
- Computationl Notebook
- Version Tracking
- Data Archiving
- Containers










Quick Poll

- Statistical language:
- Laboratory Notebook
- Computationl Notebook
- Version Tracking
- Data Archiving
- Containers
- Experimental Engine

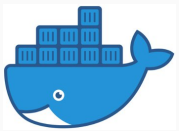













Quick Poll

- Statistical language:
- Laboratory Notebook
- Computationl Notebook
- Version Tracking
- Data Archiving
- Containers
- Experimental Engine

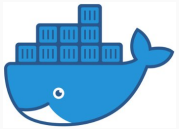






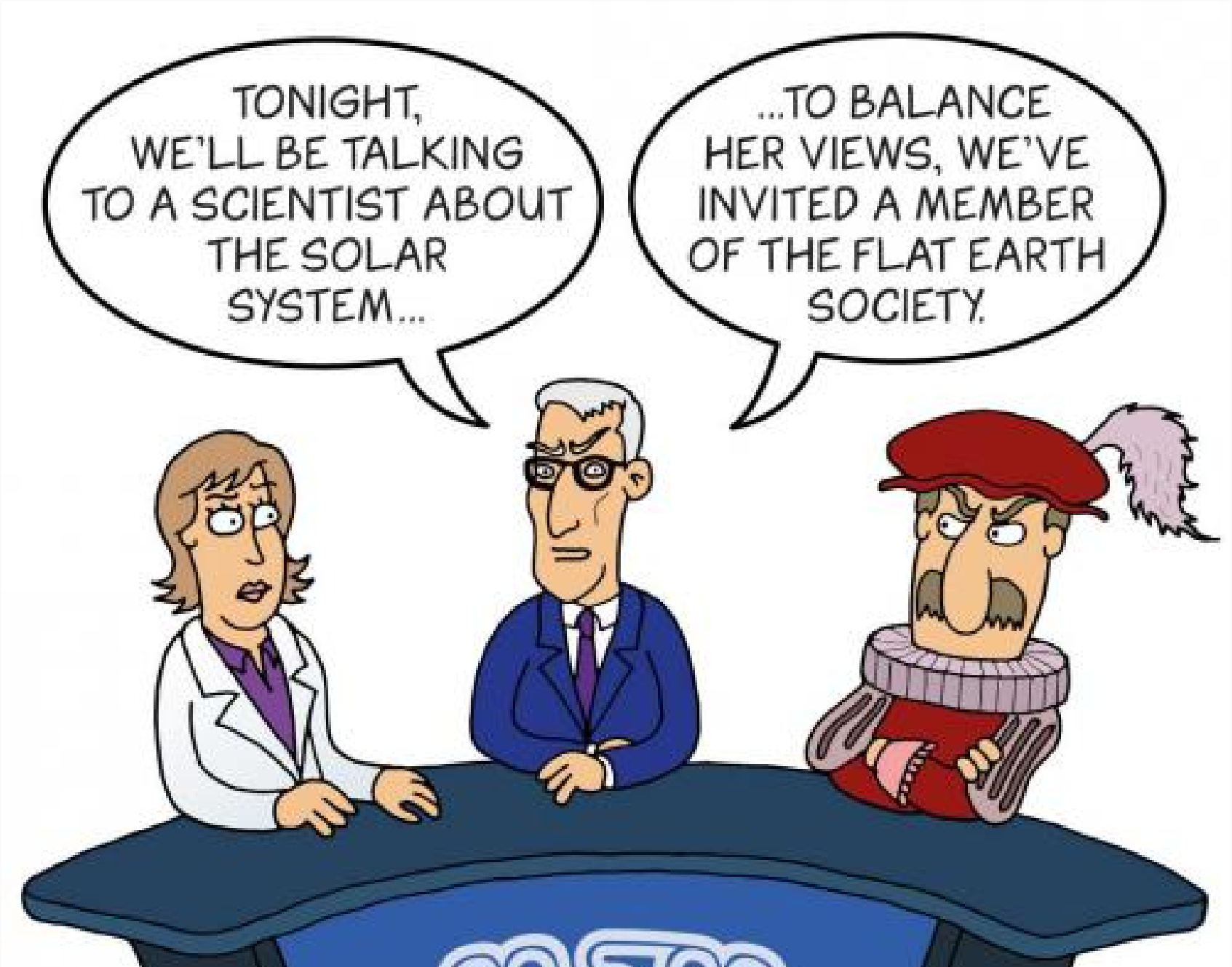




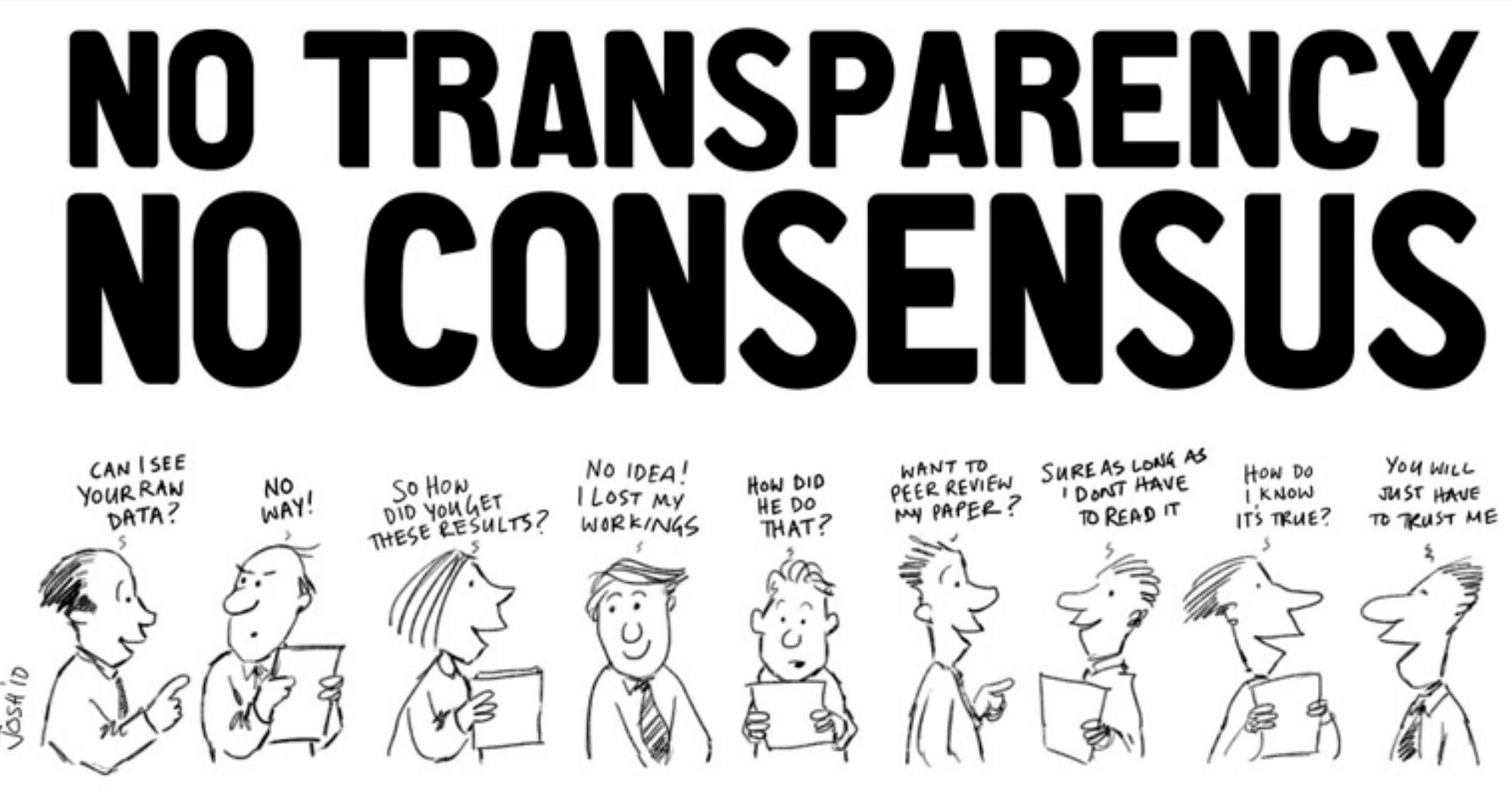



EnOSlib

Scientific Consensus vs. Democracy and Freedom of Speech

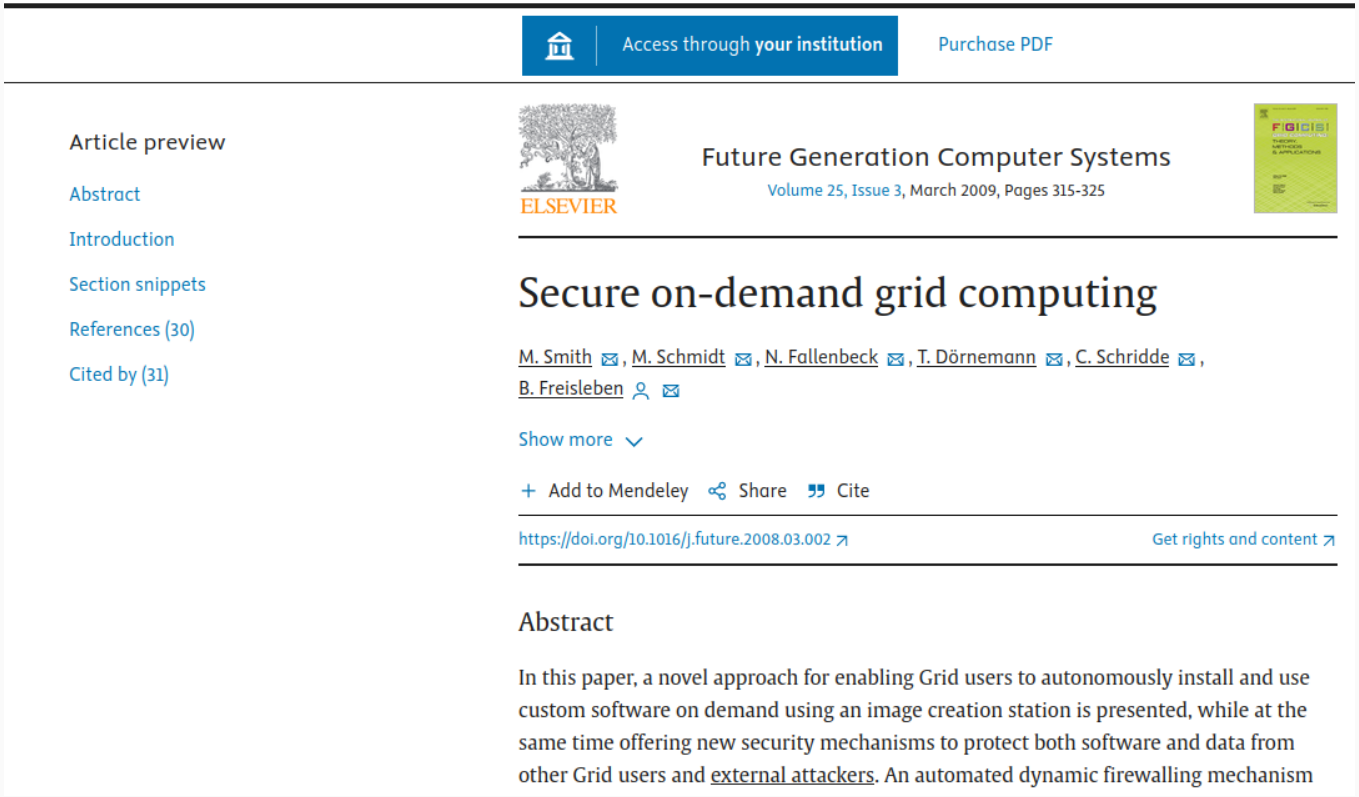


Scientific Consensus



Common Reproducibility Pitfalls

Go read the paper by Smith et. al. 2009



Access through your institution Purchase PDF

Article preview

Abstract

Introduction

Section snippets

References (30)

Cited by (31)

ELSEVIER

Future Generation Computer Systems

Volume 25, Issue 3, March 2009, Pages 315-325

Secure on-demand grid computing

M. Smith, M. Schmidt, N. Fallenbeck, T. Dörnemann, C. Schridde, B. Freisleben

Show more

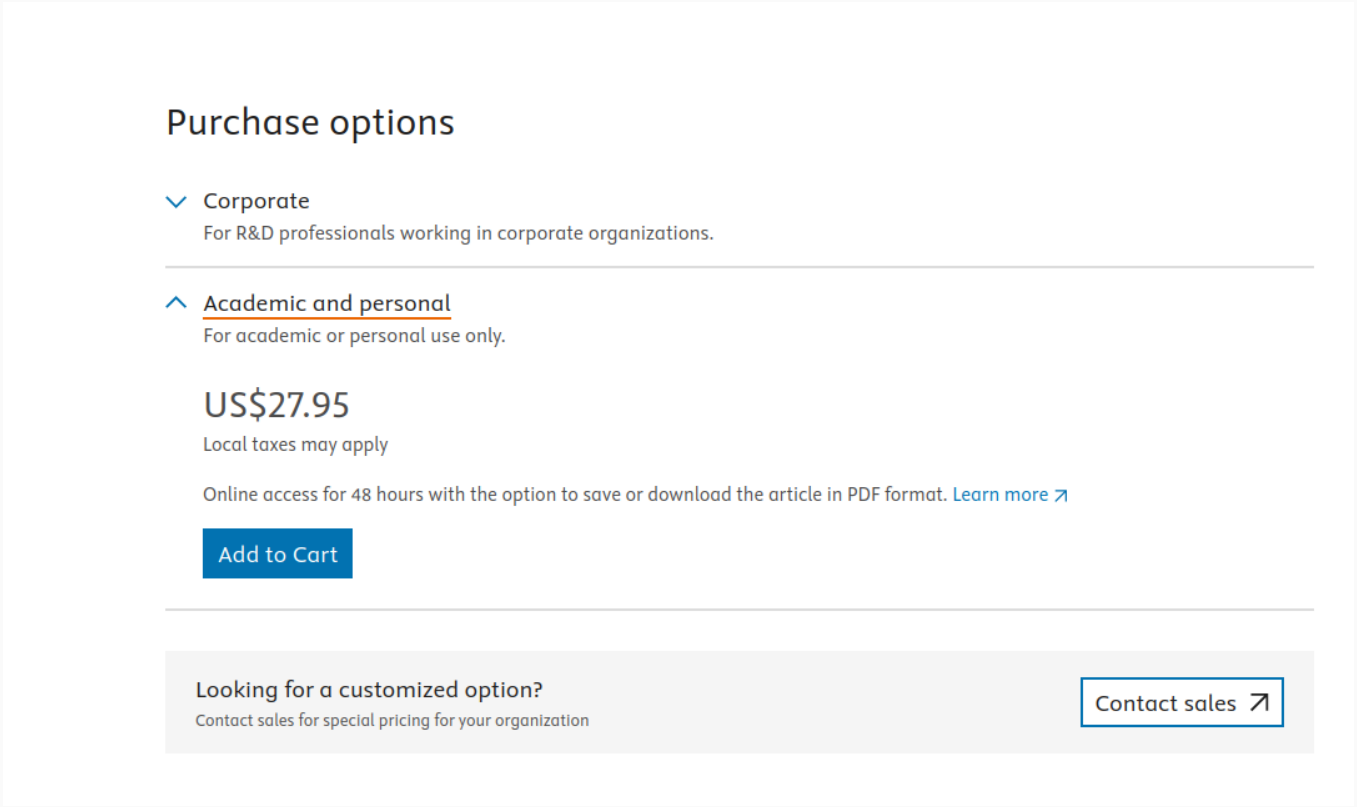
+ Add to Mendeley Share Cite

https://doi.org/10.1016/j.future.2008.03.002 Get rights and content

Abstract

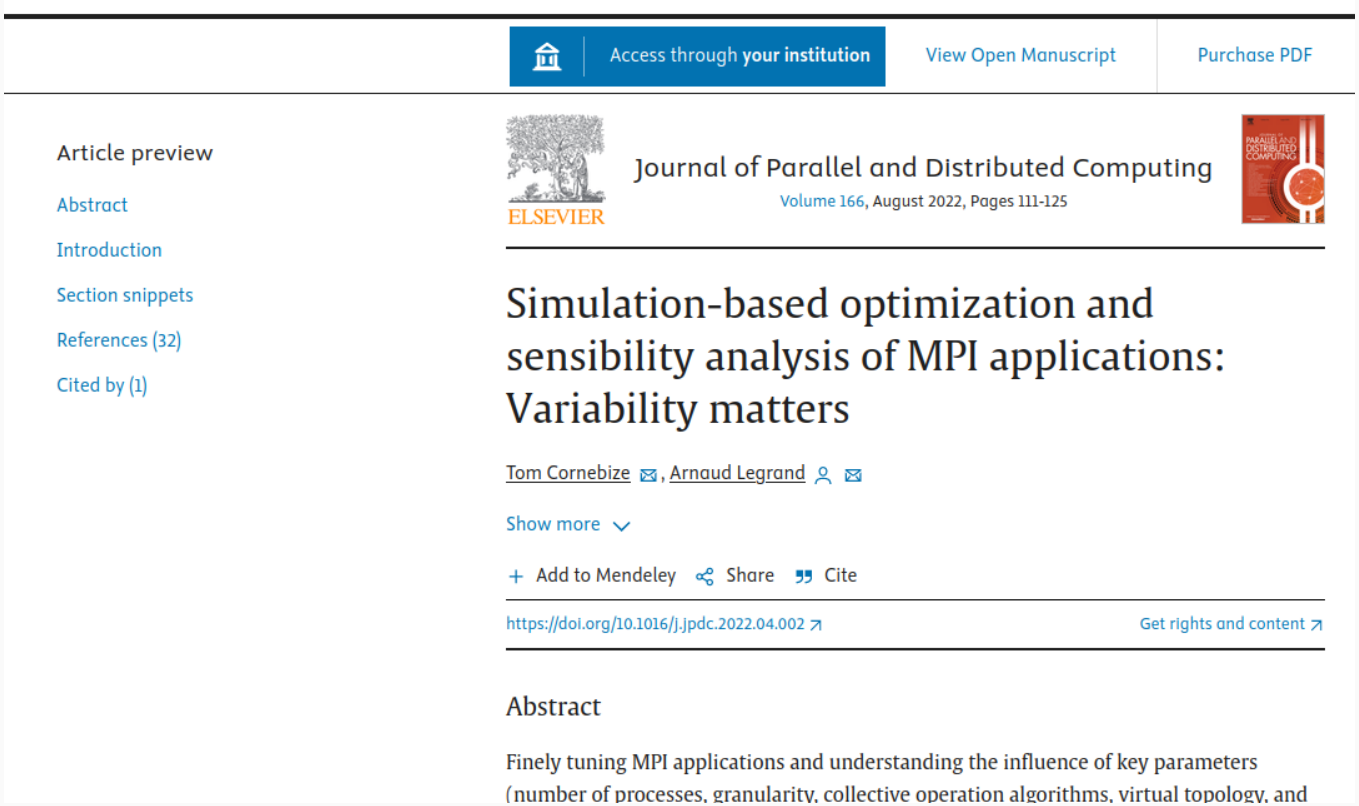
In this paper, a novel approach for enabling Grid users to autonomously install and use custom software on demand using an image creation station is presented, while at the same time offering new security mechanisms to protect both software and data from other Grid users and external attackers. An automated dynamic firewalling mechanism

Go read the paper by Smith et. al. 2009



- Use your institution subscription... or Sci-Hub 😉,

Go read the paper by Smith et. al. 2009



Access through your institution | View Open Manuscript | Purchase PDF

Article preview

Abstract

Introduction

Section snippets

References (32)

Cited by (1)

ELSEVIER

Journal of Parallel and Distributed Computing

Volume 166, August 2022, Pages 111-125

Simulation-based optimization and sensibility analysis of MPI applications: Variability matters

Tom Cornebize, Arnaud Legrand

Show more

+ Add to Mendeley | Share | Cite

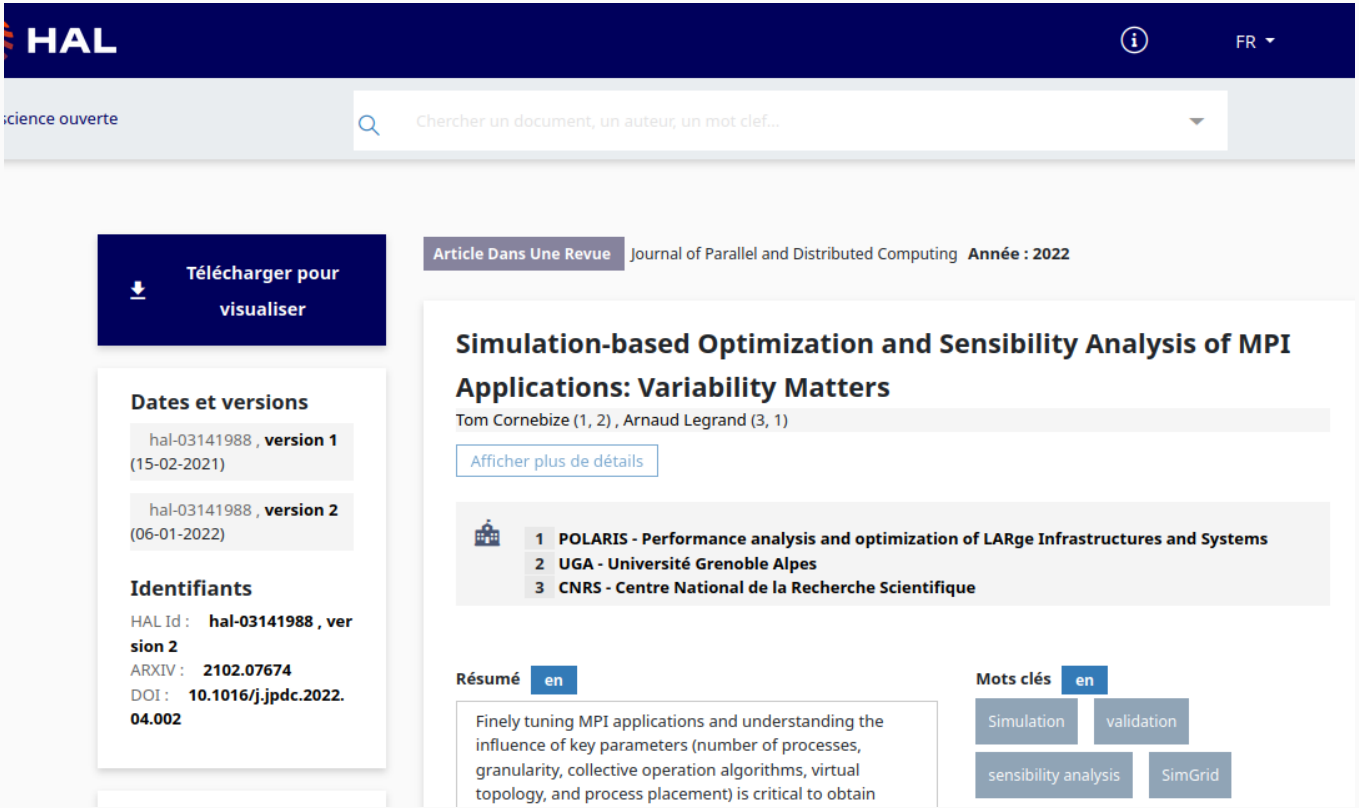
https://doi.org/10.1016/j.jpdc.2022.04.002 | Get rights and content

Abstract

Finely tuning MPI applications and understanding the influence of key parameters (number of processes, granularity, collective operation algorithms, virtual topology, and

- Use your institution subscription... or Sci-Hub 😉, or HAL/Arxiv

Go read the paper by Smith et. al. 2009



- Use your institution subscription... or Sci-Hub 😉, or HAL/Arxiv

Just compare to the Algorithm they proposed

Rodriguez et al., CONCUR'15

Unfolding-based Partial Order Reduction*

César Rodríguez[1], Marcelo Sousa[2], Subodh Sharma[3], and Daniel Kroening[4]

1 Université Paris 13, Sorbonne Paris Cité, LIPN, CNRS, France
2,4 Department of Computer Science, University of Oxford, UK
3 Indian Institute of Technology Delhi, India

Abstract

Partial order reduction (POR) and net unfoldings are two alternative methods to tackle state-space explosion caused by concurrency. In this paper, we propose the combination of both approaches in an effort to combine their strengths. We first define, for an abstract execution model, unfolding semantics parameterized over an arbitrary independence relation. Based on it, our main contribution is a novel stateless POR algorithm that explores at most one execution per Mazurkiewicz trace, and in general, can explore exponentially fewer, thus achieving a form of *super-optimality*. Furthermore, our unfolding-based POR copes with non-terminating executions and incorporates state-caching. Over benchmarks with busy-waits, among others, our experiments show a dramatic reduction in the number of executions when compared to a

Jul 2015

Just compare to the Algorithm they proposed

Rodriguez et al., CONCUR'15

Algorithm 1: An unfolding-based POR exploration algorithm.

```
1  Initially, set U := {⊥}, set G := ∅, and call Explore({⊥}, ∅, ∅).
2  Procedure Explore(C, D, A)
3      Extend(C)
4      if en(C) = ∅ return
5      if A = ∅
6          Choose e from en(C)
7      else
8          Choose e from A ∩ en(C)
9      Explore(C ∪ {e}, D, A ∖ {e})
10     if ∃J ∈ Alt(C, D ∪ {e})
11         Explore(C, D ∪ {e}, J ∖ C)
12     Remove(e, C, D)
13 Procedure Extend(C)
14     Add ex(C) to U
15 Procedure Remove(e, C, D)
16     Move {e} ∖ Q_{C,D,U} from U to G
17     foreach ê ∈ #^i_U(e)
18         Move [ê] ∖ Q_{C,D,U} from U to G
```

- Looks good!

Just compare to the Algorithm they proposed

Rodriguez et al., CONCUR'15

We give some new definitions. Let C be a configuration of $\mathcal{U}$. The *extensions* of C, written $ex(C)$, are all those events outside C whose causes are included in C. Formally, $ex(C) := \{e \in E : e \notin C \wedge \lceil e \rceil \subseteq C\}$. We let $en(C)$ denote the set of events *enabled* by C, i.e., those corresponding to the transitions enabled at $state(C)$, formally defined as $en(C) := \{e \in ex(C) : C \cup \{e\} \in conf(\mathcal{U})\}$. All those events in $ex(C)$ which are not in $en(C)$ are the *conflicting extensions*, $cex(C) := \{e \in ex(C) : \exists e' \in C,\ e \#^i e'\}$. Clearly, sets $en(C)$ and $cex(C)$ partition the set $ex(C)$. Lastly, we define $\#^i(e) := \{e' \in E : e \#^i e'\}$, and $\#^i_U(e) := \#^i(e) \cap U$. The difference between both is that $\#^i(e)$ contains events from *anywhere* in the unfolding structure, while $\#^i_U(e)$ can only *see* events in U.

The algorithm is given in Alg. 1. `Explore`(C, D, A), the main procedure, is given the configuration that is to be explored as the parameter C. The parameter D (for *disabled*) is the set of set of events that have already been explored and prevents that `Explore()` repeats work. It can be seen as a *sleep set* [7]. Set A (for *add*) is occasionally used to guide the direction of the exploration.

Additionally, a global set U stores all events presently known to the algorithm. Whenever some event can safely be discarded from memory, `Remove` will move it from U to G (for *garbage*). Once in G, it can be discarded at any time, or be preserved in G in order to save work when it is re-inserted in U. Set G is thus our *cache memory* of events.

The key intuition in Alg. 1 is as follows. A call to `Explore`(C, D, A) visits all maximal configurations of $\mathcal{U}$ which contain C and do not contain D; and the first one explored will

- Looks good! Err... not so simple. Depending on how you do this, you quickly move from polynomial to exponential.

