Reproductibilité des expérimentations (REP)

Mathieu Acher @acherm


Université de Rennes
INSA

institut universitaire de France

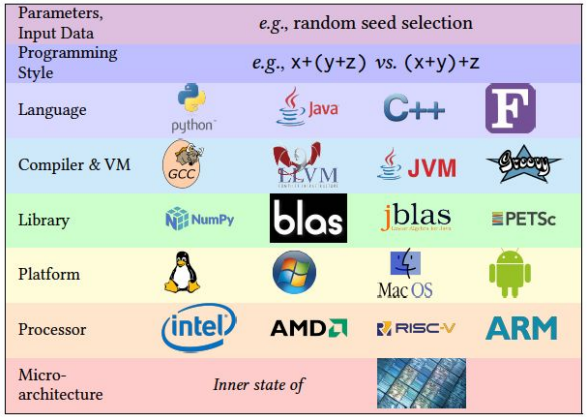

Reproductibilité des expérimentations (REP)

(TP1) À quelle fréquence x+(y+z) == (x+y)+z ? On commencera par répondre à cette question en utilisant différents langages de programmation, librairies, machines, OS, compilateurs, ou LLMs… Pour s'apercevoir que reproduire une expérience et obtenir des résultats robustes/consistants, ce n'est pas si simple, tant d'un point de vue technique que méthodologique.

```python
from random import random, seed
def associativity_test() -> bool:
    x = random(); y = random(); z = random()
    return x + (y + z) == (x + y) + z

def proportion(number: int) -> int:
    ok = 0
    for i in range(number):
        ok += associativity_test()
    return ok * 100 // number

seed(int(input('Seed: ')))
print(str(proportion(1000)) + "%")
```



Maitrise d'outils et langages (package manager, git, Docker/Nix, Notebook, etc.) pour permettre la reproductibilité (utile/nécessaire en sciences et en développement logiciel)

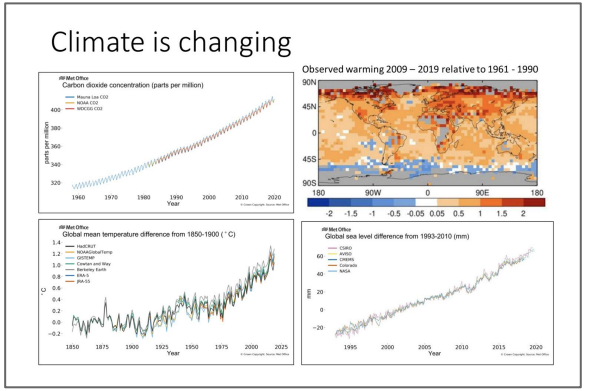
Revisite d'articles scientifiques en reproduisant des résultats, puis en faisant varier quelques aspects de l'expérience… et ainsi évaluer la robustesse/généralité des résultats originaux.

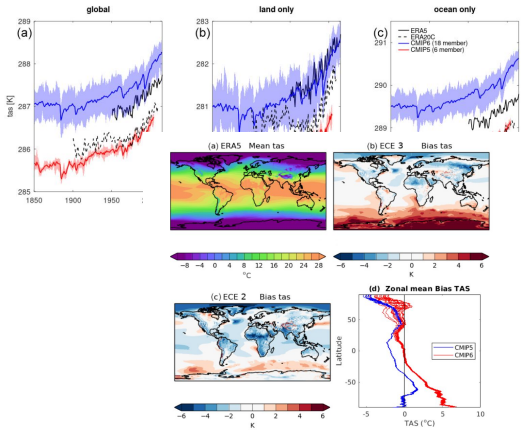
Reproductibilité des expérimentations (REP)

Travaux pratiques pour maîtriser de nombreux outils/langages

Revisite d'articles scientifiques avec REP

- Automatisation
- Polyglotte
- Maîtrise de la pile logicielle
- Qualité
- Produire ses propres outils
- Abstraction
- Gestion de projets
- Méthodes scientifiques
- Valeurs et éthiques





INSA

Evaluation

50% TP (rendu ~début novembre)

50% reproduction et réplication d’article (rendu ~mi-décembre)

REP aka
Reproducible Science and Software Engineering

Abstract: One of the main promises of software is that a result obtained from an experiment (e.g. a simulation) can be reproduced with a high degree of concordance. The quest for reproducibility has an impact on different scientific fields, takes different forms and requires making all the data and code available so that calculations can be run again with identical results. In this course, we will first review terminologies (e.g. reproducible vs replication vs repeatability), basic tools (e.g. versioning systems, build systems, package managers, notebooks), and techniques (e.g. automated testing, continuous integration and deployment, configuration management) related to reproducible science.

We will then program a relatively simple feature in different variants and demonstrate that many factors (including programming languages, library versions, compilers, variable types, randomness management, etc.) can have an impact on the final result. Through this exercise, we aim to present and discuss software techniques, methodologies and tools that developers or scientists can use to address and mitigate reproducibility issues, hopefully leading to more robust and general results.

AGENDA

Interactive course: Reproducible Science and (Deep) Software (Variability)

Is (x+y)+z == x+(y+z)? (TP1)

SOFTWARE VARIANTS ARE EATING THE WORLD

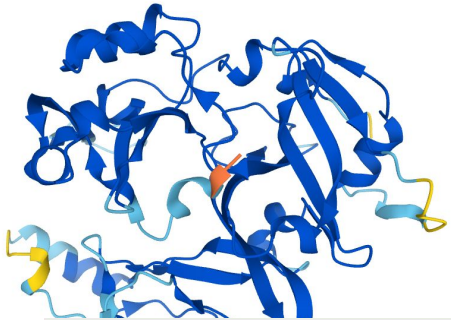
OBS Open Broadcaster Software

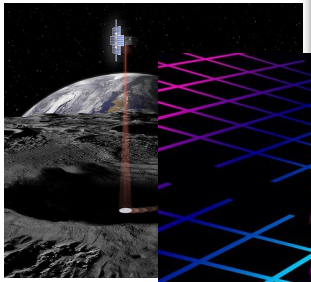
NETFLIX twitch zoom

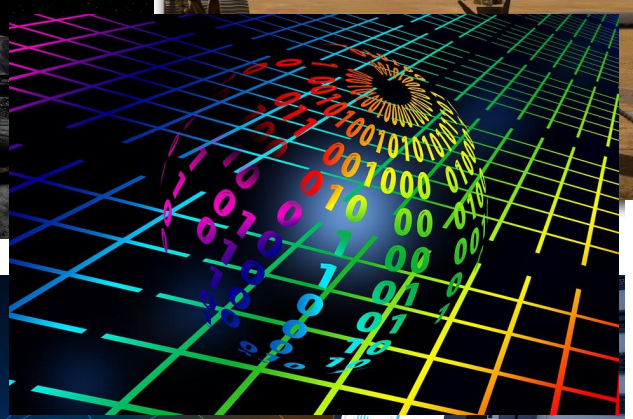
android

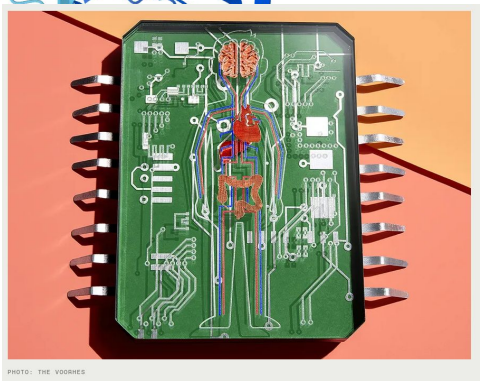
Lichess 2021

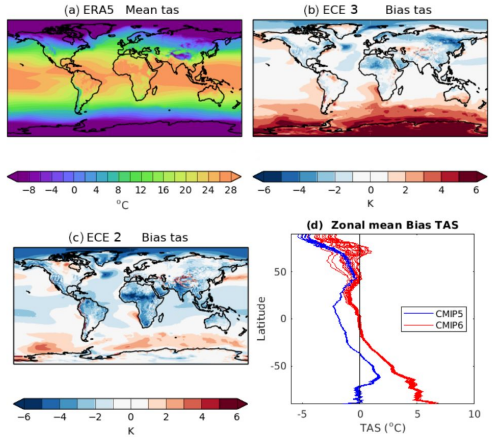
Science is changing: Computation-based research











