

# Reproducible Science and Deep Software Variability

Mathieu Acher @acherm

Special thanks to Luc Lesoil, Jean-Marc Jézéquel, Arnaud Blouin, and Benoit Combemale

VaMoS keynote, 24 february 2022

<https://hal.inria.fr/hal-03528889>

**Abstract** : Biology, medicine, physics, astrophysics, chemistry: all these scientific domains need to process large amount of data with more and more complex software systems. For achieving reproducible science, there are several challenges ahead involving multidisciplinary collaboration and socio-technical innovation with software at the center of the problem. Despite the availability of data and code, several studies report that the same data analyzed with different software can lead to different results. I am seeing this problem as a manifestation of deep software variability: many factors (operating system, third-party libraries, versions, workloads, compile-time options and flags, etc.) themselves subject to variability can alter the results, up to the point it can dramatically change the conclusions of some scientific studies. In this keynote, I argue that deep software variability is a threat and also an opportunity for reproducible science. I first outline some works about (deep) software variability, reporting on preliminary evidence of complex interactions between variability layers. I then link the ongoing works on variability modelling and deep software variability in the quest for reproducible science.

**References** at the ~end of the slides!

# AGENDA

## **Reproducible Science and (Deep) Software (Variability)**

Deep Software Variability

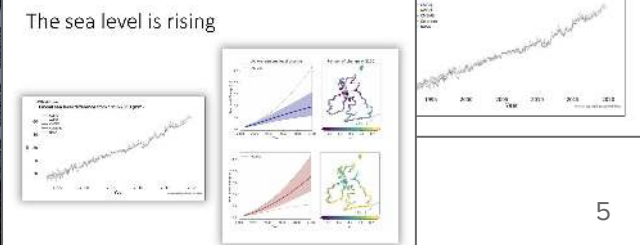
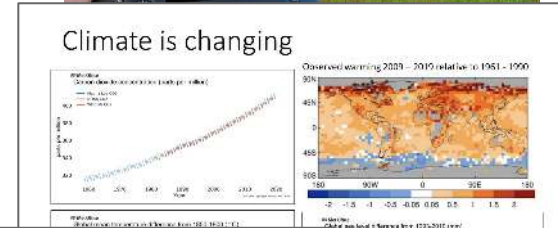
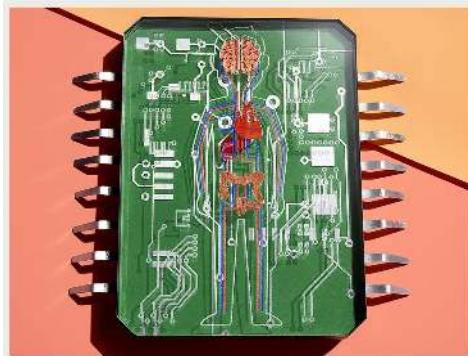
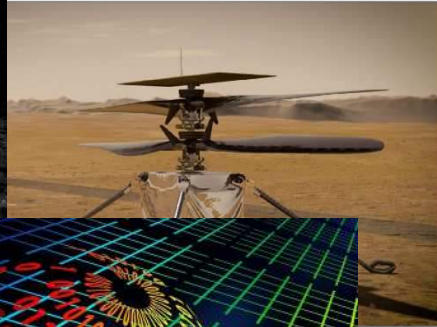
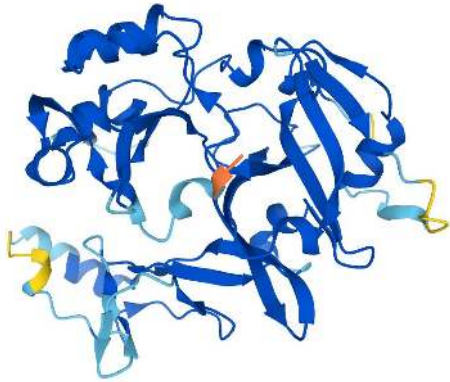
Evidence of Deep Software Variability in Science

Threats and Opportunities

# SOFTWARE VARIANTS ARE EATING THE WORLD



# SOFTWARE IS EATING SCIENCE

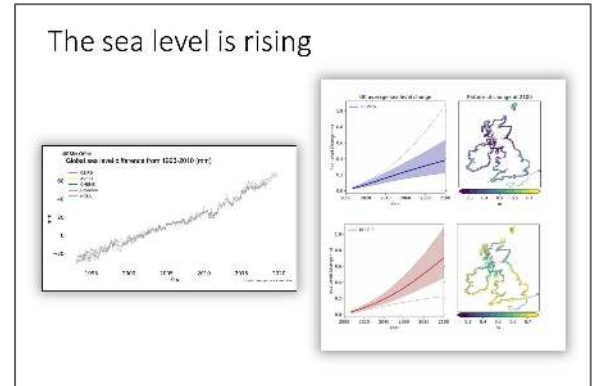
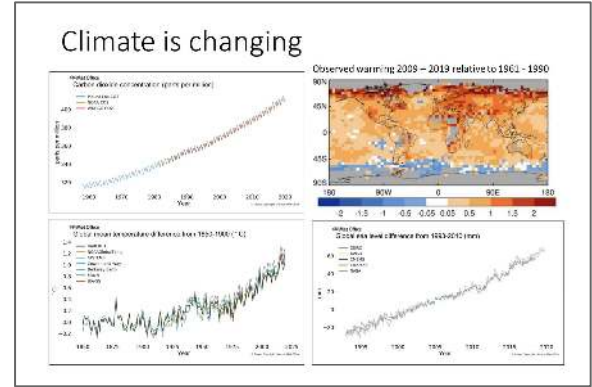


# Computational science depends on software and its engineering



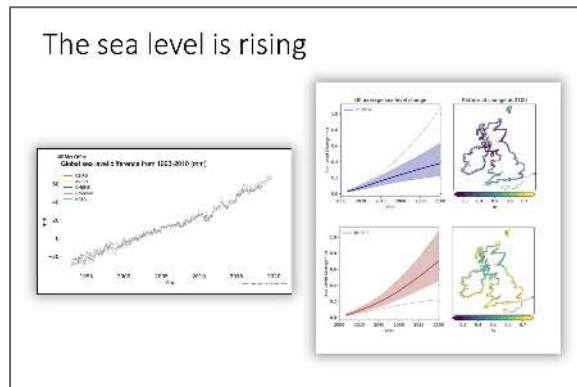
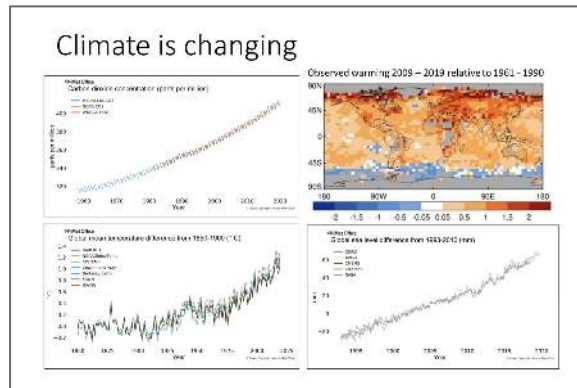
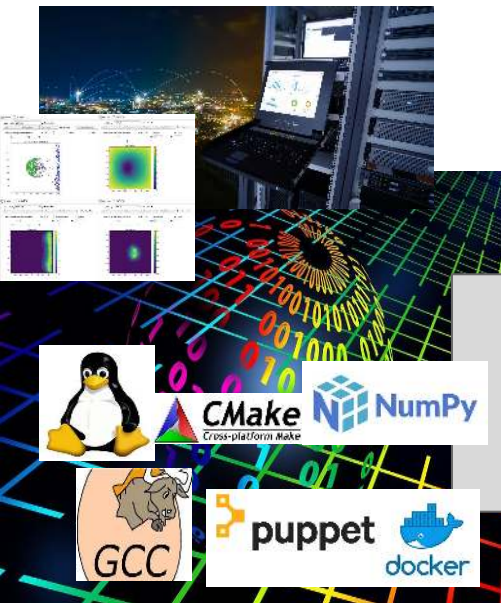
design of mathematical model  
mining and analysis of data  
executions of large simulations  
problem solving  
executable paper

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

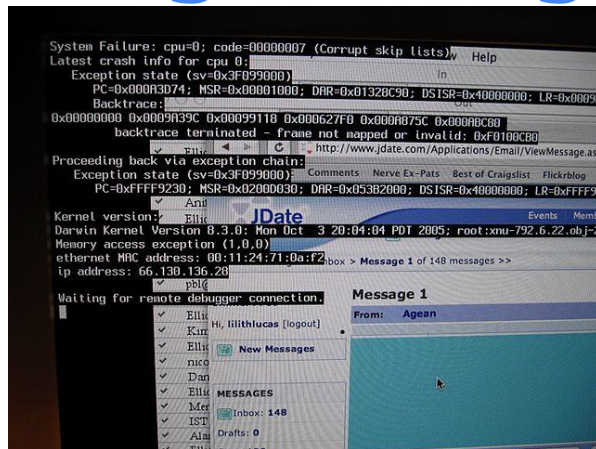
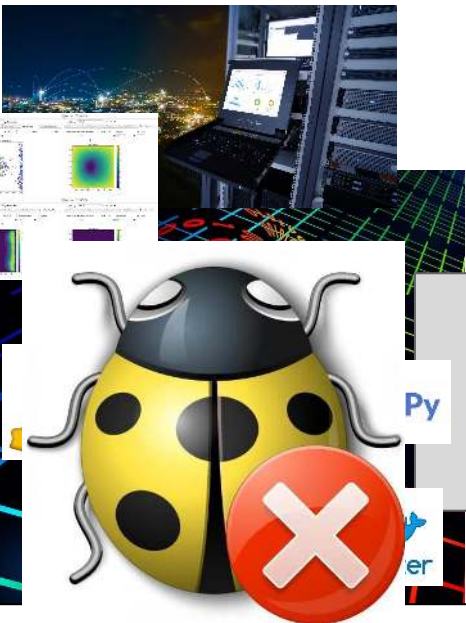
multi-million line of code base  
multi-dependencies  
multi-systems  
multi-layer  
multi-version  
multi-person  
multi-variant



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



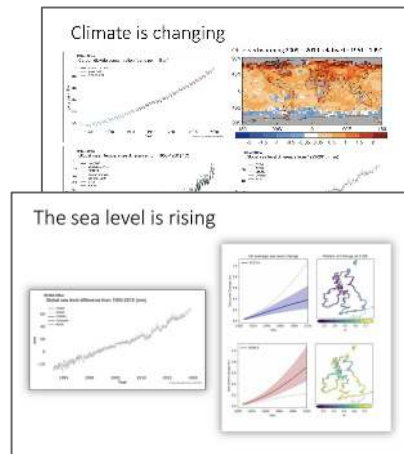
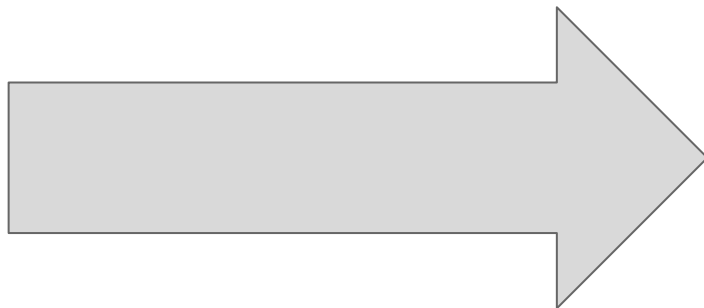
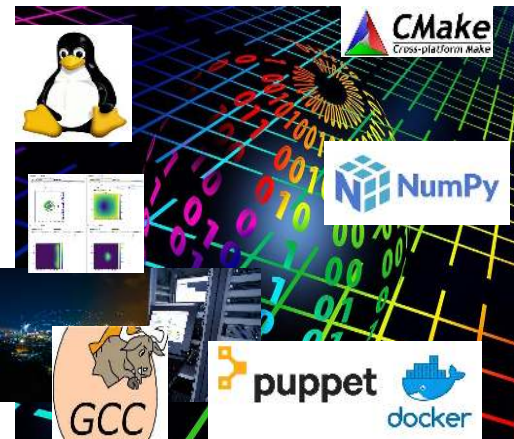
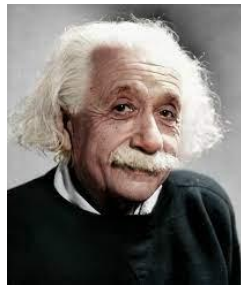
```
In [6]: sess.run(tf.svd(tf_matrix))
Out[6]: (array([[ 9.99999987e-01,  1.48747023e-03,  4.88133628e-06,
  4.69611084e-06,  4.37980998e-06,  3.45290823e-06,
  1.14686304e-06,  3.10980795e-06,  2.97525912e-06,
  2.65099743e-06,  1.91537106e-06,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00], dtype=float32),
array([[ 1.00000000e+00,  9.82503479e-05, -2.52892733e-06,
  6.43756945e-07, -2.61201495e-06, -3.18651830e-06,
  0.00000000e+00,  nan,  nan,  nan,
  8.60062854e-10,  1.93595340e-09,  0.00000000e+00,
 -1.24836730e-09,  3.83645737e-09, -3.90316446e-09,
  nan, -7.00323994e-07],
 [ -7.27595761e-11,  7.15255737e-07, -2.68207733e-02,
 -6.68754578e-01,  1.68675050e-01, -2.37232931e-02,
 ...])
```

Dealing with software collapse: software stops working eventually  
Konrad Hinsin 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>

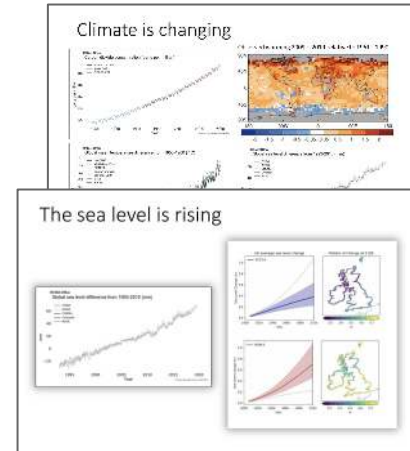
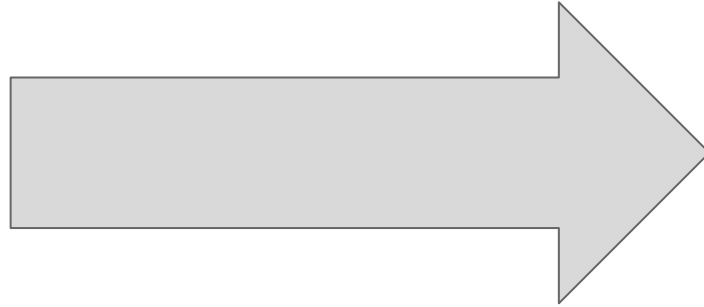
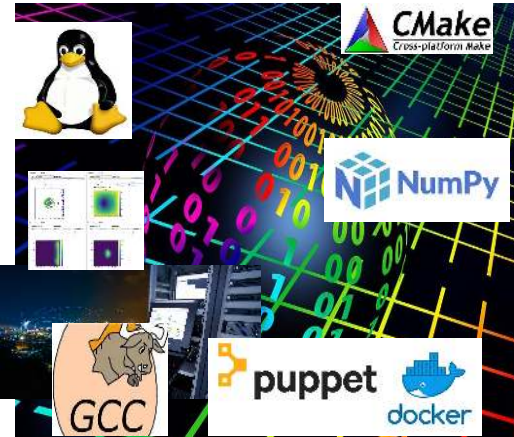


# Reproducibility

“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)

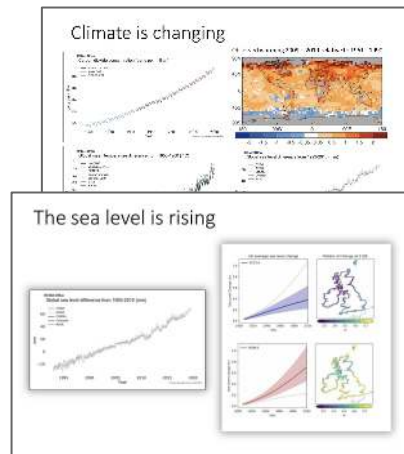
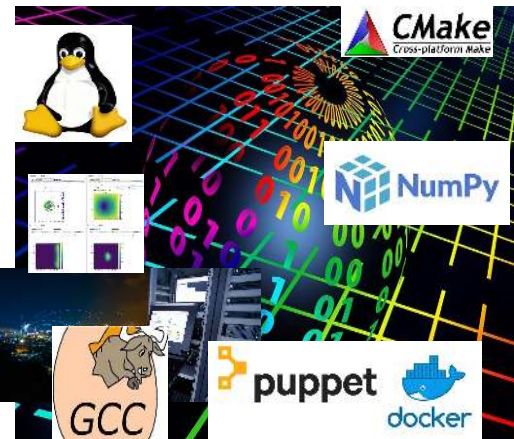