Just compare to the Algorithm they proposed

Rodriguez et al., CONCUR'15

POSIX threads. The analyzer accepts deterministic programs, implements a variant of Alg. 1 where the computation of the alternatives is memoized, and supports cutoffs events with the criteria defined in § 5.

We ran Poet on a number of multi-threaded C programs. Most of them are adapted from benchmarks of the Software Verification Competition [17]; others are used in related works [8, 19, 2]. We investigate the characteristics of average program unfoldings (depth, width, etc.) as well as the frequency and impact of cutoffs on the exploration. We also compare Poet with Nidhugg [1], a state-of-the-art stateless model checking for multi-threaded C programs that implements Source-DPOR [2], an efficient but non-optimal DPOR. All experiments were run on an Intel Xeon CPU with 2.4 GHz and 4 GB memory. Tables 1 and 2 give our experimental data for programs with acyclic and non-acyclic state spaces, respectively.

For programs with acyclic state spaces (Table 1), Poet with and without cutoffs seems to perform the same exploration when the unfolding has no cutoffs, as expected. Furthermore, the number of explored executions also coincides with Nidhugg when the latter reports 0 sleep-set blocked executions (cf., § 4), providing experimental evidence of Poet's optimality.

The unfoldings of most programs in Table 1 do not contain cutoffs. All these programs are deterministic, and many of them highly sequential (Stf, Spin08, Fib), features known to make cutoffs unlikely. Ccnf(n) are concurrent programs composed of $n-1$ threads where thread i and $i+1$ race on writing one variable, and are independent of all remaining

[3] Source code and benchmarks available from: http://www.cs.ox.ac.uk/people/marcelo.sousa/poet/.

- Looks good! Err... not so simple. Depending on how you do this, you quickly move from polynomial to exponential.
- Experiments!

Just compare to the Algorithm they proposed

Rodriguez et al., CONCUR'15

Table 1 Programs with acyclic state space. Columns are: $|P|$: nr. of threads; $|I|$: nr. of explored traces; $|B|$: nr. of sleep-set blocked executions; $t(s)$: running time; $|E|$: nr. of events in $\mathcal{U}$; $|E_{\text{cut}}|$: nr. of cutoff events; $|\Omega|$: nr. of maximal configurations; $\langle|U_\Omega|\rangle$: avg. nr. of events in U when exploring a maximal configuration. A $*$ marks programs containing bugs. <7K reads as "*fewer than 7000*".

Benchmark		Nidhugg			Poet (without cutoffs)				Poet (with cutoffs)				
Name	$\|P\|$	$\|I\|$	$\|B\|$	$t(s)$	$\|E\|$	$\|\Omega\|$	$\langle\|U_\Omega\|\rangle$	$t(s)$	$\|E\|$	$\|E_{\text{cut}}\|$	$\|\Omega\|$	$\langle\|U_\Omega\|\rangle$	$t(s)$
Stf	3	6	0	0.06	121	6	79	0.04	121	0	6	79	0.06
Stf*	3	-	-	0.05	-	-	-	0.02	-	-	-	-	0.03
Spin08	3	84	0	0.08	2974	84	1506	2.04	2974	0	84	1506	2.93
Fib	3	8953	0	3.36	<185K	8953	92878	305	<185K	0	8953	92878	704
Fib*	3	-	-	0.74	-	-	-	81.0	-	-	-	-	133
Ccnf(9)	9	16	0	0.05	49	16	46	0.07	49	0	16	46	0.06
Ccnf(17)	17	256	0	0.15	97	256	94	5.76	97	0	256	94	6.09
Ccnf(19)	19	512	0	0.28	109	512	106	22.5	109	0	512	106	22.0
Ssb	5	4	2	0.05	48	4	38	0.03	46	1	4	37	0.03
Ssb(1)	5	22	14	0.06	245	23	143	0.11	237	4	23	140	0.11
Ssb(3)	5	169	67	0.12	2798	172	1410	3.51	1179	48	90	618	0.90
Ssb(4)	5	336	103	0.15	<7K	340	3333	20.3	2179	74	142	1139	2.07
Ssb(8)	5	2014	327	0.85	<67K	2022	32782	4118	<12K	240	470	6267	32.1

Lastly, we note that this cutoff approach imposes no liability on what events shall be kept in the prefix, set G can be cleaned at discretion. Also, redefining (7) to use adequate orders [5] is straightforward, cf. App. C.1 (in our proofs we actually assume adequate orders).

- Looks good! Err... not so simple. Depending on how you do this, you quickly move from polynomial to exponential.
- Experiments! ⚠Possible *404, code not found!* ahead!!!

Just compare to the Algorithm they proposed

Rodriguez et al., CONCUR'15

- Looks good! Err... not so simple. Depending on how you do this, you quickly move from polynomial to exponential.
- Experiments! ⚠Possible *404, code not found!* ahead!!!
- Wait, what's this language? Did this ever run one day?

Just compare to the Algorithm they proposed

Rodriguez et al., CONCUR'15

- Looks good! Err... not so simple. Depending on how you do this, you quickly move from polynomial to exponential.
- Experiments! ⚠Possible *404, code not found!* ahead!!!
- Wait, what's this language? Did this ever run one day?
- Wow, I'll need `CPLEX` and `Gurobi` but all I have is `lpsolve` or `glpk`

Just compare to the Algorithm they proposed

Rodriguez et al., CONCUR'15

- Looks good! Err... not so simple. Depending on how you do this, you quickly move from polynomial to exponential.
- Experiments! ⚠Possible *404, code not found!* ahead!!!
- Wait, what's this language? Did this ever run one day?
- Wow, I'll need `CPLEX` and `Gurobi` but all I have is `lpsolve` or `glpk`
- Sweet, they provided a binary! Oh, wait, MacOSX in 2015 ?!?

Just compare to the Algorithm they proposed

Rodriguez et al., CONCUR'15

- Looks good! Err... not so simple. Depending on how you do this, you quickly move from polynomial to exponential.
- Experiments! ⚠Possible *404, code not found!* ahead!!!
- Wait, what's this language? Did this ever run one day?
- Wow, I'll need `CPLEX` and `Gurobi` but all I have is `lpsolve` or `glpk`
- Sweet, they provided a binary! Oh, wait, MacOSX in 2015 ?!?
- The GitHub webpage says it requires Foo, Bar, and Baz, but none of the versions I find appear to work.

Just compare to the Algorithm they proposed

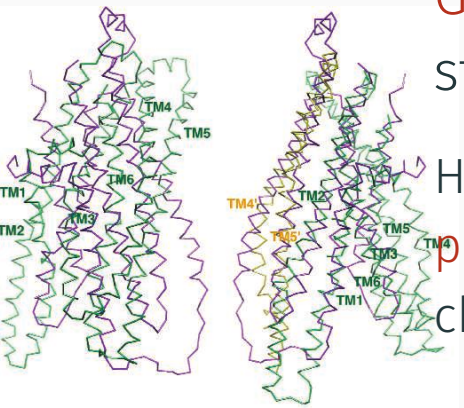
Rodriguez et al., CONCUR'15

- Looks good! Err... not so simple. Depending on how you do this, you quickly move from polynomial to exponential.
- Experiments! ⚠Possible *404, code not found!* ahead!!!
- Wait, what's this language? Did this ever run one day?
- Wow, I'll need `CPLEX` and `Gurobi` but all I have is `lpsolve` or `glpk`
- Sweet, they provided a binary! Oh, wait, MacOSX in 2015 ?!?
- The GitHub webpage says it requires Foo, Bar, and Baz, but none of the versions I find appear to work.
- With which parameters and data set do you run this code? And Why?

In the end, one new thesis to understand this paper and contribute.

Blaming "Computer Science"

How computers broke science



Geoffrey Chang (Scripps, UCSD) works on crystalography and studies the structure of cell membrane proteins.

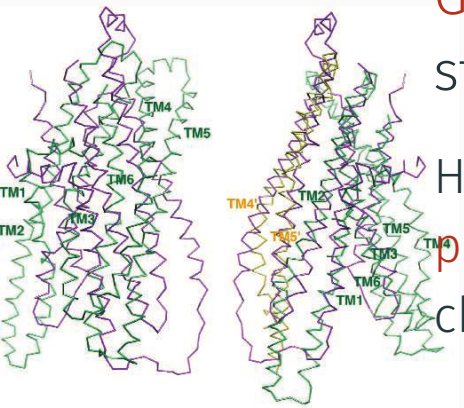
He specialized in structures of multidrug resistant transporter proteins in bacteria: MsbA de Escheria Choli (Science, 2001), Vibrio cholera (Mol. Biology, 2003), Salmonella typhimurium (Science, 2005)

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 comp. biology

How computers broke science



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 comp. biology

Computers...

How computers broke science –

and what we can do about it

> *Most modern science is so complicated, and most journal articles so brief, it's impossible for the article to include details of many important methods and decisions made by the researcher as he analyzed his data on his computer. How, then, can another researcher judge the reliability of the results, or reproduce the analysis?*

– Ben Marwick,
The conversation, 2015

Point-and-click procedures are rampant but they hinder reproducibility.

Spreadsheets are generalized and intensively uses in biology:

- `Membrane-Associated Ring Finger (C3HC4) 1`,
  `E3 Ubiquitin Protein Ligase` → `MARCH1` → 2016-03-01 → 1456786800
- `2310009E13` → 2.31E+19

And more recently, we had the COVID tracing failure.

... and Statistics/Machine Learning

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".*
>
> *– Alex "Sandy" Pentland*

The Reproducibility Crisis in ML-based science (Princeton workshop 2022)

> *Reproducibility failures in ML-based science are systemic. We found 20 reviews across 17 scientific fields (medicine, neuroimaging, autism diagnosis, genomics, computer security, ...) that find errors in a total of 329 papers that use ML-based science and in some cases leading to wildly overoptimistic conclusion. [...] complex ML models don't perform substantively better than decades-old LR models.*
>
> *Data leakage: spurious relationship between the independent variables and the target variable that arises as an artifact of the data collection, sampling, or pre-processing strategy.*
>
> *– S. Kapoor and A. Narayanan*

This is about Computational Science. Should mathematicians care?

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

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

This is about Computational Science. Should mathematicians care?

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

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

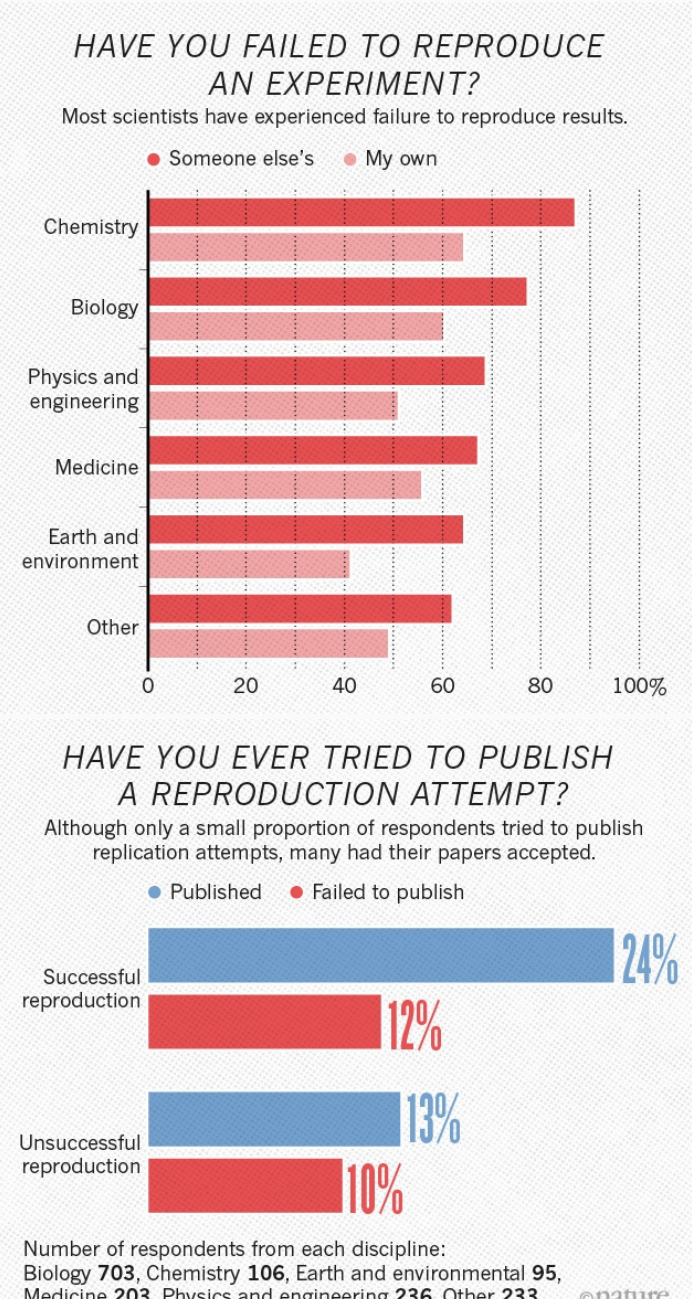
Yet, incoherencies are common, especially in a fast moving field:

- E.g., definitions/concepts in book/article A and B are *slightly different* and the resulting theorems cannot be mixed
- Have a look at Vladimir Voevodsky's talk in 2014 at Princeton 😉
- ERC Nano bubbles: how, when and why does science fail to correct itself?

  Flagging incorrect nucleotide sequence reagents in biomedical papers: To what extent does the leading publication format impede automatic error detection? (Labbe et al., 2020)

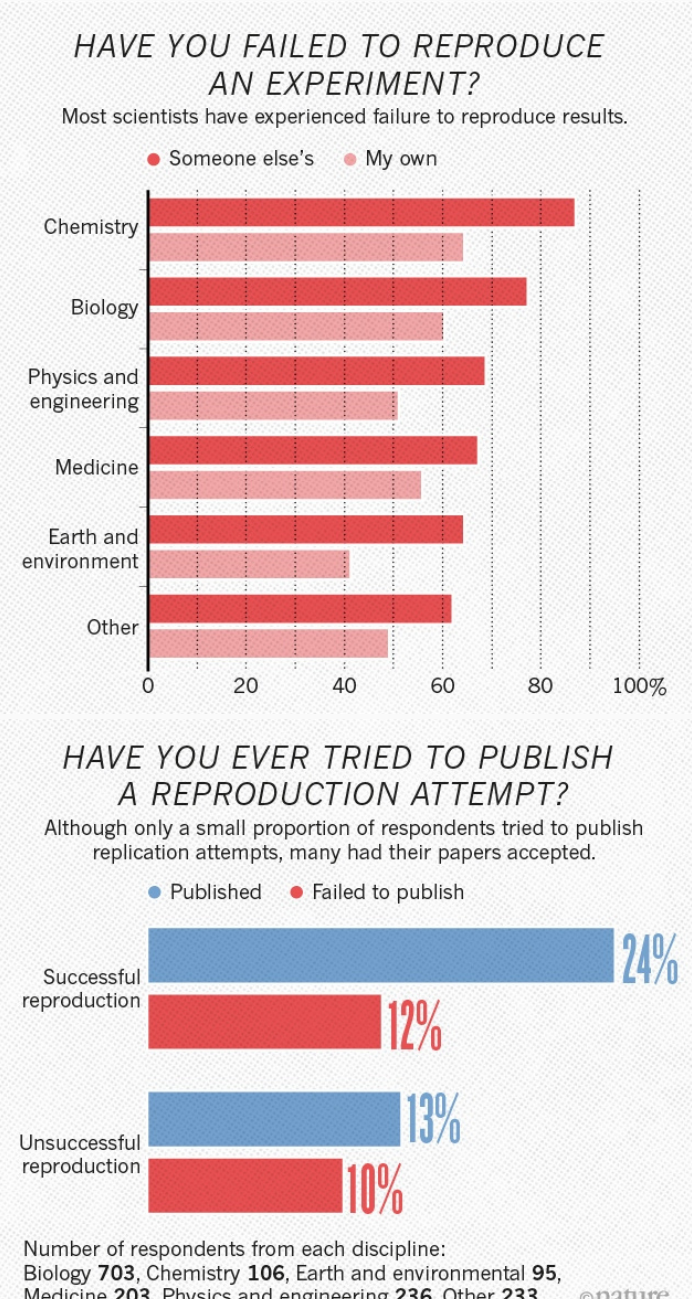
Different kinds of reproducibility

Socio-technical Challenges



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

Socio-technical Challenges

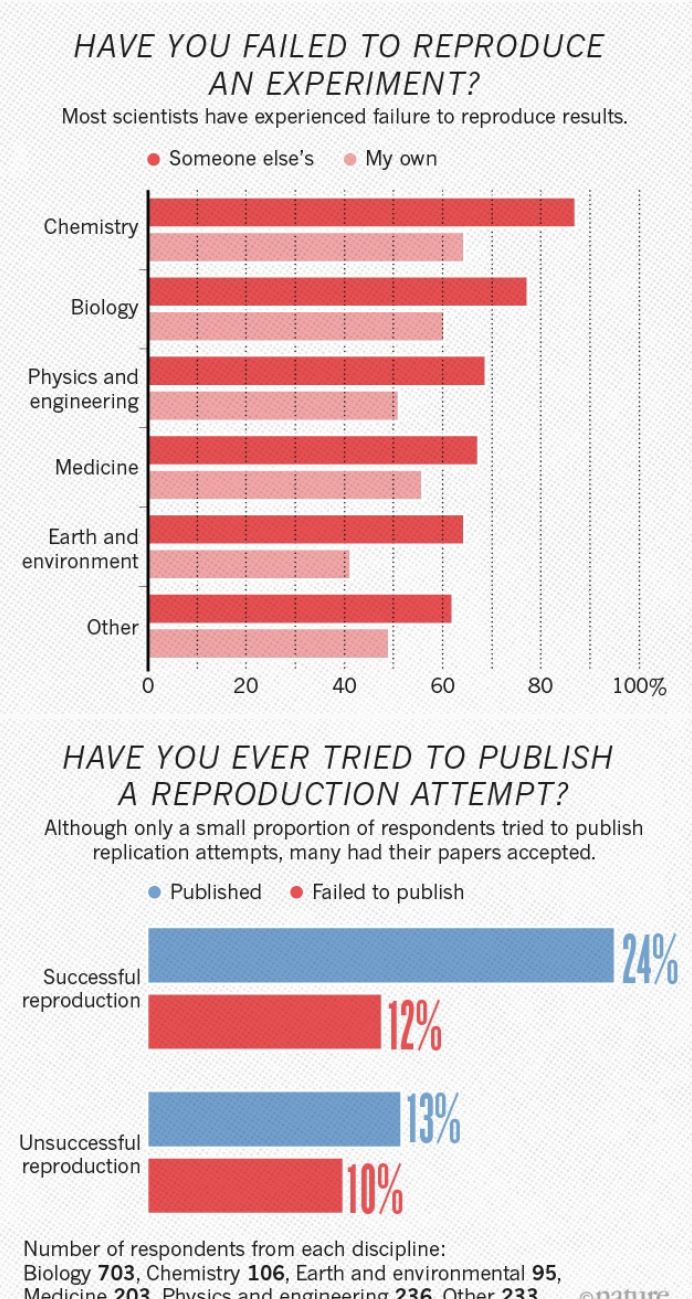


1,500 scientists lift the lid on reproducibility, Nature, May 2016

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

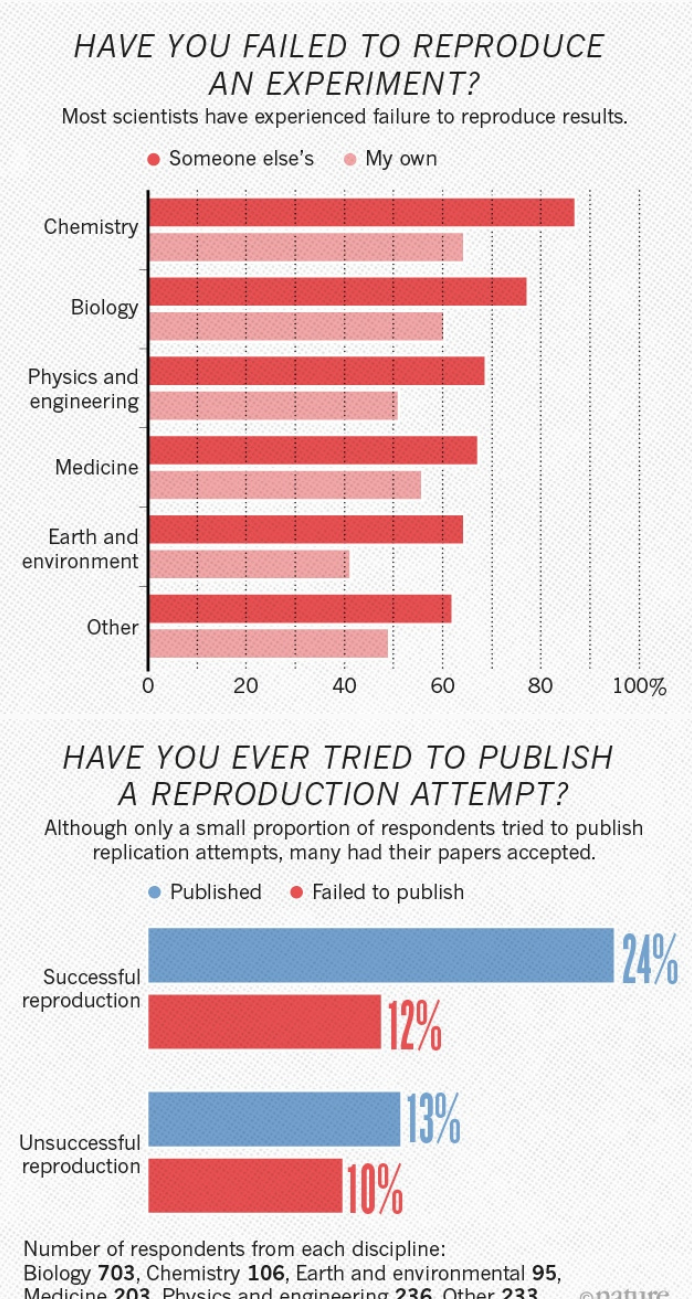


1,500 scientists lift the lid on reproducibility, Nature, May 2016

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

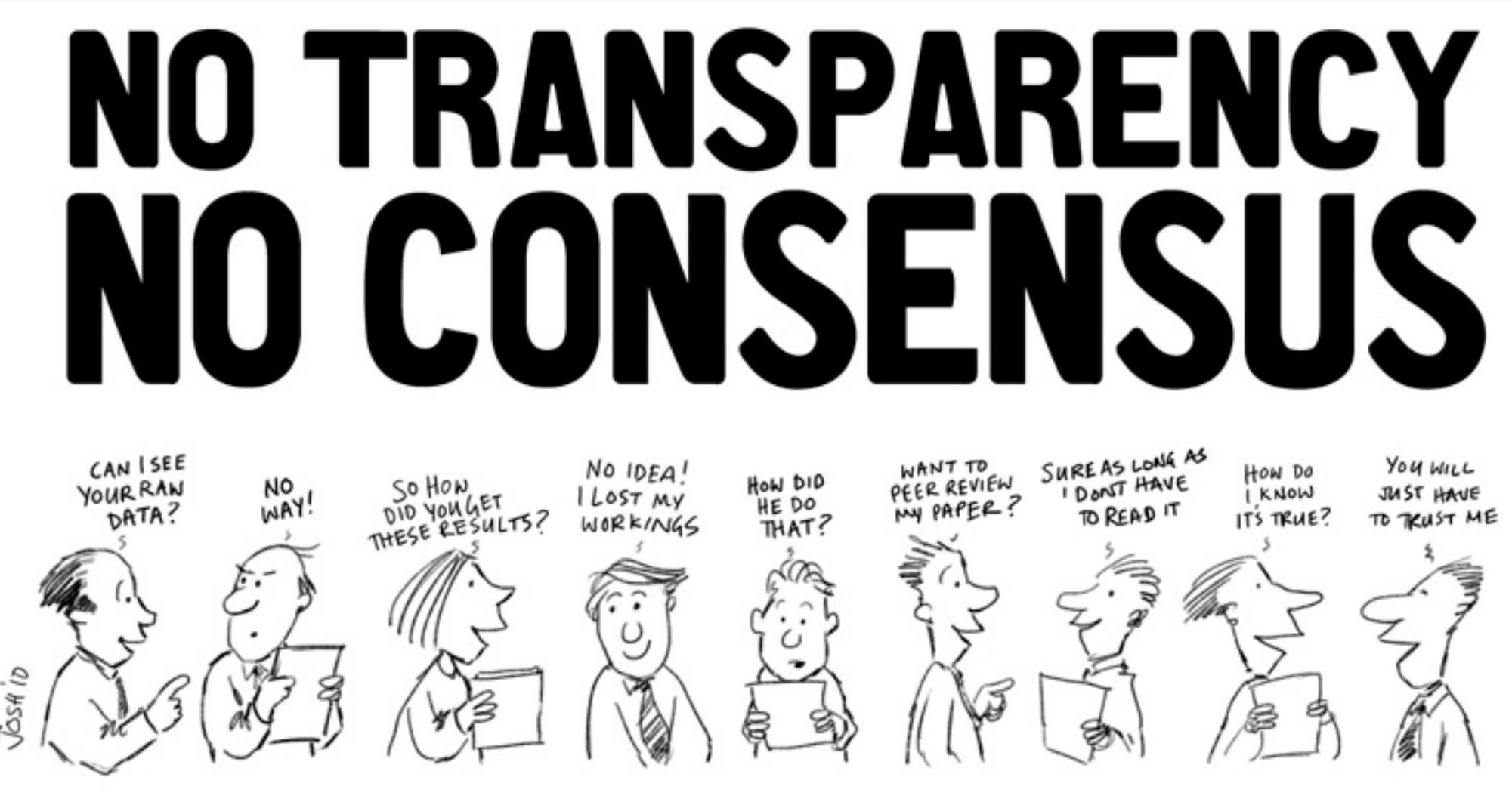
Social causes

- 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

Scientific Consensus



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 ⇝ standardized protocols

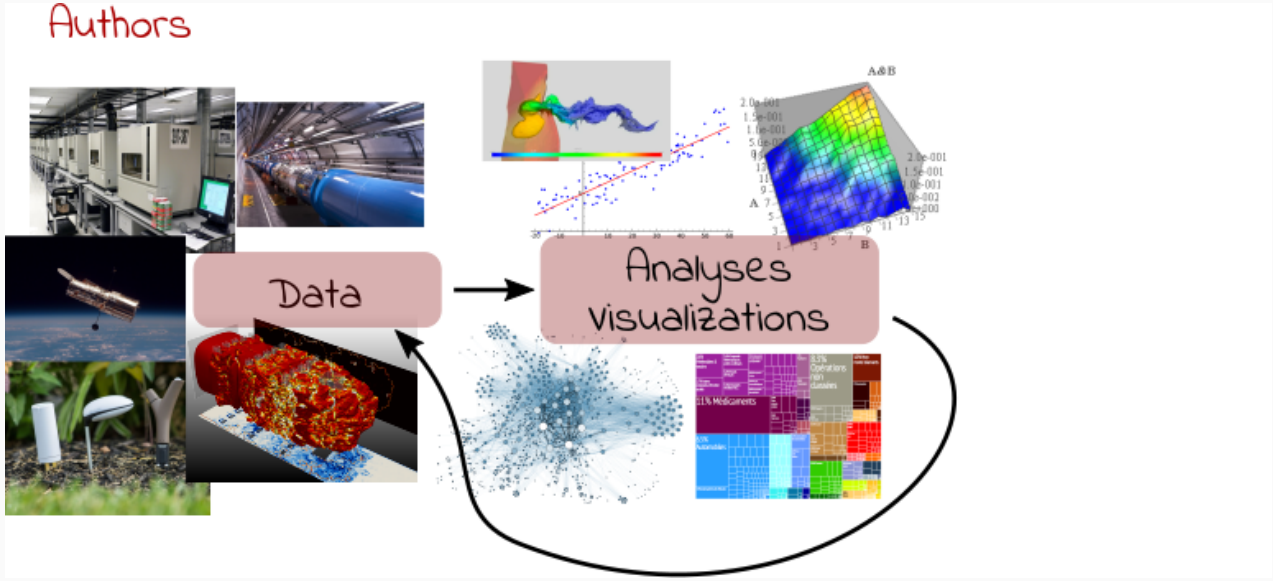
Psychology, Nutrition HARKING, p-hacking ⇝ pre-registration