Computational science depends on software and its engineering

design of mathematical model

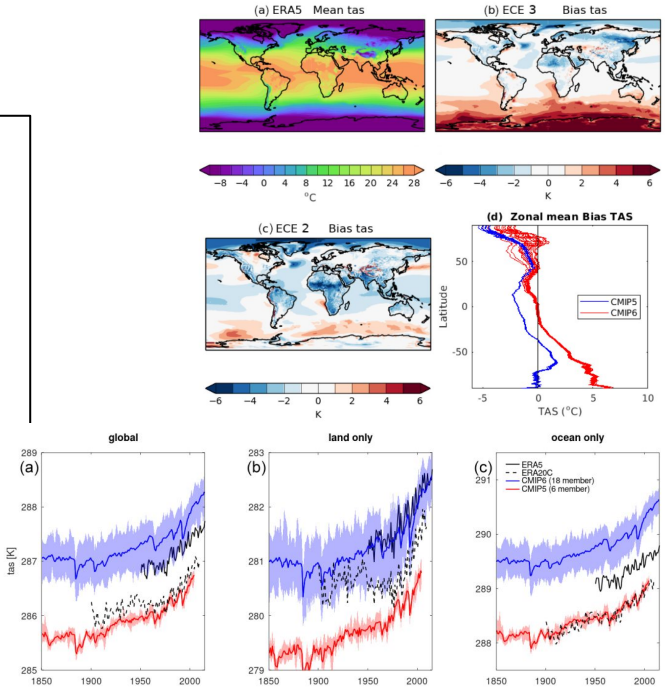
mining and analysis of data

executions of large simulations

problem solving

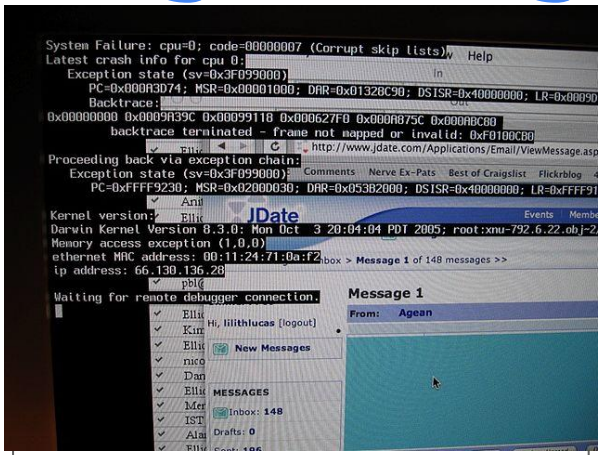
executable paper

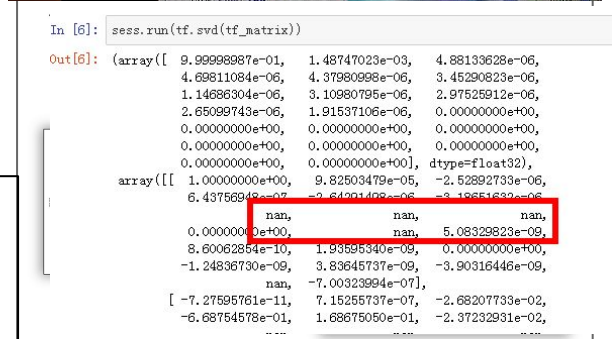
from a set of scripts to automate the deployment to... a comprehensive system containing several features that help researchers exploring various hypotheses



Computational science depends on software and its engineering

multi-million line of code base
multi-dependencies
multi-systems
multi-layer
multi-version
multi-person
multi-variant



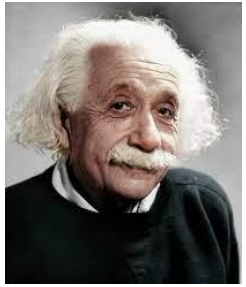


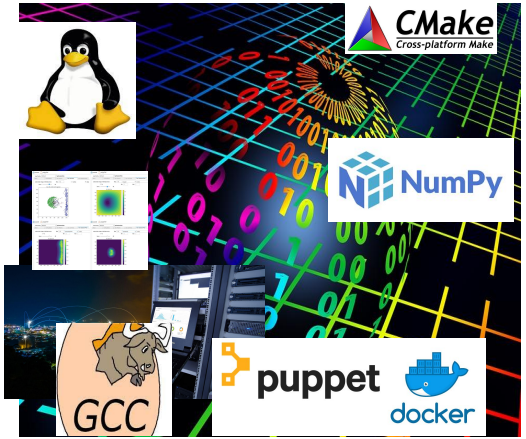
Dealing with software collapse: software stops working eventually
Konrad Hinsen 2019

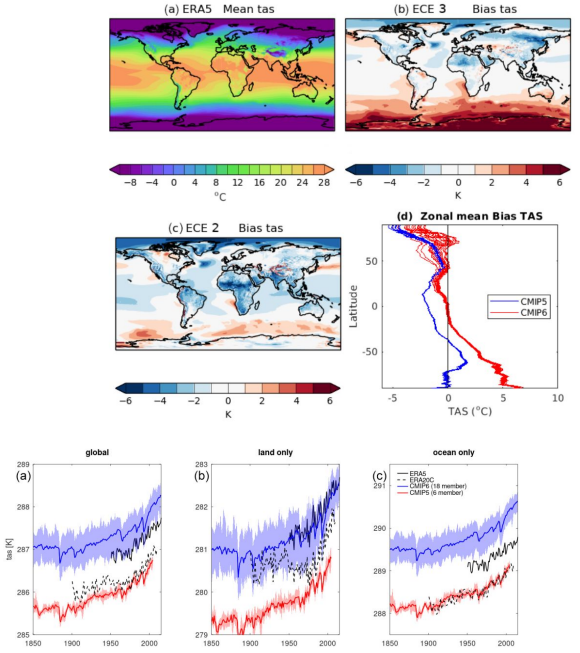
Configuration failures represent one of the most common types of software failures Sayagh et al. TSE 2018

“Insanity is doing the same thing over and over again and expecting different results”

http://throwgrammarfromthetrain.blogspot.com/2010/10/definition-of-insanity.html



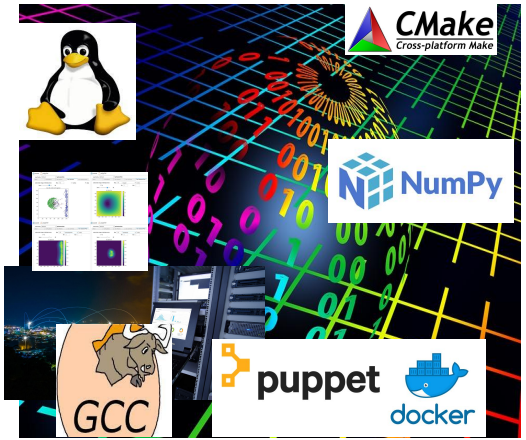


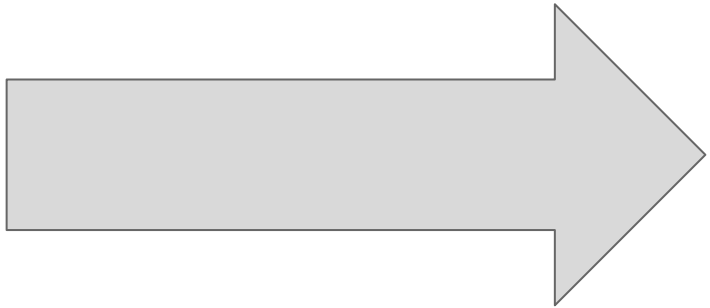


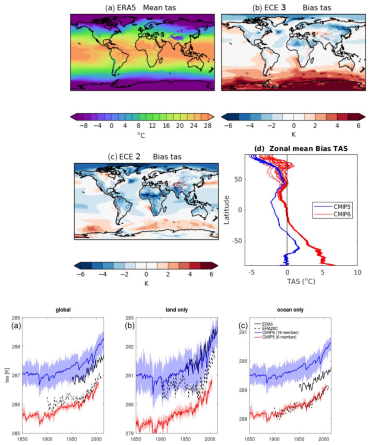
Reproducibility

"Authors provide all the necessary data and the computer codes to run the analysis again, re-creating the results."
(Claerbout/Donoho/Peng definition)