# Reproducibility and Replicability

**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

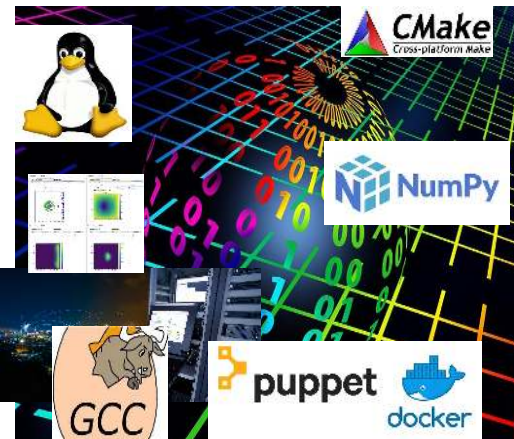


# Reproducibility and Replicability

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

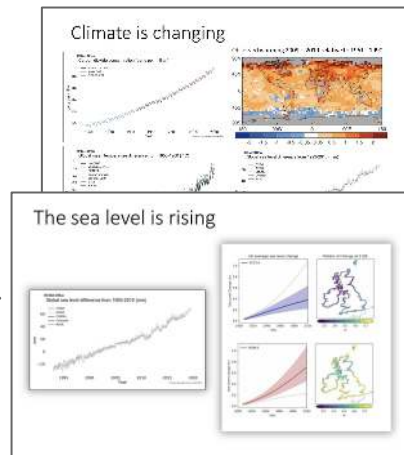
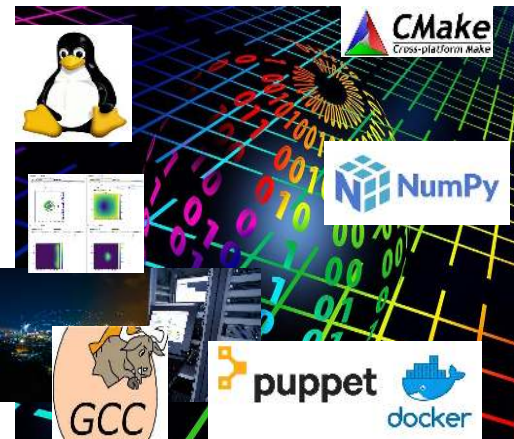
# Reproducibility and Replicability

**Methods Reproducibility:** A method is reproducible if reusing the original code leads to the same results.

**Results Reproducibility:** A result is reproducible if a reimplementaion 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.

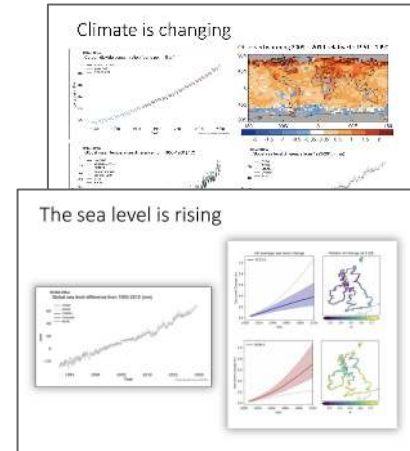
“Unreproducible Research is Reproducible”, Bouthillier et al., ICML 2019



# 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 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.

## EMSE Open Science Initiative

Openness in science is key to fostering progress via transparency, reproducibility, and replicability. Especially open data and open source are two fundamental pillars in open science as both build the core for excellence in evidence-based research. The Empirical Software Engineering journal (EMSE) has therefore decided to explicitly foster open science and reproducible research by encouraging and supporting authors to share their (anonymised and curated) empirical data and source code in form of replication packages. The overall goals are:

- Increasing the transparency, reproducibility, and replicability of research endeavours. This supports the immediate credibility of authors' work, and it also provides a common basis for joint community efforts grounded on shared data.
- Building up an overall body of knowledge in the community leading to widely accepted and well-formed software engineering theories in the long run.

<https://github.com/emsejournal/openscience>

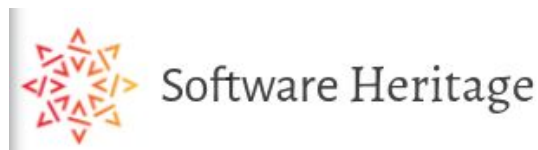
Reproducible Science is good. Replicated Science is better.

ReScience C is a *platinum open-access* peer-reviewed journal that targets computational research and encourages the explicit **replication** of already published research, promoting new and open-source implementations in order to ensure that the original research is **reproducible**. You can read about the ideas behind ReScience C in the article [Sustainable computational science: the ReScience initiative](#)

<https://rescience.github.io/>

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

OpenReview.net



GitHub  
Enterprise



# 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.



## CALL FOR CHALLENGE CASES

[Home / Call for Papers / Call for challenge cases](#)

# ICPE 2022

13th ACM/SPEC International Conference on Performance Engineering

## ARTIFACT EVALUATION

Authors of accepted research papers are invited to submit the artifacts associated with their paper for evaluation. To do so, they should submit a PDF via EasyChair (select the Research Artifacts track). The PDF should contain a stable URL (or DOI) to the artifacts. The URL should contain the steps or general instructions to execute/analyze the artifact. Each artifact 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."

The idea of the challenge track is to provide participants with a set of case studies that tackle relevant problems and challenge the state of the art. The challenge track happens in two phases. In the first phase, there will be a call for cases. Submitted cases will be reviewed by the challenge co-chairs to ensure that the information is clearly described. Accepted cases will be part of the official conference proceedings.

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.



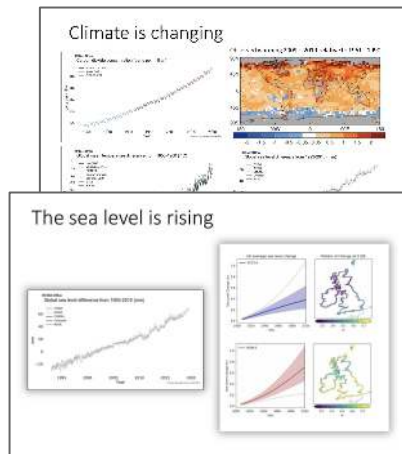
# Reproducible science

*“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





deep software variability

software application variability

input data variability

build variability

compiler variability

container variability

hypervisor variability

operating system variability

hardware variability

version variability

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!

# AGENDA

Reproducible Science and (Deep) Software (Variability)

**Deep Software Variability**

Evidence of Deep Software Variability in Science

Threats and Opportunities



**deep software variability**

**hardware variability**

**15,000+ options**

**thousands of compiler flags  
and compile-time options**

**dozens of preferences**

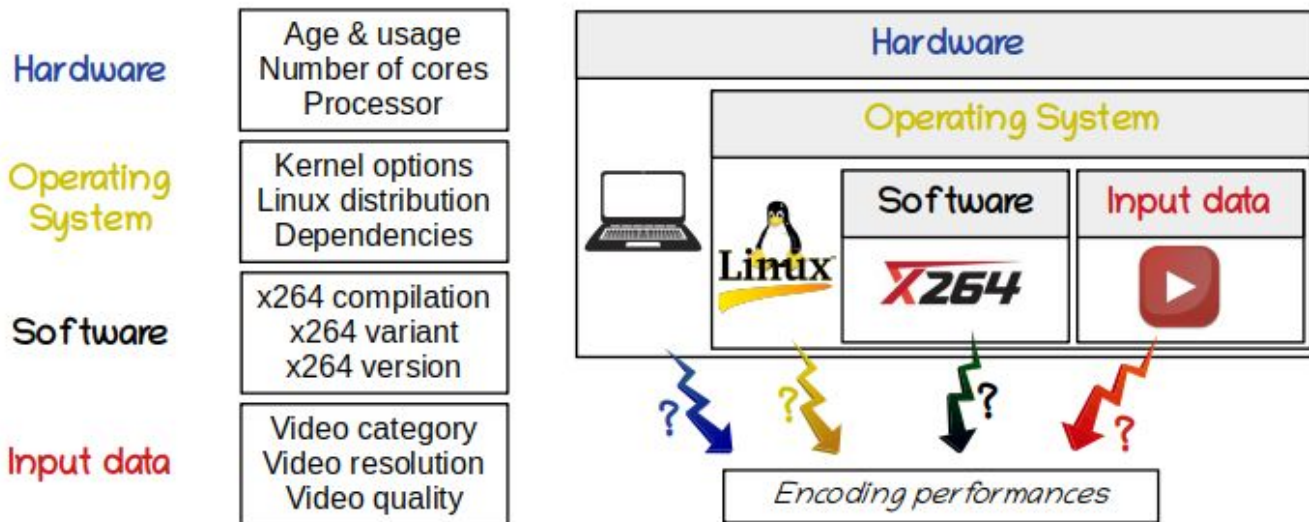
**100+ command-line parameters**

**1000+ feature toggles**

# Deep software variability: does it matter? *i.e.* Are layers/features orthogonal or are there interactions?

*Variability layers*

**264** environment



Luc Lesoil, Mathieu Acher, Arnaud Blouin, Jean-Marc Jézéquel:  
*Deep Software Variability: Towards Handling Cross-Layer Configuration.*

# Configuration is hard: numerous options, informal knowledge



```
--bframes 1 --ref 3 --cabac DiverSE-meeting.mp4 -o meeting13.webm
```

??????



```
mathieuacher localhost.localdomain ~ x264 --fullhelp | wc -l
```

480

## Lossless:

```
x264 --qp 0 -o <output> <input>
```

## Maximum PSNR at the cost of speed and visual quality:

```
x264 --preset placebo --tune psnr -o <output> <input>
```

## Constant bitrate at 1000kbps with a 2 second-buffer:

```
x264 --vbv-bufsize 2000 --bitrate 1000 -o <output> <input>
```

## Presets:

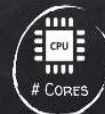
```
--profile <string> Force the limits of an H.264 profile
                    Overrides all settings.
                    - baseline, main, high, high10, high422
--preset <string> Use a preset to select encoding settings [medium]
                  Overridden by user settings.
                  - ultrafast,superfast,veryfast,faster,fast
                  - medium,slow,slower,veryslow,placebo
--tune <string> Tune the settings for a particular type of source
                or situation
                Overridden by user settings.
                Multiple tunings are separated by commas.
                Only one psy tuning can be used at a time.
```

```
-I, --keyint <integer or "infinite"> Maximum GOP size [250]
--tff Enable interlaced mode (top field first)
--bff Enable interlaced
--pulldown <string> Use soft pulldown
                  - none, 22, 3
```

## Ratecontrol:

```
-B, --bitrate <integer> Set bitrate (kbit/s)
--crf <float> Quality-based VBR
--vbv-maxrate <integer> Max local bitrate
--vbv-bufsize <integer> Set size of the VBV buffer
-p, --pass <integer> Enable multipass
                  - 1: First pass
                  - 2: Last pass
```