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 ⇝ standardized protocols

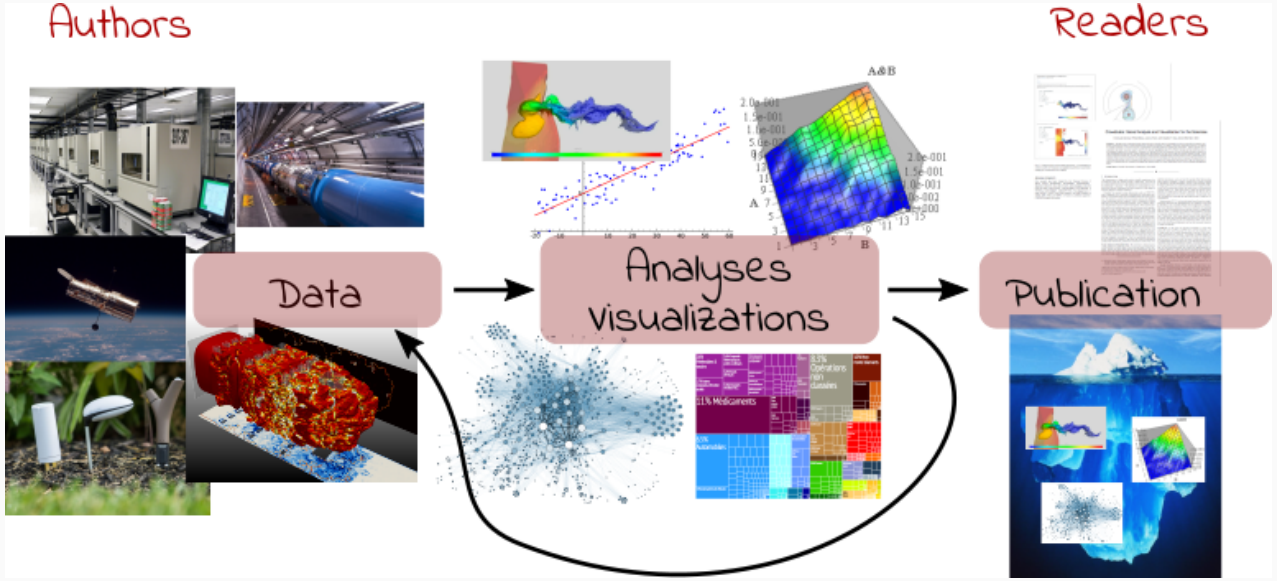
Psychology, Nutrition HARKING, p-hacking ⇝ pre-registration

Genomics software engineering, computational reproducibility, provenance

Computational fluid dynamics numerical chaos, parallel architectures

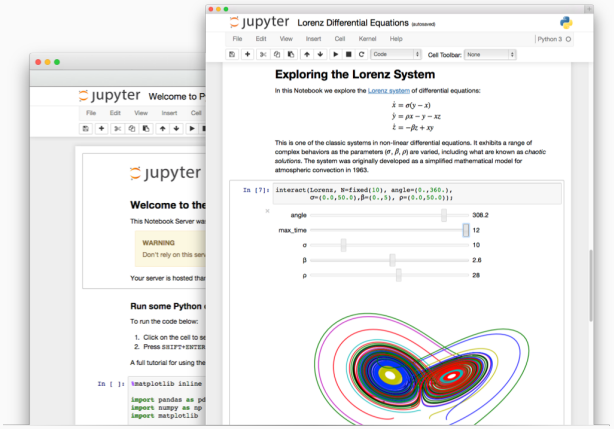
Artificial Intelligence most of the above 😉

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



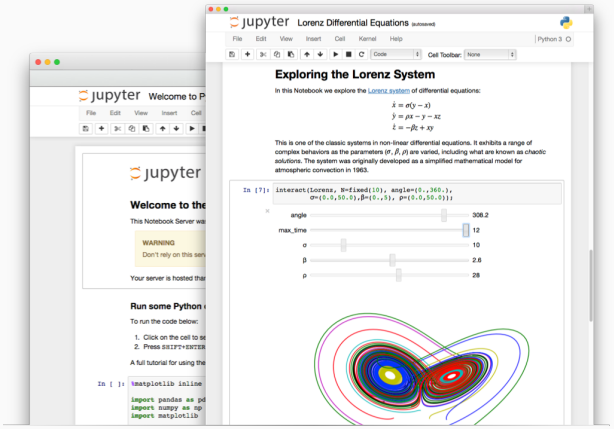
Reproducible Research = Bridging the Gap by working Transparently

Computation provenance: notebooks and workflows

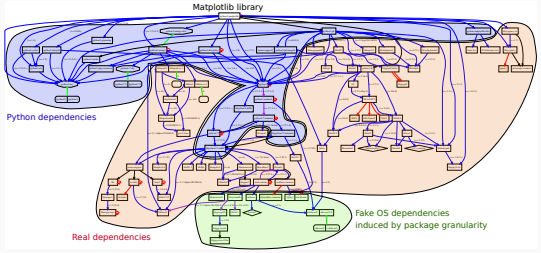


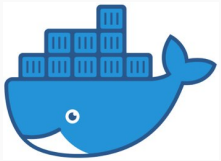
Reproducibility Issues Related to the use of Computers

Computation provenance: notebooks and workflows



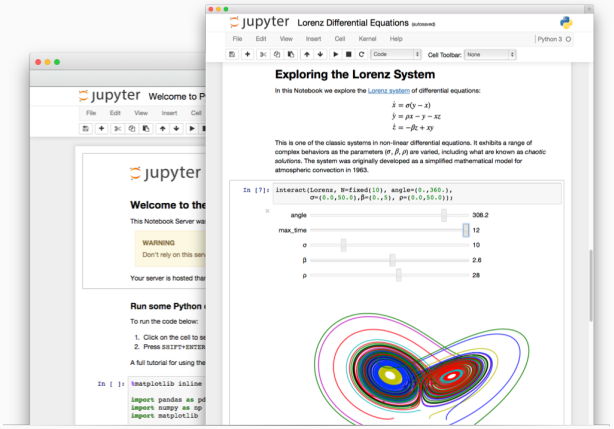
Software environments



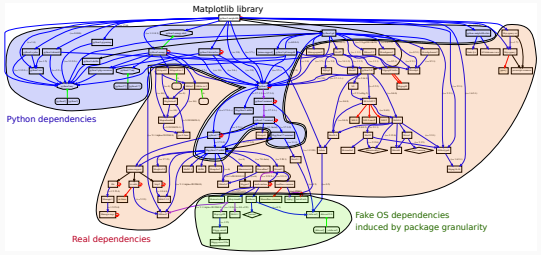


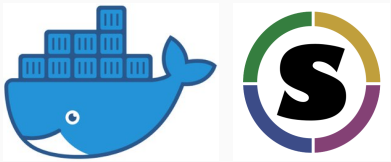
Reproducibility Issues Related to the use of Computers

Computation provenance: notebooks and workflows

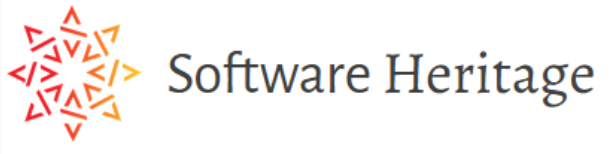


Software environments



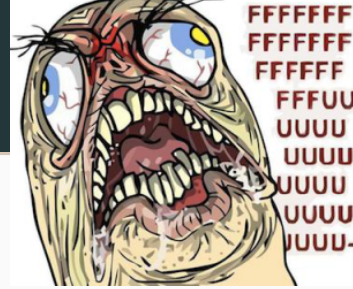


Sharing and Archiving



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*

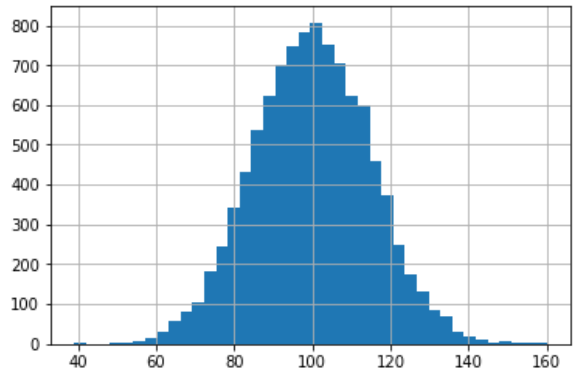
```
3.141592653589793
```

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... ☺).
```

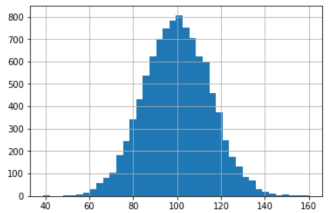
In [3]:

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*

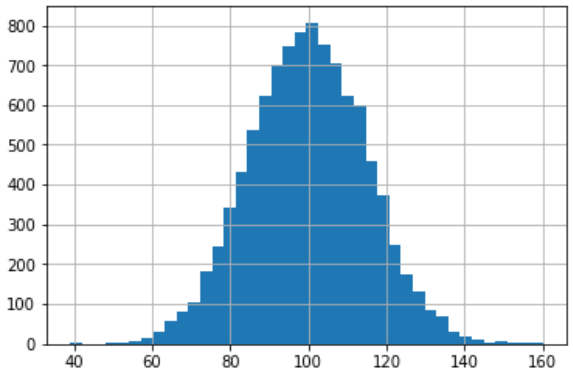
```
3.141592653589793
```

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

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 π 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 π (si ce n'est une constante de normalisation... ☺).

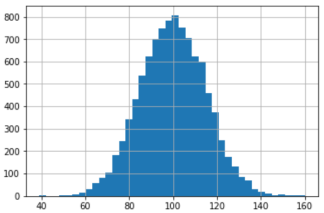
In [3]:

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

Un document computationnel

Mon ordinateur m'indique que π vaut *approximativement*

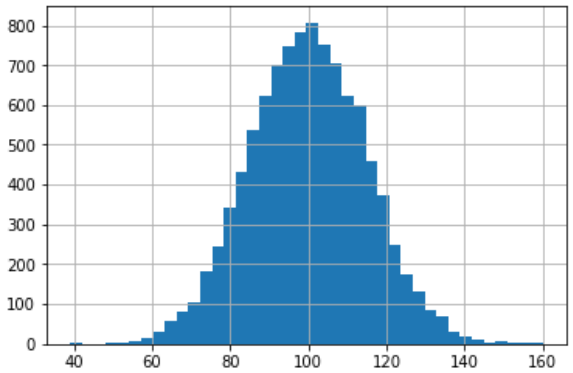
```
3.141592653589793
```

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... ☺).
```

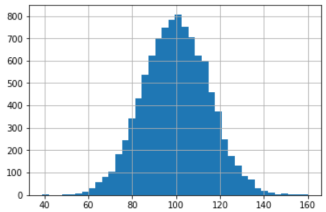
In [3]:

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()
```



Code

Document final

Un document computationnel

Mon ordinateur m'indique que π vaut *approximativement*

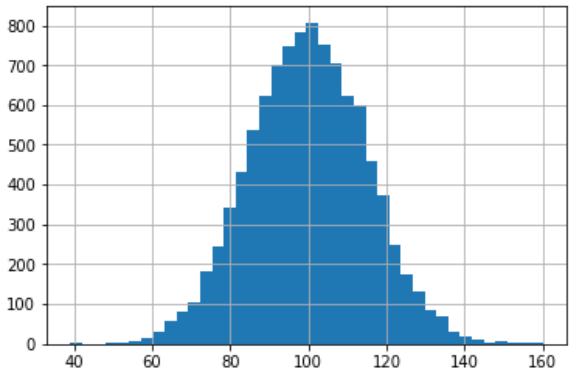
```
3.141592653589793
```

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... ☺).



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... ☺).
```

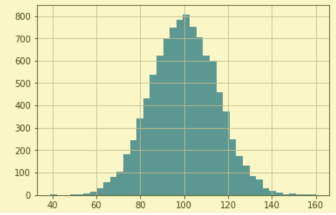
In [3]:

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()
```



Résultats

Document final

Un document computationnel

Mon ordinateur m'indique que π vaut *approximativement*

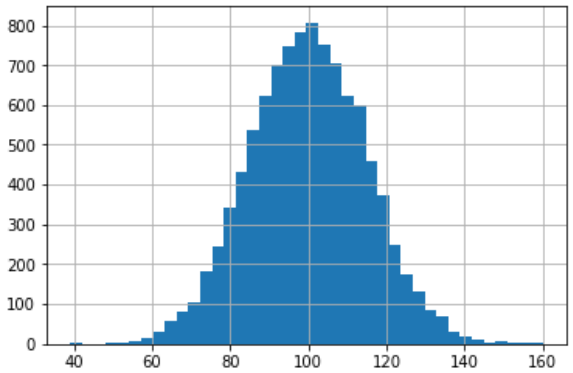
```
3.141592653589793
```

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... ☺).



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... ☺).
```

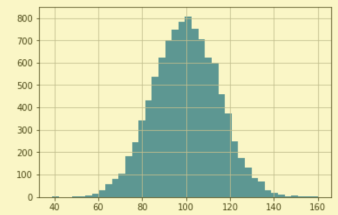
In [3]:

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*

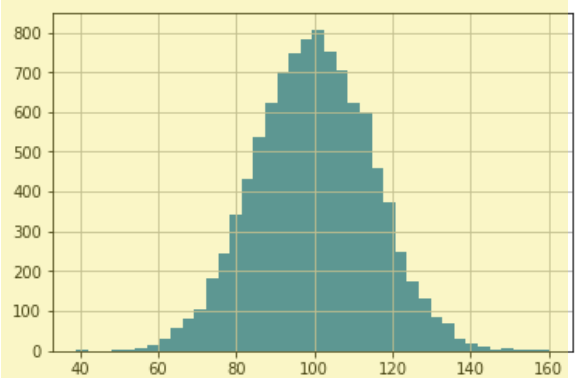
```
3.141592653589793
```

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... ☺).
```

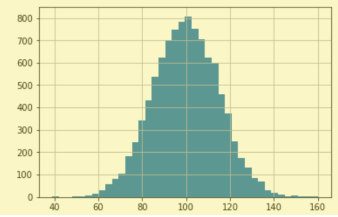
In [3]:

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*

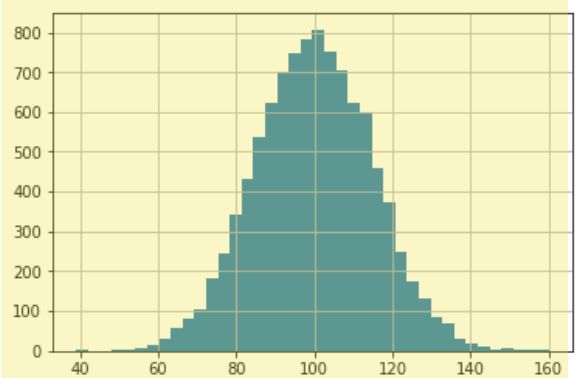
```
3.141592653589793
```

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

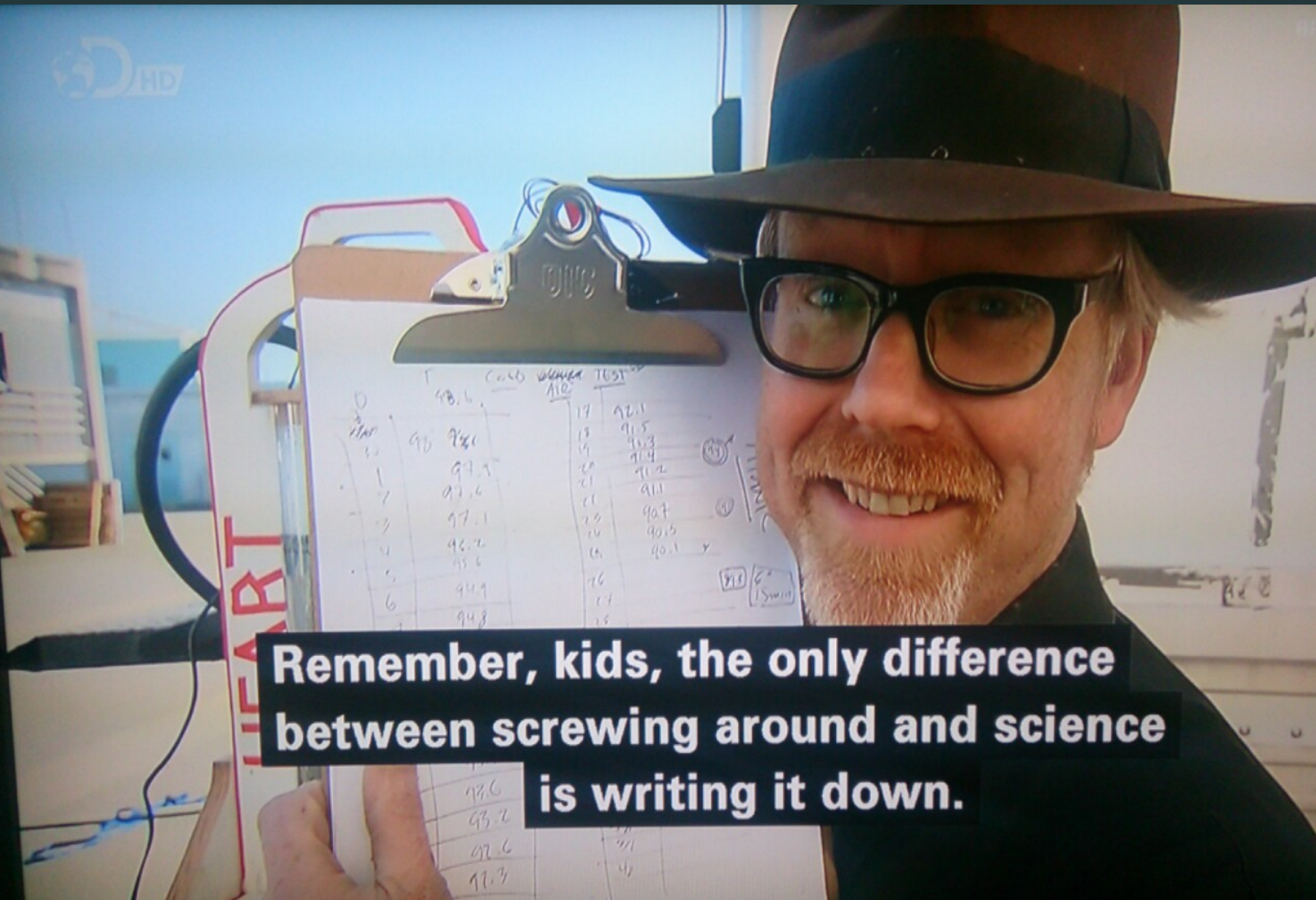
```
import numpy as np
N = 1000000
x = np.random.uniform(size=N, low=0, high=1)
theta = np.random.uniform(size=N, low=0, high=pi/2)
2/(sum((x+np.sin(theta))>1)/N)
```

```
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... ☺).