"The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures." (~executable paper)





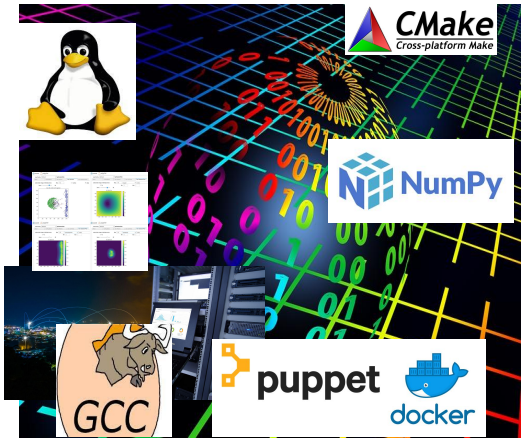


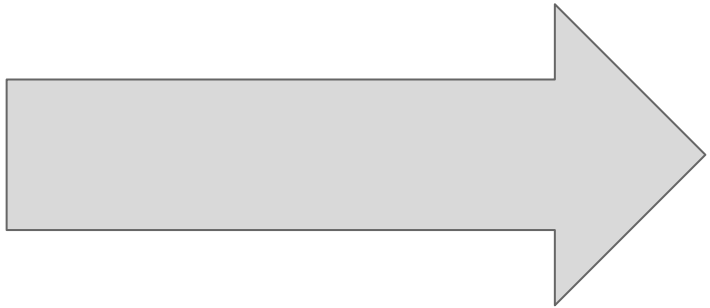
Reproducibility and Replicability

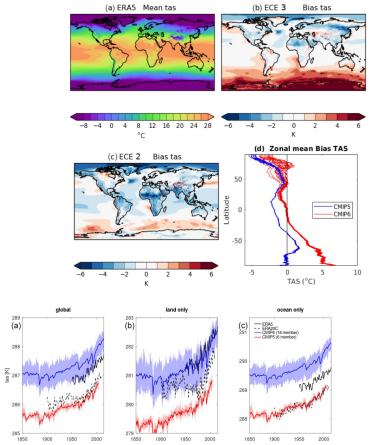
Reproducible: Authors provide all the necessary data and the computer codes to run the analysis again, re-creating the results.

Replication: A study that arrives at the same scientific findings as another study, collecting new data (possibly with different methods) and completing new analyses.

"Terminologies for Reproducible Research", Lorena A. Barba, 2018



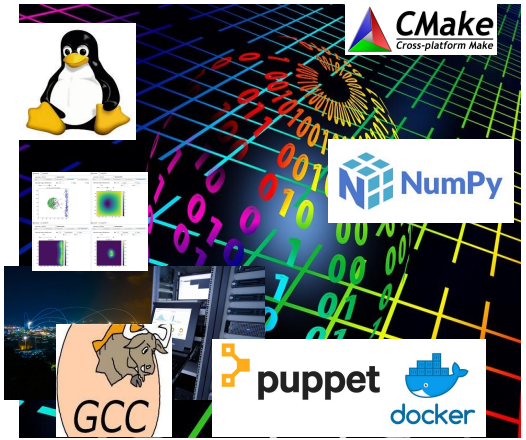




Reproducibility and Replicability

Reproducible: Authors provide all the necessary data and the computer codes to run the analysis again, re-creating the results.
Replication: A study that arrives at the same scientific findings as another study, collecting new data (possibly with different methods) and completing new analyses.

"Terminologies for Reproducible Research", Lorena A. Barba, 2018



The Claerbout/Donoho/Peng terminology is broadly disseminated across disciplines (see Table 2). But the recent adoption of an opposing terminology by two large professional groups—ACM and FASEB—make standardization awkward. The ACM publicizes its rationale for adoption as based on the International Vocabulary of Metrology, but a close reading of the sources makes this justification tenuous. The source of the FASEB adoption is unclear, but there's a chance that Casadevall and Fang (2010) had an influence there. They, in turn, based their definitions on the emphatic but essentially flawed work of Drummond (2009).

Table 2: Grouping of terminologies, as in Table 1, but by discipline.

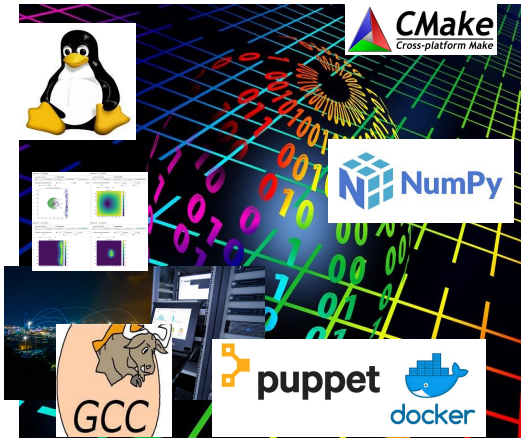
A	*B1*	*B2*
political science economics	signal processing scientific computing econometry epidemiology clinical studies internal medicine physiology (neuro) computational biology biomedical research statistics	microbiology, immunology (FASEB) computer science (ACM)

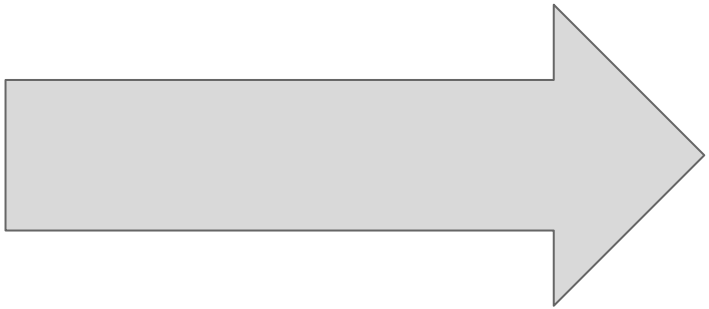
*As a result of discussions with the National Information Standards Organization (NISO), it was recommended that ACM harmonize its terminology and definitions with those used in the broader scientific research community, and ACM agreed with NISO's recommendation to swap the terms "reproducibility" and "replication" with the existing definitions used by ACM as part of its artifac review and badging initiative. ACM took action to update all prior badging to ensure consistency.

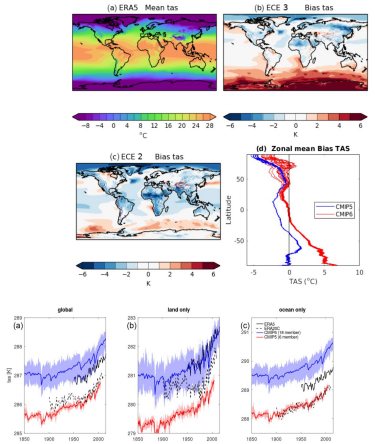
Reproducibility and Replicability

Methods Reproducibility: A method is reproducible if reusing the original code leads to the same results.
Results Reproducibility: A result is reproducible if a reimplementation of the method generates statistically similar values.
Inferential Reproducibility: A finding or a conclusion is reproducible if one can draw it from a different experimental setup.