## HARDWARE



## OPERATING SYSTEM



## SOFTWARE



## INPUT DATA



# REAL WORLD EXAMPLE (x264)

**HARDWARE**

**OPERATING SYSTEM**

**SOFTWARE**

**INPUT DATA**



(A) DELL LATITUDE 7400



(B) RASPBERRY PI 4 MODEL B



20.04



10.4

x264 (1) --mbtree

(2) x264 --no-mbtree

x264 (1) --mbtree

(2) x264 --no-mbtree



animation

vertical

animation

vertical

animation

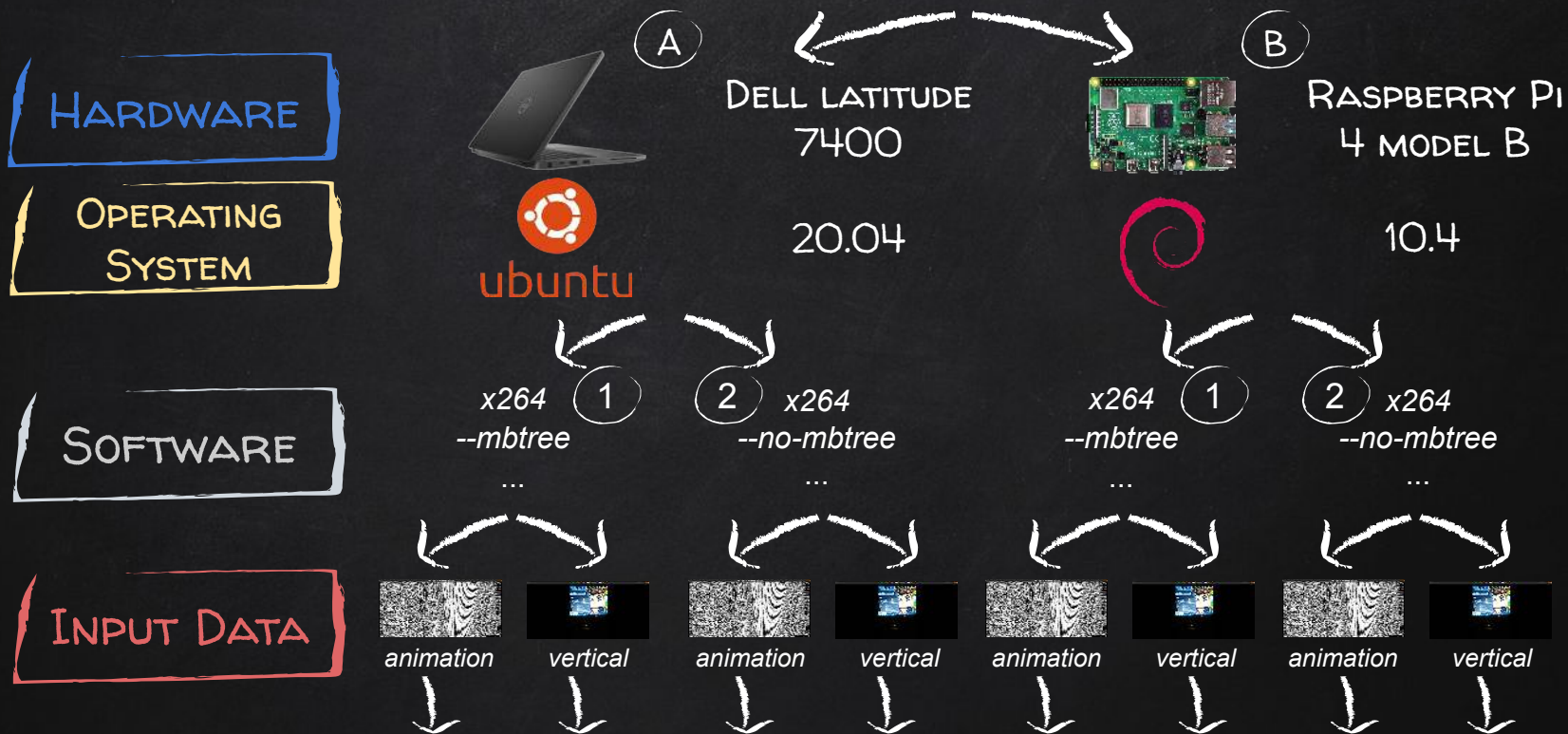
vertical

animation

vertical

DURATION (s)	6	22	6	25	73	351	72	359
SIZE (MB)	33	28	21	34	33	28	21	34

# REAL WORLD EXAMPLE (x264)



DURATION (s)	6	22	6	25	73	351	72	359
SIZE (MB)	33	28	21	34	33	28	21	34



# REAL WORLD EXAMPLE (x264)

**HARDWARE**

**OPERATING SYSTEM**

**SOFTWARE**

**INPUT DATA**



(A) DELL LATITUDE 7400



(B) RASPBERRY PI 4 MODEL B



20.04



10.4

x264 (1) --mbtree

(2) x264 --no-mbtree

x264 (1) --mbtree

(2) x264 --no-mbtree



animation

vertical

animation

vertical

animation

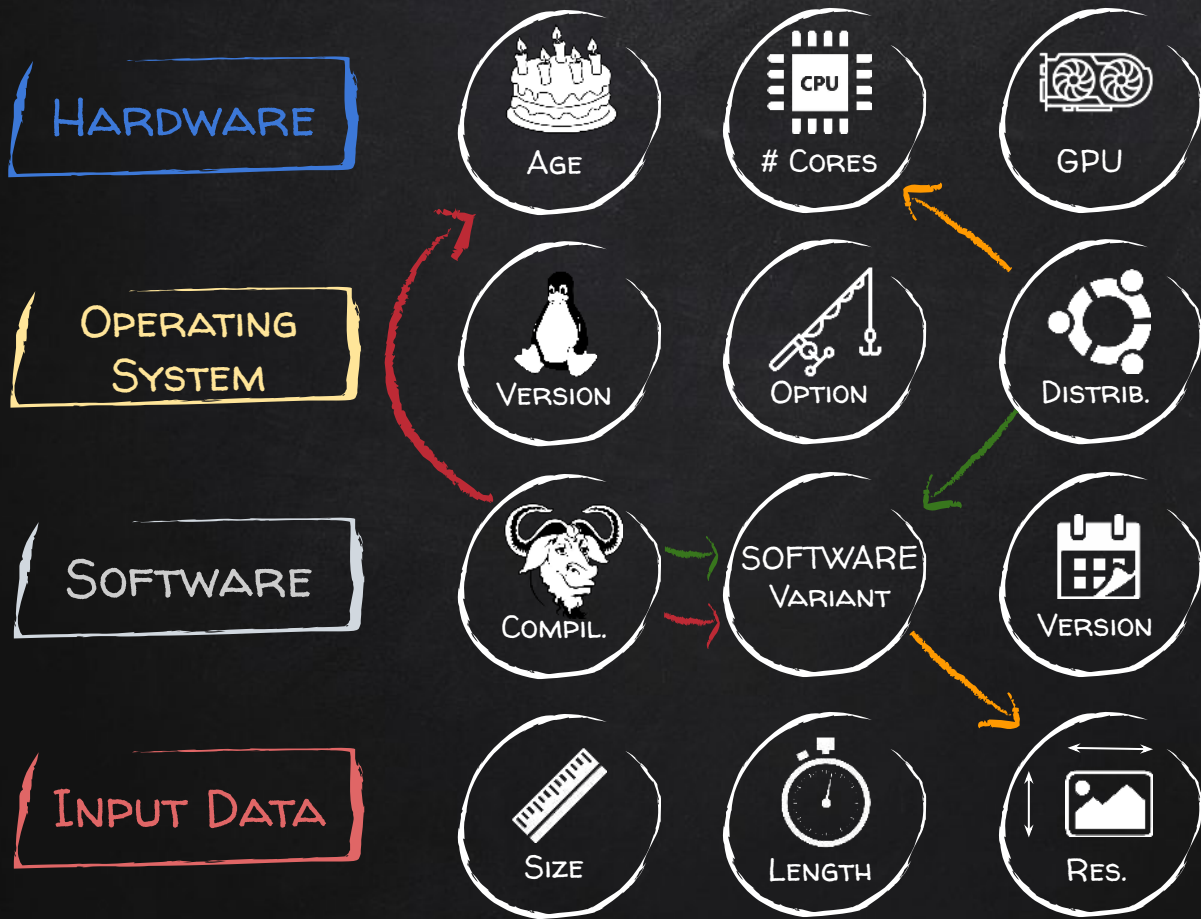
vertical

animation

vertical

DURATION (s)	6	22	6	25	73	351	72	359	≈*16
SIZE (MB)	33	28	21	34	33	28	21	34	≈*12

# DEEP VARIABILITY



The “best”/default software variant might be a bad one.

Influential software options and their interactions vary.

Performance prediction models and **variability knowledge may not generalize**



L. Lesoil, M. Acher, A. Blouin and J.-M. Jézéquel, “Deep Software Variability: Towards Handling Cross-Layer Configuration” in VaMoS 2021

# Transferring Performance Prediction Models Across Different Hardware Platforms

Valov et al. ICPE 2017

Table 3: Summary of measured systems;  $N_f$  – Number of features; NM – Number of machines on which systems were measured; NMC – Number of measured configurations

System	$N_f$	NM	NMC
XZ	7	7	154
x264	7	11	165
SQLite	5	10	32

Table 2: Summary of hardware platforms on which configurable software systems were measured; MID – Machine ID in DataMill cluster; NC – Number of CPUs; IS – Instruction set; CCR – CPU clock rate (MHz); RAM – RAM memory size (MB)

Systems			Machines				
XZ	x264	SQLite	MID	NC	IS	CCR	RAM
✓			73	2	i686	1733	1771
✓	✓	✓	75	2	i686	3200	977
✓			77	2	i686	2992	2024
✓			78	1	i686	1495	755
✓			79	4	x86_64	3291	7961
✓			80	8	x86_64	3401	7907
✓	✓		81	16	x86_64	2411	32193
	✓		87	1	i686	1595	249
	✓		88	1	i686	1700	978
		✓	90	2	i686	3200	977
	✓		91	1	i686	2400	1009
	✓		97	2	i686	2992	873
	✓	✓	98	2	i686	2992	873
		✓	99	2	i686	2793	880
	✓		103	2	i686	3200	881
	✓		104	1	i686	1800	502
	✓		105	2	i686	3200	881
	✓		106	2	i686	3192	494
		✓	125	4	x86_64	3301	7960
	✓		128	2	i686	2993	2024
		✓	130	2	i686	3198	880
		✓	146	2	i686	2998	872
		✓	157	36	x86_64	2301	15954

“**Linear model** provides a good approximation of transformation between performance distributions of a system deployed in **different hardware environments**”

what about  
**variability of  
input data?**

**compile-time options?**

**version?**



# Transfer Learning for Software Performance Analysis: An Exploratory Analysis

Jamshidi et al. ASE 2017

SPEAR (SAT Solver)	X264 (video encoder)	SQLite (DB engine)	SaC (Compiler)
<b>Analysis time</b>	<b>Encoding time</b>	<b>Query time</b>	<b>Execution time</b>
14 options	16 options	14 options	50 options
16,384 configurations	4,000 configurations	1,000 configurations	71,267 configurations
SAT problems	Video quality/size	DB Queries	10 Demo programs
3 hardware	2 hardware	2 hardware	
2 versions	3 versions	2 versions	

$ec_1 : [h_2 \rightarrow h_1, w_3, v_3]$	SM	0.97
$ec_2 : [h_2 \rightarrow h_1, w_1, v_3]$	S	0.96
$ec_3 : [h_1, w_1 \rightarrow w_2, v_3]$	M	0.65
$ec_4 : [h_1, w_1 \rightarrow w_3, v_3]$	M	0.67
$ec_5 : [h_1, w_3, v_2 \rightarrow v_3]$	L	0.05
$ec_6 : [h_1, w_3, v_1 \rightarrow v_3]$	L	0.06
$ec_7 : [h_1, w_1 \rightarrow w_3, v_2 \rightarrow v_3]$	L	0.08
$ec_8 : [h_2 \rightarrow h_1, w_1 \rightarrow w_3, v_2 \rightarrow v_3]$	VL	0.09



**Insight.** For non-severe hardware changes, we can linearly transfer performance models across environments.

**Insight.** The strength of the influence of configuration options is typically preserved across environments.

**Insight.** A large percentage of configurations are typically invalid in both source and target environments.

# Transfer Learning for Software Performance Analysis: An Exploratory Analysis

Jamshidi et al. ASE 2017

SPEAR (SAT Solver)	X264 (video encoder)	SQLite (DB engine)	SaC (Compiler)
<b>Analysis time</b>	<b>Encoding time</b>	<b>Query time</b>	<b>Execution time</b>
14 options	16 options	14 options	50 options
16,384 configurations	4,000 configurations	1,000 configurations	71,267 configurations
SAT problems	Video quality/size	DB Queries	10 Demo programs
3 hardware	2 hardware	2 hardware	
2 versions	3 versions	2 versions	

$ec_1 : [h_2 \rightarrow h_1, w_3, v_3]$	SM	0.97
$ec_2 : [h_2 \rightarrow h_1, w_1, v_3]$	S	0.96
$ec_3 : [h_1, w_1 \rightarrow w_2, v_3]$	M	0.65
$ec_4 : [h_1, w_1 \rightarrow w_3, v_3]$	M	0.67
$ec_5 : [h_1, w_3, v_2 \rightarrow v_3]$	L	0.05
$ec_6 : [h_1, w_3, v_1 \rightarrow v_3]$	L	0.06
$ec_7 : [h_1, w_1 \rightarrow w_3, v_2 \rightarrow v_3]$	L	0.08
$ec_8 : [h_2 \rightarrow h_1, w_1 \rightarrow w_3, v_2 \rightarrow v_3]$	VL	0.09



**mixing deep variability:** hard to assess the specific influence of each layer

**very few hardware, version, and input data...** but lots of runtime configurations (variants)

**Let's go deep with input data!**

# Let's go deep with **input data!**



Intuition: video encoder behavior (and thus **runtime configurations**) hugely depends on the **input video** (different compression ratio, encoding size/type etc.)

Is the best software configuration still the best?

Are influential options always influential?

Does the configuration knowledge generalize?

two performance models  $f_1$  and  $f_2$

$$f_1 = \beta \times f_2 + \alpha$$



YouTube User General Content dataset: **1397 videos**  
Measurements of **201 soft. configurations** (with same hardware, compiler, version, etc.): encoding time, bitrate, etc.



# Do x264 software performances stay consistent across inputs?

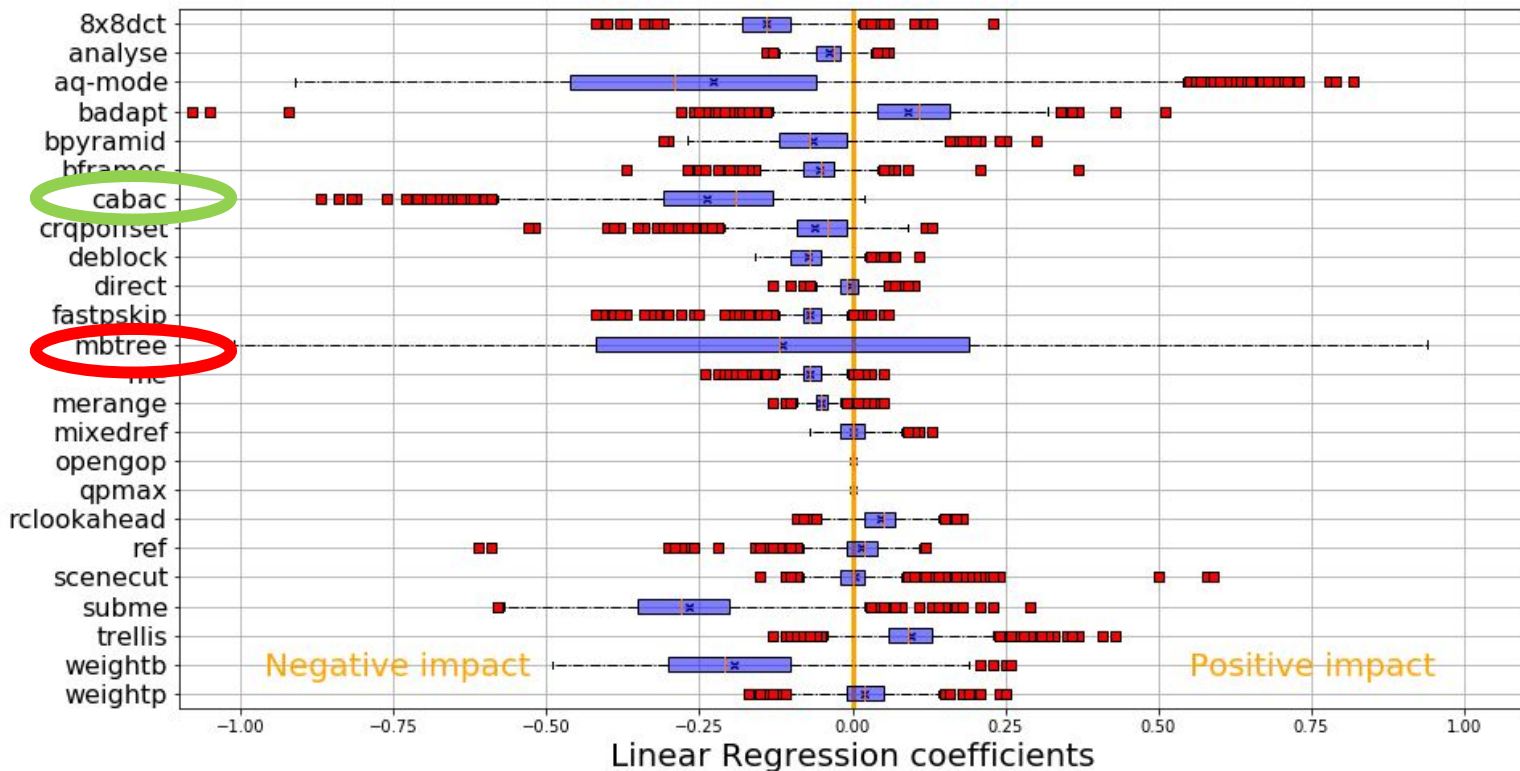


**1397 videos x 201 software configurations**

- Encoding time: very strong correlations
  - low input sensitivity
- FPS: very strong correlations
  - low input sensitivity
- CPU usage : moderate correlation, a few negative correlations
  - medium input sensitivity
- Bitrate: medium-low correlation, many negative correlations
  - High input sensitivity
- Encoding size: medium-low correlation, many negative correlations
  - High input sensitivity

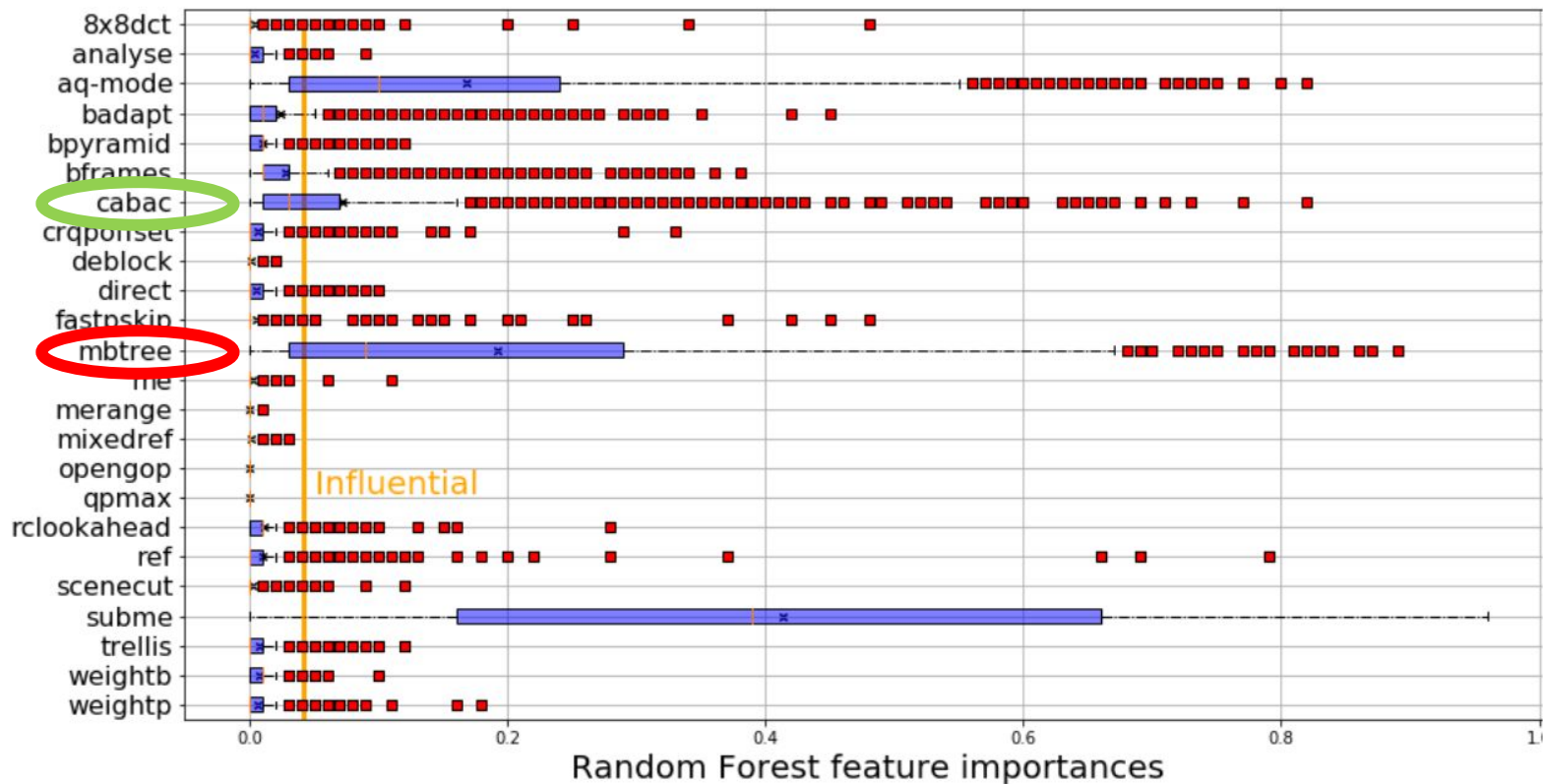
two performance models  $f_1$  and  $f_2$   $f_1 = \beta \times f_2 + \alpha$  ?