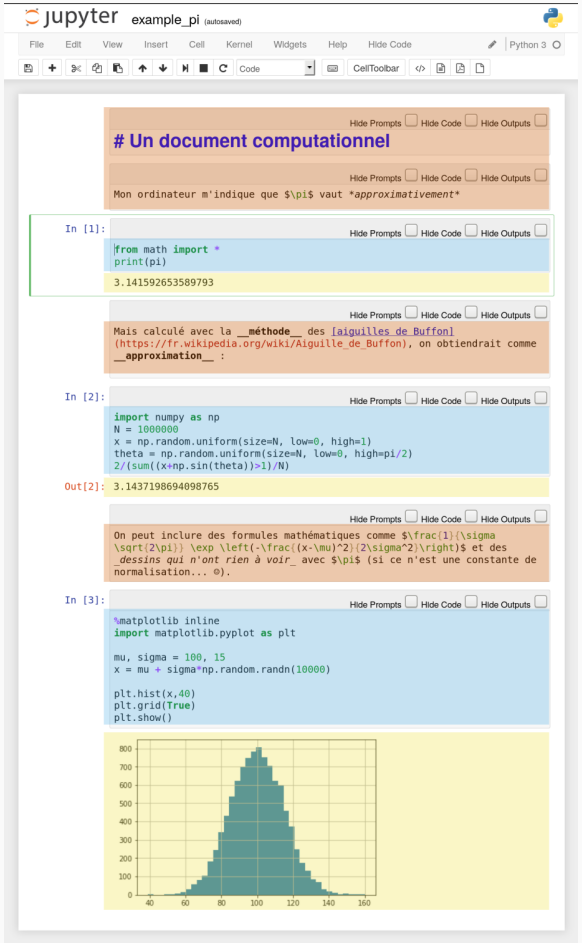


Tool 1 bis: Laboratory Notebooks, Computational Documents



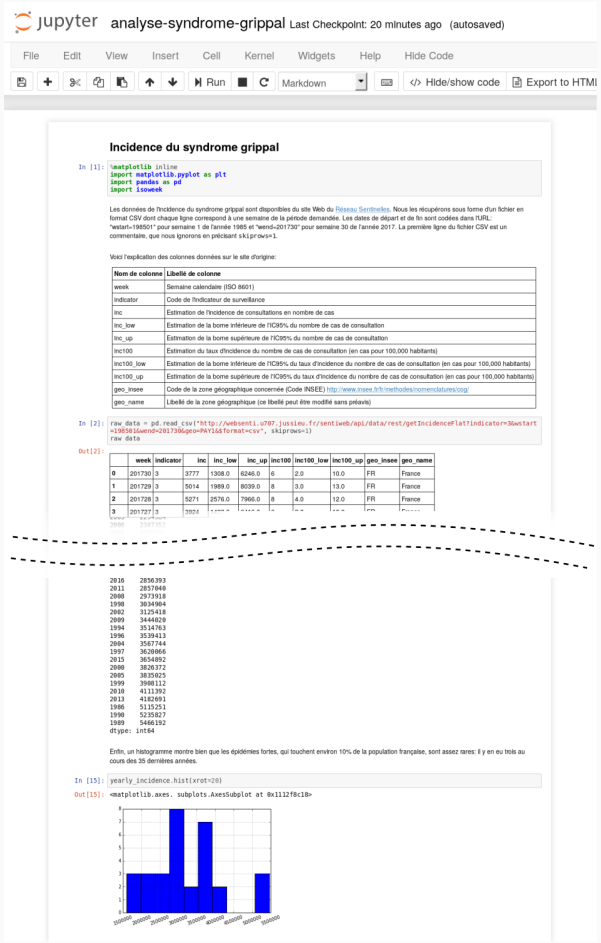
Tool 1 ter: Workflows

Notebooks are no panacea and do not help developing clean code



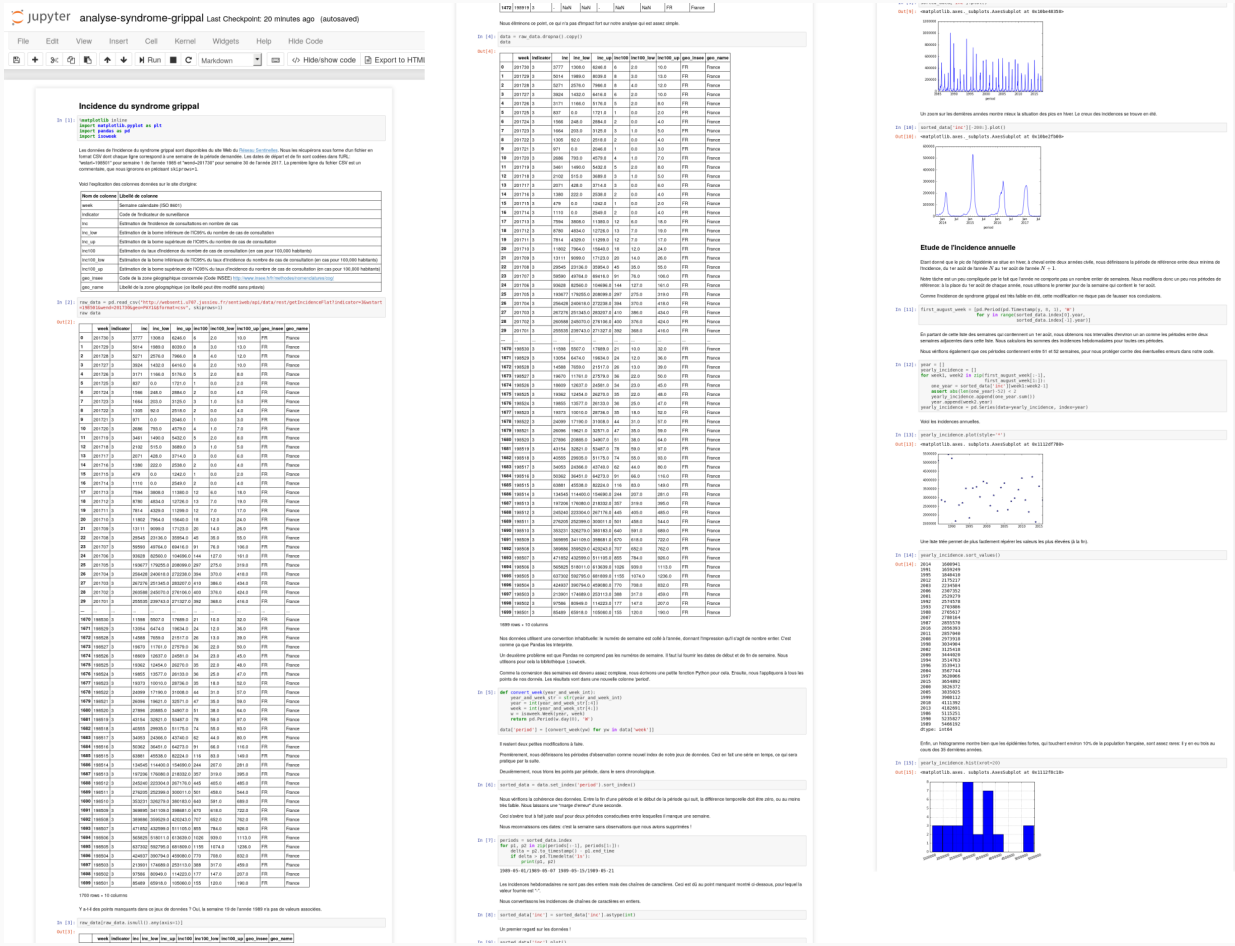
Tool 1 ter: Workflows

Notebooks are no panacea and do not help developing clean code



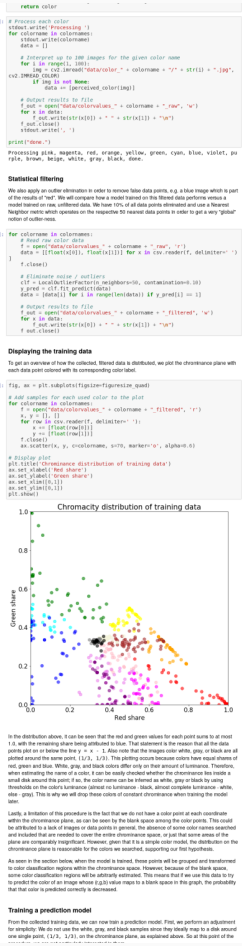
Tool 1 ter: Workflows

Notebooks are no panacea and do not help developing clean code

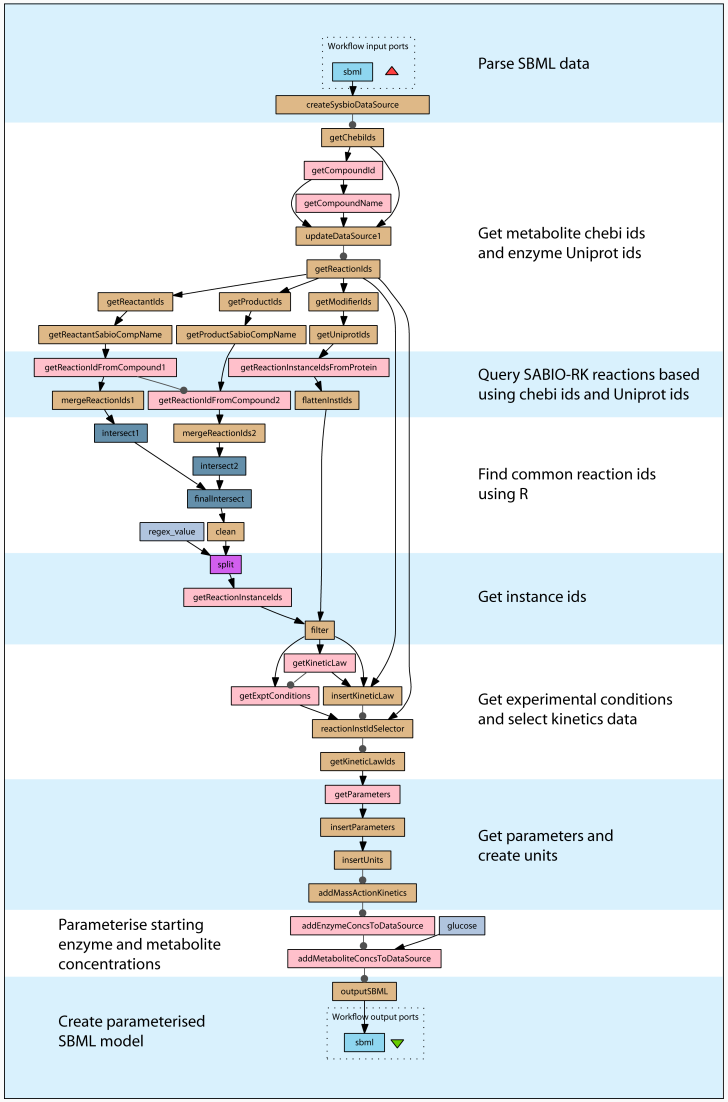


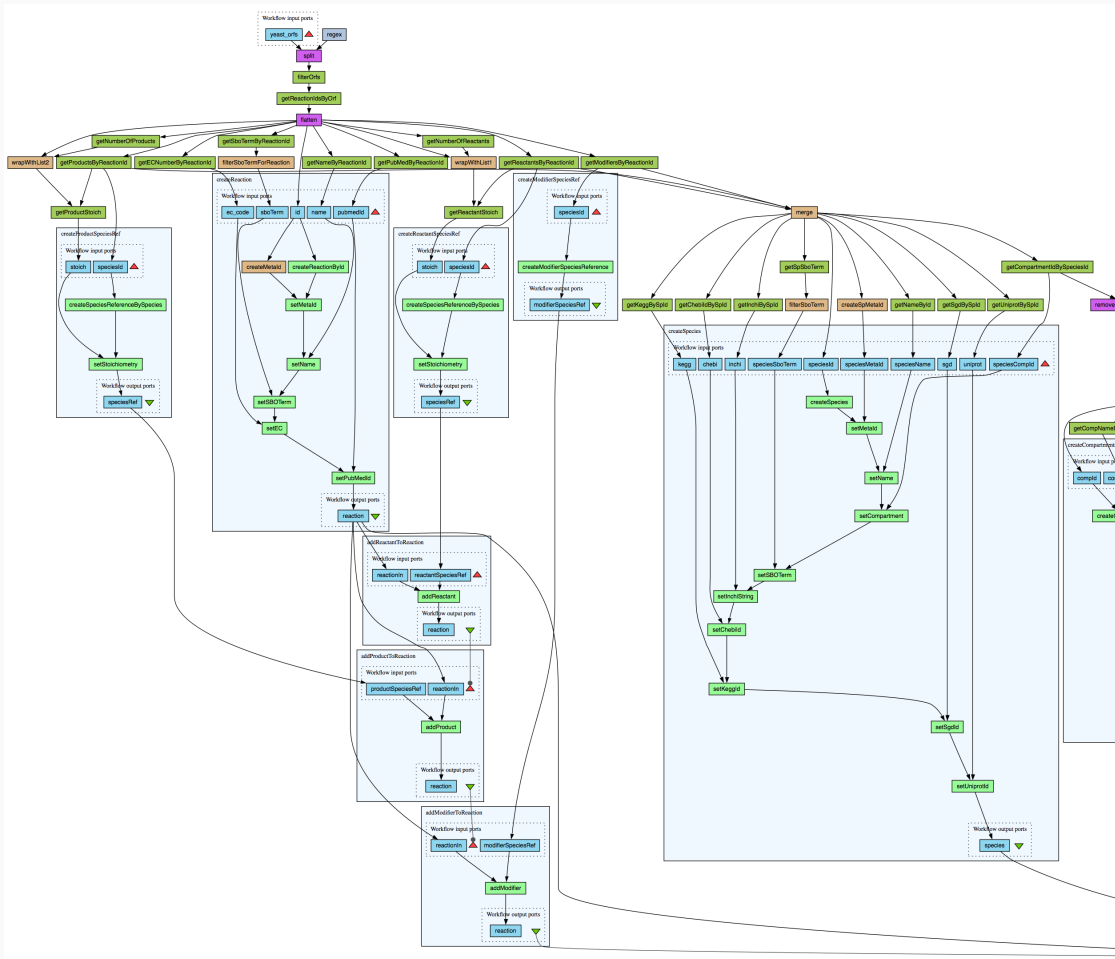
Tool 1 ter: Workflows

Notebooks are no panacea and do not help developing clean code



Tool 1 ter: Workflows





Tool 1 ter: 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 ☹

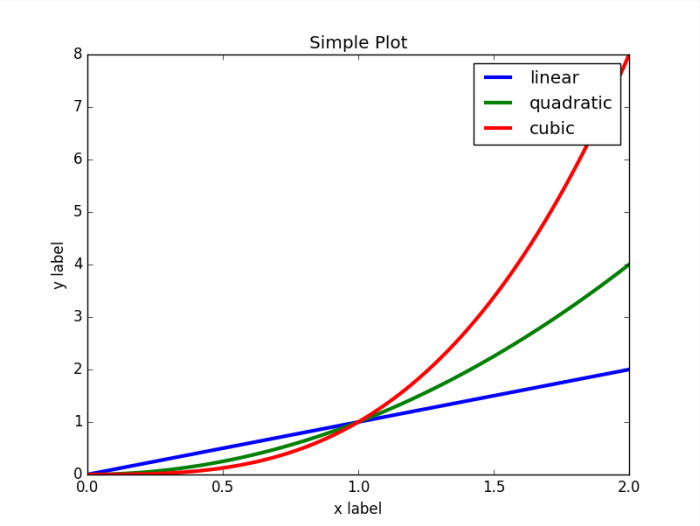
- Whenever trying the code of my colleague, I had to install `libFoo-1.5c` and `pip install blah` but I broke everything and now neither his code nor mine works! ☹

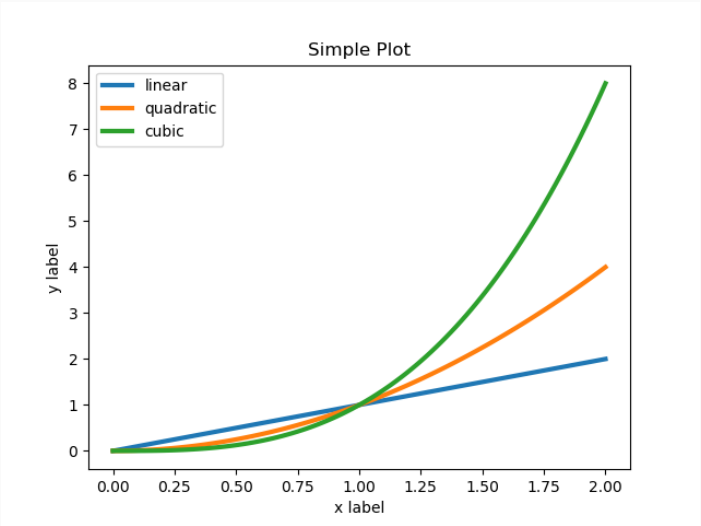
- But hey! Here is my code. It's on GitHub so feel free to play with it! I'm doing open science 😉
  1. No one will ever run/use your code if it isn't easy to install
  2. No one will ever manage to run your code if you don't document how to run it
  3. Others (even you) are unlikely to get the same results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

Software dependencies: horror stories

Software dependencies: horror stories

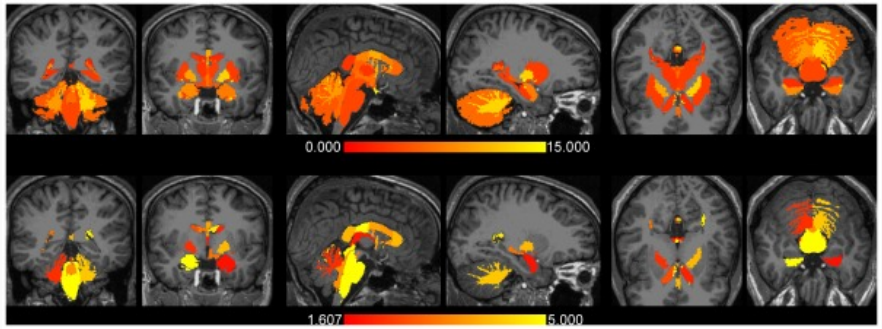
- Software environment evolution





Software dependencies: horror stories

- Software environment evolution
- OS heterogeneity

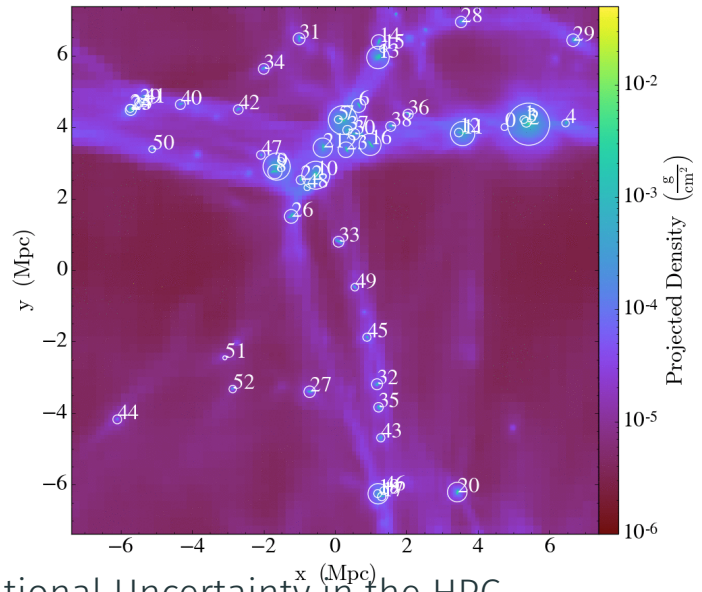


The Effects of FreeSurfer Version, Workstation Type, and Macintosh Operating System Version on Anatomical Volume and Cortical Thickness Measurements (PLOS ONE, 2012)

> *Significant differences in volume and cortical thickness were revealed across FreeSurfer versions:*
>
> - *volume:* *8.8 ± 6.6% (range 1.3-64.0%)*
> - *cortical thickness:* *2.8 ± 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

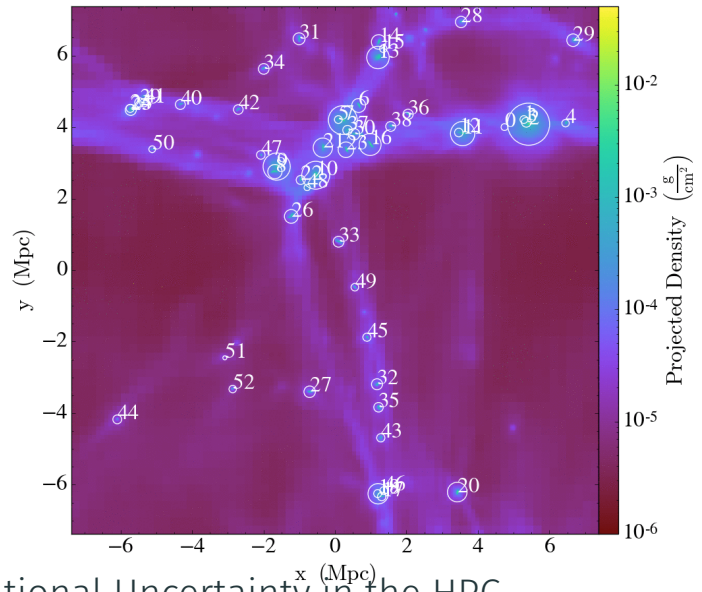


Assessing Reproducibility: An Astrophysical Example of Computational Uncertainty in the HPC Context (ResCuE-HPC, 2018)

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

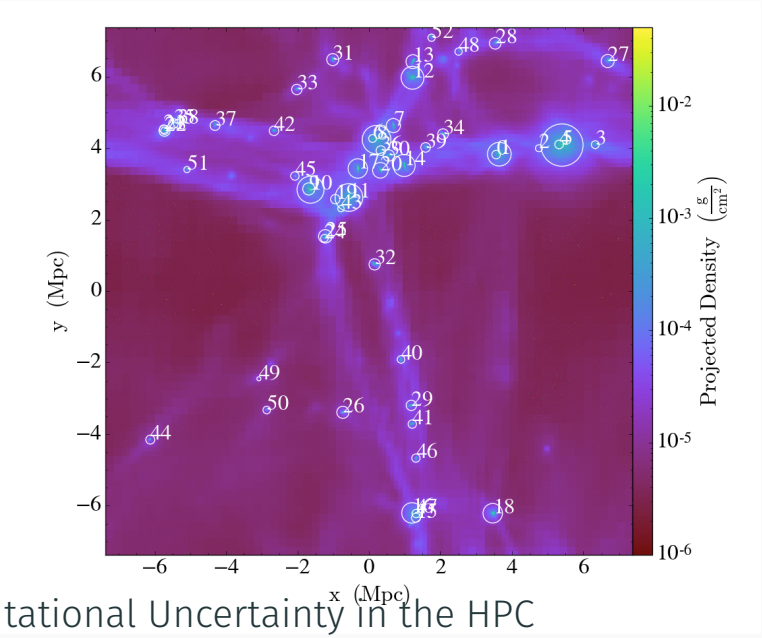


Assessing Reproducibility: An Astrophysical Example of Computational Uncertainty in the HPC Context (ResCuE-HPC, 2018)

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

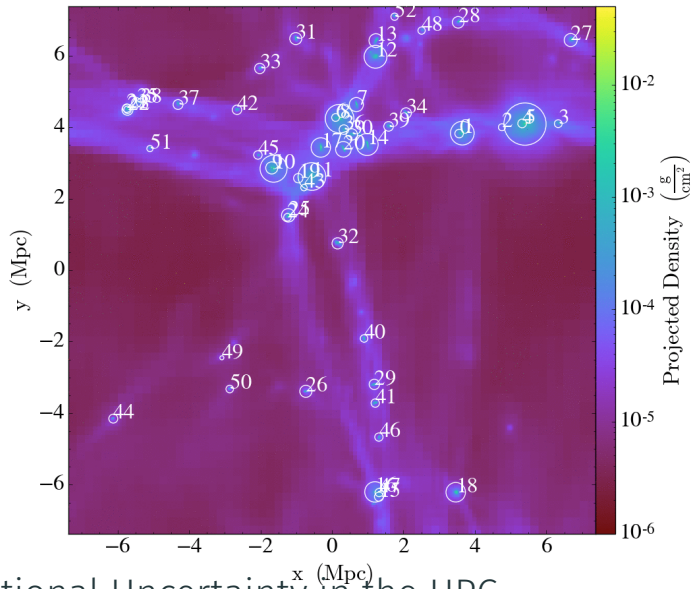


Assessing Reproducibility: An Astrophysical Example of Computational Uncertainty in the HPC Context (ResCuE-HPC, 2018)

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

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

```
3.5.1
```

Complex ecosystems

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

```
3.5.1
```

```
apt show python3-matplotlib
```

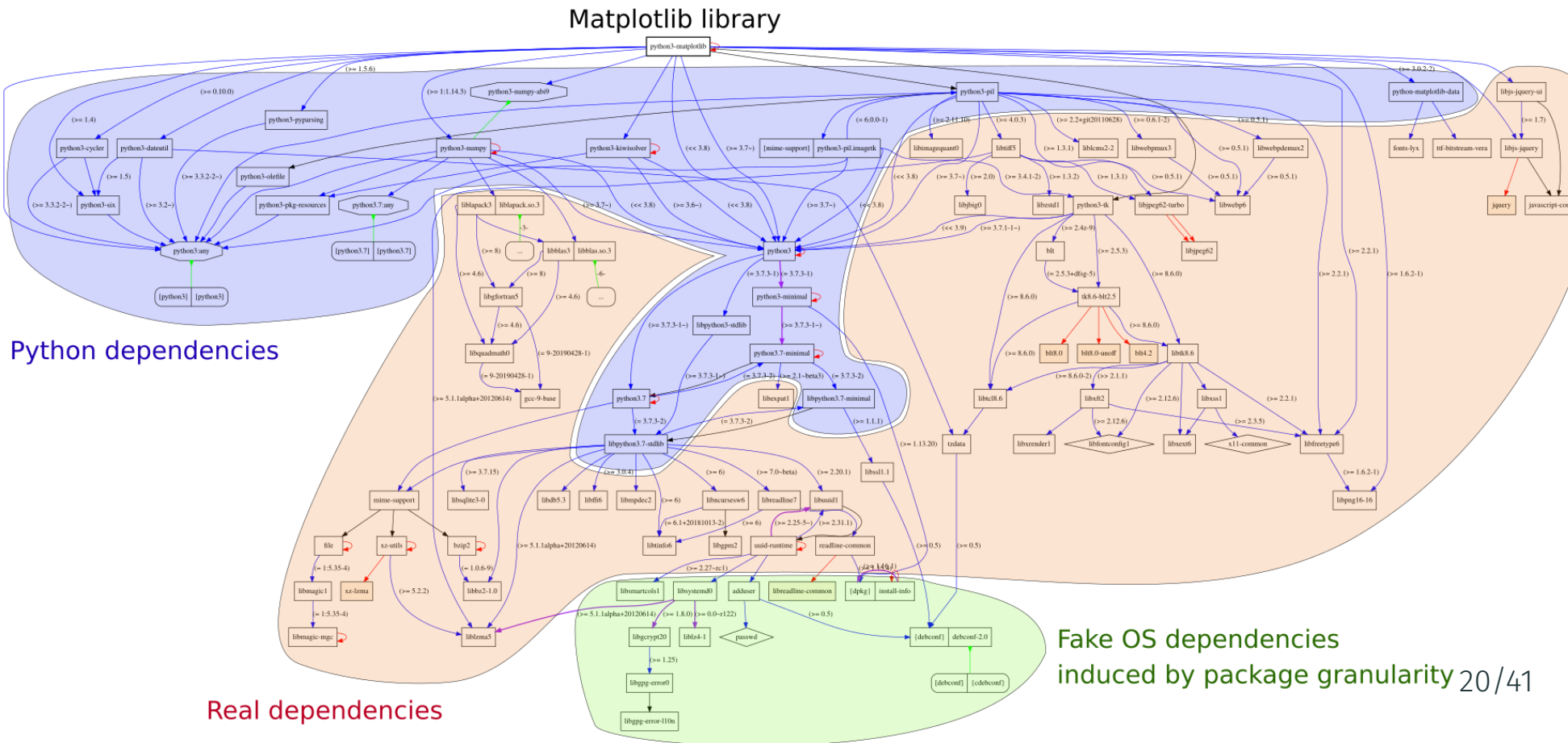
```
Package: python3-matplotlib
Version: 3.5.1-2+b1
Source: matplotlib (3.5.1-2)
Maintainer: Sandro Tosi <morph@debian.org>
Installed-Size: 27.6 MB
Depends: libjs-jquery, libjs-jquery-ui, python-matplotlib-data (>= 3.5.1),
         python3-dateutil, python3-pil.imagetk, python3-pyparsing (>= 1.5.6),
         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,
          python3-scipy, python3-sip, python3-tornado, texlive-extra-utils
```

Complex ecosystems

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

```
3.5.1
```

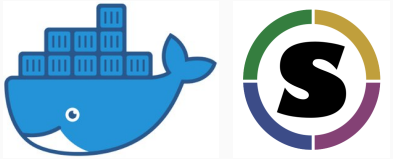
```
apt show python3-matplotlib
```



Tool 2: Containers and Package Managers

The good

The bad



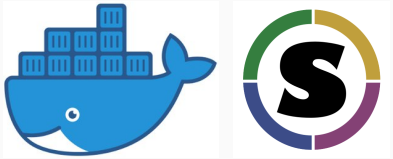
The uggly

Automatic tracking

Tool 2: Containers and Package Managers

The good

The bad



The uggly

Automatic tracking

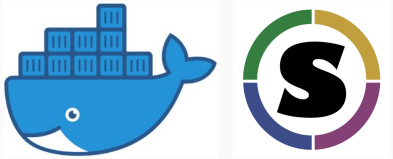
Containers

- 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*

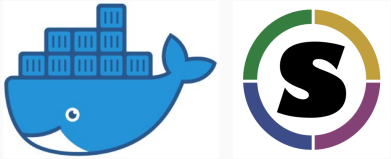
```
1  FROM ubuntu:20.04
2  RUN apt-get update
3       && apt-get upgrade -y
4       && apt-get install -y ...
```

- 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/Bioconductor`
  - 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)

Debian dependencies

```
1 dpkg --status python3-matplotlib
```

```
Package: python3-matplotlib
Version: 3.6.3-1+b1
Source: matplotlib (3.6.3-1)
Depends: libjs-jquery, libjs-jquery-ui, python-matplotlib-data (>= 3.6.3),
         python3-dateutil, python3-pil.imagetk, python3-pyparsing (>= 1.5.6),
         python3-six (>= 1.4), python3-numpy (>= 1:1.22.0), python3-contourpy,
         python3 (<< 3.12), python3 (>= 3.11~), python3-numpy-abi9,
         python3-cycler (>= 0.10.0), python3-fonttools, python3-kiwisolver,
         python3-packaging, python3-pil, python3:any, libc6 (>= 2.34),
         libfreetype6 (>= 2.2.1), libgcc-s1 (>= 3.3.1),
         libqhull-r8.0 (>= 2020.1), libstdc++6 (>= 11)
```

Debian dependencies

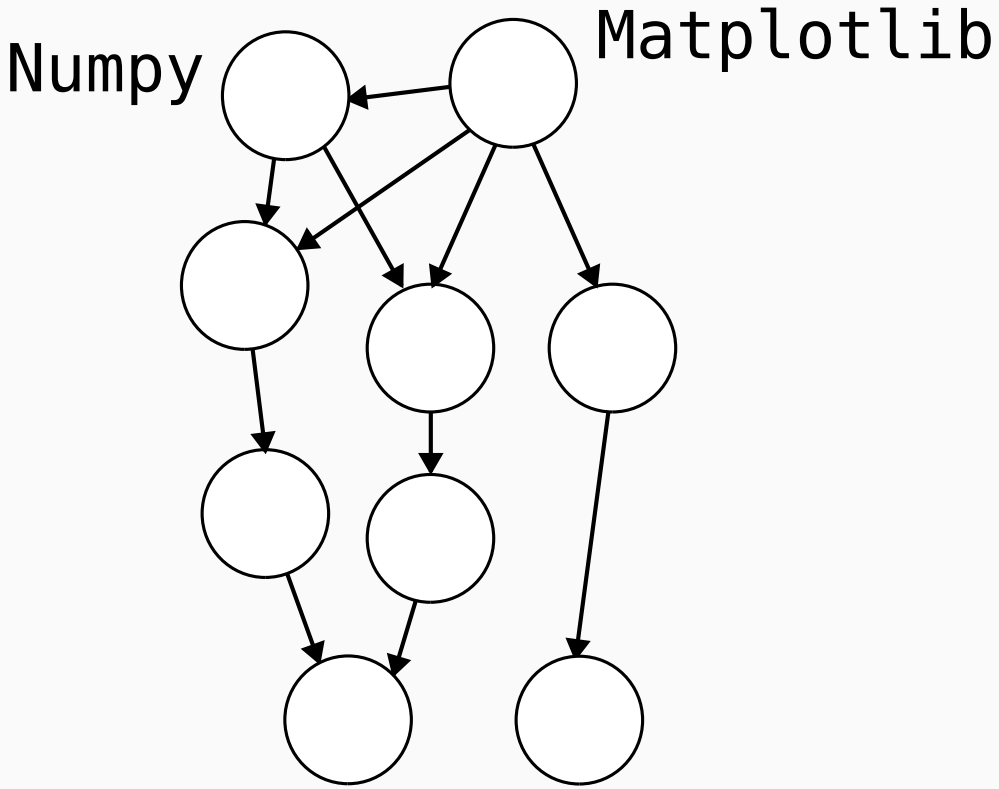
```
dpkg --status python3-matplotlib
```

```
Package: python3-matplotlib
Version: 3.6.3-1+b1
Source: matplotlib (3.6.3-1)
Depends: libjs-jquery, libjs-jquery-ui, python-matplotlib-data (>= 3.6.3),
         python3-dateutil, python3-pil.imagetk, python3-pyparsing (>= 1.5.6),
         python3-six (>= 1.4), python3-numpy (>= 1:1.22.0), python3-contourpy,
         python3 (<< 3.12), python3 (>= 3.11~), python3-numpy-abi9,
         python3-cycler (>= 0.10.0), python3-fonttools, python3-kiwisolver,
         python3-packaging, python3-pil, python3:any, libc6 (>= 2.34),
         libfreetype6 (>= 2.2.1), libgcc-s1 (>= 3.3.1),
         libqhull-r8.0 (>= 2020.1), libstdc++6 (>= 11)
```

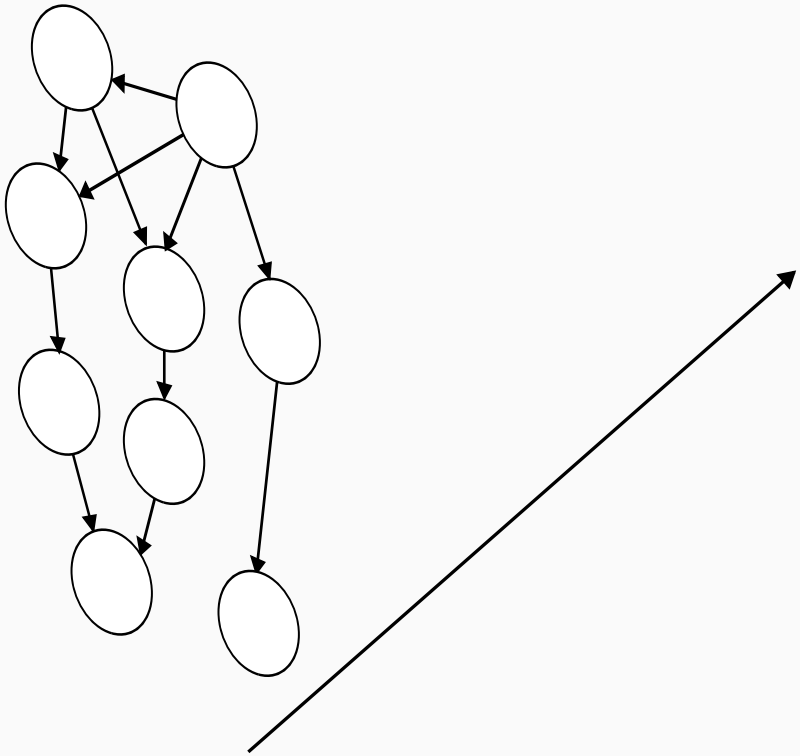
On a given day:

- Several versions of each package are available on the server
- Installing the latest version of a package may require upgrading some other packages

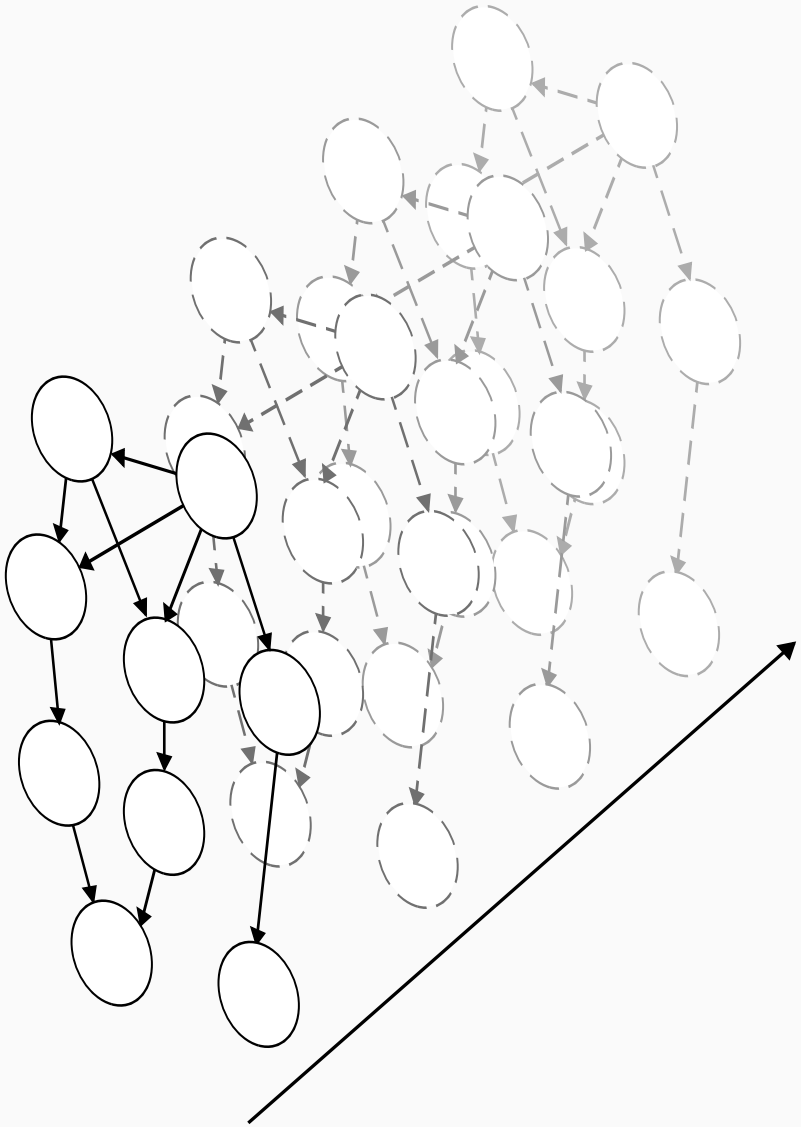
Loose vs. strict dependencies in picture



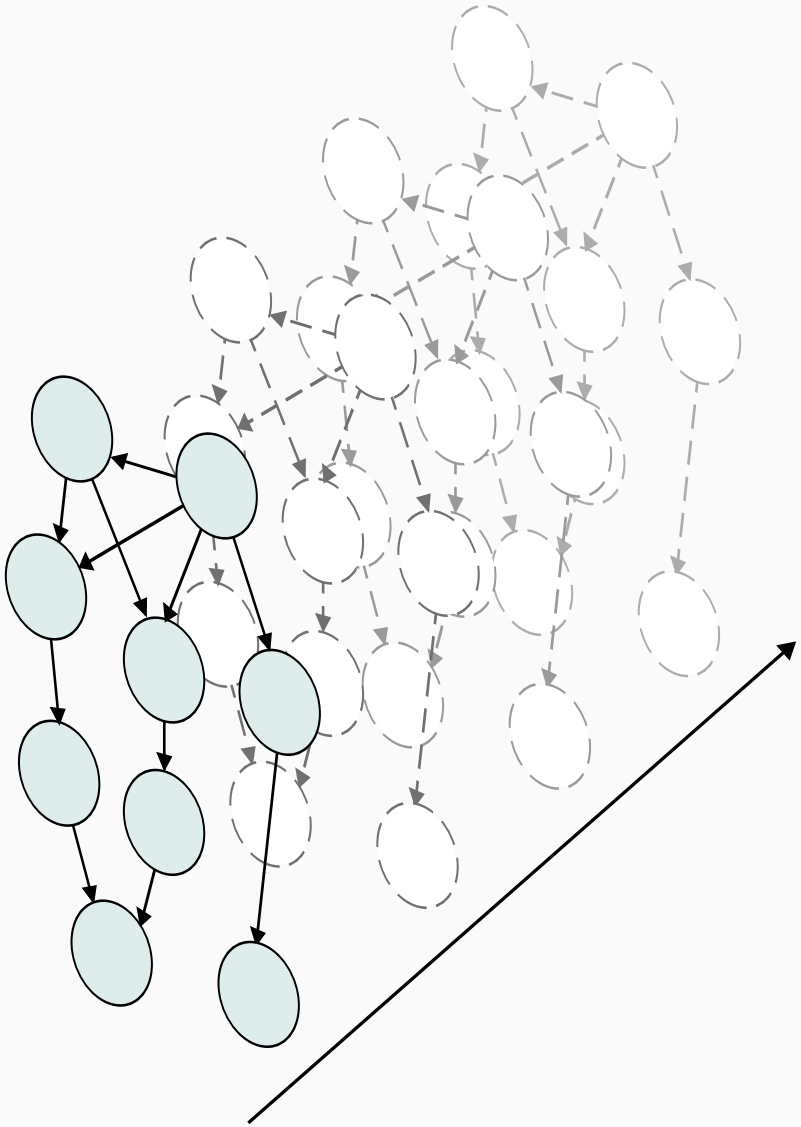
Loose vs. strict dependencies in picture



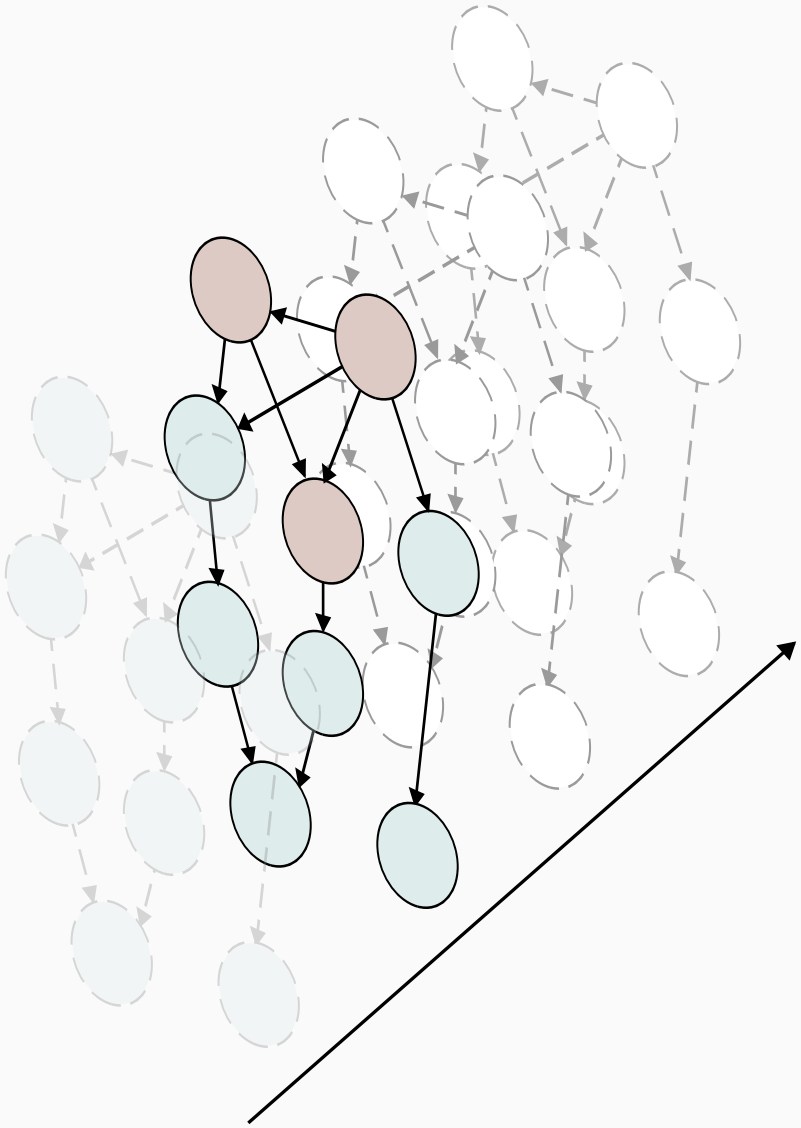
Loose vs. strict dependencies in picture



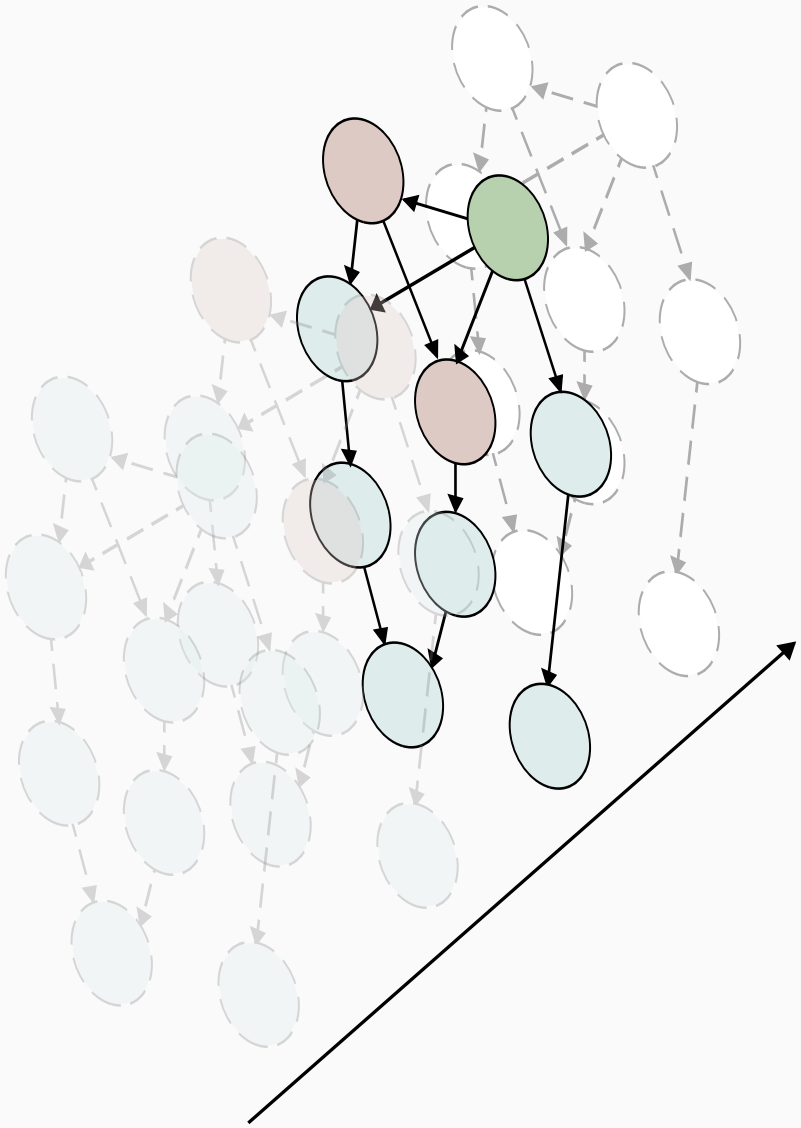
Loose vs. strict dependencies in picture



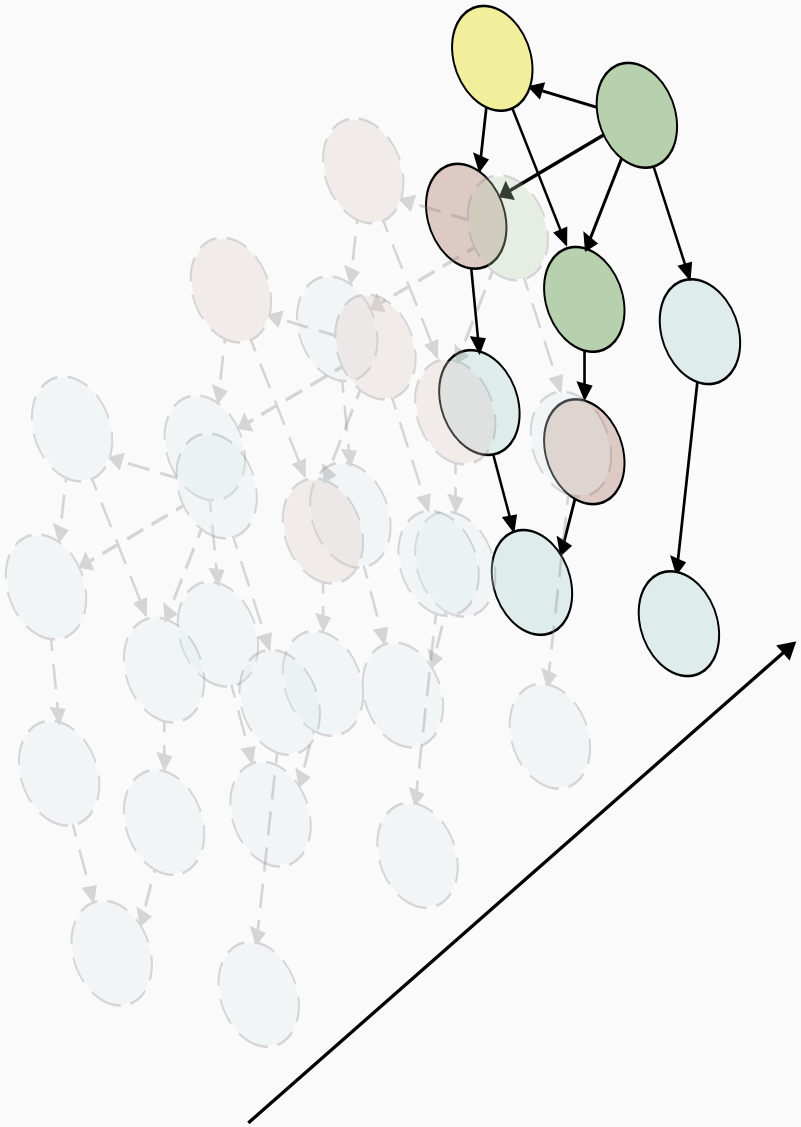
Loose vs. strict dependencies in picture



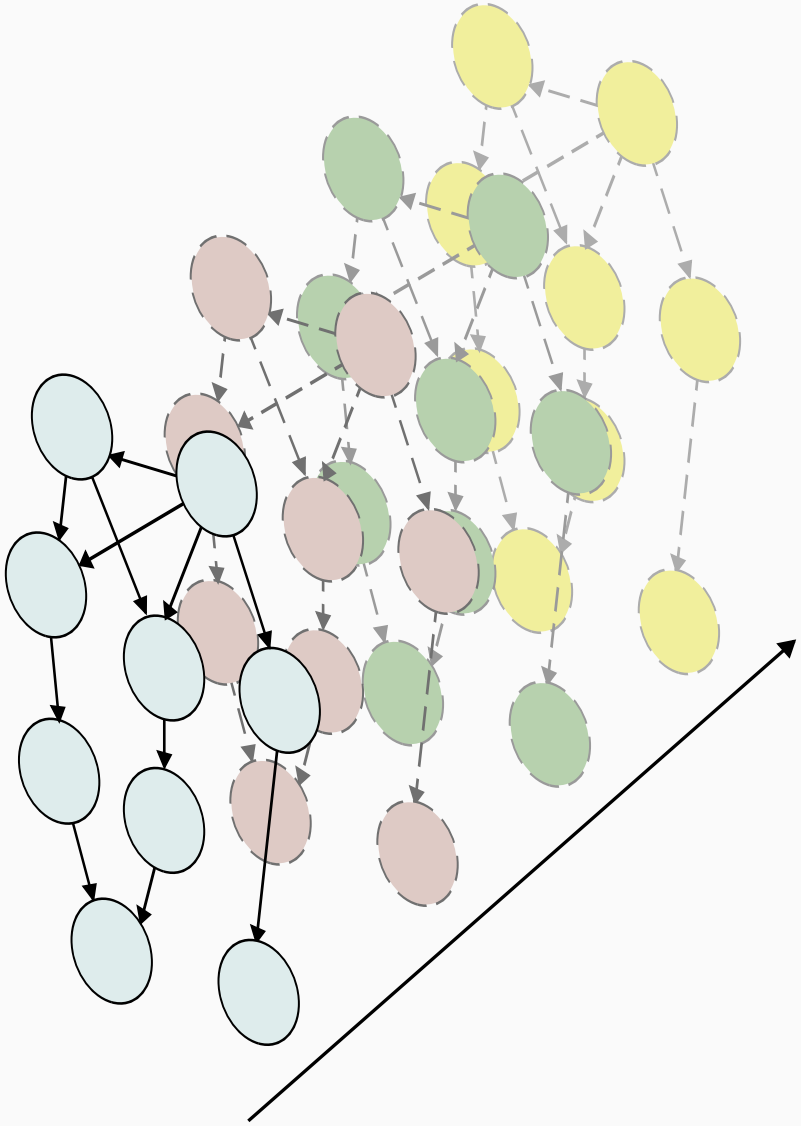
Loose vs. strict dependencies in picture



Loose vs. strict dependencies in picture

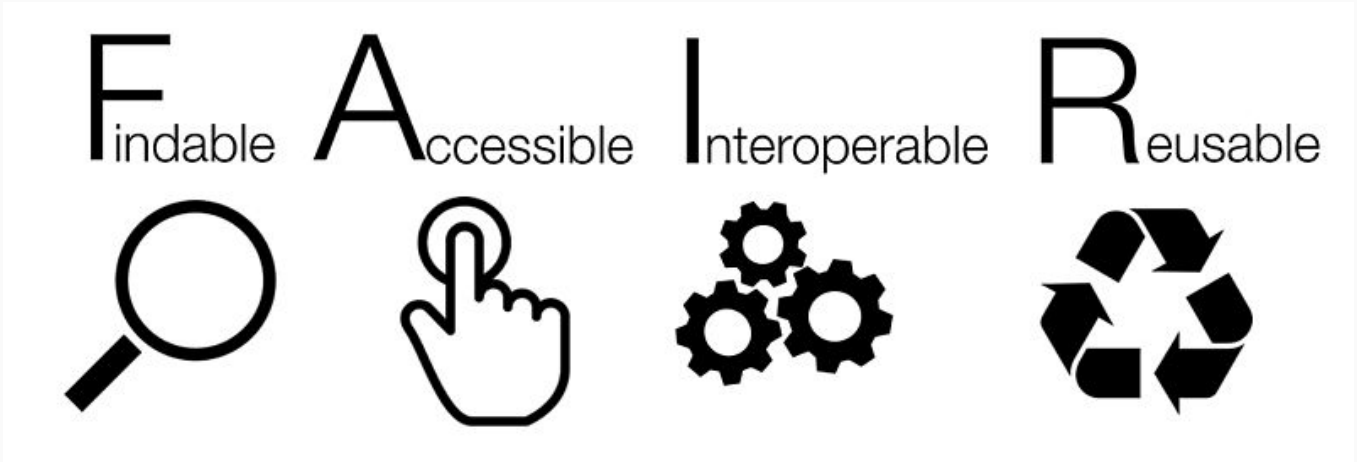


Loose vs. strict dependencies in picture



Good Practice #3
Version Control and Archiving

FAIR principles



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

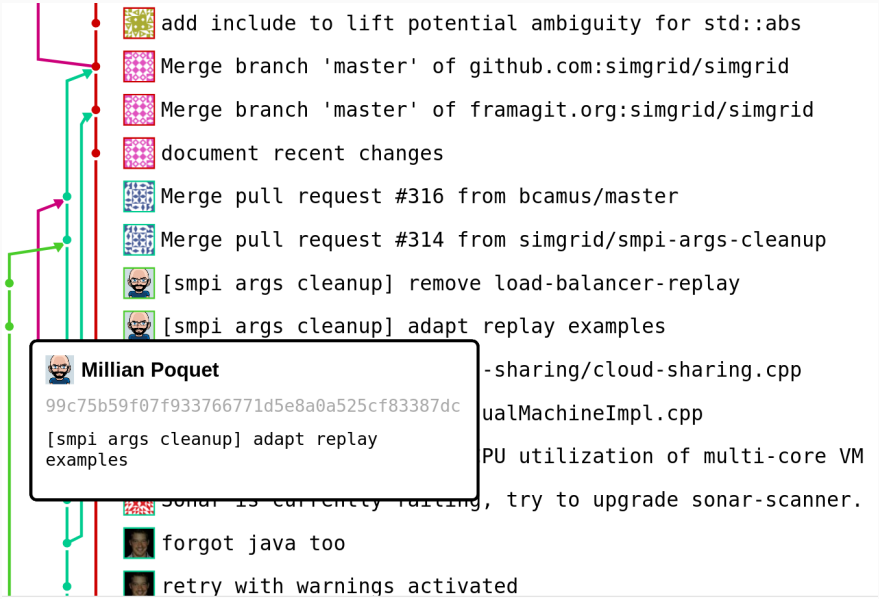
- *"Open as much as possible and close as much as necessary"*
- Management, publication, annotation (metadata), archiving
- Source code = specific data with specific consideration

Let's go beyond general principles!

Tool 3: Version Control and Forge

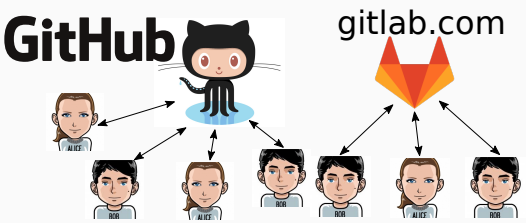
Git = version control

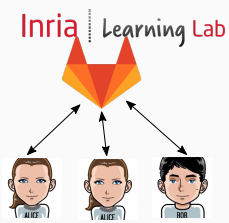
- Developed in 2005 by Linus Torvalds for the kernel development
- Local and efficient rollbacks
- Distributed: everyone has a full copy of the history

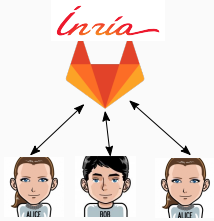


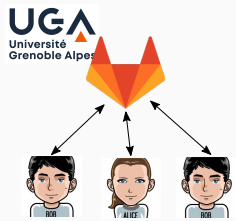
GitHub, GitLab, and Co

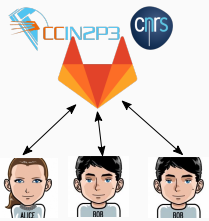
- Free hosting of public projects, social network











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 (≠ 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

Data archives figshare zenodo

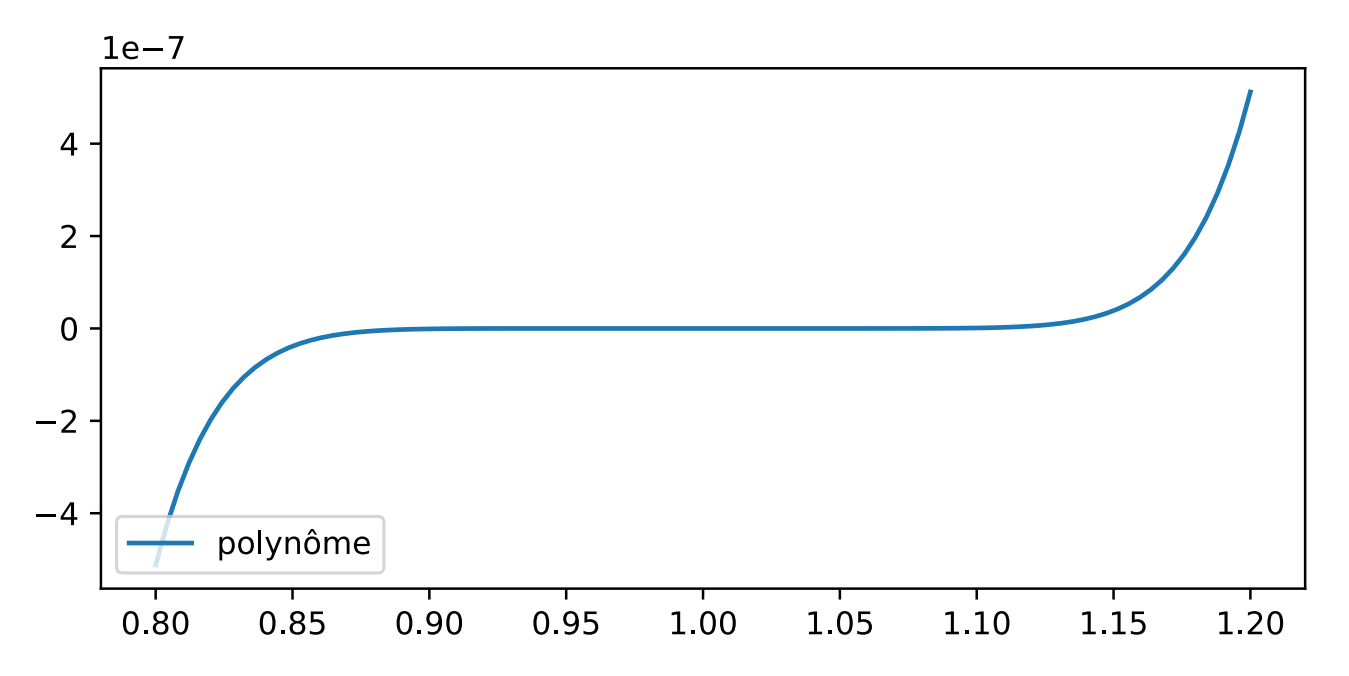
Software Archive Software Heritage

INTERNET ARCHIVE

Collect/Preserve/Share

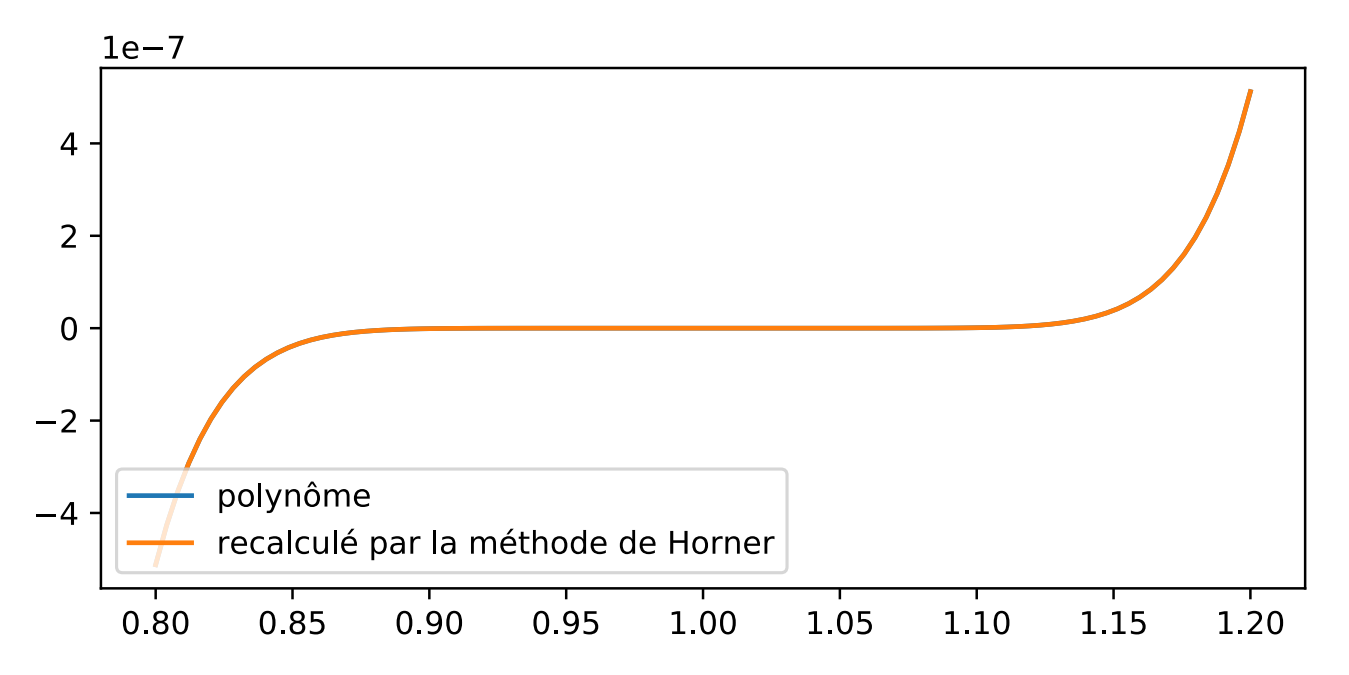
Controling the whole software/compiling stack is not sufficient