"Unreproducible Research is Reproducible", Bouthillier et al., ICML 2019



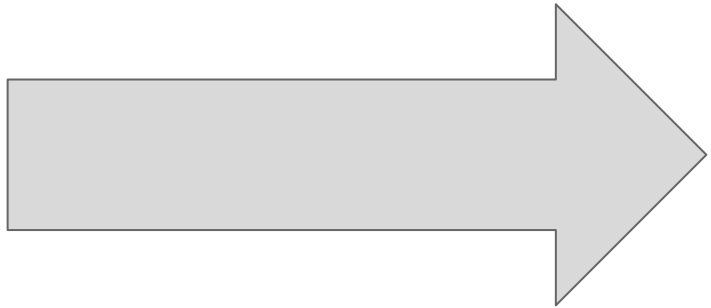


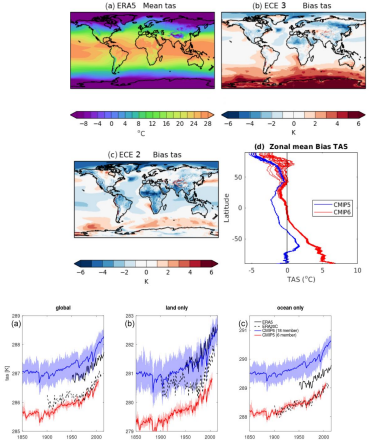


Reproducible science

“Authors provide all the necessary data and the computer codes to run the analysis again, re-creating the results.”

Socio-technical issues: open science, open source software, multi-disciplinary collaboration, incentives/rewards, initiatives, etc.
with many challenges related to data acquisition, knowledge organization/sharing, etc.





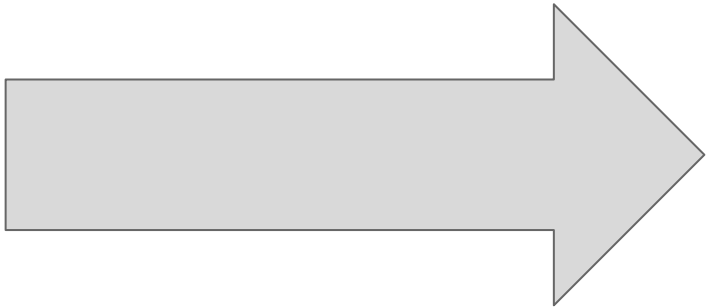
Reproducible Builds

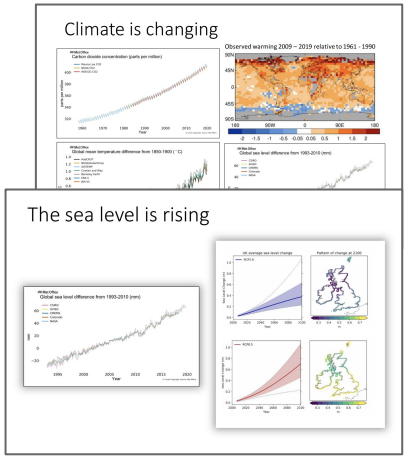
Reproducible builds are a set of software development practices that create an independently-verifiable path from source to binary code. (more)

Reproducible science

“Authors provide all the necessary data and the computer codes to run the analysis again, re-creating the results.”

Socio-technical issues: open science, open source software, multi-disciplinary collaboration, incentives/rewards, initiatives, etc.
with many challenges related to data acquisition, knowledge organization/sharing, etc.

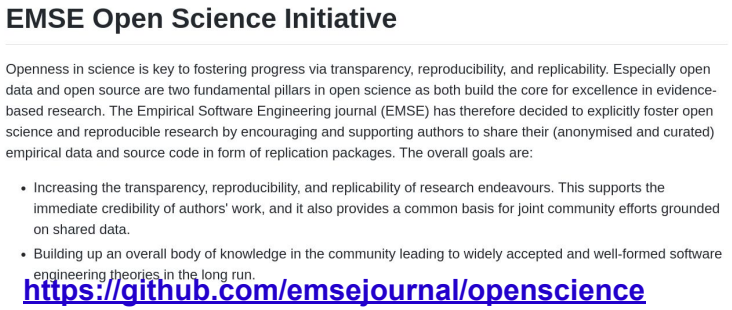




Reproducible science

“Authors provide all the necessary data and the computer codes to run the analysis again, re-creating the results.”

Socio-technical issues: open science, open source software, multi-disciplinary collaboration, incentives/rewards, initiatives, etc.
with many challenges related to data acquisition, knowledge organization/sharing, etc.



https://github.com/emsejournal/openscience

https://rescience.github.io/

https://reproducible-research.inria.fr/

OpenReview.net

Software Heritage

GitLab

GitHub Entreprise

Reproducible science

"Authors provide all the necessary data and the computer codes to run the analysis again, re-creating the results."

Socio-technical issues: open science, open source software, multi-disciplinary collaboration, incentives/rewards, initiatives, etc.
with many challenges related to data acquisition, knowledge organization/sharing, etc.

ARTIFACT EVALUATION

Authors of accepted research papers are invited to submit the artifacts associatec paper for evaluation. To do so, they should submit a PDF via Easychair (select the R Artifacts track). The PDF should contain a stable URL (or DOI) to the artifacts. The UR contain the steps or general instructions to execute/analyze the artifact. Each artif submission will be reviewed by at least two reviewers.

According to ACM's "Result and Artifact Review and Badging" policy, an "artifact" is "a digital object that was either created by the authors to be used as part of the study or generated by the experiment itself [... which can include] software systems, scripts used to run experiments, input datasets, raw data collected in the experiment, or scripts used to analyze results."

CALL FOR CHALLENGE CASES

Home / Call for Papers / Call for challenge cases

ICPE 2022
13th ACM/SPEC International Conference on Performance Engineering

In this track, an industrial performance dataset will be provided. The participants are invited to come up with research questions about the dataset, and study those. The challenge is open-ended: participants can choose the research questions that they find most interesting. The proposed approaches and/or tools and their findings are discussed in short papers, and presented in the main conference.

Why should you make your software (work) reproducible and replicable?

Why not: hard to trust your results; hard to innovate on top of your work; impossible to refute hypothesis or verify new theories

Easier to redo computation when eg inputs change

Easier to inspect and document

Easier to evolve and maintain

Openness and transparency boost innovation and collaboration

Security and trust (release)

It can be seen as engineering high-quality software

It is an important effort, it’s very hard! But worth, not to say mandatory…

Science is about reproducibility

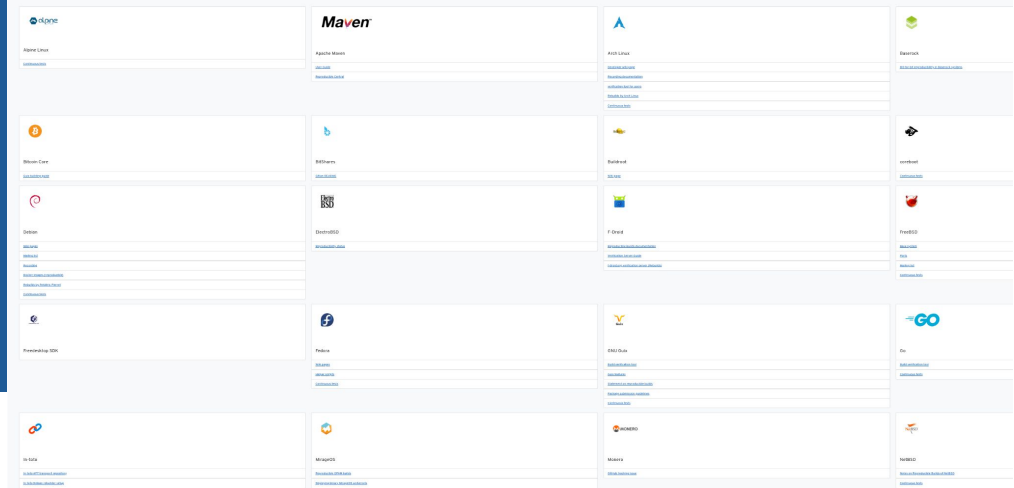
https://x.com/acherm/status/1808019516070261086

"Science is about reproducibility. I can have the most brilliant, crazy, fun idea ever and if I perform an experiment and no one else can duplicate that experiment it belongs in a trash heap [...] That's how science works" with Neil deGrasse Tyson's voice

https://www.youtube.com/clip/UgkxxpU_O7czAnUlUbjgiPbK7ItZAwHMLIuY

Reproducible Builds

Reproducible builds are a set of software development practices that create an independently-verifiable path from source to binary code. (more)



Why does it matter?