# Are there some configuration options more sensitive to input videos? (bitrate)





# Are there some configuration options more sensitive to input videos? (bitrate)



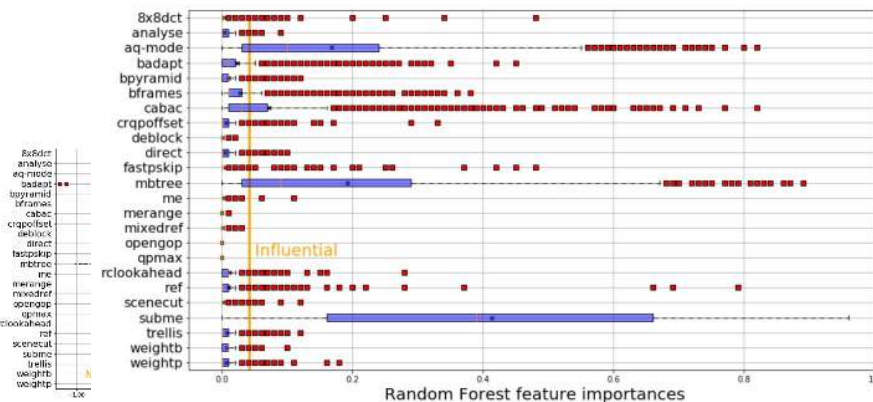
# Practical impacts for users, developers, scientists, and self-adaptive systems

Threats to variability knowledge for performance property bitrate



- optimal configuration is specific to an input; a good configuration can be a bad one
- some options' values have an opposite effect depending on the input
- effectiveness of sampling strategies (random, 2-wise, etc.) is input specific (somehow confirming Pereira et al. ICPE 2020)
- predicting, tuning, or understanding configurable systems

without being aware of inputs can be inaccurate and... pointless



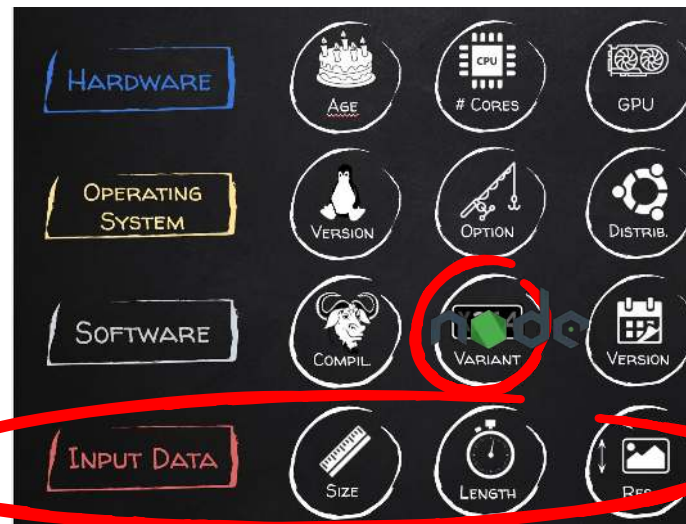
# Practical impacts for users, developers, scientists, and self-adaptive systems

**Threats** to **variability knowledge**: predicting, tuning, or understanding configurable systems without being aware of inputs can be inaccurate and... pointless

**Opportunities**: for some performance properties (P) and subject systems, some stability is observed and performance remains consistent!

System	Domain	Commit	Configs #C	Inputs I	#I
gcc	Compilation	ccb4e07	80	.c programs	30
ImageMagick	Image processing	5ee49d6	100	images	1000
lingeling	SAT solver	7d5db72	100	SAT formulae	351
nodeJS	JS runtime env.	78343bb	50	.js scripts	1939
poppler	PDF rendering	42dde68	16	.pdf files	1480
SQLite	DBMS	53fa025	50	databases	150
x264	Video encoding	e9a5903	201	videos	1397
xz	Data compression	e7da44d	30	system files	48

System	#M	Performance(s) P	Docker	Dataset
gcc	2400	size, ctime, exec	Link	Link
ImageMagick	100 000	size, time	Link	Link
lingeling	35 100	#confl.,#reduc.	Link	Link
nodeJS	96 950	#operations/s	Link	Link
poppler	23 680	size, time	Link	Link
SQLite	7500	15 query times q1-q15	Link	Link
x264	280 797	cpu, fps, kbs, size, time	Link	Link
xz	1440	size, time	Link	Link



L. Lesoil, M. Acher, A. Blouin and J.-M. Jézéquel “The Interaction between Inputs and Configurations fed to Software Systems: an Empirical Study”  
<https://arxiv.org/abs/2112.07279>

# x264 video encoder (compilation/build)

~ x264 --bframes 1 --ref 3 --cabac DiverSE-meeting.mp4 -o meeting13.webm

mathieuacher localhost.localdomain ../x264-SLIMFAST/x264 x264-gcov □ ./configure --help

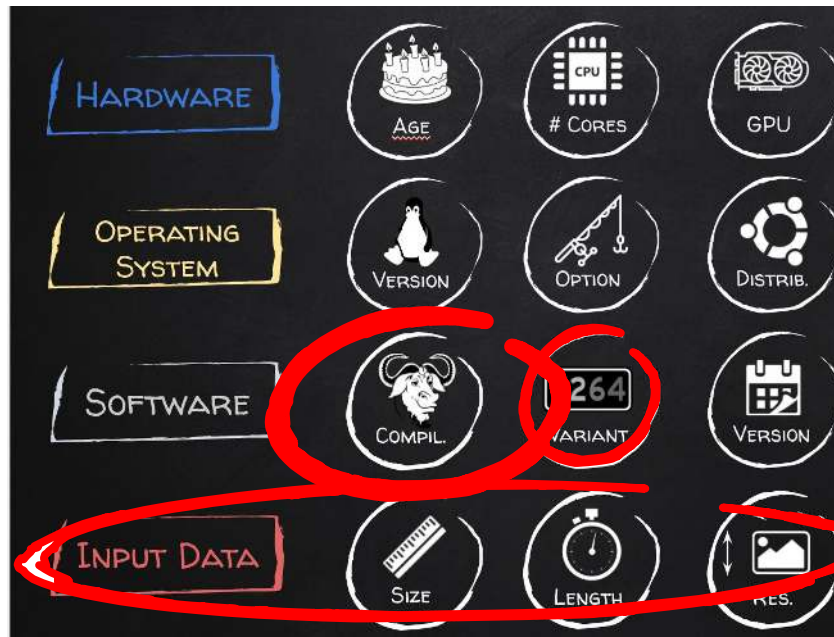
```
--disable-thread      disable multithreaded encoding
--disable-win32thread  disable win32threads (windows only)
--disable-interlaced   disable interlaced encoding support
--bit-depth=BIT_DEPTH set output bit depth (8, 10, all) [all]
--chroma-format=FORMAT output chroma format (400, 420, 422, 444, all) [all]
```

```
Advanced options:
--disable-asm         disable platform-specific assembly optimizations
--enable-lto          enable link-time optimization
--enable-debug        add -g
--enable-gprof        add -pg
--enable-strip        add -s
--enable-pic          build position-independent code
```

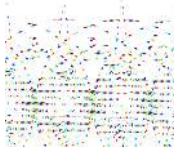
```
Cross-compilation:
--host=HOST           build programs to run on HOST
--cross-prefix=PREFIX use PREFIX for compilation tools
--sysroot=SYSROOT    root of cross-build tree
```

```
External library support:
--disable-avs         disable avisynth support
--disable-awscale    disable swscale support
--disable-lavf        disable libavformat support
--disable-ffms        disable ffmpegsource support
--disable-gpac        disable gpac support
--disable-lsmash     disable lsmash support
```

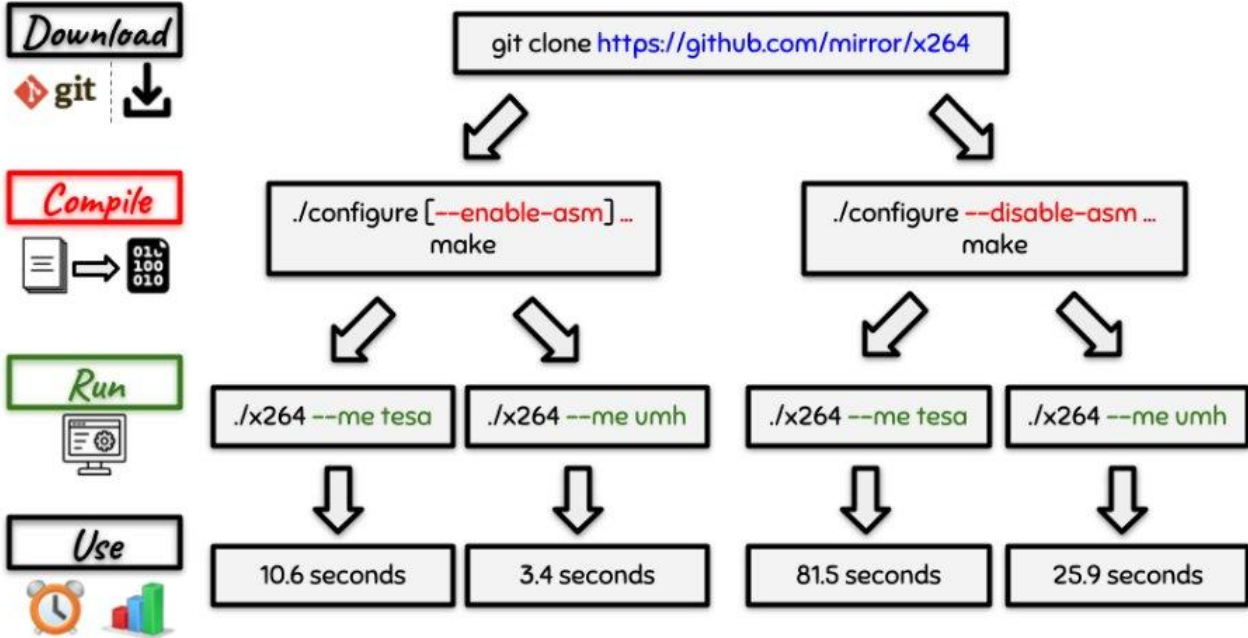
compile-time options



# Is there an interplay between **compile-time** and **runtime** options?



L. Lesoil, M. Acher, X. Těrnava, A. Blouin and J.-M. Jézéquel "The Interplay of Compile-time and Run-time Options for Performance Prediction" in SPLC '21



This paper investigates how compile-time options can affect software performances and how compile-time options interact with run-time options.

Figure 1: Cross-layer variability of x264



## Key results (for x264)

**Worth tuning software at compile-time:** gain about 10 % of execution time with the tuning of compile-time options (compared to the default compile-time configuration). The improvements can be larger for some inputs and some runtime configurations.

**Stability of variability knowledge:** For all the execution time distributions of x264 and all the input videos, the worst correlation is greater than 0.97. If the compile-time options change the scale of the distribution, they do not change the rankings of run-time configurations (i.e., they do not truly interact with the run-time options).

Reuse of configuration knowledge:  $f_1 = \beta \times f_2 + \alpha$

- Linear transformation among distributions
- Users can also trust the documentation of run-time options, consistent whatever the compile-time configuration is.



# Key results (Poppler)

First good news: Worth it

Second good news: correlation is weak. It does not change the rankings

It has three practical implications:



# XZ

performances!

**interplay between compile-time and runtime options and even input!**

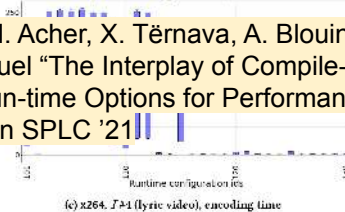
transfer learning of options from one input to another. Users can customize the configuration at run time.

Best compile-time configuration among all the possible ones allows configuration at run time. We can remove away a default compile-time configuration, use

performance for



L. Lesoil, M. Acher, X. Těrnava, A. Blouin and J.-M. Jézéquel "The Interplay of Compile-time and Run-time Options for Performance Prediction" in SPLC '21



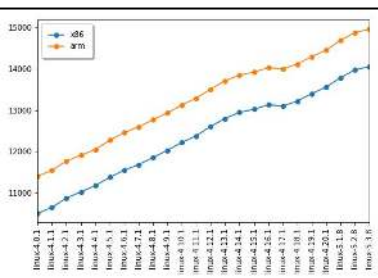
all the input videos, the worst scale of the distribution, they do not interact with the runtime options).



- Linux as a subject software system (not as an OS interacting with other layers)
- Targeted non-functional, quantitative property: binary size
  - interest for maintainers/users of the Linux kernel (embedded systems, cloud, etc.)
  - challenging to predict (cross-cutting options, interplay with compilers/build systems, etc./.)
- Dataset: version 4.13.3 (september 2017), x86\_64 arch, measurements of 95K+ random configurations
  - paranoiac about deep variability since 2017, Docker to control the build environment and scale
  - diversity of binary sizes: from 7Mb to 1.9Gb
  - 6% MAPE errors: quite good, though costly...



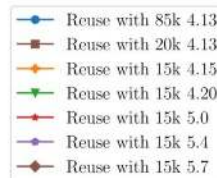
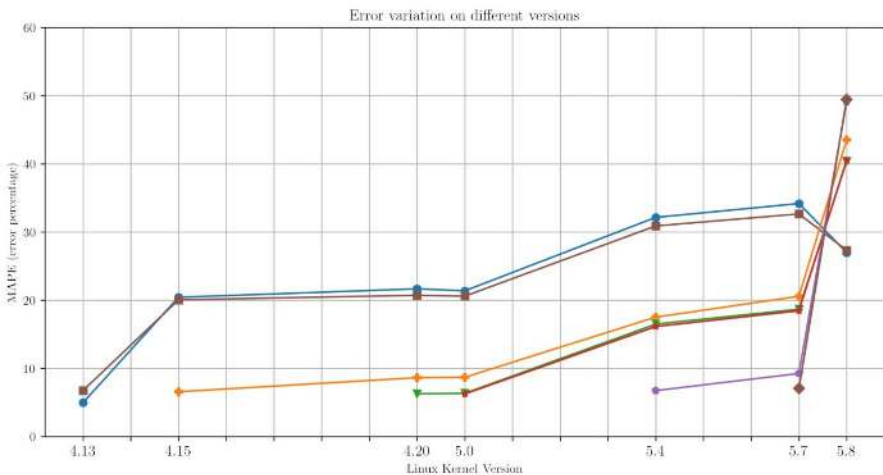




Version	Release Date	LOC	Files	Examples	Seconds/config	Options	Features	Deleted features	New features	$\Delta$ Commits	Files changes
4.13	2017/09/03	16,616,534	60,530	92,562	not available	12,776	9,468	-	-	-	-
4.15	2018/01/28	17,073,368	62,249	39,391	not available	12,998	9,425	342	299	31,052	934,628
4.20	2018/12/23	17,526,171	62,423	23,489	225	13,533	10,189	468	1,189	104,691	1,972,020
5.0	2019/03/03	17,679,372	63,076	19,952	247	13,673	10,293	494	1,319	118,778	2,170,935
5.4	2019/10/24	19,358,903	67,915	25,847	285	14,159	10,813	663	2,008	181,308	3,827,025
5.7	2020/05/31	19,358,903	67,915	20,159	258	14,586	11,338	715	2,585	225,804	4,393,117
5.8	2020/08/02	19,729,197	69,303	21,923	289	14,817	11,530	730	2,792	242,381	4,681,313

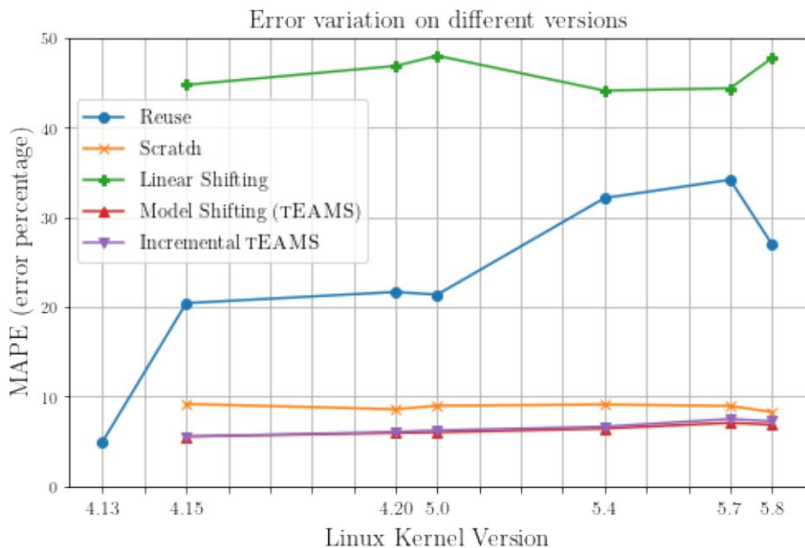
Table I: Dataset properties for each version. The number of deleted/new features, delta commits, files changes are w.r.t. 4.13.