All I care about is the algorithm output (FP)



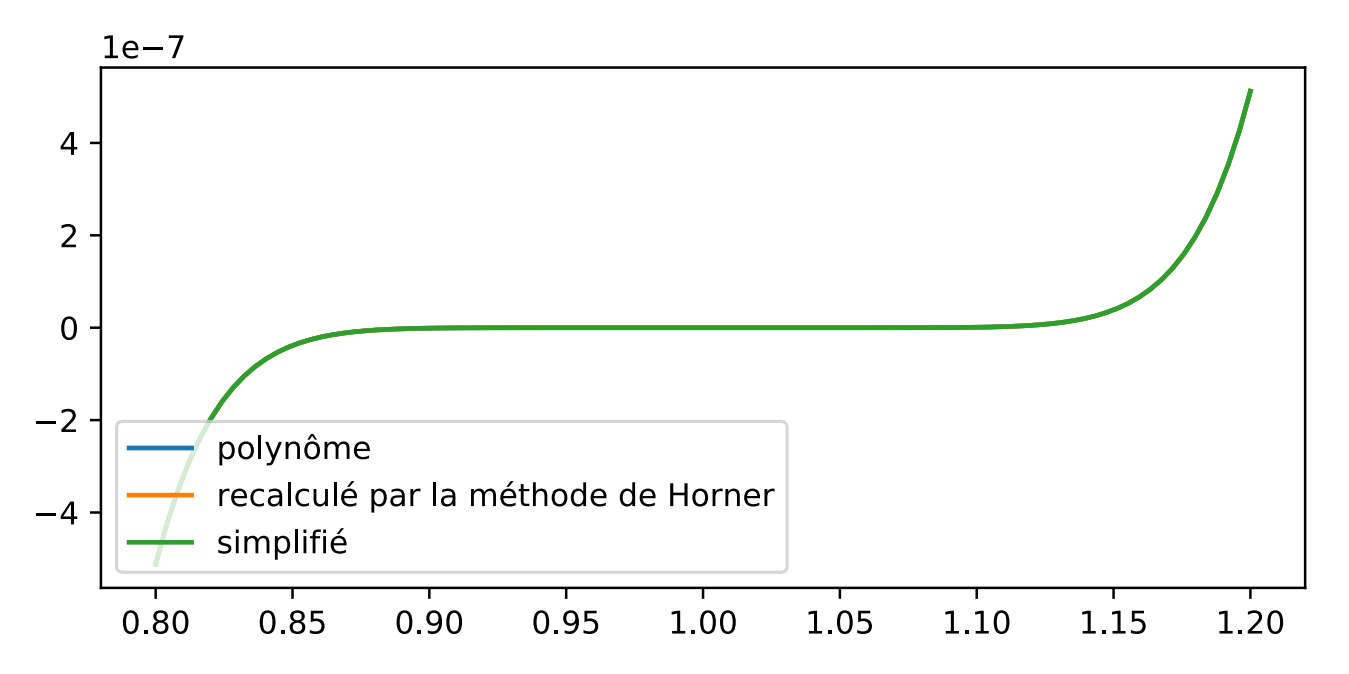

```
1  def polynome(x):
2      return x**9 - 9.*x**8 + 36.*x**7 - 84.*x**6 + 126.*x**5 \
3             - 126.*x**4 + 84.*x**3 - 36.*x**2 + 9.*x - 1.
```

Floating-point arithmetic



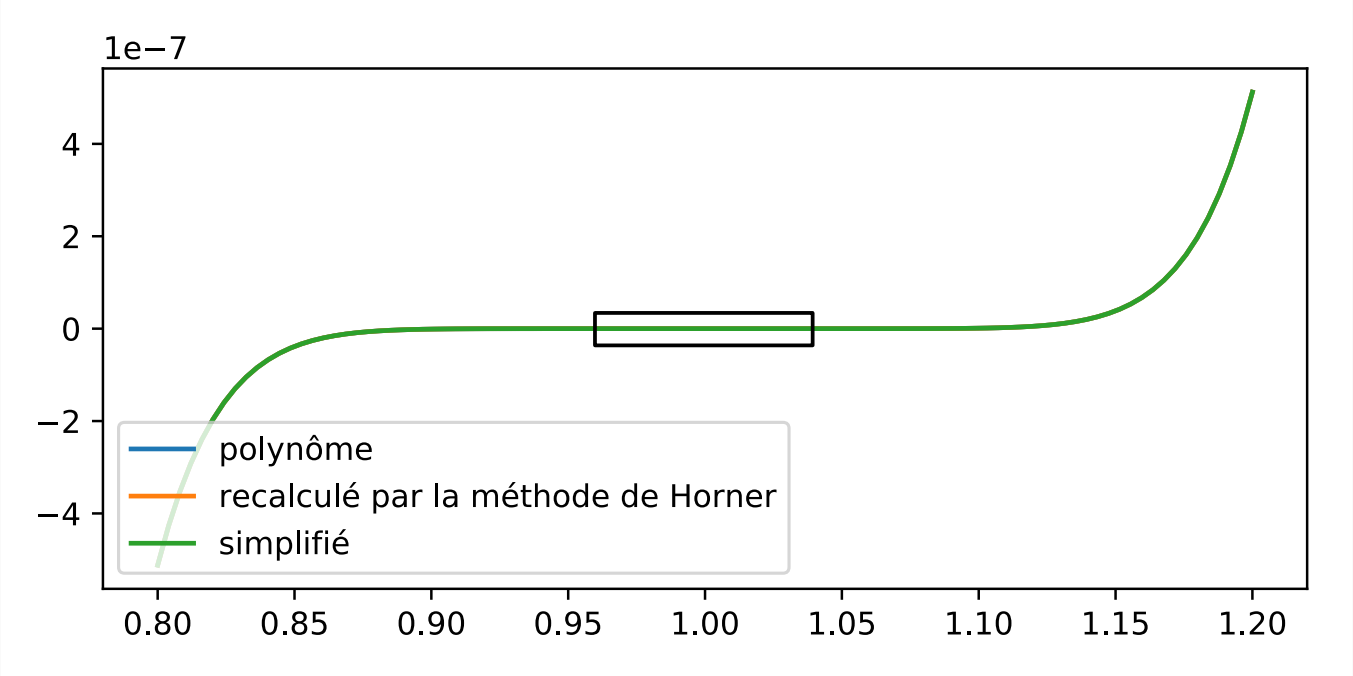

```
1  def horner(x):
2      return x*(x*(x*(x*(x*(x*(x*(x*(x - 9.) + 36.) - 84.) + 126.) \
3             - 126.) + 84.) - 36.) + 9.) - 1.
```

Floating-point arithmetic

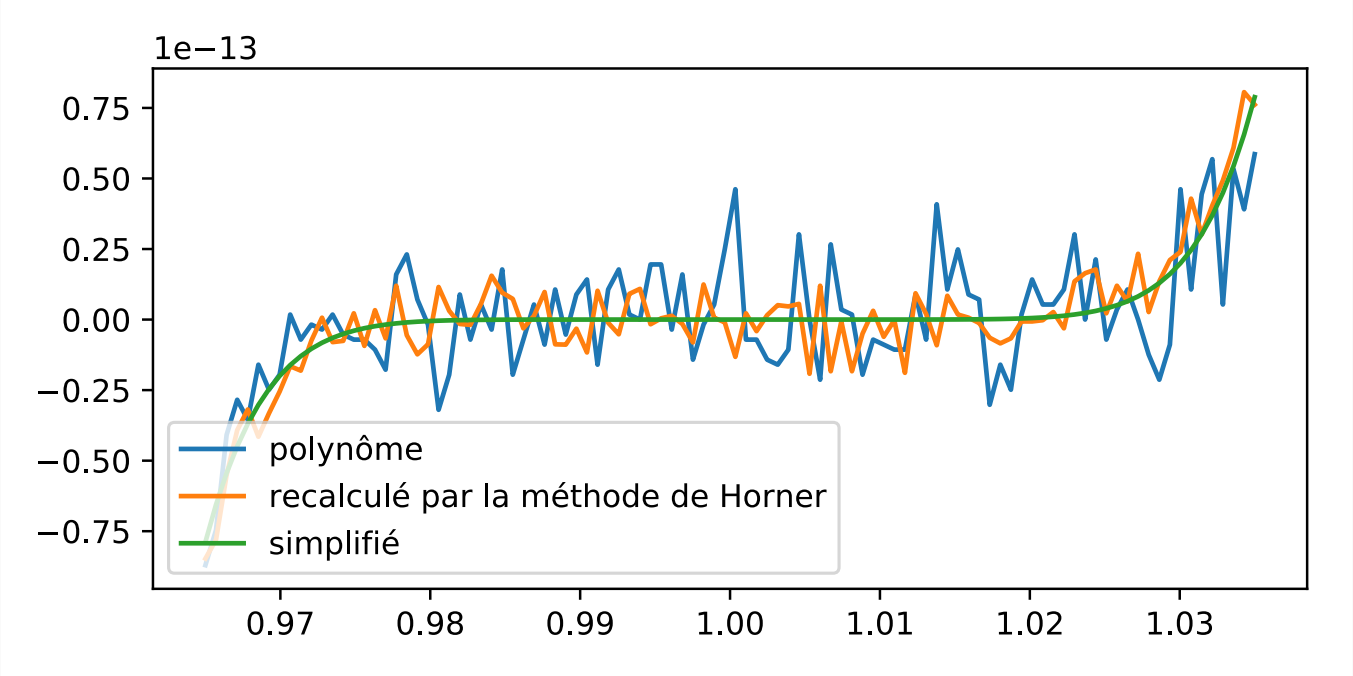



```
1  def simple(x):
2      return (x-1.)**9
3  # Easy! ;)
```

Floating-point arithmetic



Floating-point arithmetic



Rounding

- Every operation includes implicit rounding.
- a+b is actually round(a+b).
- Unfortunately:

$$\texttt{round}(\texttt{round}(a+b)+c) \neq \texttt{round}(a+\texttt{round}(b+c)).$$

- Operation order therefore matters.

For a reproducible computation, operation order must be preserved!!!

How to explain it to my compiler?

To speed up computations, compilers may change operation order, and thus results.

Two options for computing reproducibly:

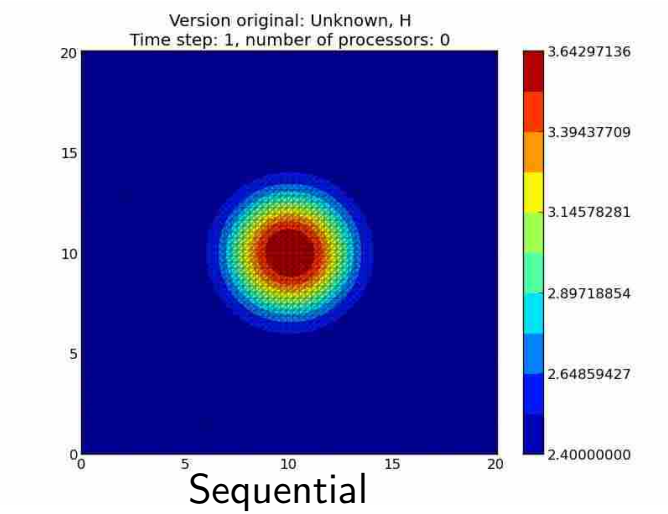
1. Insist on the preservation of operation order,
   - if the language permits it.
   - Example: Module ‘ieee_arithmetic‘ in Fortran 2003
2. Make compilation reproducible:
   - Record the precise compiler version
   - Record all compilation options

Telemac2D: the simplest gouttedo simulation

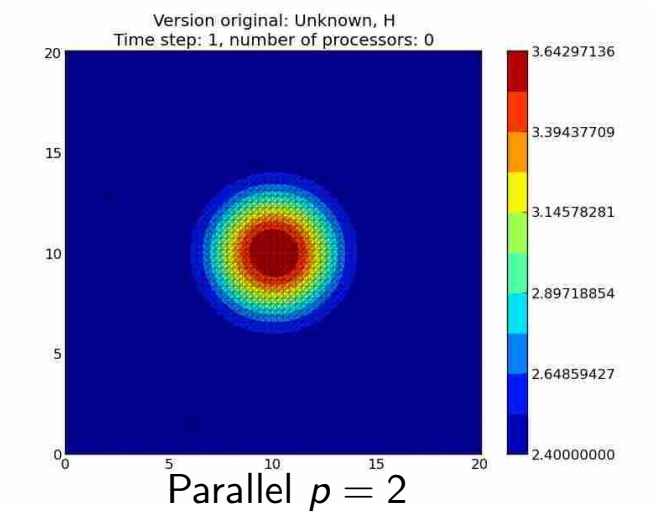
The gouttedo test case

- 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



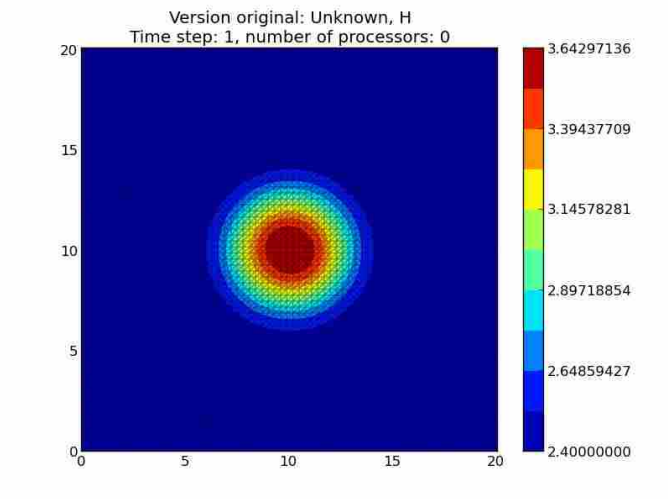
Parallel $p = 2$

Courtesy of P. Langlois and R. Nheili

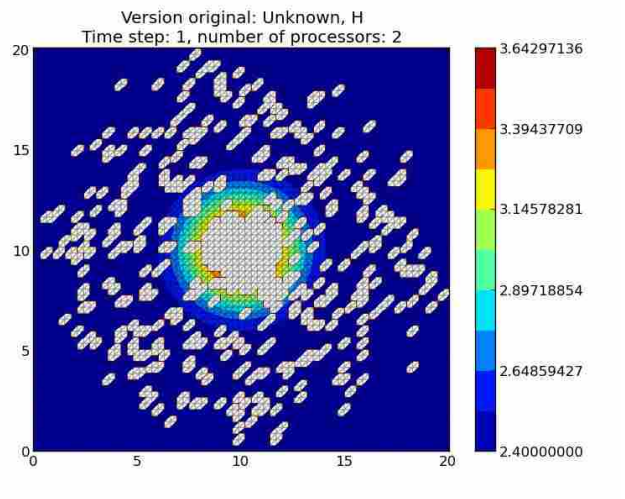
A white plot displays a non-reproducible value

Numerical reproducibility?

time step = 1



Sequential



Parallel $p = 2$

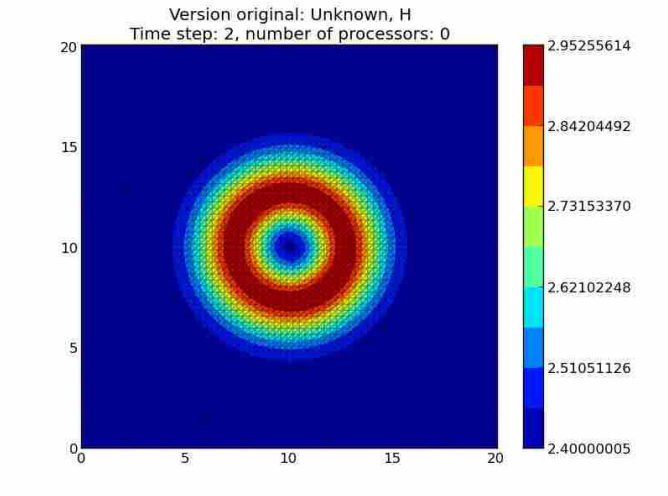
14 / 54

Courtesy of P. Langlois and R. Nheili

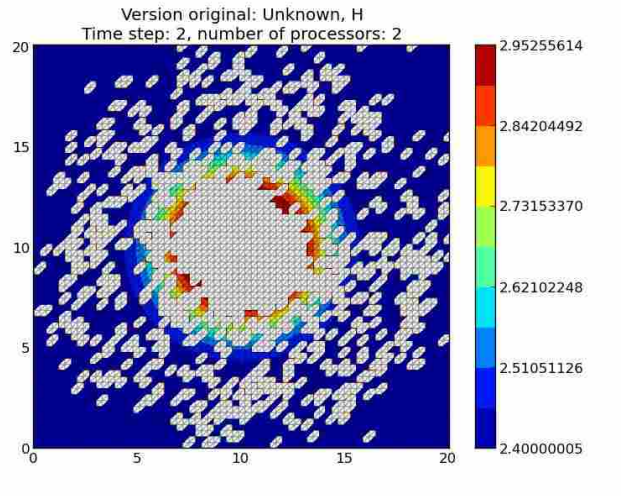
A white plot displays a non-reproducible value

Numerical reproducibility?

time step = 2



Sequential



Parallel $p = 2$

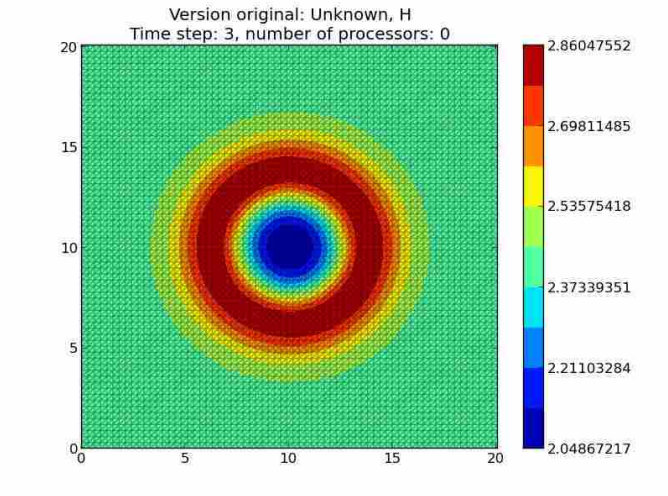
14 / 54

Courtesy of P. Langlois and R. Nheili

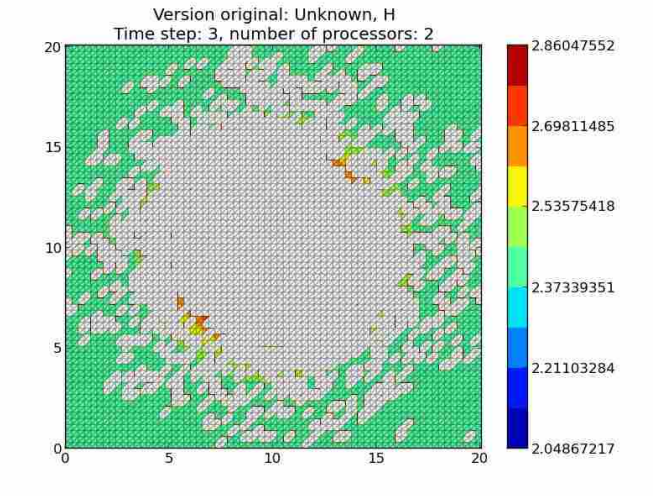
A white plot displays a non-reproducible value

Numerical reproducibility?

time step = 3



Sequential



Parallel $p = 2$

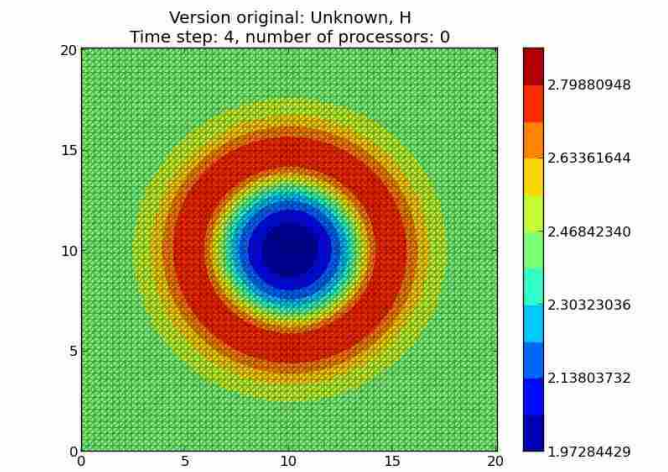
Courtesy of P. Langlois and R. Nheili

Did I mention we have parallel machines nowadays?

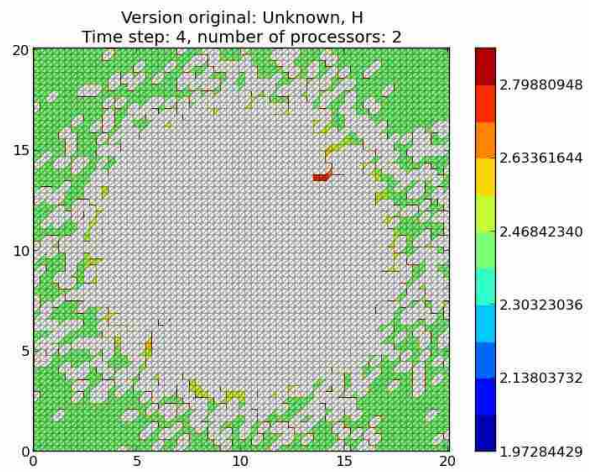
A white plot displays a non-reproducible value

Numerical reproducibility?

time step = 4



Sequential



Parallel $p = 2$

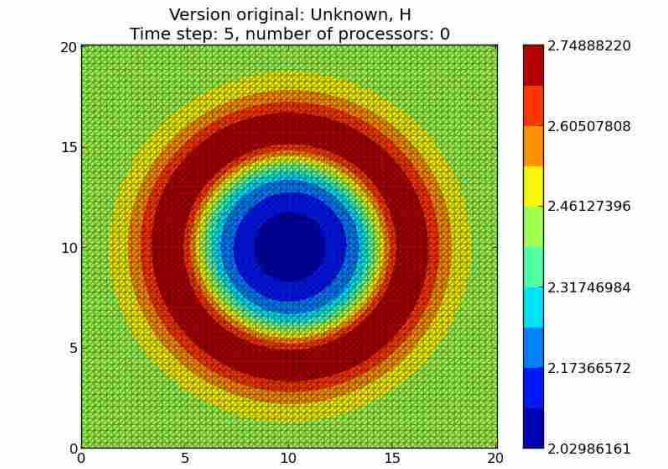
14 / 54

Courtesy of P. Langlois and R. Nheili

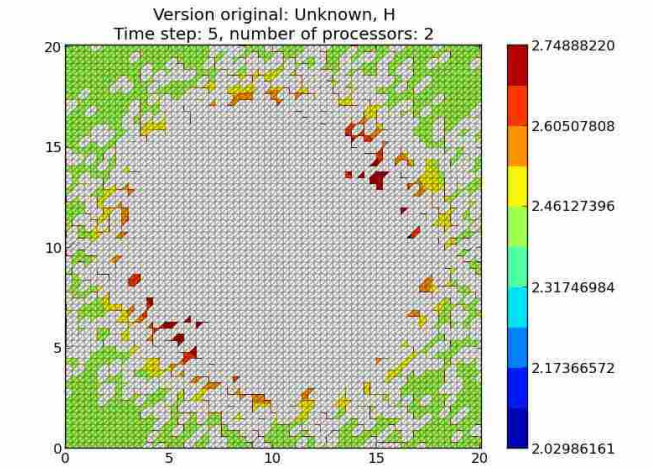
A white plot displays a non-reproducible value

Numerical reproducibility?

time step = 5



Sequential



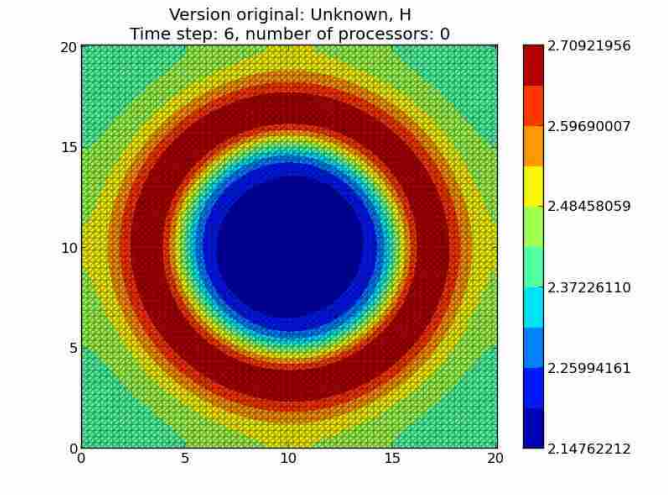
Parallel $p = 2$

Courtesy of P. Langlois and R. Nheili

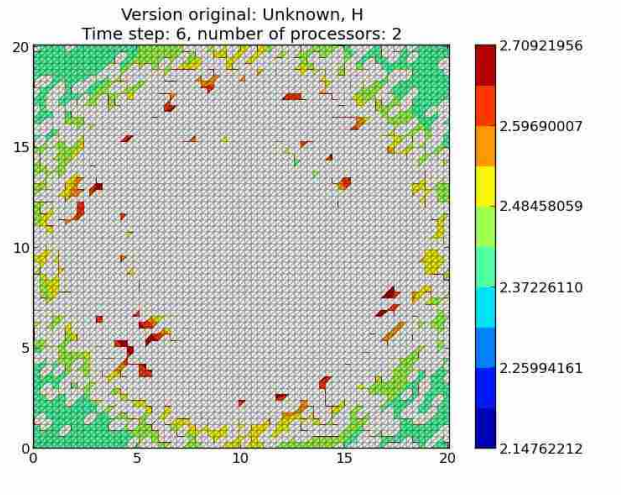
A white plot displays a non-reproducible value

Numerical reproducibility?