Whilst anyone may inspect the source code of free and open source software for malicious flaws, most software is distributed pre-compiled with no method to confirm whether they correspond.

This incentivises attacks on developers who release software, not only via traditional exploitation, but also in the forms of political influence, blackmail or even threats of violence.

This is particularly a concern for developers collaborating on privacy or security software: attacking these typically result in compromising particularly politically-sensitive targets such as dissidents, journalists and whistleblowers, as well as anyone wishing to communicate securely under a repressive regime.

Whilst individual developers are a natural target, it additionally encourages attacks on build infrastructure as a successful attack would provide access to a large number of downstream computer systems. By modifying the generated binaries here instead of modifying the upstream source code, illicit changes are essentially invisible to its original authors and users alike.

The motivation behind the **Reproducible Builds** project is therefore to allow verification that no vulnerabilities or backdoors have been introduced during this compilation process. By promising identical results are always generated from a given source, this allows multiple third parties to come to a consensus on a “correct” result, highlighting any deviations as suspect and worthy of scrutiny.

This ability to notice if a developer or build system has been compromised then prevents such threats or attacks occurring in the first place, as any compromise can be quickly detected. As a result, front-liners cannot be threatened/coerced into exploiting or exposing their colleagues.

Several free software projects already, or will soon, provide reproducible builds.

What’s your experience with reproducible software so far?

Lamb and Zacchiroli “Reproducible Builds: Increasing the Integrity of Software Supply Chains” IEEE Software 2022

https://arxiv.org/pdf/2104.06020

(best paper award IEEE Software for year 2022)

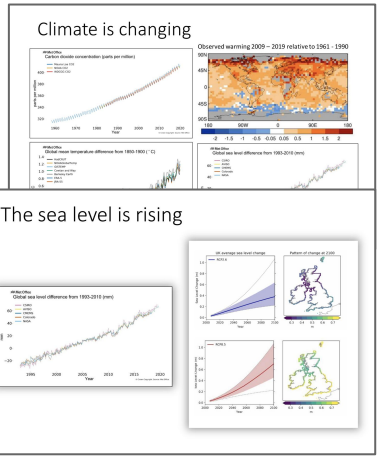
“The build process of a software product is reproducible if, after designating a specific version of its source code and all of its build dependencies, every build produces bit-for-bit identical artifacts, no matter the environment in which the build is performed.”

Reproducible science

“Authors provide all the necessary data and the computer codes to run the analysis again, re-creating the results.”

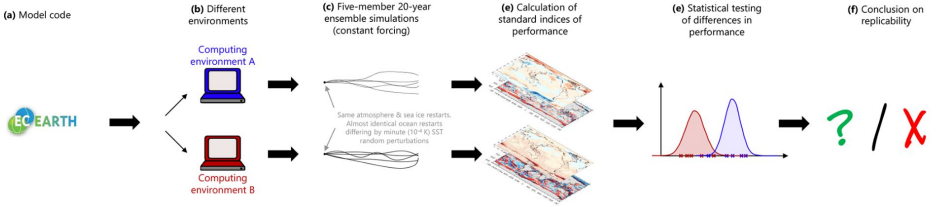
Despite the availability of data and code, several studies report that the same data analyzed with different software can lead to different results.

from a set of scripts to automate the deployment to... a comprehensive system containing several features that help researchers exploring various hypotheses

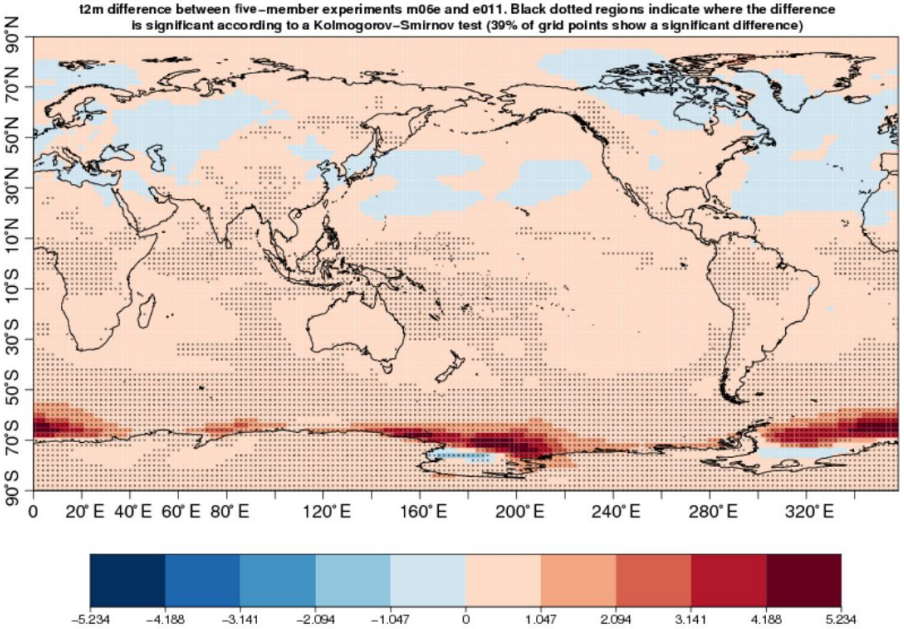


Replicability of the EC-Earth3 Earth system model under a change in computing environment

François Massonnet[1,2], Martin Ménégoz[2,3], Mario Acosta[2], Xavier Yepes-Arbós[2], Eleftheria Exarchou[2], and Francisco J. Doblas-Reyes[2,4]

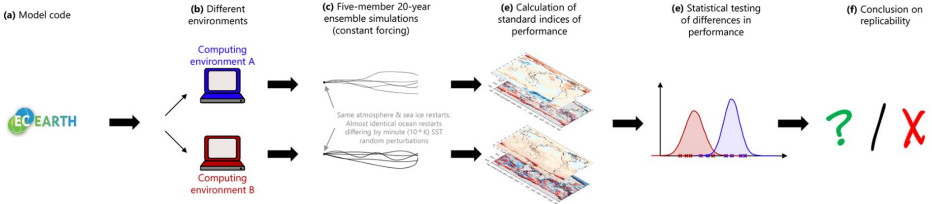


Can a coupled ESM simulation be restarted from a different machine without causing climate-changing modifications in the results? Using two versions of EC-Earth: one **"non-replicable" case** (see below) and one replicable case.



Replicability of the EC-Earth3 Earth system model under a change in computing environment

François Massonnet[1,2], Martin Ménégoz[2,3], Mario Acosta[2], Xavier Yepes-Arbós[2], Eleftheria Exarchou[2], and Francisco J. Doblas-Reyes[2,4]



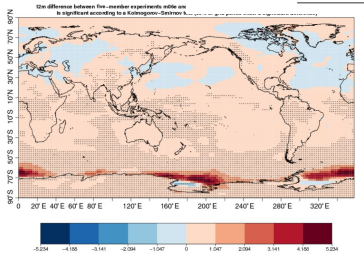
Can a coupled ESM simulation be restarted from a different machine without causing climate-changing modifications in the results? Using two versions of EC-Earth: one "**non-replicable" case** (see below) and one replicable case.