4.13 version (sep 2017): 6%. What about **evolution?** Can we reuse the 4.13 Linux prediction model? No, accuracy quickly decreases: **4.15 (5 months after): 20%; 5.7 (3 years after): 35%**



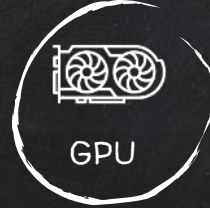
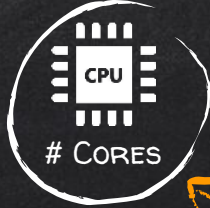
# Transfer learning to the rescue

- Mission Impossible: Saving variability knowledge and prediction model 4.13 (15K hours of computation)
- Heterogeneous transfer learning: the feature space is different
- TEAMS: transfer evolution-aware model shifting



# DEEP VARIABILITY

HARDWARE



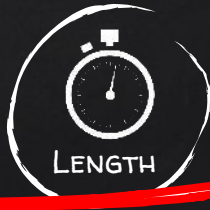
OPERATING SYSTEM



SOFTWARE



INPUT DATA



Sometimes, variability is consistent/stable and knowledge transfer is immediate.

But there are also interactions among variability layers and **variability knowledge may not generalize**

- BUG
- PERF. ↓
- PERF. ↗

# AGENDA

Reproducible Science and (Deep) Software (Variability)

Deep Software Variability

**Evidence of Deep Software Variability in Computational Science**

Threats and Opportunities

## DEEP VARIABILITY



**It's all about software...  
Where is  
(computational)  
science?**



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

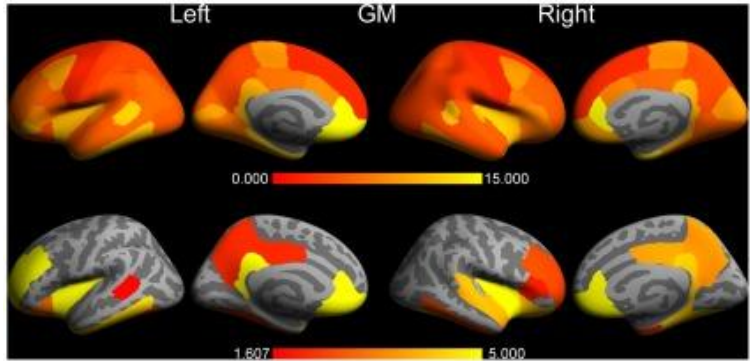
# “Neuroimaging pipelines are known to generate different results depending on the computing platform where they are compiled and

> [PLoS One. 2012;7\(6\):e38234. doi: 10.1371/journal.pone.0038234. Epub 2012 Jun 1.](https://doi.org/10.1371/journal.pone.0038234)

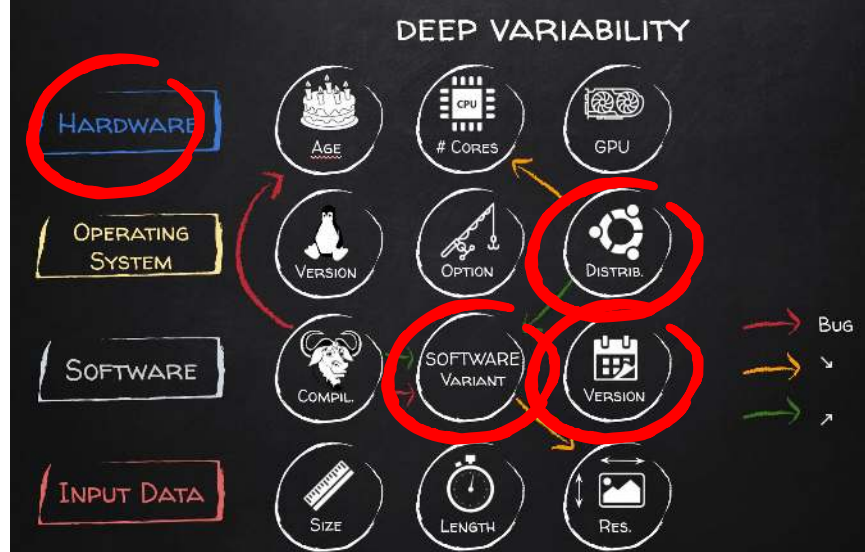
## The effects of FreeSurfer version, workstation type, and Macintosh operating system version on anatomical volume and cortical thickness measurements

Ed H B M Gronenschild <sup>1</sup>, 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.**

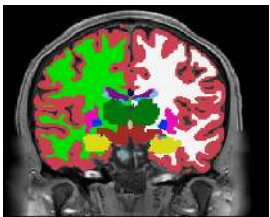


see also Krefting, D., Scheel, M., Freing, A., Specovius, S., Paul, F., and Brandt, A. (2011). “Reliability of quantitative neuroimage analysis using freesurfer in distributed environments,” in *MICCAI Workshop on High-Performance and Distributed Computing for Medical Imaging*. (Toronto, ON).



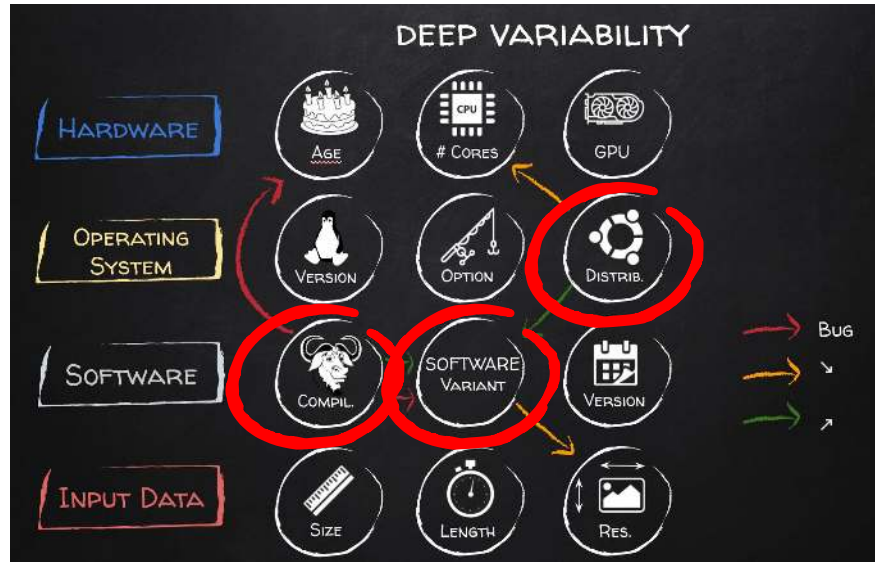
“Neuroimaging pipelines are known to generate different results depending on the computing platform where they are compiled and executed.”

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.

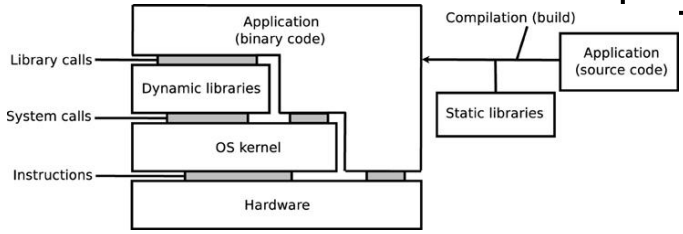
	Cluster A	Cluster B
Applications	Freesurfer 5.3.0, build 1 FSL 5.0.6, build 1 CIVET 1.1.12-UCSF, build 1	Freesurfer 5.3.0, build 1 and 2 FSL 5.0.6, build 1 and 2 CIVET 1.1.12-UCSF, build 1
Interpreters	Python 2.4.3, bash 3.2.25, Perl 5.8.8, tcsh 6.14.00	Python 2.7.5, bash 4.2.47, Perl 5.18.2, tcsh 6.18.01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



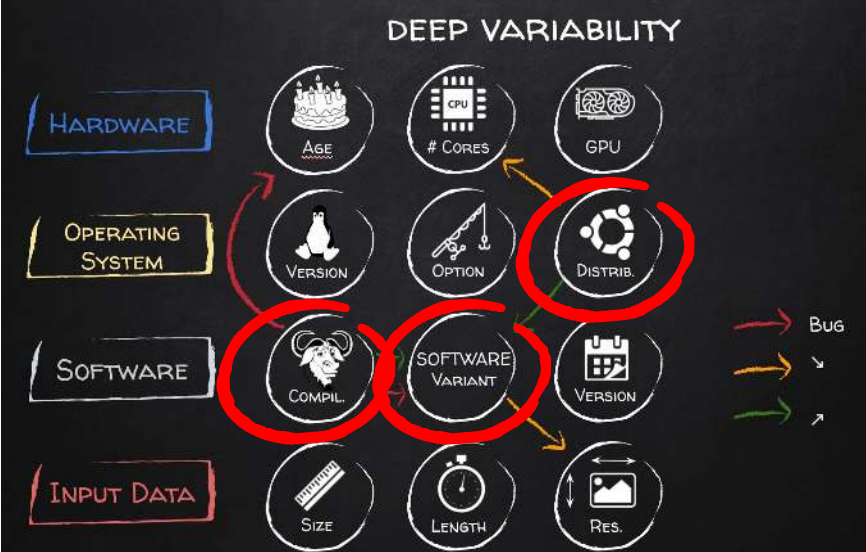
“Neuroimaging pipelines are known to generate different results depending on the computing platform where they are compiled and executed.”

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 FSL 5.0.6, build 1 CIVET 1.1.12-UCSF, build 1	Freesurfer 5.3.0, build 1 and 2 FSL 5.0.6, build 1 and 2 CIVET 1.1.12-UCSF, build 1
Interpreters	Python 2.4.3, bash 3.2.25, Perl 5.8.8, tcsh 6.14.00	Python 2.7.5, bash 4.2.47, 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)





Explanatory variable	Meaning
<i>entcoef</i>	Entrainment coefficient
<i>ct</i>	Accretion constant
<i>rhcrit</i>	Critical relative humidity
<i>vf1</i>	Ice fall speed through clouds
<i>eacf</i>	Empirically adjusted cloud fraction
<i>cw</i>	Threshold for precipitation
<i>dtice</i>	Temperature range of ice albedo variation
<i>ice</i>	Nonspherical ice
<i>middleware</i>	Client middleware
<i>ice_size</i>	Ice particle size
<i>alphan</i>	Albedo at melting point of ice
<i>processor_name</i>	CPU classification
<i>clock_classic</i>	Processor clock speed recorded under classic middleware
<i>ram_size</i>	Hardware RAM
<i>clock_boinc_i</i>	Integer processor clock speed recorded under BOINC middleware
<i>clock_boinc_f</i>	Floating point processor clock speed recorded under BOINC middleware
<i>os_name</i>	Operating system
<i>dtheta</i>	Perturbations to initial conditions on a given level

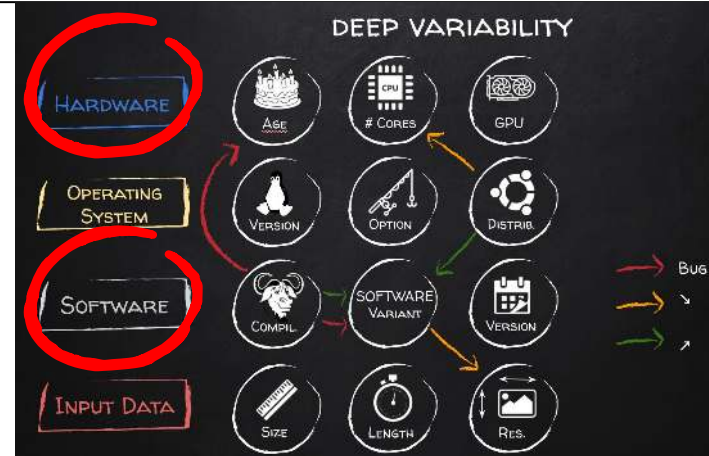
## Association of parameter, software, and hardware variation with large-scale behavior across 57,000 climate models

Christopher G. Knight, Sylvia H. E. Knight, Neil Massey, Tolu Aina, Carl Christensen, Dave J. F...

[+ See all authors and affiliations](#)

PNAS July 24, 2007 104 (30) 12259-12264; <https://doi.org/10.1073/pnas.0608144104>

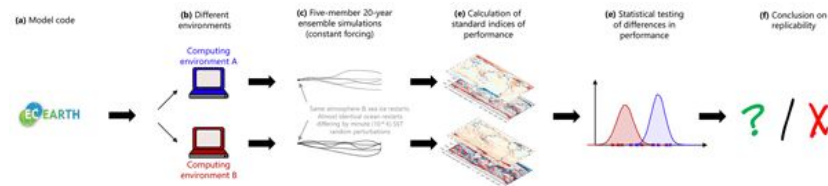
We demonstrate that **effects of parameter, hardware, and software variation are detectable, complex, and interacting**. However, we find most of the effects of parameter variation are caused by a small subset of parameters. Notably, the entrainment coefficient in clouds is associated with 30% of the variation seen in climate sensitivity, although both low and high values can give high climate sensitivity. **We demonstrate that the effect of hardware and software is small relative to the effect of parameter variation** and, over the wide range of systems tested, may be treated as equivalent to that caused by changes in initial conditions.