time step = 6



Sequential



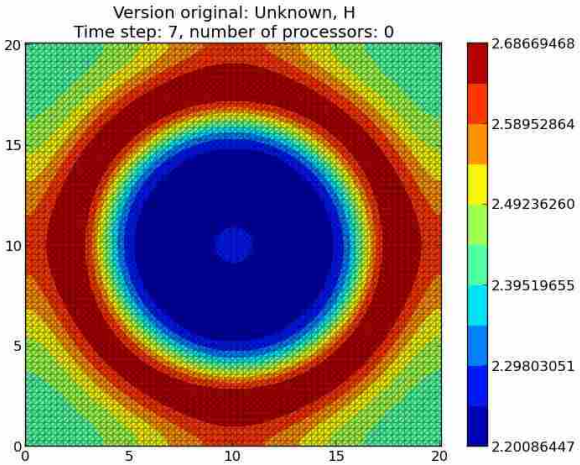
Parallel $p = 2$

Courtesy of P. Langlois and R. Nheili

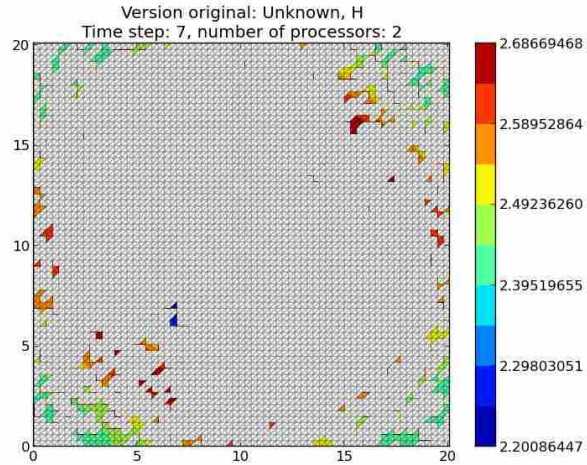
A white plot displays a non-reproducible value

Numerical reproducibility?

time step = 7



Sequential



Parallel $p = 2$

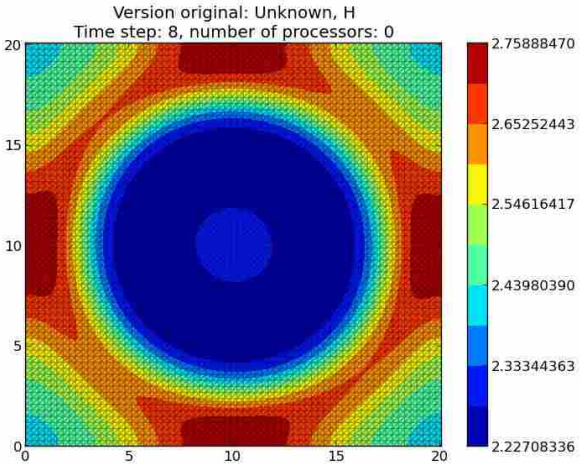
14 / 54

Courtesy of P. Langlois and R. Nheili

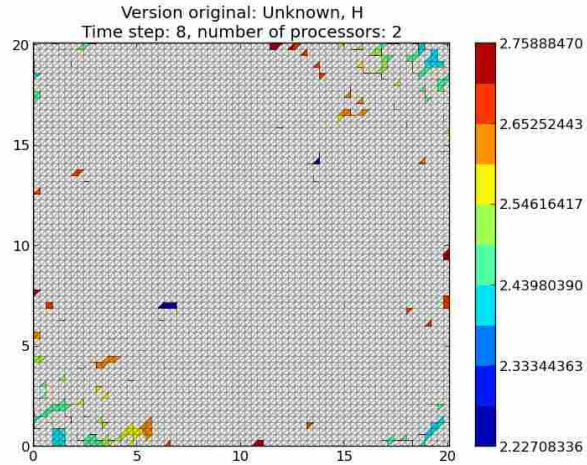
A white plot displays a non-reproducible value

Numerical reproducibility?

time step = 8



Sequential



Parallel $p = 2$

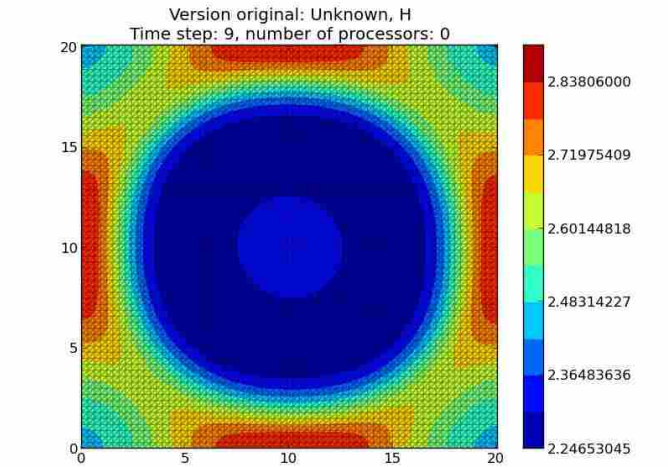
14 / 54

Courtesy of P. Langlois and R. Nheili

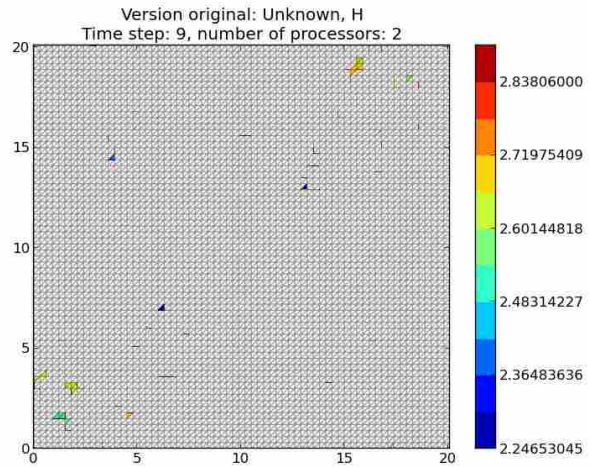
A white plot displays a non-reproducible value

Numerical reproducibility?

time step = 9



Sequential



Parallel $p = 2$

14 / 54

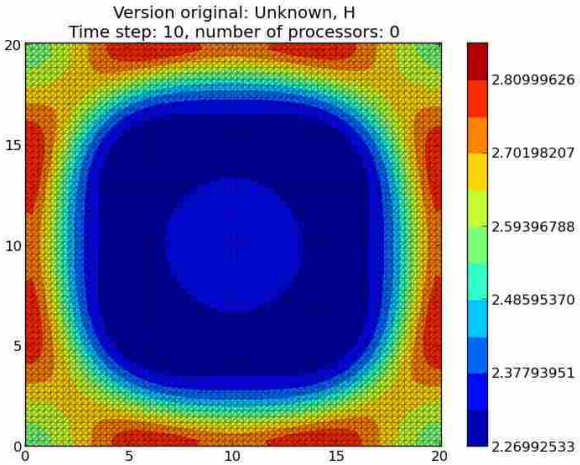
Courtesy of P. Langlois and R. Nheili

Did I mention we have parallel machines nowadays?

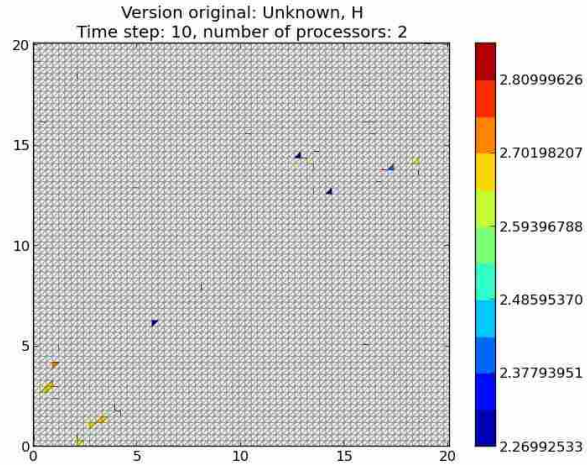
A white plot displays a non-reproducible value

Numerical reproducibility?

time step = 10



Sequential



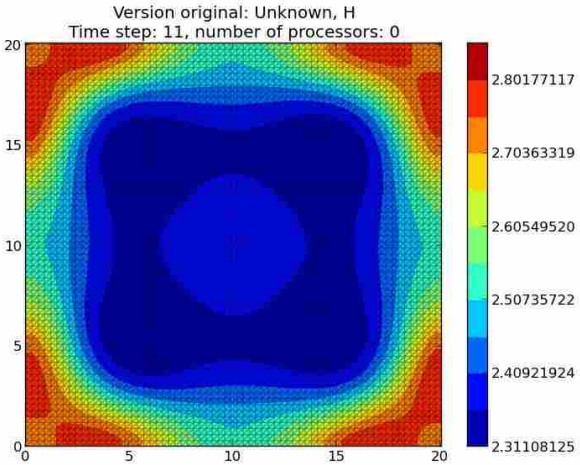
Parallel $p = 2$

Courtesy of P. Langlois and R. Nheili

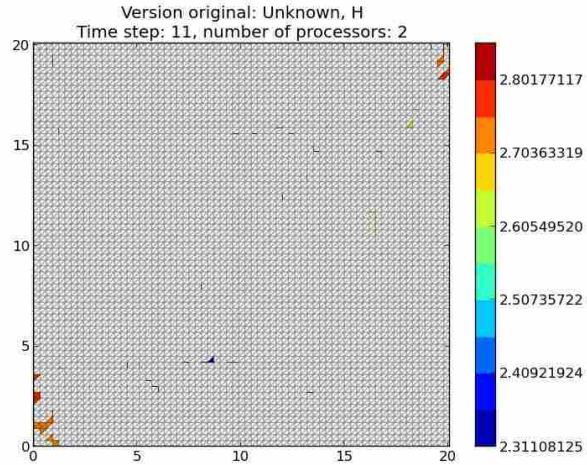
A white plot displays a non-reproducible value

Numerical reproducibility?

time step = 11



Sequential



Parallel $p = 2$

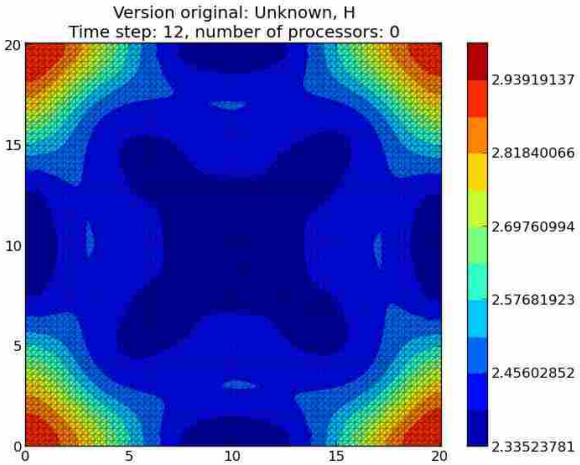
Courtesy of P. Langlois and R. Nheili

Did I mention we have parallel machines nowadays?

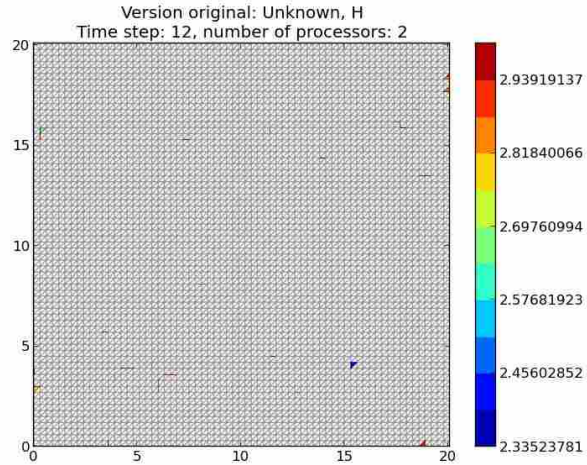
A white plot displays a non-reproducible value

Numerical reproducibility?

time step = 12



Sequential



Parallel $p = 2$

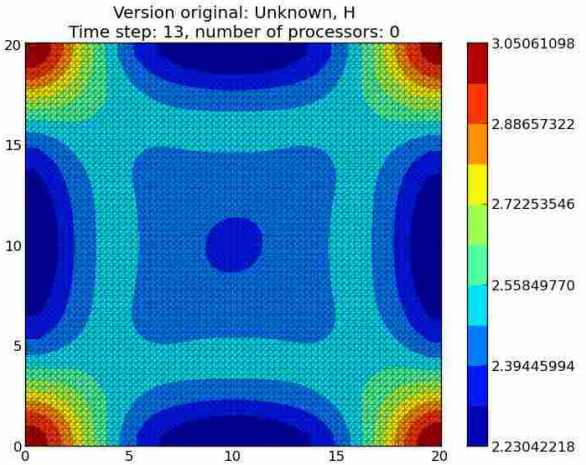
14 / 54

Courtesy of P. Langlois and R. Nheili

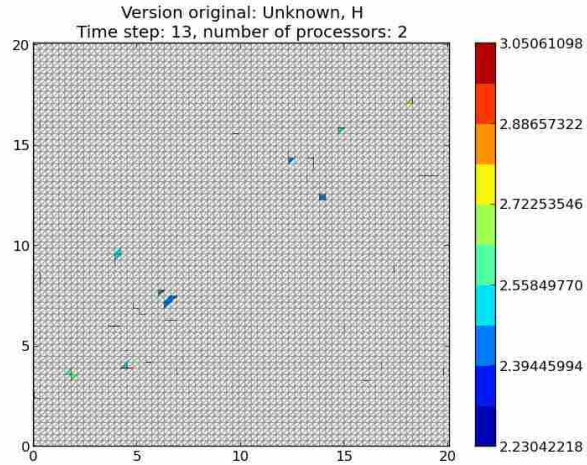
A white plot displays a non-reproducible value

Numerical reproducibility?

time step = 13



Sequential



Parallel $p = 2$

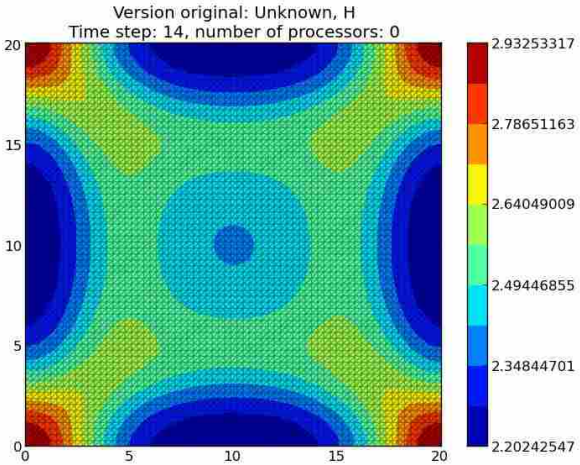
14 / 54

Courtesy of P. Langlois and R. Nheili

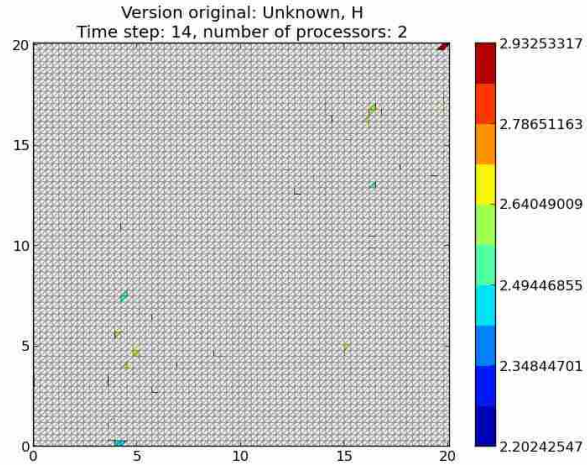
A white plot displays a non-reproducible value

Numerical reproducibility?

time step = 14



Sequential



Parallel $p = 2$

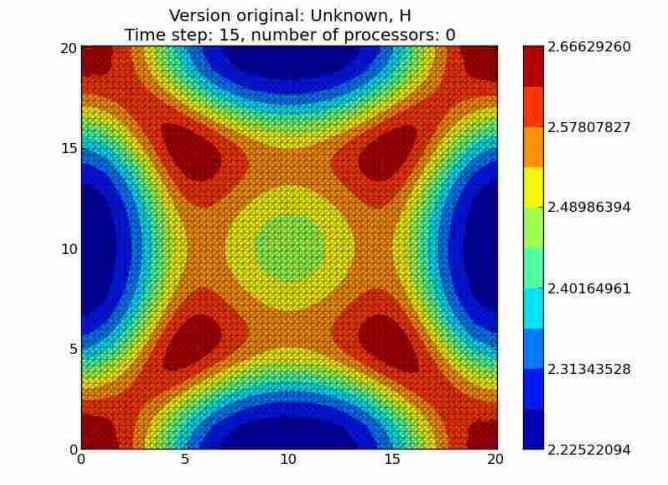
14 / 54

Courtesy of P. Langlois and R. Nheili

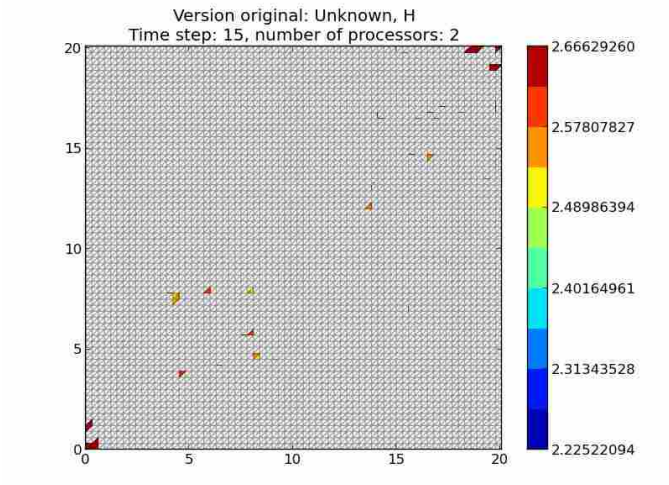
A white plot displays a non-reproducible value

NO numerical reproducibility!

time step = 15



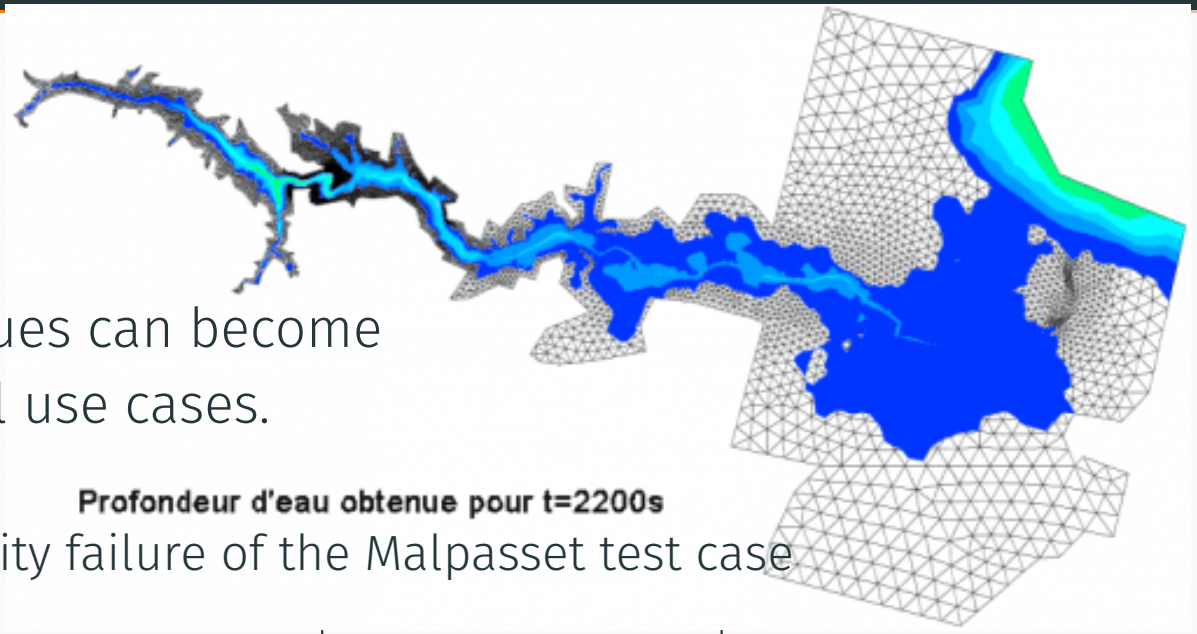
Sequential



Parallel $p = 2$

Courtesy of P. Langlois and R. Nheili

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 the algorithm, in its implementation, in its execution.

The whole chain needs to be revisited.

Courtesy of P. Langlois and R. Nheili

Software/Hardware dependencies

Runtime dependencies interpreter, libraries, other programs

Build dependencies compilers, headers, `autotools/cmake`, etc.

Software/Hardware dependencies

Runtime dependencies interpreter, libraries, other programs

Build dependencies compilers, headers, `autotools/cmake`, etc.

Version a loose description

- `v_1.4.2`

Software/Hardware dependencies

Runtime dependencies interpreter, libraries, other programs

Build dependencies compilers, headers, `autotools/cmake`, etc.

Version a loose description

- `v_1.4.2` (commit `78c293967830979f988b39cea2587a7c26b`),

Software/Hardware dependencies

Runtime dependencies interpreter, libraries, other programs

Build dependencies compilers, headers, `autotools/cmake`, etc.

Version a loose description

- `v_1.4.2` (commit `78c293967830979f988b39cea2587a7c26b`), (built/run in *this* environment with *such and such* options)

Software/Hardware dependencies

Runtime dependencies interpreter, libraries, other programs

Build dependencies compilers, headers, `autotools/cmake`, etc.

Version a loose description

- `v_1.4.2` (commit `78c293967830979f988b39cea2587a7c26b`), (built/run in *this* environment with *such and such* options)
- `v_1.5` or `v_1.4.3` are often *better* than `v_1.4.2`

Software/Hardware dependencies

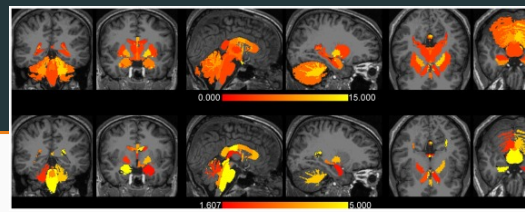
Runtime dependencies interpreter, libraries, other programs

Build dependencies compilers, headers, `autotools/cmake`, etc.

Version a loose description

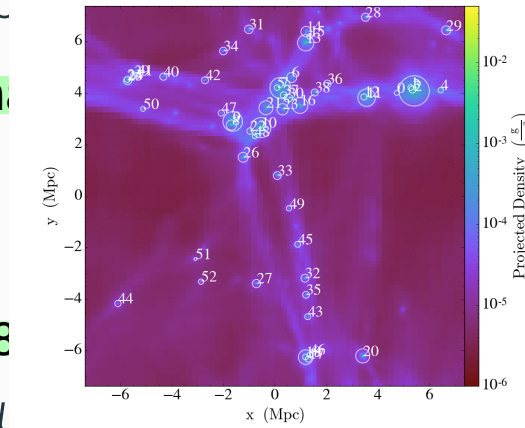
- `v_1.4.2` (commit `78c293967830979f988b39cea2587a7c26b`), (built/run in *this* environment with *such and such* options)
- `v_1.5` or `v_1.4.3` are often *better* than `v_1.4.2`
- Convenient but very problematic for reproducibility

Software/Hardware dependencies



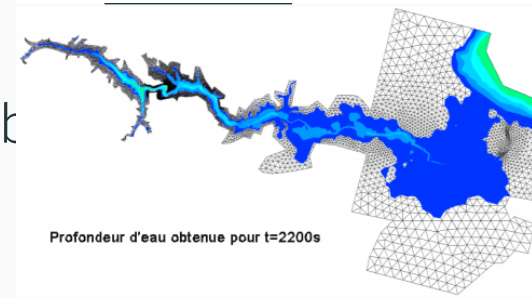
Runtime dependencies interpreter, libraries, other pro

Build dependencies compilers, headers, `autotools/cm`



Version a loose description

- `v_1.4.2` (commit `78c293967830979f988b39cea258`
  (built/run in *this* environment with *such and su*
- `v_1.5` or `v_1.4.3` are often *better* than `v_1.4.2`
- Convenient but very problematic for reproducib



It is not a good sign if your code is sensitive.

Software/Hardware dependencies

Runtime dependencies interpreter, libraries, other programs

Build dependencies compilers, headers, `autotools/cmake`, etc.