Table 1. The two computing environments considered in this study.

Computing environment	ECMWF-CCA	MareNostrum3
Location	Reading, UK	Barcelona, Spain
Motherboard	Cray XC30 system	IBM dx360 M4
Processor	Dual 12-core E5-2697 v2 (Ivy Bridge) series processors (2.7 GHz), 24 cores per node	2x Intel SandyBridge-EP E5-2670/1600 20M 8-core at 2.6 GHz, 16 cores per node
Operating system	Cray Linux Environment (CLE) 5.2	Linux – SuSe distribution 11 SP2
Compiler	Intel(R) 64 Compiler XE for applications running on Intel(R) 64, version 14.0.1.106 build 20131008	Intel(R) 64 Compiler XE for applications running on Intel(R) 64, version 13.0.1.117 build 20121010
MPI version	Cray mpich2 v6.2.0	Intel MPI v4.1.3.049
LAPACK version	Cray libsci v12.2.0	Intel MKL v11.0.1
SZIP, HDF5, NetCDF4	v2.1, v1.8.11, v4.3.0	v2.1, v1.8.14, v4.2
GribAPI, GribEX	v1.13.0, v000395	v1.14.0, v000370

Table 2. The four experiments considered in this study.

Experiment ID	e011	m06e	a0gi	a0go
Computing environment	ECMWF-CCA	MareNostrum3	ECMWF-CCA	MareNostrum3
EC-Earth version	3.1	3.1	3.2	3.2
Processors (IFS+NEMO+OASIS)	598 (480 + 96 + 22)	512 (384 + 96 + 22)	432 (288 + 144) (OASIS: library)	416 (288 + 128) (OASIS:library)
F flags	-O2 -g -traceback -vec-report0 -r8	-O2 -g -traceback -vec-report0 -r8	-O2 -g -traceback -r8 -fp-model strict -fp-model strict -xHost	-O2 -fp-model precise -xHost -g -traceback -g -traceback -r8
C flags	-O2 -g -traceback	-O2 -g -traceback	-O2 -g -traceback -fp model strict -xHost	-O2 -fp-model precise -xHost -g -traceback
LD flags	-O2 -g -traceback	-O2 -g -traceback	-O2 -g -traceback -fp-model strict -xHost	-O2 -fp-model precise -xHost -g -traceback
Output size	141.8 GB	141.6 GB	101.3 GB	101.3 GB

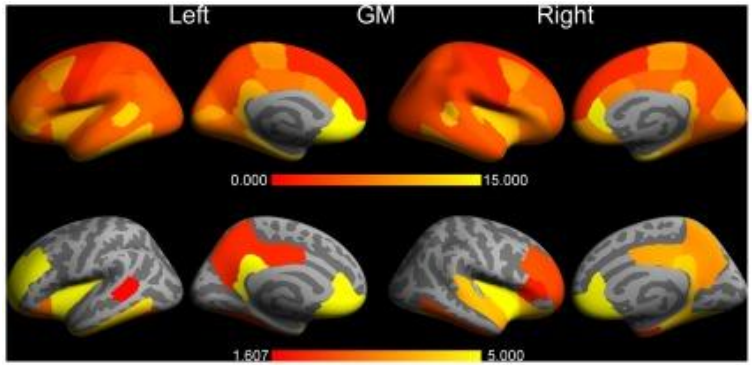


Should software version numbers determine science?

PLoS One. 2012;7(6):e38234. doi: 10.1371/journal.pone.0038234. Epub 2012 Jun 1.

The effects of FreeSurfer version, workstation type, and Macintosh operating system version on anatomical volume and cortical thickness measurements

Ed H B M Gronenschild [1], Petra Habets, Heidi I L Jacobs, Ron Mengelers, Nico Rozendaal, Jim van Os, Machteld Marcelis



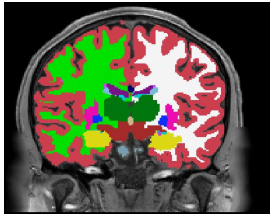
Significant differences were revealed between FreeSurfer version v5.0.0 and the two earlier versions. [...] About a factor two smaller differences were detected between Macintosh and Hewlett-Packard workstations and between OSX 10.5 and OSX 10.6. The observed differences are similar in magnitude as effect sizes reported in accuracy evaluations and neurodegenerative studies.

see also Krefting, D., Scheel, M., Freing, A., Specovius, S., Paul, F., and Brandt, A. (2011). "Reliability of quantitative neuroimage analysis using freesurfer in distributed environments," in *MICCAI Workshop on High-Performance and Distributed Computing for Medical Imaging*. (Toronto, ON).

“Neuroimaging pipelines are known to generate different results depending on the computing platform where they are compiled and executed.”

Reproducibility of neuroimaging analyses across operating systems, Glatard et al., Front. Neuroinform., 24 April 2015

The implementation of mathematical functions manipulating single-precision **floating-point numbers in libmath** has evolved during the last years, leading to numerical differences in computational results. While these differences have little or no impact on simple analysis pipelines such as brain extraction and cortical tissue classification, their **accumulation creates important differences in longer pipelines** such as the subcortical tissue classification, RSfMRI analysis, and cortical thickness extraction.

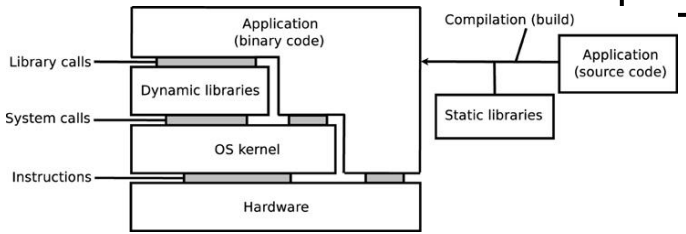


	Cluster A	Cluster B
Applications	Freesurfer 5.3.0, build **1**	Freesurfer 5.3.0, build **1** and **2**
	FSL 5.0.6, build **1**	FSL 5.0.6, build **1** and **2**
	CIVET 1.1.12-UCSF, build **1**	CIVET 1.1.12-UCSF, build **1**
Interpreters	`Python` 2.4.3, `bash` 3.2.25,	`Python` 2.7.5, `bash` 4.2.47,
	`Perl` 5.8.8, `tcsh` 6.14.00	`Perl` 5.18.2, `tcsh` 6.18.01
`glibc` version	2.5	2.18
OS	CentOS 5.10	Fedora 20
Hardware	x86_64 CPUs (Intel Xeon)	x86_64 CPUs (Intel Xeon)

“Neuroimaging pipelines are known to generate different results depending on the computing platform where they are compiled and executed.”

Reproducibility of neuroimaging analyses across operating systems, Glatard et al., Front. Neuroinform., 24 April 2015

Statically building programs improves reproducibility across OSes, but small differences may still remain when dynamic libraries are loaded by static executables[...]. When static builds are not an option, software heterogeneity might be addressed using virtual machines. However, such solutions are only workarounds: differences may still arise between **static executables built on different OSes**, or between **dynamic executables executed in different VMs**.