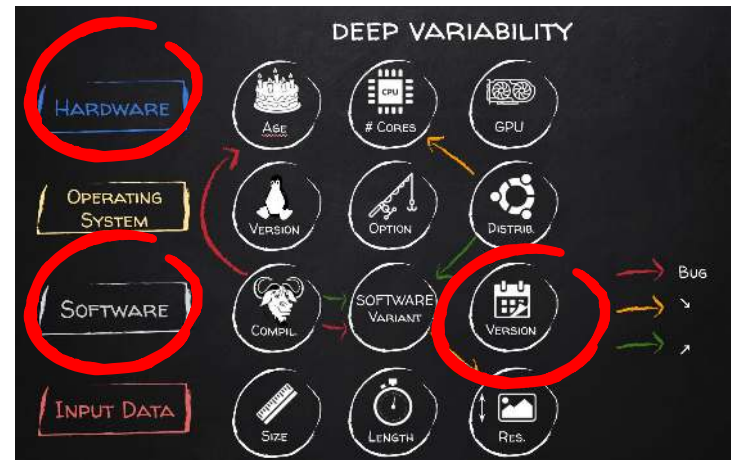
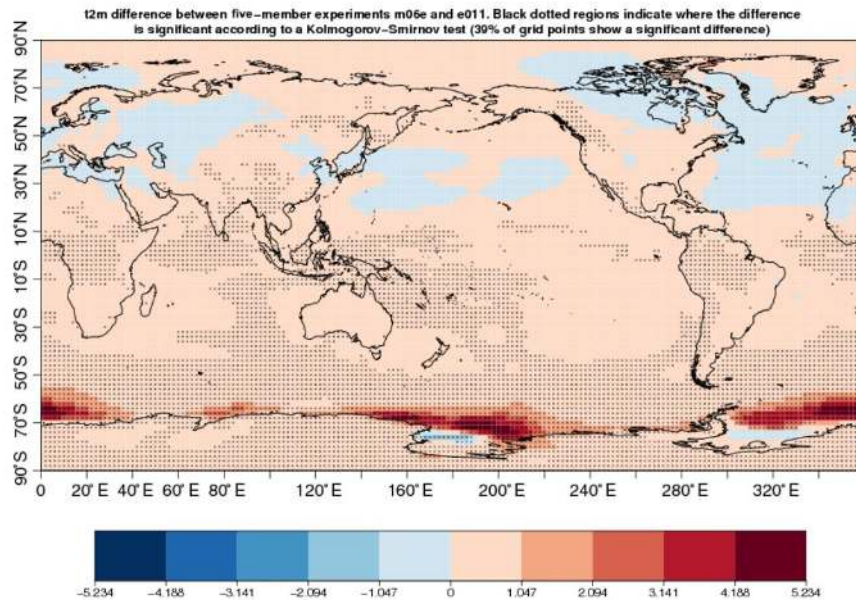
57,067 climate model runs. These runs sample parameter space for 10 parameters with between two and four levels of each, covering 12,487 parameter combinations (24% of possible combinations) and a range of initial conditions

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

François Massonnet<sup>1,2</sup>, Martin Ménégoz<sup>2,3</sup>, Mario Acosta<sup>2</sup>, Xavier Yepes-Arbós<sup>2</sup>, Eleftheria Exarchou<sup>2</sup>, and Francisco J. Doblas-Reyes<sup>2,4</sup>

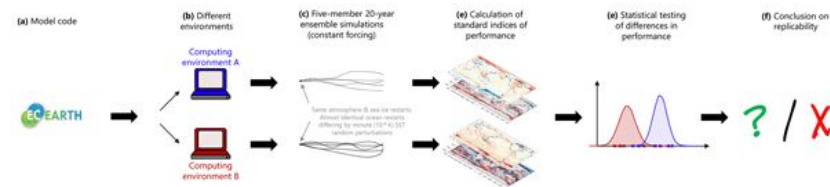


**Can a coupled ESM simulation be restarted from a different machine without causing climate-changing modifications in the results?** Using two versions of EC-Earth: one “non-replicable” case (see below) and one replicable case.



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

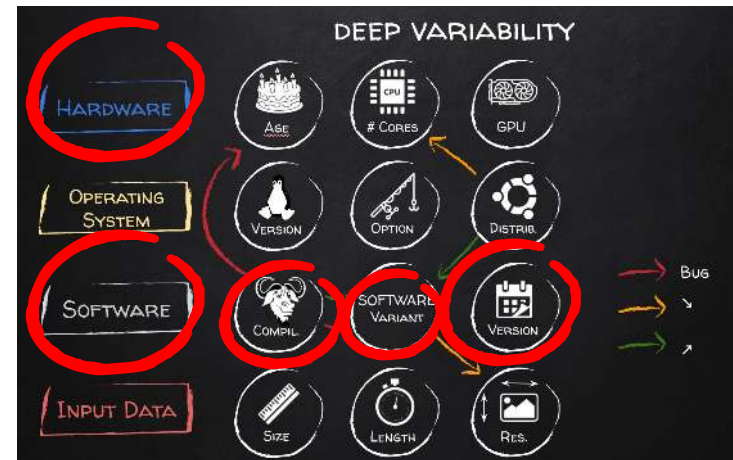
François Massonnet<sup>1,2</sup>, Martin Ménégoz<sup>2,3</sup>, Mario Acosta<sup>1,2</sup>, Xavier Yepes-Arbós<sup>1,2</sup>, Eleftheria Exarchou<sup>1,2</sup>, and Francisco J. Doblas-Reyes<sup>1,2,4</sup>



## Can a coupled ESM simulation be restarted from a different machine without causing climate-changing modifications in the results?

A study involving eight institutions and seven different supercomputers in Europe is currently ongoing with EC-Earth. This ongoing study aims to do the following:

- evaluate different **computational environments** that are used in collaboration to produce CMIP6 experiments (can we safely create large ensembles composed of subsets that emanate from different partners of the consortium?);
- detect if the same **CMIP6 configuration** is replicable among platforms of the EC-Earth consortium (that is, can we safely exchange restarts with EC-Earth partners in order to initialize simulations and to avoid long spin-ups?); and
- systematically evaluate the impact of **different compilation flag options** (that is, what is the highest acceptable level of optimization that will not break the replicability of EC-Earth for a given environment?).



Joelle Pineau “Building Reproducible, Reusable, and Robust Machine Learning Software” ICSE’19 keynote “[...] results can be brittle to even minor perturbations in the domain or experimental procedure”

## Deep Reinforcement Learning that Matters

Peter Henderson<sup>1\*</sup>, Riashat Islam<sup>1,2\*</sup>, Philip Bachman<sup>2</sup>  
Joelle Pineau<sup>1</sup>, Doina Precup<sup>1</sup>, David Meger<sup>1</sup>

What is the magnitude of the effect

**hyperparameter** settings can have on baseline performance?

How does the choice of **network architecture** for the policy and value function approximation affect performance?

How can the **reward scale** affect results?

Can **random seeds** drastically alter performance?

How do the **environment properties** affect variability in reported RL algorithm performance?

Are commonly used baseline **implementations** comparable?

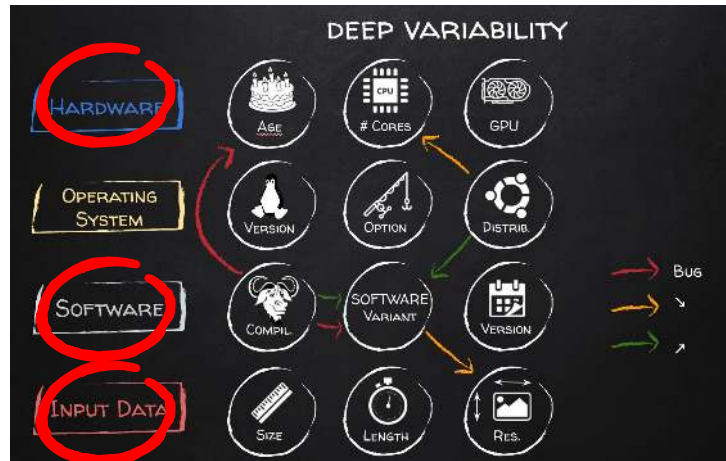
torch.manual\_seed(3407) is all you need: On the influence of random seeds in deep learning architectures for computer vision

David Picard  
LIGM, École des Ponts, 77455 Marnes la vallée, France

DAVID.PICARD@ENPC.FR

### Abstract

In this paper I investigate the effect of random seed selection on the accuracy when using popular deep learning architectures for computer vision. I scan a large amount of seeds (up to  $10^4$ ) on CIFAR 10 and I also scan fewer seeds on Imagenet using pre-trained models to investigate large scale datasets. The conclusions are that even if the variance is not very large, it is surprisingly easy to find an outlier that performs much better or much worse than the average.



# Reproducible and replicable CFD: it's harder than you think

Olivier Mesnard, Lorena A. Barba

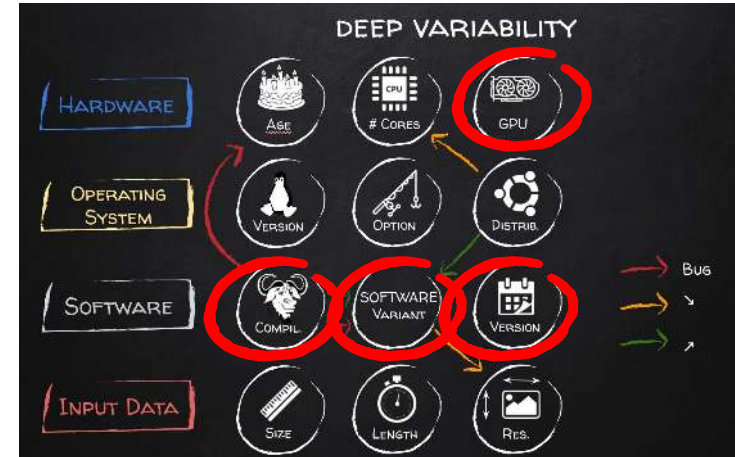
Mechanical and Aerospace Engineering, George Washington University, Washington DC 20052

“Completing a full replication study of our previously published findings on bluff-body aerodynamics was harder than we thought. Despite the fact that we have good reproducible-research practices, sharing our code and data openly.”

**Story 1: Meshing and boundary conditions can ruin everything**

**Story 3: All linear algebra libraries are not created equal**

**Story 4: Different versions of your code, external libraries or even compilers may challenge reproducibility**



# AGENDA

Reproducible Science and (Deep) Software (Variability)

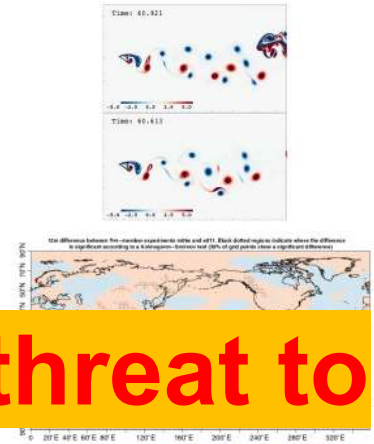
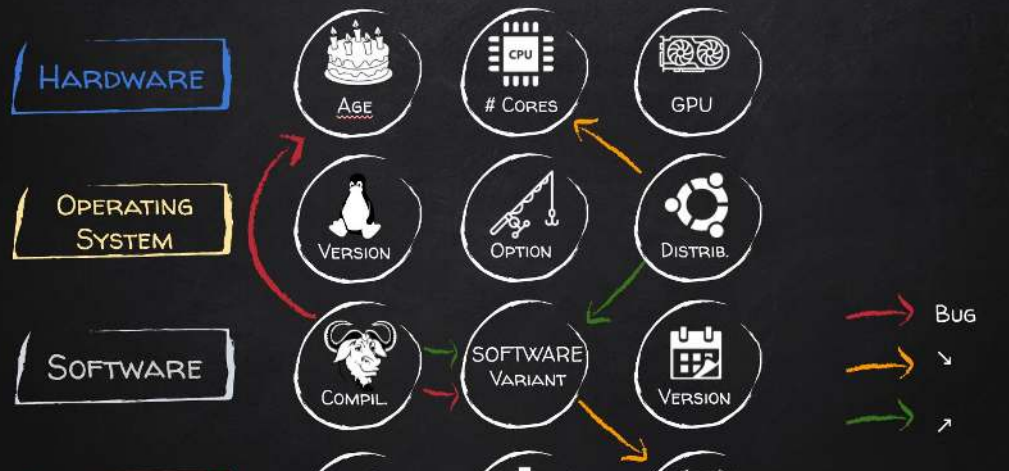
Deep Software Variability

Evidence of Deep Software Variability in Computational Science

**Threats and Opportunities**

# (computational) science

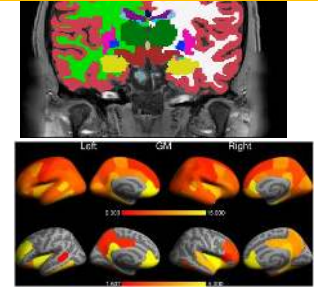
## DEEP VARIABILITY



**Deep software variability is a threat to scientific, software-based experiments**

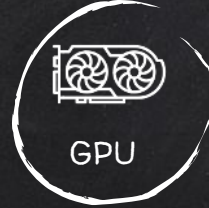
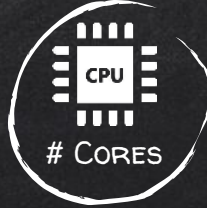


from a set of scripts to automate the deployment to... a comprehensive system containing several features that help researchers exploring various hypotheses



# DOES DEEP SOFTWARE VARIABILITY AFFECT PREVIOUS SCIENTIFIC, SOFTWARE-BASED STUDIES? (A GRAPHICAL TEMPLATE)

HARDWARE



OPERATING SYSTEM



SOFTWARE



INPUT DATA



LIST ALL DETAILS...

AND QUESTIONS:

WHAT IF WE RUN THE EXPERIMENTS ON DIFFERENT:

OS?

VERSION/COMMIT?

PARAMETERS?

INPUT?



# What can we do? (#1 studies)

## Empirical studies about deep software variability

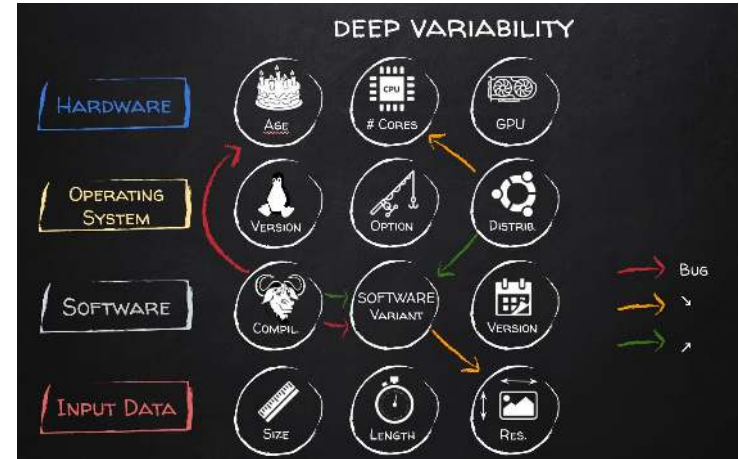
- more subject systems
- more variability layers, including interactions
- more quantitative (e.g., performance) properties

with challenges for gathering measurements data:

- how to scale experiments? Variant space is huge!
- how to fix/isolate some layers? (eg hardware)
- how to measure in a reliable way?

Expected outcomes:

- significance of deep software variability in the wild
- identification of **stable** layers: sources of variability that should not affect the conclusion and that can be eliminated/forgotten
- identification/quantification of **sensitive** layers and interactions that matter
- **variability knowledge**

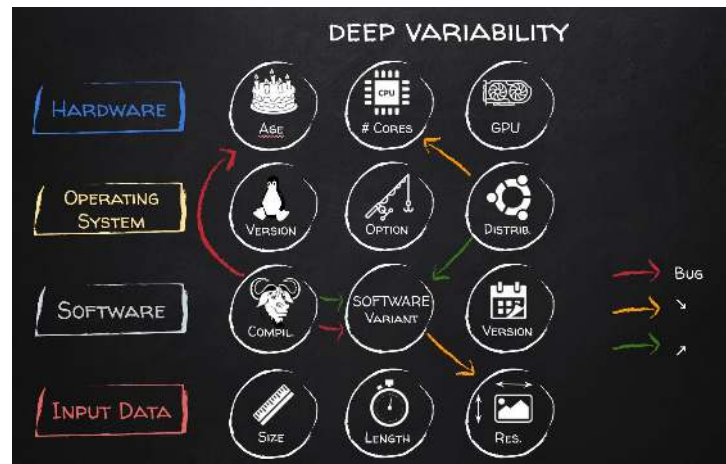


# What can we do? (#2 cost)

## Reducing the cost of exploring the variability spaces

Many directions here (references at the end of the slides):

- learning
  - many algorithms/techniques with tradeoffs interpretability/accuracy
  - transfer learning (instead of learning from scratch)
- sampling strategies
  - uniform random sampling? t-wise? distance-based? ...
  - sample of hardware? input data?
- incremental build of configurations
- white-box approaches
- ...



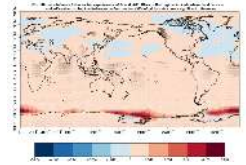
# What can we do? (#3 modelling)

## Modelling variability

- Abstractions are definitely needed to...
  - reason about logical constraints and interactions
  - integrate domain knowledge
  - synthesize domain knowledge
  - automate and guide the exploration of variants
  - scope and prioritize experiments
- Challenges:
  - Multiple systems, layers, concerns
  - Different kinds of variability: technical vs domain, accidental vs essential, implicit vs explicit... when to stop modelling?
  - reverse engineering



# What can we do? (#4 robustness)



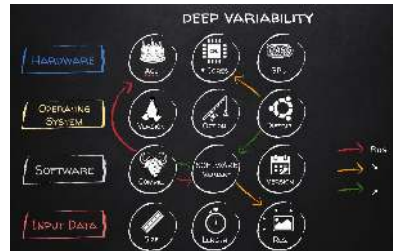
**Robustness (trustworthiness) of scientific results to sources of variability**

I have shown many examples of sources of variations and non-robust results...