Version a loose description

- `v_1.4.2` (commit `78c293967830979f988b39cea2587a7c26b`), (built/run in *this* environment with *such and such* options)
- `v_1.5` or `v_1.4.3` are often *better* than `v_1.4.2`
- Convenient but very problematic for reproducibility

It is not a good sign if your code is sensitive. You need:

1. Variation generation (test)
2. Perfect control (debug)

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

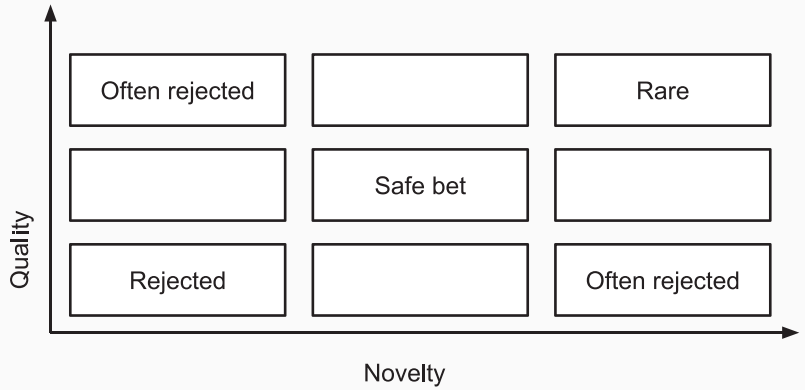
Joelle Pineau @ NeurIPS'18

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

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

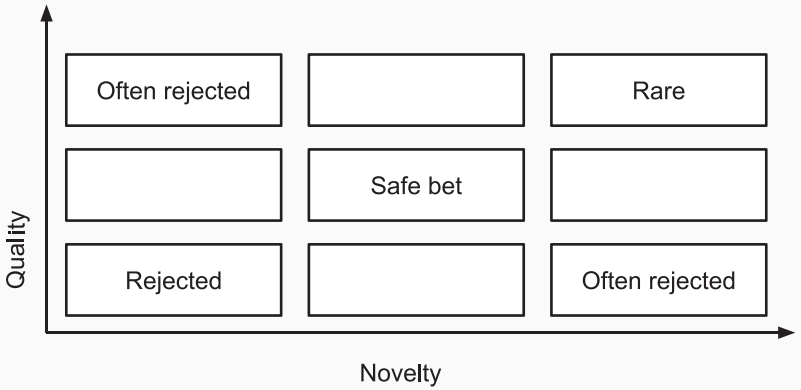
Changing Academic Practices (~~Publish or Perish~~)

- 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 (×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*

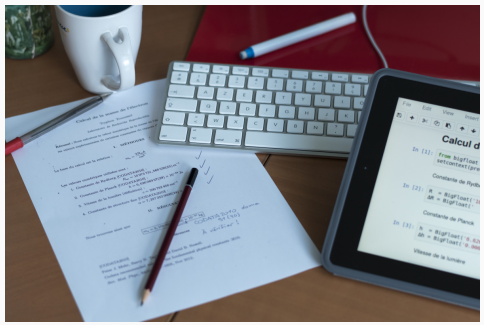
Reproducible Research = Rigor and Transparency

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

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

- Konrad Hinsen, Christophe Pouzat
- 3rd Edition: March 2020 – … (16,800+)

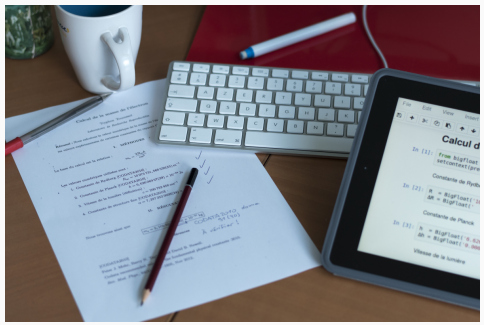


Reproducible Research = Rigor and Transparency

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

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



- Konrad Hinsen, Christophe Pouzat
- 3rd Edition: March 2020 – … (16,800+)

MOOC "Advanced RR" planned for May 2024

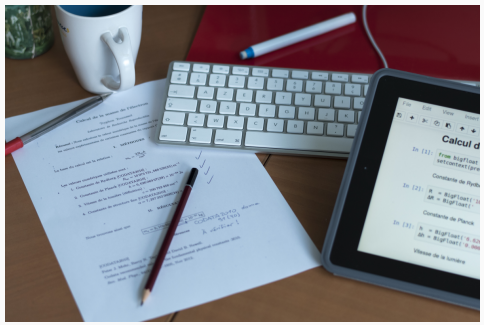
- Managing data (`FITS/HDF5`, `git annex`)
- Software environment control (`docker`, `singularity`, `guix`)
- Scientific workflow (`make`, `snakemake`)

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

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



- Konrad Hinsen, Christophe Pouzat
- 3rd Edition: March 2020 – … (16,800+)

MOOC "Advanced RR" planned for May 2024

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

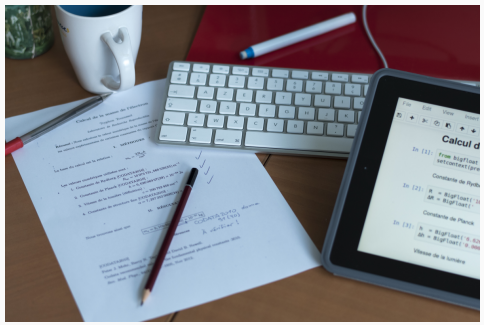
2. **Change the norm:** make publication practices evolve
3. **Incentive**: consider RR/open science when hiring/promoting

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

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

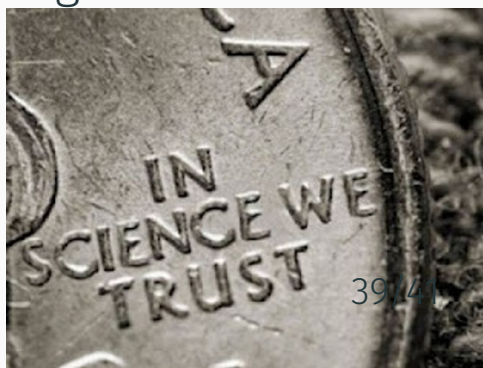


- Konrad Hinsen, Christophe Pouzat
- 3rd Edition: March 2020 – … (16,800+)

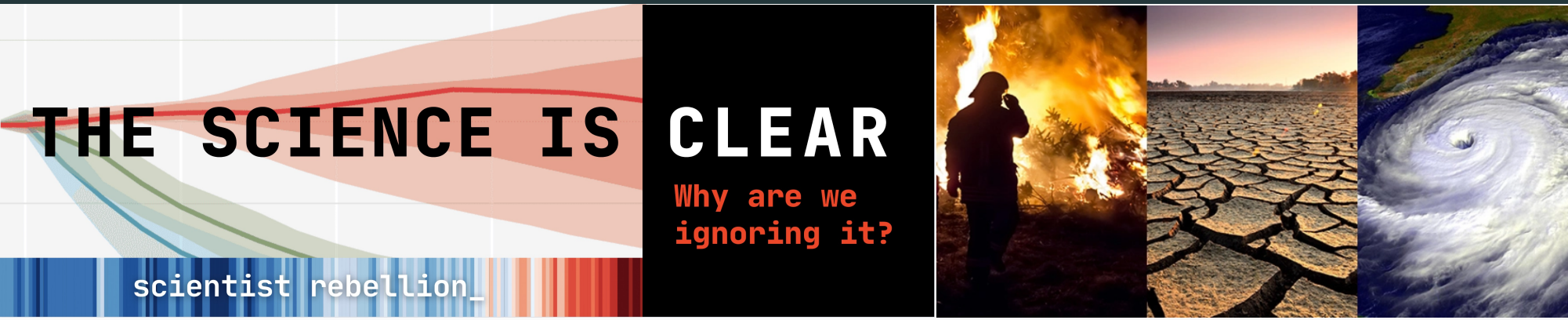
MOOC "Advanced RR" planned for May 2024

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

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 ?

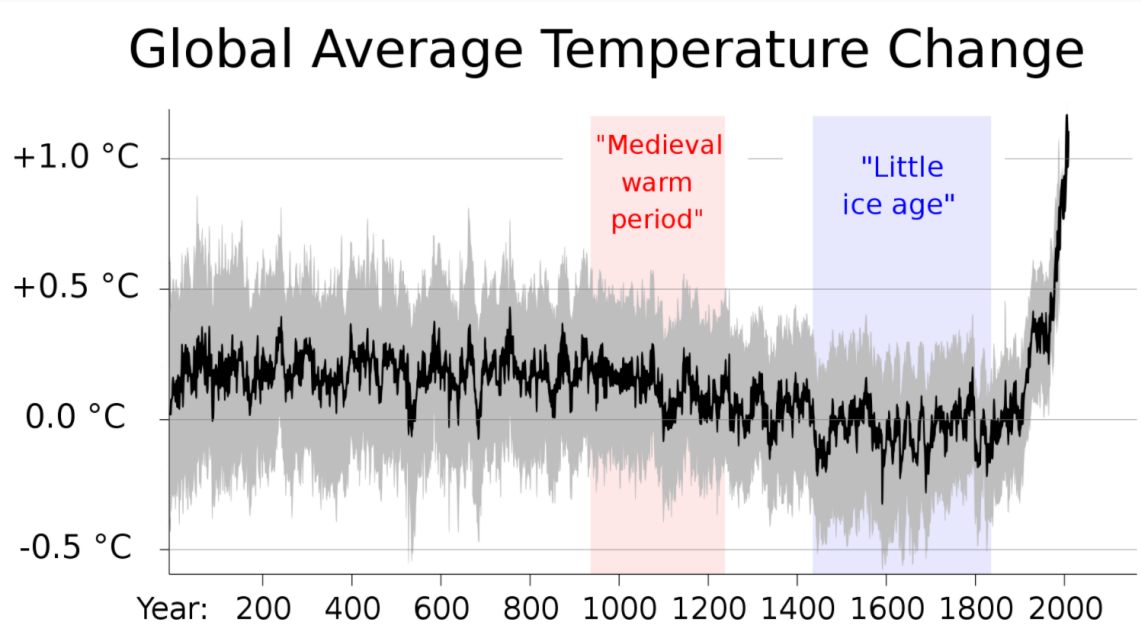


The Elephant in the Room: Climate Change

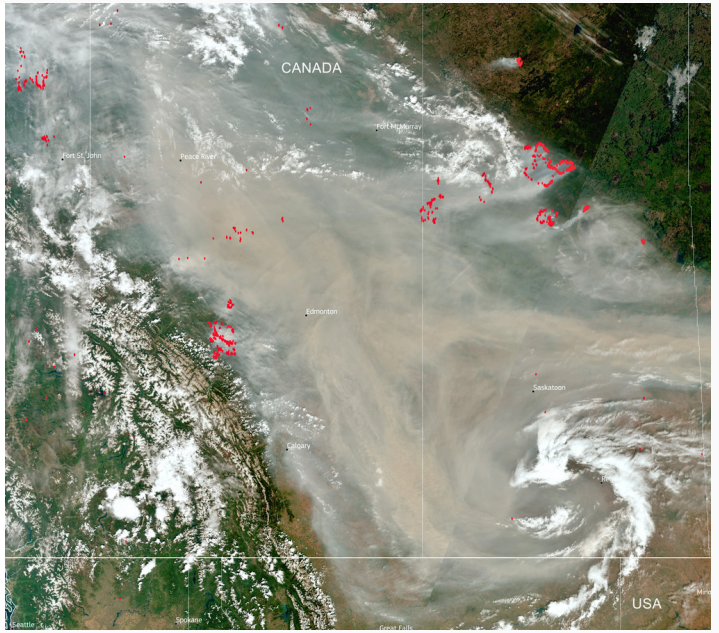


IPCC, IPBES, https://climate.nasa.gov/

1. Global climate change is **not** a future problem

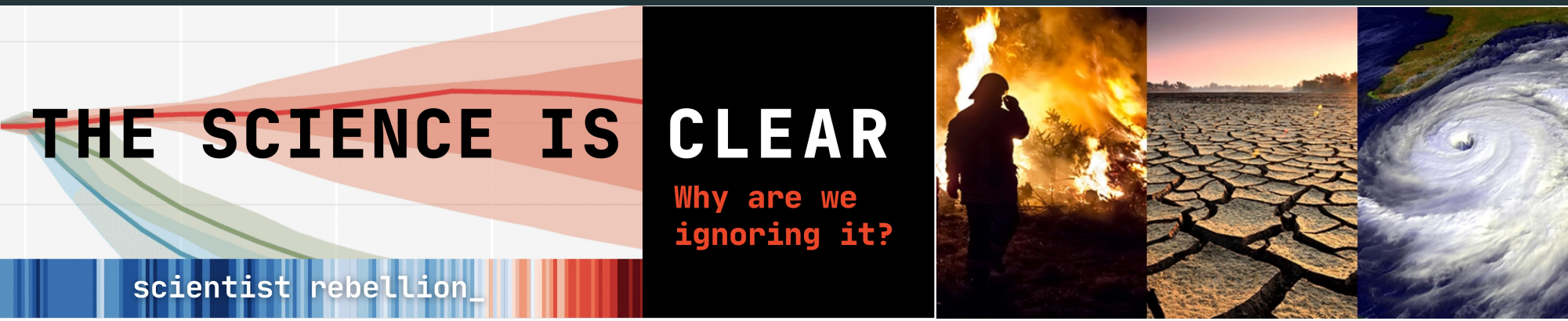


https://en.wikipedia.org/wiki/Global_temperature_record



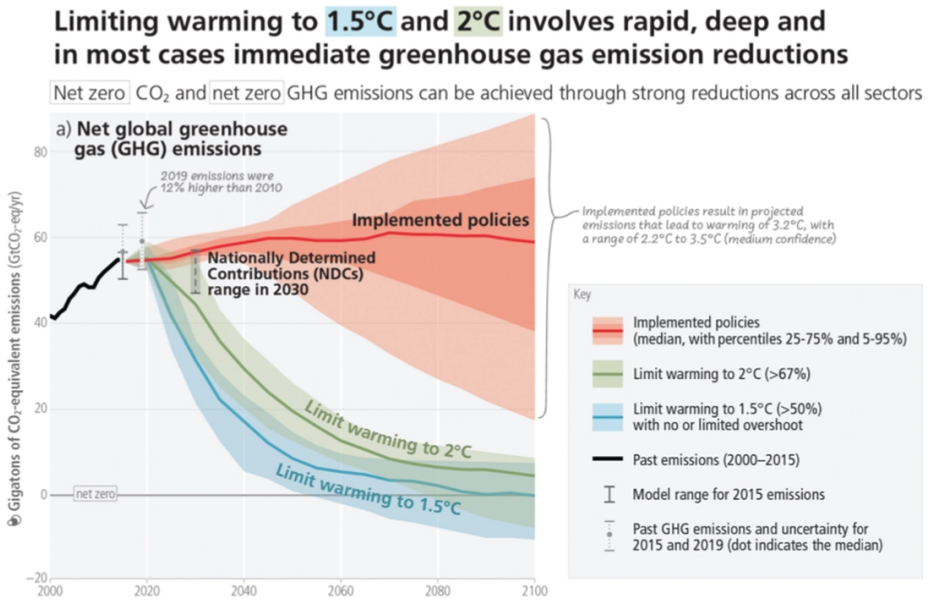
2023 Alberta wildfires (> 1 Mha)

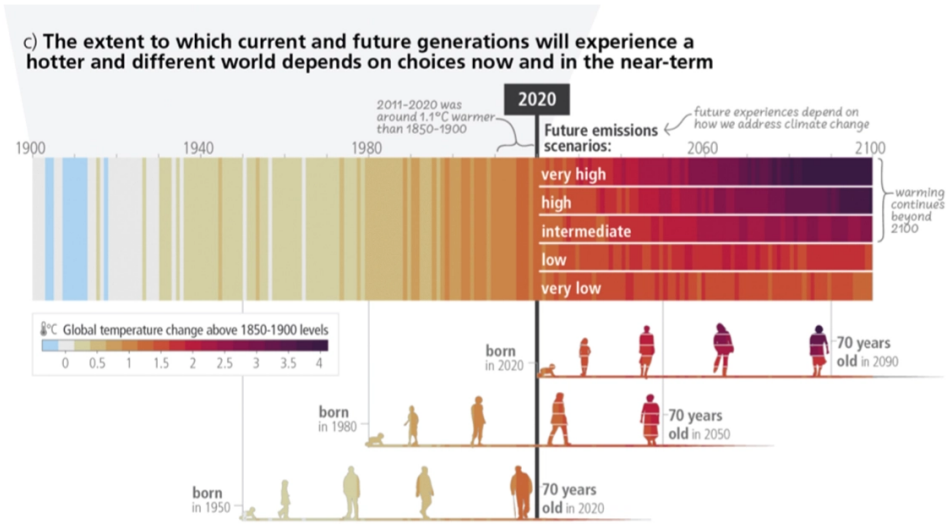
The Elephant in the Room: Climate Change



IPCC, IPBES, https://climate.nasa.gov/

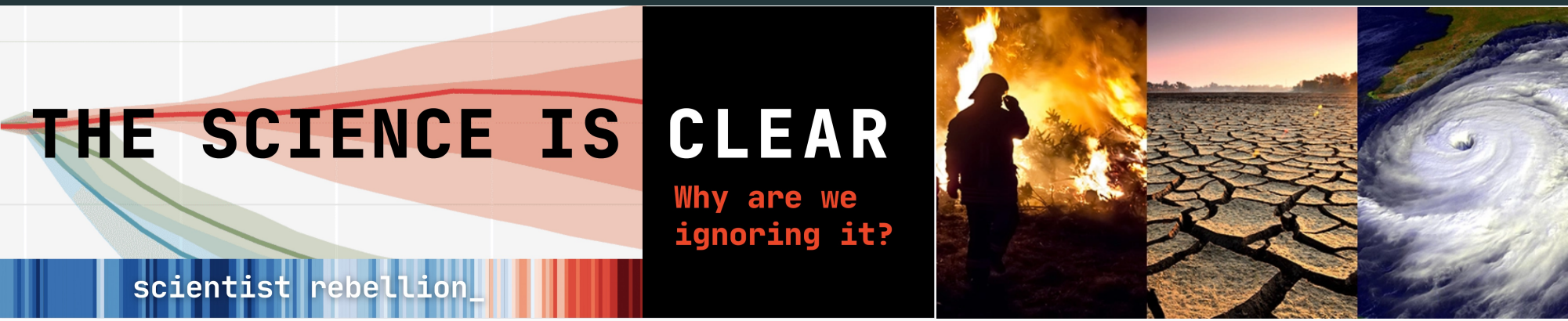
1. Global climate change is **not** a future problem
2. It is **entirely** due to human activity





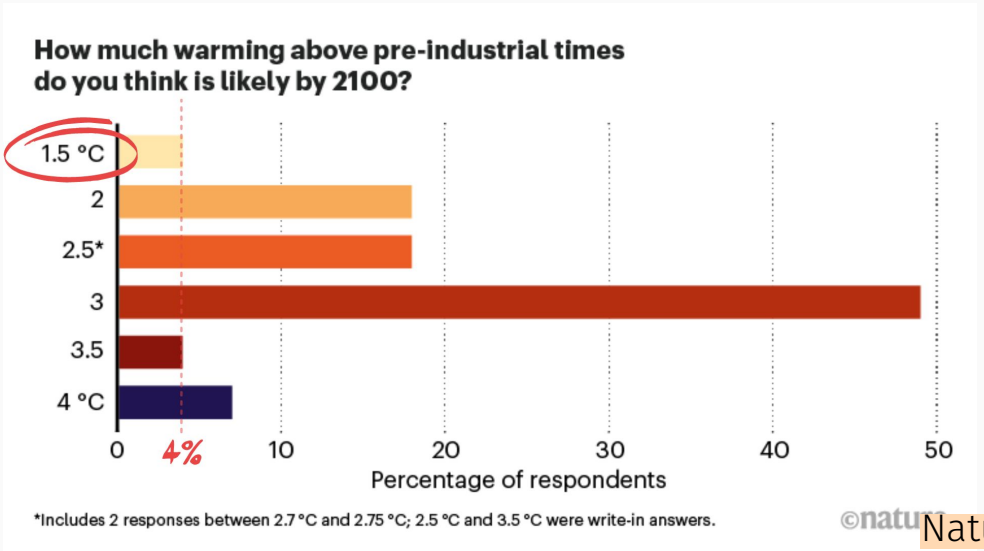
Paris Agreement'15 ∼ Net Zero by 2050

Latest IPCC report



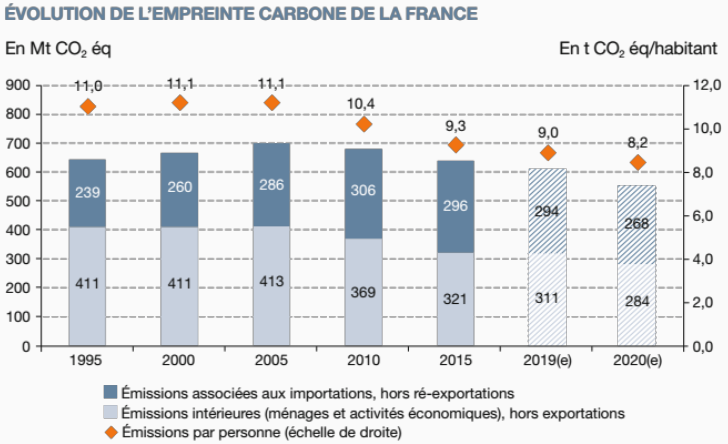
IPCC, IPBES, https://climate.nasa.gov/

1. Global climate change is **not** a future problem
2. It is **entirely** due to human activity
3. *9 out of 10 IPCC scientists believe overshoot is likely*

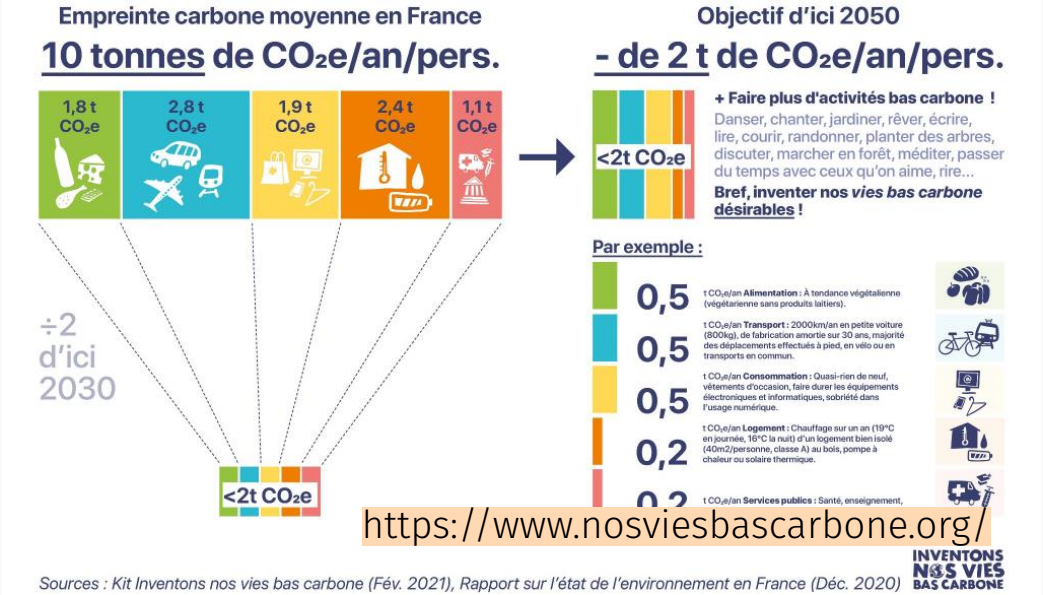


Nature survey, Nov. 2021

Put aside biodiversity loss, pollution, freshwater, land system change...

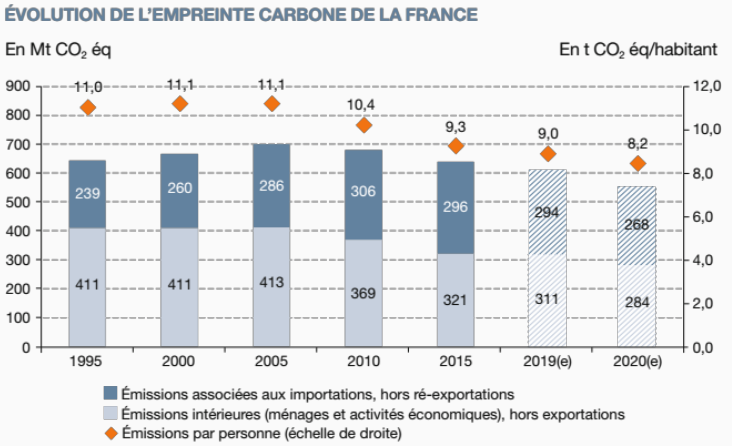


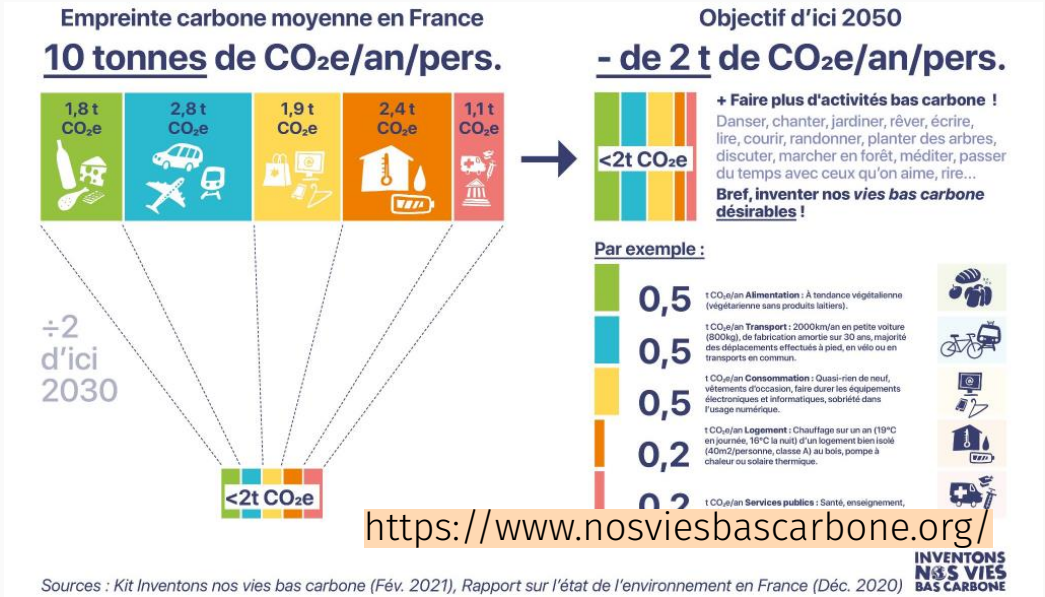
(e) = estimations.
Note : l'empreinte carbone porte sur les trois principaux gaz à effet de serre (CO_2, CH_4, N_2O). En 2021, la méthodologie a été ajustée afin de mieux tenir compte de l'évolution des coûts du pétrole brut, du gaz et du charbon ; l'ensemble de la série a ainsi été révisée, l'essentiel des ajustements portant sur les émissions importées de CH_4.
Champ : périmètre Kyoto (métropole et outre-mer appartenant à l'UE).
***Sources :** Citepa ; AIE ; FAO ; Douanes ; Eurostat ; Insee. Traitement : SDES, 2021*



Sources : Kit Inventons nos vies bas carbone (Fév. 2021), Rapport sur l'état de l'environnement en France (Déc. 2020)

Put aside biodiversity loss, pollution, freshwater, land system change...

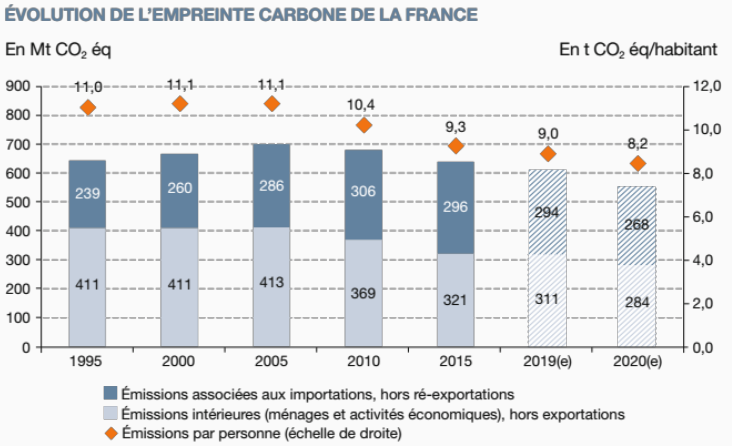


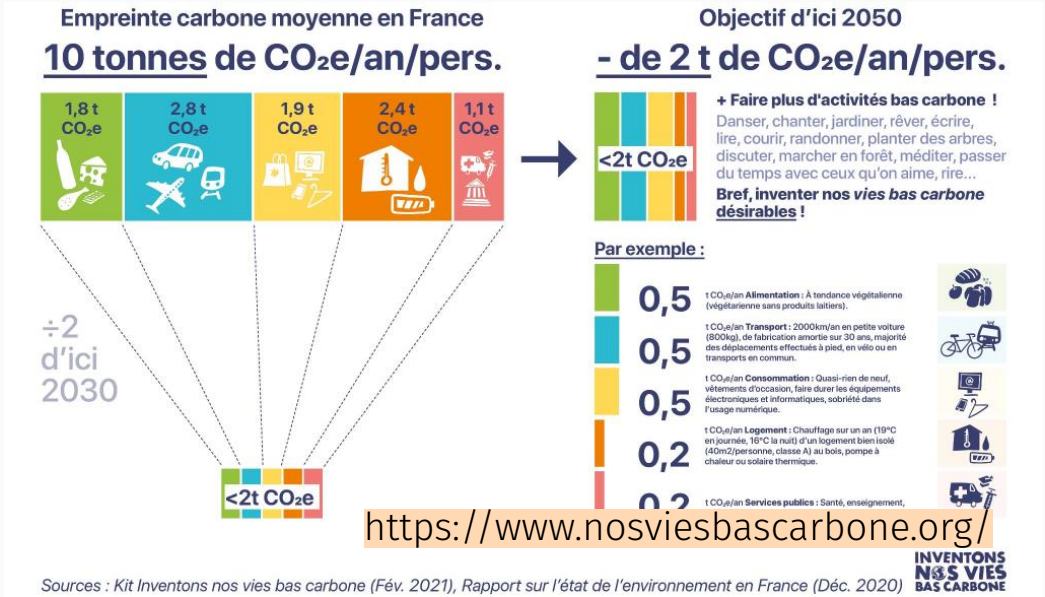


French government response

- *Verdissement de l'industrie: « pause » sur les normes environnementales*
- *Loi de programmation militaire (+41%)*
- *Nous devons préparer la France à une élévation de la température de 4 °C*
- Academia ? PEPR 5G, Cloud, NUMPEX, Quantique, IA, Agroécologie et numérique

Put aside biodiversity loss, pollution, freshwater, land system change...





French government response

- *Verdissement de l'industrie: « pause » sur les normes environnementales*
- *Loi de programmation militaire (+41%)*
- *Nous devons préparer la France à une élévation de la température de 4 °C*
- Academia ? PEPR 5G, Cloud, NUMPEX, Quantique, IA, Agroécologie et numérique

~~Several~~ scenarios on the table

- What will research/CS look like/be used for in such a world?
- Energy optimization/saving ≠ sobriety and frugality