	Cluster A	Cluster B
Applications	Freesurfer 5.3.0, build **1**	Freesurfer 5.3.0, build **1** and **2**
	FSL 5.0.6, build **1**	FSL 5.0.6, build **1** and **2**
	CIVET 1.1.12-UCSF, build **1**	CIVET 1.1.12-UCSF, build **1**
Interpreters	`Python` 2.4.3, `bash` 3.2.25,	`Python` 2.7.5, `bash` 4.2.47,
	`Perl` 5.8.8, `tcsh` 6.14.00	`Perl` 5.18.2, `tcsh` 6.18.01
`glibc` version	2.5	2.18
OS	CentOS 5.10	Fedora 20
Hardware	x86_64 CPUs (Intel Xeon)	x86_64 CPUs (Intel Xeon)

Can Machine Learning Pipelines Be Better Configured? Wang et al. FSE'2023

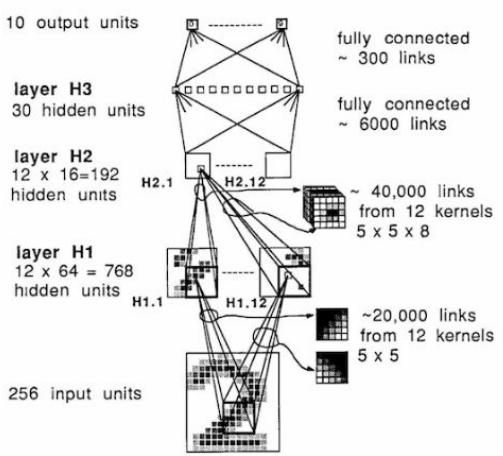
"A pipeline is subject to misconfiguration if it exhibits significantly inconsistent performance upon changes in the versions of its configured libraries or the combination of these libraries. We refer to such performance inconsistency as a pipeline configuration (PLC) issue."

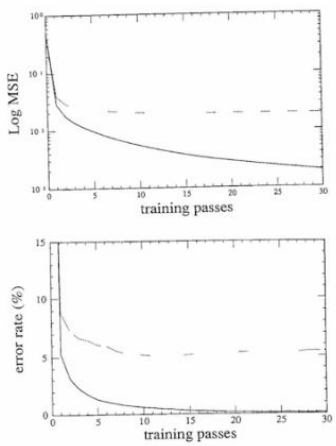
In this paper, we empirically studied 11,363 ML pipelines from diverse competitions on KAGGLE to explore the impacts of different ML library version combinations on their performances. Our study reveals the pervasiveness and severity of *PLC* issues in ML pipelines. Our findings can motivate the establishment of a symbiotic ecosystem where researchers, tool builders, and library vendors work together to assist developers in combating *PLC* issues.

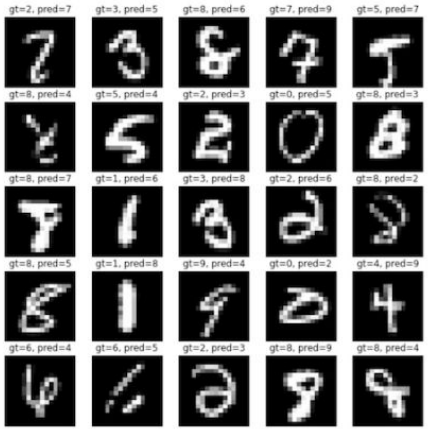
{Keras, Tensorflow} Versions	Score (AUC)	Ranking	Time (ms)	Memory (MB)
{2.7.1, 2.7.0}	Crash	Crash	Crash	Crash
{2.4.3, 2.4.1}	0.768	522	1895.656	1244.446
{2.4.3, 2.3.1}	0.737	521	1882.099	1241.001
{2.4.3, 2.2.0}	0.559	523	1926.980	1248.518
{2.3.1, 2.4.1}	Crash	Crash	Crash	Crash
{2.3.1, 2.3.1}	Crash	Crash	Crash	Crash
{2.3.1, 2.2.0}	0.997	1	1877.330	1199.130
{2.3.1, 2.1.0}	0.997	1	1888.612	1202.602
{2.3.1, 2.0.0}	Crash	Crash	Crash	Crash
{2.3.1, 1.15.2}	0.997	1	1989.861	1194.425
{2.3.1, 1.14.0}	0.997	1	1853.423	1196.269
{2.3.1, 1.13.1}	0.997	1	1901.693	1183.123

Is it possible to reproduce a paper from 1987?

lecun1989-repro







This code tries to reproduce the 1989 Yann LeCun et al. paper: Backpropagation Applied to Handwritten Zip Code Recognition. To my knowledge this is the earliest real-world application of a neural net trained with backpropagation (now 33 years ago).

https://github.com/karpathy/lecun1989-repro

Is it possible to reproduce a paper from 1987?

input data

run

Since we don't have the exact dataset that was used in the paper, we take MNIST and randomly pick examples from it to generate an approximation of the dataset, which contains only 7291 training and 2007 testing digits, only of size 16x16 pixels (standard MNIST is 28x28).

https://github.com/karpathy/lecun1989-repro

Is it possible to reproduce a paper from 1987?

same results? well...

Now we can attempt to reproduce the paper. The original network trained for 3 days, but my (Apple Silicon M1) MacBook Air 33 years later chunks through it in about 90 seconds. (non-emulated arm64 but CPU only, I don't believe PyTorch and Apple M1 are best friends ever just yet, but anyway still about 3000X speedup). So now that we've run prepro we can run repro! (haha):

```
$ python repro.py
```

Running this prints (on the 23rd, final pass):

```
eval: split train. loss 4.073383e-03. error 0.62%. misses: 45
eval: split test . loss 2.838382e-02. error 4.09%. misses: 82
```

This is close but not quite the same as what the paper reports. To match the paper exactly we'd expect the following instead:

```
eval: split train. loss 2.5e-3. error 0.14%. misses: 10
eval: split test . loss 1.8e-2. error 5.00%. misses: 102
```

https://github.com/karpathy/lecun1989-repro

Is it possible to reproduce a paper from 1987?

huge variants/decisions space!

hyperparameter?

algorithm?

implementation?

NN architecture?

Open questions:

- The 12 -> 8 connections from H2 to H1 are not described in this paper... I will assume a sensible block structure connectivity
- Not clear what exactly is the "MSE loss". Was the scaling factor of 1/2 included to simplify the gradient calculation? Will assume no.
- What is the learning rate? I will run a sweep to determine the best one manually.
- Was any learning rate decay used? Not mentioned, I am assuming no.
- Was any weight decay used? not mentioned, assuming no.
- Is there a bug in the pdf where in weight init the fan in should have a square root? The pdf's formatting is a bit messed up. Assuming yes.
- The paper does not say, but what exactly are the targets? Assuming they are +1/-1 for pos/neg, as the output units have tanh too...

One more notes on the weight init conundrum. Eg the "Kaiming init" is:

```
a = gain * sqrt(3 / fan_in)
~U(-a, a)
```

For tanh neurons the recommended gain is 5/3. So therefore we would have `a = sqrt(3) * 5 / 3 *  sqrt(1 / fan_in) = 2.89 * sqrt(1 / fan_in)`, which is close to what the paper does (gain 2.4). So if the original work in fact did use a sqrt and the pdf is just formatted wrong, then the (modern) Kaiming init and the originally used init are pretty close.

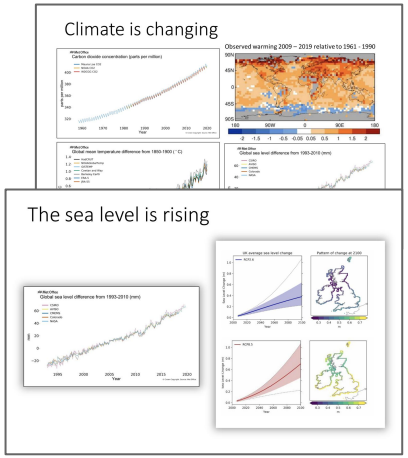
https://github.com/karpathy/lecun1989-repro

Reproducible science as a (deep) software variability problem

“Authors provide all the necessary data and the computer codes to run the analysis again, re-creating the results.”