Robustness should be rigorously defined (hint: it's not the definition as given in computer science)

How to verify the effect of sources of variations on the robustness of given conclusions?

- actionable metrics?
- methodology? (eg when to stop?)
- variability can actually be leveraged to augment confidence



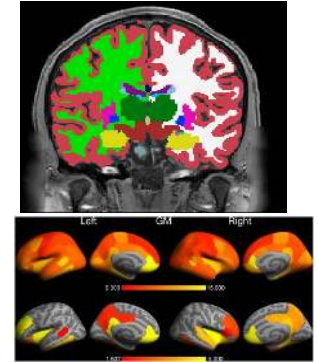
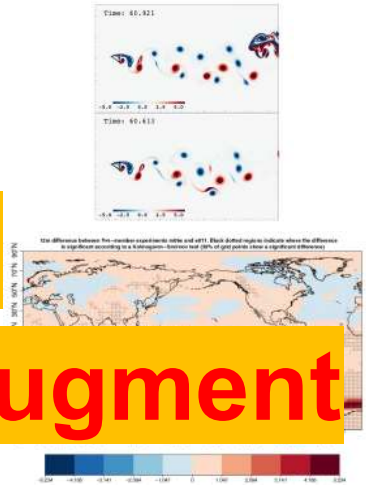
# (computational) science

## DEEP VARIABILITY



**Deep software variability is an opportunity to robustify and augment scientific knowledge**

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



different data

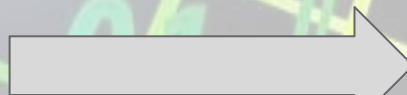


deep

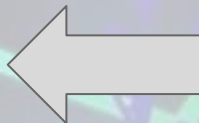
software

variability

different methods



different assumptions



### Variability in the analysis of a single neuroimaging dataset by many teams

Rotem Botvink-Nezer, Felix Holzmeister, ... Tom Schonberg + Show authors

Nature 582, 84–88 (2020) | [Cite this article](#)

42k Accesses | 203 Citations | 2056 Altmetric | [Metrics](#)

### Increasing Transparency Through a Multiverse Analysis

Sara Steegen <sup>1</sup>, Francis Tuerlinckx <sup>1</sup>, Andrew Gelman <sup>2</sup>, Wolf Vanpaemel <sup>3</sup>

Affiliations + expand

PMID: 27694465 DOI: 10.1177/1745691616658637

### Many Analysts, One Data Set: Making Transparent How Variations in Analytic Choices Affect Results

R. Silberzahn, E. L. Uhlmann, D. P. Martin, more...

Show all authors

First Published August 23, 2018 | Research Article |

<https://doi.org/10.1177/2515245917747646>

different analyses



“When you are a researcher, you want to be the first. Research is about discovery, inventing things that others have not done. It is by definition a form of competition. You have to accept that.” CNRS CEO



At a time of global pandemics, global warming and unprecedented public distrust of politics and democracy, research is anything but an individualistic competition.

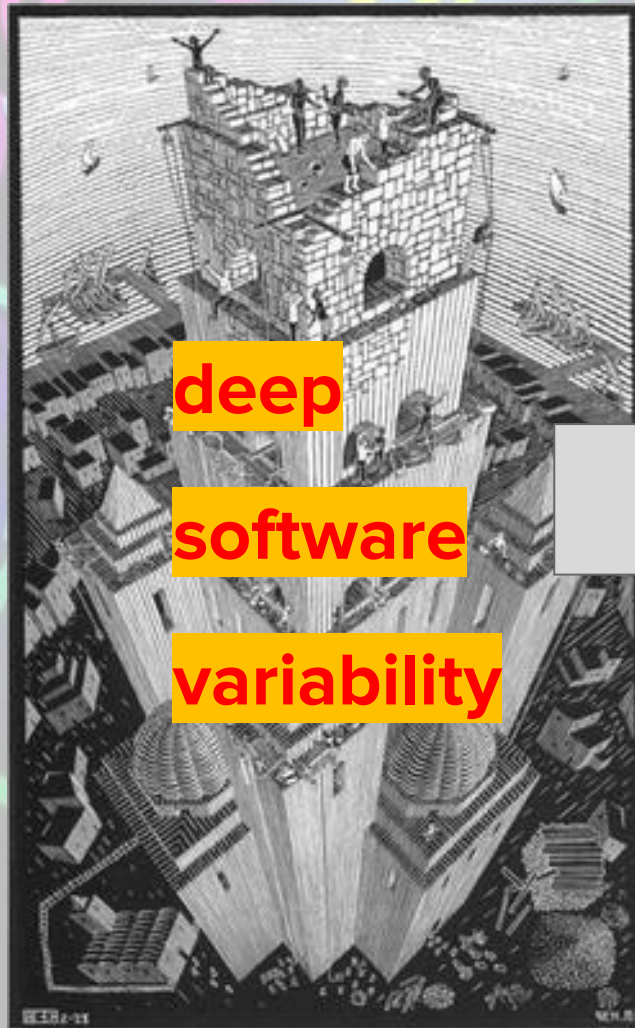
Above all, let us value the need for reproducibility in research. Let us prefer the "discovery" that will be verified or even falsified and encourage collaboration.

**Deeply exploring the variability space cannot be done alone. Collaborative, distributed effort needed.**

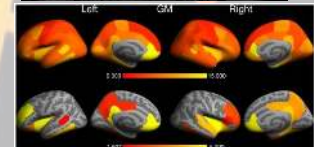
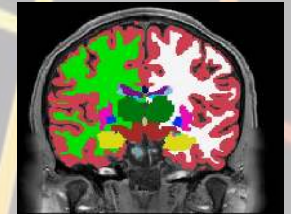
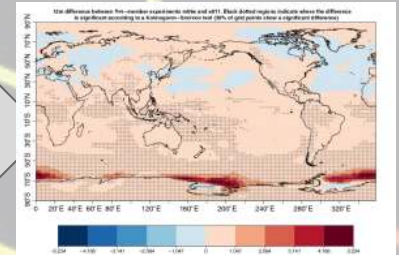
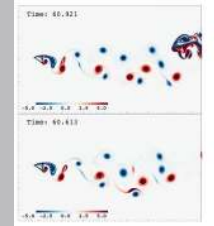
Are we in ivory towers?

I mean: We're studying software... for the sake of improving software engineering (and it's nice!).

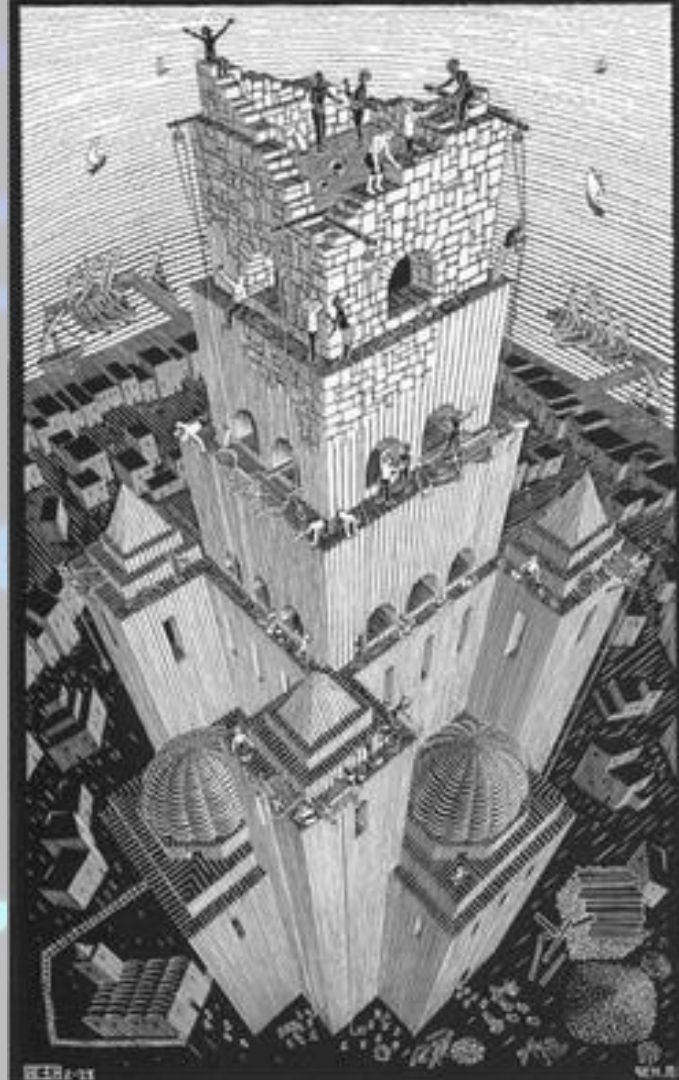
But should we stay?



**(computational)  
science**







**Deep software variability** is...

a **threat** for reproducible research

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

an **opportunity** for replication

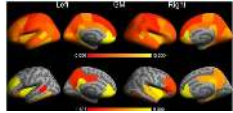
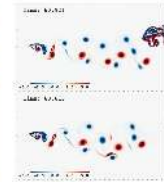
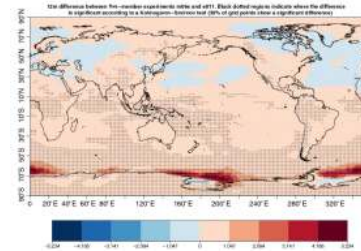
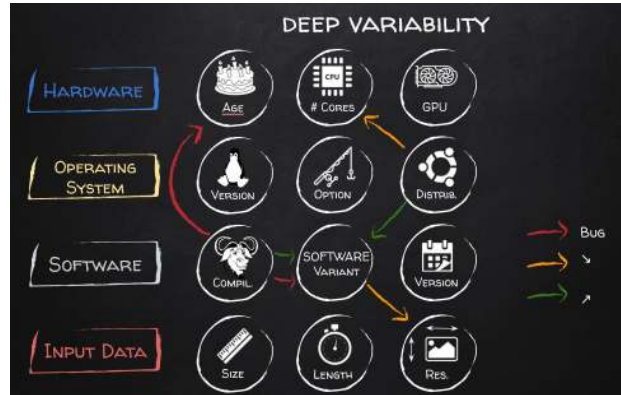
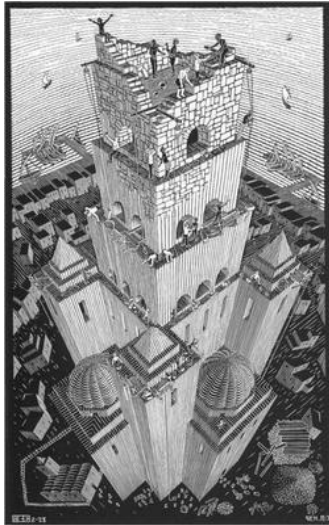
“A study that arrives at the same scientific findings as another study, collecting new data (possibly with different methods) and completing new analyses.”

“A study that refutes some scientific findings of another study, through the collection of new data (possibly with different methods) and completion of new analyses.”

**robustifying and augmenting**

**scientific knowledge**

# Reproducible Science and Deep Software Variability



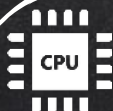
# BACKUP SLIDES

# (BEYOND x264) EMPIRICAL AND FUNDAMENTAL QUESTION

HARDWARE



AGE



# CORES



GPU

HOW DOES DEEP  
SOFTWARE VARIABILITY  
MANIFEST IN THE WILD?

OPERATING  
SYSTEM



VERSION



OPTION



DISTRIB.

SOFTWARE  
SCIENTISTS

SOFTWARE



COMPIL.

VARIANT



VERSION

SHOULD  
OBSERVE THE  
JUNGLE/  
GALAXY!

INPUT DATA



SIZE



LENGTH



RES.

# DEEP SOFTWARE VARIABILITY

## 4 CHALLENGES AND OPPORTUNITIES

Luc Lesoil, Mathieu Acher, Arnaud Blouin, Jean-Marc Jézéquel:

*Deep Software Variability: Towards Handling Cross-Layer Configuration. VaMoS 2021: 10:1-10:8*

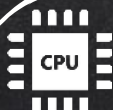
# IDENTIFY THE INFLUENTIAL LAYERS

1

HARDWARE



AGE



# CORES



GPU

OPERATING  
SYSTEM



VERSION



OPTION



DISTRIB.

SOFTWARE



COMPIL.



VARIANT



VERSION

INPUT DATA



SIZE



LENGTH



RES.

PROBLEM  
≠ LAYERS,  
≠ IMPORTANCES ON  
PERFORMANCES

CHALLENGE  
ESTIMATE THEIR EFFECTS

OPPORTUNITY  
LEVERAGE THE USEFUL  
VARIABILITY LAYERS  
& VARIABLES

# TEST & BENCHMARK ENVIRONMENTS

HARDWARE



PROBLEM  
COMBINATORIAL EXPLOSION AND COST

OPERATING SYSTEM



CHALLENGE  
BUILD A  
REPRESENTATIVE, CHEAP  
SET OF ENVIRONMENTS

SOFTWARE



INPUT DATA



OPPORTUNITY  
DIMENSIONALITY REDUCTION

# TRANSFER PERFORMANCES ACROSS ENVIRONMENTS

3



(A)




DELL LATITUDE  
7400




20.04

ubuntu

(B)



RASPBERRY PI  
4 MODEL B



10.4

## PROBLEM

OPTIONS' IMPORTANCES  
CHANGE WITH ENVIRONMENTS

## CHALLENGE

TRANSFER PERFORMANCES  
ACROSS ENVIRONMENTS

## OPPORTUNITY

REDUCE COST OF MEASURE



# CROSS-LAYER TUNING

- BUG
- PERF. ↓
- PERF. ↑

HARDWARE

OPERATING SYSTEM

SOFTWARE

INPUT DATA



PROBLEM  
(NEGATIVE) INTERACTIONS  
OF LAYERS

CHALLENGE  
FIND & FIX VALUES TO IMPROVE  
PERFORMANCES

OPPORTUNITY  
SPECIALIZE THE ENVIRONMENT  
FOR A USE CASE

# CHALLENGES FOR DEEP SOFTWARE VARIABILITY

IDENTIFY THE INFLUENTIAL LAYERS

TEST & BENCHMARK ENVIRONMENTS

TRANSFER PERFORMANCES ACROSS ENVIRONMENTS

CROSS-LAYER TUNING

# Wrap-up

Deep software variability is a thing...

miracle and smooth fit of variability layers?

or subtle interactions among layers?

software scientists should **systematically study** the phenomenon

Many challenges and opportunities

**cross-layer tuning** at affordable cost

configuration **knowledge** that generalizes to any context and usage

Dimensionality reduction and **transfer** learning



Variability layers (in time and in space)

# Impacts of deep software variability

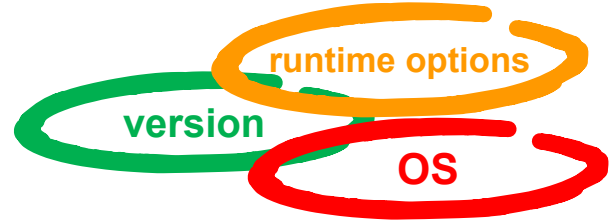
Users/developers/scientists: missed opportunities due to (accidental) variability/complexity

Deep software variability can be seen as a **threat** to knowledge/validity

**Claim: Deep software variability is a threat to scientific, software-based experiments**

Example #1: in the results of neuroimaging studies

- applications of different analysis pipelines (Krefting et al. 2011)
- alterations in software version (Glatard et al. 2015)
- changes in operating system (Gronenschild2012 et al.)



have both shown to cause variation (up to the point it can change the conclusions).

Example #2: on a modest scale, our ICPE 2020 (Pereira et al.) showed that a variation in inputs could change the conclusion about the effectiveness of a sampling strategy for the same software system.