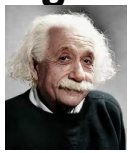
Despite the availability of data and code, several studies report that the same data analyzed with different software can lead to different results.

from a set of scripts to automate the deployment to… a comprehensive system containing several features that help researchers exploring various hypotheses



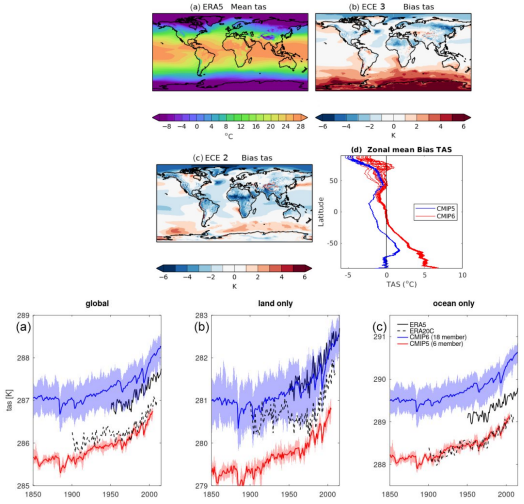
“Insanity is doing the same thing over and over again and expecting different results”

http://throwgrammarfromthetrain.blogspot.com/2010/10/definition-of-insanity.html



Overoptimistic reaction: we are not insane in computational science; we just live with the fact that many variability factors can lead to “different” results.

Reality check: we are not insane in CS; we ~~fix everything and~~ explore a tiny portion and ~~ignore which variability factors have a significant effect on results~~ hope that variability factors won’t refute our findings

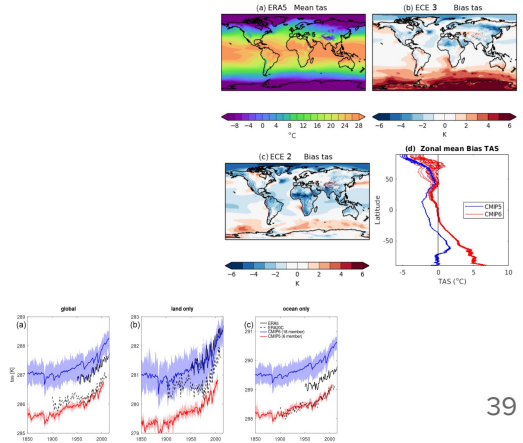


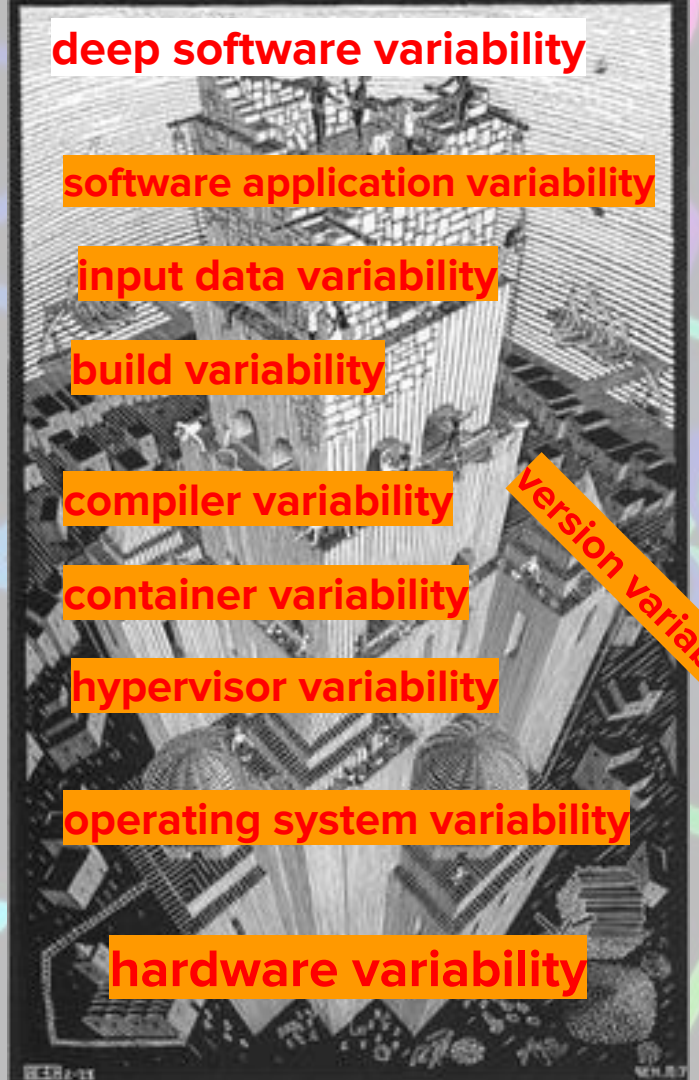
Reproducible science with variability

“Authors provide all the necessary data and the computer codes to run the analysis again, re-creating the results.”

Yet, despite the availability of data and code, several studies report that **unexplored variability in software can lead to varying results** up to the point **discrepancies can radically change the conclusions and contradict established knowledge**

from a set of scripts to automate the deployment to... a comprehensive system containing several features that help researchers exploring various hypotheses





Despite the availability of data and code, several studies report that the same data analyzed with **different software** can lead to **different results**

Many ***layers*** (operating system, third-party libraries, versions, workloads, compile-time options and flags, etc.) themselves subject to variability can alter the results.

Reproducible science and deep software variability: a threat and opportunity for scientific knowledge!

Does deep software variability affect previous scientific, software-based studies? (a graphical template)

- **Hardware**
  - Age
  - # Cores
  - GPU
- **Operating System**
  - Version
  - Option
  - Distrib.
- **Software**
  - Compil.
  - Software Variant
  - Version
- **Input Data**
  - Size
  - Length
  - Res.

List all details...

and questions:

What if we run the experiments on different:

- OS?
- version/commit?
- parameters?
- input?

Revisite d'article scientifique (50%)

- Read a paper (choose 1 out of 3, see next slide)
  - read/study ASAP
- Reproduce
  - reuse code/data
  - understand; re-run
  - conclusions: is it reproducible? do you confirm original results?
- Replicate
  - identify variability factors and threats
  - make a deviation/variation
  - analyse
  - conclude: do you confirm original results?
- Present key results and lessons learned

Choose 1 out of 3

- "Debunking the Chessboard: Confronting GPTs Against Chess Engines to Estimate Elo Ratings and Assess Legal Move Abilities"
  - https://blog.mathieuacher.com/GPTsChessEloRatingLegalMoves/
- "COVID and Home Advantage in Football: An Analysis of Results and xG Data in European Leagues"
  - https://blog.mathieuacher.com/FootballAnalysis-xG-COVIDHome/
- “Ranking Programming Languages by Energy Efficiency” SCP 2021
  - https://github.com/greensoftwarelab/RosettaExamples

=> XXX (link)

Debunking the Chessboard: Confronting GPTs Against Chess Engines to Estimate Elo Ratings and Assess Legal Move Abilities

https://blog.mathieuacher.com/GPTsChessEloRatingLegalMoves/
https://www.youtube.com/watch?v=6D1XIbkm4JE (video with MonsieurPhi)

- what about GPT4o or Claude or Llama?
- prompt sensitivity?
- temperature?
- what about confronting to other chess engines?
- ...

COVID and Home Advantage in Football: An Analysis of Results and xG Data in European Leagues

https://blog.mathieuacher.com/FootballAnalysis-xG-COVIDHome/

- what about 2023/2024?
- xG: instead of understat, another source of data
- other leagues (amateur)
- other sports
- ...

Ranking Programming Languages by Energy Efficiency SCP 2021

https://github.com/greensoftwarelab/RosettaExamples

- other programs/workloads?
- measurements?
- other programming languages?
- other compilers or interpreters?
- container effect?
- other hardware? ...
- ...

Time to practice! TP1