Example #3: I'm reading Zakaria Ournani et al. "Taming Energy Consumption Variations In Systems Benchmarking" ICPE 2020 as an excellent inquiry to control deep variability factors

Example #4: Similar observations have been made in the machine learning community (Henderson et al. 2018)

# Impacts of deep software variability

Users/developers/scientists: missed opportunities due to (accidental) complexity

Deep software variability can be seen as a **threat** to knowledge/validity

Claim: Deep software variability is a threat to scientific, software-based experiments

I propose an exercise that might be slightly disturbing:

**Does deep software variability affect (your|a known) previous scientific, software-based studies?**

(remark: it's mainly how we've investigated deep software variability so far... either as a threat to our own experiments or as a threat identified in papers)

Applications of different analysis pipelines, alterations in software version, and even changes in operating system have both shown to cause variation in the results of a neuroimaging study. Example:

### Reproducibility of neuroimaging analyses across operating systems

Tristan Glatard, Lindsay B. Lewis, Rafael Ferreira da Silva, Roza Adalat, Natacha Beck, Claude Lepage, Pierre Rioux, Marc-Etienne Roussou, Tarek Sherif, Ewa Doolman, Najmeh Khalili-Mahani and Alan C. Evans

Neuroimaging pipelines are known to generate different results depending on the computing platform where they are compiled and executed. We quantify these differences for brain tissue classification, fMRI analysis, and cortical thickness (CT) extraction, using three of the main neuroimaging packages (FSL, Freesurfer and CIVET) and different versions of GNU/Linux. We also identify some causes of these differences using library and system call interception. We find that these packages use mathematical functions based on single-precision floating-point arithmetic whose implementations in operating systems continue to evolve. 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 subcortical tissue classification, fMRI analysis, and cortical thickness extraction. With FSL, most Dice coefficients between subcortical

**DEEP SOFTWARE  
VARIABILITY**

Deep software variability is a threat to scientific, software-based experiments

torch.manual.seed(3407) is all you need: On the influence of random seeds in deep learning architectures on computer vision

David Picard  
LIGM, École des Ponts, 77455 Marne la vallée, France

DAVID.PICARD@ENPC.FR

**DEEP SOFTWARE  
VARIABILITY**

#### Abstract

In this paper I investigate the effect of random seed selection on the accuracy when using popular deep learning architectures for computer vision. I scan a large amount of seeds (up to  $10^4$ ) on CIFAR 10 and I also scan fewer seeds on Imagenet using pre-trained models to investigate large scale datasets. The conclusions are that even if the variance is not very large, it is surprisingly easy to find an outlier that performs much better or much worse than the average.

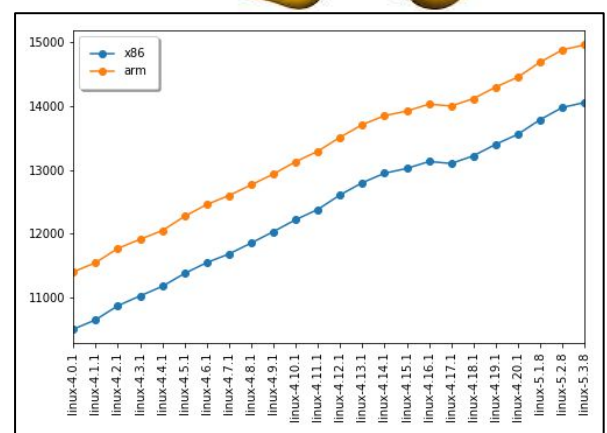
DEEP QUESTIONS?



android



15,000+ options



TRISTATE	61.63	3 <sup>9000</sup>
BOOL	36.40	
INT	1.54	2 <sup>6000</sup>
STRING	0.29	
HEX	0.14	

Linux 5.2.8, arm  
(% of types' options)

≈ 10<sup>6000</sup> variants

(without constraints)





## REFERENCES

- [1] 2021. Hail, software! *Nature Computational Science* 1, 2 (01 Feb 2021), 89–89. <https://doi.org/10.1038/s43588-021-00037-8>
- [2] Marc Andreessen. 2011. Why software is eating the world. *Wall Street Journal* 20, 2011 (2011), C2.
- [3] Lorena A Barba. 2018. Terminologies for reproducible research. *arXiv preprint arXiv:1802.03311* (2018).
- [4] Rotem Botvinik-Nezer, Felix Holzmeister, Colin F Camerer, Anna Dreber, Juergen Huber, Magnus Johannesson, Michael Kirchler, Roni Iwanir, Jeanette A Mumford, R Alison Adcock, et al. 2020. Variability in the analysis of a single neuroimaging dataset by many teams. *Nature* 582, 7810 (2020), 84–88.
- [5] Xavier Bouthillier, César Laurent, and Pascal Vincent. 2019. Unreproducible research is reproducible. In *International Conference on Machine Learning*. PMLR, 725–734.
- [6] Joshua Carp. 2012. On the plurality of (methodological) worlds: estimating the analytic flexibility of fMRI experiments. *Frontiers in neuroscience* 6 (2012), 149.
- [7] Roberto Di Cosmo and Stefano Zacchiroli. 2017. Software heritage: Why and how to preserve software source code. In *iPRES 2017-14th International Conference on Digital Preservation*. 1–10.
- [8] Steve M Easterbrook and Timothy C Johns. 2009. Engineering the software for understanding climate change. *Computing in science & engineering* 11, 6 (2009), 65–74.
- [9] Tristan Glatard, Lindsay B Lewis, Rafael Ferreira da Silva, Reza Adalat, Natacha Beck, Claude Lepage, Pierre Rioux, Marc-Etienne Rousseau, Tarek Sherif, Ewa Deelman, et al. 2015. Reproducibility of neuroimaging analyses across operating systems. *Frontiers in neuroinformatics* 9 (2015), 12.
- [10] Ed HBM Gronenschild, Petra Habets, Heidi IL Jacobs, Ron Mengelers, Nico Rozendaal, Jim Van Os, and Machteld Marcelis. 2012. The effects of FreeSurfer version, workstation type, and Macintosh operating system version on anatomical volume and cortical thickness measurements. *PLoS one* 7, 6 (2012), e38234.
- [11] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. 2018. Deep reinforcement learning that matters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.
- [12] Ruben Heradio, David Fernandez-Amoros, José A Galindo, David Benavides, and Don Batory. 2022. Uniform and scalable sampling of highly configurable systems. *Empirical Software Engineering* 27, 2 (2022), 1–34.
- [13] <https://www.wcrp.climate.org/>. [n.d.]. World climat research program (WCRP).
- [14] Pooyan Jamshidi, Norbert Siegmund, Miguel Velez, Akshay Patel, and Yuvraj Agarwal. 2017. Transfer Learning for Performance Modeling of Configurable Systems: An Exploratory Analysis. In *IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE Press, 497–508.
- [15] Pooyan Jamshidi, Miguel Velez, Christian Kästner, and Norbert Siegmund. 2018. Learning to Sample: Exploiting Similarities Across Environments to Learn Performance Models for Configurable Systems. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM, 71–82.
- [16] Christian Kaltenecker, Alexander Grebhahn, Norbert Siegmund, and Sven Apel. 2020. The Interplay of Sampling and Machine Learning for Software Performance Prediction. *IEEE Softw.* 37, 4 (2020), 58–66. <https://doi.org/10.1109/MS.2020.2987024>
- [17] Christopher G Knight, Sylvia HE Knight, Neil Massey, Tolu Aina, Carl Christensen, Dave J Frame, Jamie A Kettleborough, Andrew Martin, Stephen Pascoe, Ben Sanderson, et al. 2007. Association of parameter, software, and hardware variation with large-scale behavior across 57,000 climate models. *Proceedings of the National Academy of Sciences* 104, 30 (2007), 12259–12264.
- [18] Dagmar Krefting, Michael Scheel, Alina Freing, Svenja Specovius, Friedemann Paul, and Alexander Brandt. 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)*.
- [19] R. Krishna, V. Nair, P. Jamshidi, and T. Menzies. 2020. Whence to Learn? Transferring Knowledge in Configurable Systems using BEETLE. *IEEE Transactions on Software Engineering* (2020), 1–1.
- [20] Dorian Leroy, June Sallou, Johann Bourcier, and Benoit Combemale. 2021. When Scientific Software Meets Software Engineering. *Computer* (2021), 1–11. <https://hal.inria.fr/hal-03318348>
- [21] Luc Lesoil, Mathieu Acher, Arnaud Blouin, and Jean-Marc Jézéquel. 2021. The Interaction between Inputs and Configurations fed to Software Systems: an Empirical Study. *CoRR abs/2112.07279* (2021). [arXiv:2112.07279](https://arxiv.org/abs/2112.07279) <https://arxiv.org/abs/2112.07279>
- [22] Luc Lesoil, Mathieu Acher, Arnaud Blouin, and Jean-Marc Jézéquel. 2021. Deep Software Variability: Towards Handling Cross-Layer Configuration. In *VaMoS 2021 - 15th International Working Conference on Variability Modelling of Software-Intensive Systems*. Krems / Virtual, Austria, 1–8. <https://hal.inria.fr/hal-03084276>
- [23] Luc Lesoil, Mathieu Acher, Xhevahire Tërnavá, Arnaud Blouin, and Jean-Marc Jézéquel. 2021. The Interplay of Compile-time and Run-time Options for Performance Prediction. In *SPLC 2021 - 25th ACM International Systems and Software Product Line Conference - Volume A*. ACM, Leicester, United Kingdom, 1–12. <https://doi.org/10.1145/3461001.3471149>
- [24] Luc Lesoil, Hugo Martin, Mathieu Acher, Arnaud Blouin, and Jean-Marc Jézéquel. 2022. Transferring Performance between Distinct Configurable Systems : A Case Study. In *VaMoS '22: 16th International Working Conference on Variability Modelling of Software-Intensive Systems, Florence, Italy, February 23 - 25, 2022*, Paolo Arcaini, Xavier Devroey, and Alessandro Fantechi (Eds.). ACM, 10:1–10:6. <https://doi.org/10.1145/3510466.3510486>
- [25] Hugo Martin, Mathieu Acher, Juliana Alves Pereira, Luc Lesoil, Jean-Marc Jézéquel, and Djamel Eddine Khelladi. 2021. Transfer Learning Across Variants and Versions: The Case of Linux Kernel Size. *IEEE Transactions on Software Engineering* (2021),

- 1–17. <https://hal.inria.fr/hal-03358817>
- [26] François Massonnet, Martin Ménégot, Mario Acosta, Xavier Yepes-Arbós, Eleftheria Exarchou, and Francisco J Doblas-Reyes. 2020. Replicability of the EC-Earth3 Earth system model under a change in computing environment. *Geoscientific Model Development* 13, 3 (2020), 1165–1178.
- [27] Olivier Mesnard and Lorena A Barba. 2017. Reproducible and replicable computational fluid dynamics: it’s harder than you think. *Computing in Science & Engineering* 19, 4 (2017), 44–55.
- [28] S. Mühlbauer, S. Apel, and N. Siegmund. 2020. Identifying Software Performance Changes Across Variants and Versions. In *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 611–622.
- [29] Juliana Alves Pereira, Mathieu Acher, Hugo Martin, and Jean-Marc Jézéquel. 2020. Sampling Effect on Performance Prediction of Configurable Systems: A Case Study. In *ICPE ’20: ACM/SPEC International Conference on Performance Engineering, Edmonton, AB, Canada, April 20-24, 2020*, José Nelson Amaral, Anne Koziolok, Catia Trubiani, and Alexandru Iosup (Eds.). ACM, 277–288. <https://doi.org/10.1145/3358960.3379137>
- [30] Juliana Alves Pereira, Mathieu Acher, Hugo Martin, Jean-Marc Jézéquel, Goetz Botterweck, and Anthony Ventresque. 2021. Learning software configuration spaces: A systematic literature review. *J. Syst. Softw.* 182 (2021), 111044. <https://doi.org/10.1016/j.jss.2021.111044>
- [31] Quentin Plazar, Mathieu Acher, Gilles Perrouin, Xavier Devroey, and Maxime Cordy. 2019. Uniform Sampling of SAT Solutions for Configurable Systems: Are We There Yet?. In *12th IEEE Conference on Software Testing, Validation and Verification, ICST 2019, Xi’an, China, April 22-27, 2019*. IEEE, 240–251. <https://doi.org/10.1109/ICST.2019.00032>
- [32] Georges Aaron Randrianaina, Djamel Eddine Khelladi, Olivier Zendra, and Mathieu Acher. 2022. Towards Incremental Build of Software Configurations. In *ICSE-NIER 2022 - 44th International Conference on Software Engineering – New Ideas and Emerging Results*. Pittsburgh, PA, United States, 1–5. <https://doi.org/10.1145/3510455.3512792>
- [33] Georges Aaron Randrianaina, Xhevahire Tërnav, Djamel Eddine Khelladi, and Mathieu Acher. 2022. On the Benefits and Limits of Incremental Build of Software Configurations: An Exploratory Study. In *ICSE 2022 - 44th International Conference on Software Engineering*. Pittsburgh, Pennsylvania / Virtual, United States, 1–12. <https://hal.archives-ouvertes.fr/hal-03547219>
- [34] Nicolas P. Rougier and Konrad Hinsén. 2019. ReScience C: A Journal for Reproducible Replications in Computational Science. In *Reproducible Research in Pattern Recognition*, Bertrand Kerautret, Miguel Colom, Daniel Lopresti, Pascal Monasse, and Hugues Talbot (Eds.). Springer International Publishing, Cham, 150–156.
- [35] Nicolas P. Rougier, Konrad Hinsén, Frédéric Alexandre, Thomas Arildsen, Lorena Barba, Fabien C. Y. Benureau, C. Titus Brown, Pierre De Buyl, Ozan Caglayan, Andrew P. Davison, Marc André Delsuc, Georgios Detorakis, Alexandra K. Diem, Damien Drix, Pierre Enel, Benoît Girard, Olivia Guest, Matt G. Hall, Rafael Neto Henriques, Xavier Hinaut, Kamil S Jaron, Mehdi Khamassi, Almar Klein, Tiina Manninen, Pietro Marchesi, Dan McGlenn, Christoph Metzner, Owen L. Petchey, Hans Ekkehard Plesser, Timothée Poisot, Karthik Ram, Yoav Ram, Etienne Roesch, Cyrille Rossant, Vahid Rostami, Aaron Shifman, Joseph Stachelek, Marcel Stimberg, Frank Stollmeier, Federico Vaggi, Guillaume Viejo, Julien Vitay, Anya Vostinar, Roman Yurchak, and Tiziano Zito. 2017. Sustainable computational science: the ReScience initiative. *PeerJ Computer Science* 3 (Dec. 2017), e142. <https://doi.org/10.7717/peerj-cs.142> 8 pages, 1 figure.
- [36] Mohammed Sayagh, Noureddine Kerzazi, Bram Adams, and Fabio Petrillo. 2018. Software Configuration Engineering in Practice: Interviews, Survey, and Systematic Literature Review. *IEEE Transactions on Software Engineering* (2018).
- [37] Raphael Silberzahn, Eric L Uhlmann, Daniel P Martin, Pasquale Anselmi, Frederik Aust, Eli Awtrey, Štěpán Bahník, Feng Bai, Colin Bannard, Evelina Bonnier, et al. 2018. Many analysts, one data set: Making transparent how variations in analytic choices affect results. *Advances in Methods and Practices in Psychological Science* 1, 3 (2018), 337–356.
- [38] Sara Steegen, Francis Tuerlinckx, Andrew Gelman, and Wolf Vanpaemel. 2016. Increasing transparency through a multiverse analysis. *Perspectives on Psychological Science* 11, 5 (2016), 702–712.
- [39] Pavel Valov, Jean-Christophe Petkovich, Jianmei Guo, Sebastian Fischmeister, and Krzysztof Czarnecki. 2017. Transferring Performance Prediction Models Across Different Hardware Platforms. In *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering, ICPE 2017, L’Aquila, Italy, April 22-26, 2017*. 39–50. <https://doi.org/10.1145/3030207.3030216>
- [40] Mahsa Varshosaz, Mustafa Al-Hajjaji, Thomas Thüm, Tobias Runge, Mohammad Reza Mousavi, and Ina Schaefer. 2018. A classification of product sampling for software product lines. In *Proceedings of the 22nd International Conference on Systems and Software Product Line - Volume 1, SPLC 2018, Gothenburg, Sweden, September 10-14, 2018*. 1–13. <https://doi.org/10.1145/3233027.3233035>
- [41] Max Weber, Sven Apel, and Norbert Siegmund. 2021. White-Box Performance-Influence models: A profiling and learning approach. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 1059–1071.
- [42] Greg Wilson, Dhavide A Aruliah, C Titus Brown, Neil P Chue Hong, Matt Davis, Richard T Guy, Steven HD Haddock, Kathryn D Huff, Ian M Mitchell, Mark D Plumbley, et al. 2014. Best practices for scientific computing. *PLoS biology* 12, 1 (2014), e1001745.