



**HAL**  
open science

## Computational Reproducibility

Sorina Camarasu-Pop

► **To cite this version:**

Sorina Camarasu-Pop. Computational Reproducibility. 3rd cycle. 12th SLEIGHT Science Event, Saint Etienne (FR), France. 2024. hal-04649287

**HAL Id: hal-04649287**

**<https://hal.science/hal-04649287v1>**

Submitted on 16 Jul 2024

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Computational Reproducibility

## An Overview Illustrated with Examples from the Medical Imaging Community

Sorina Camarasu-Pop

CREATIS, CNRS (UMR 5220), INSERM (U1294), INSA Lyon,  
Université de Lyon, France

12th SLEIGHT Science Event  
10/11/2024

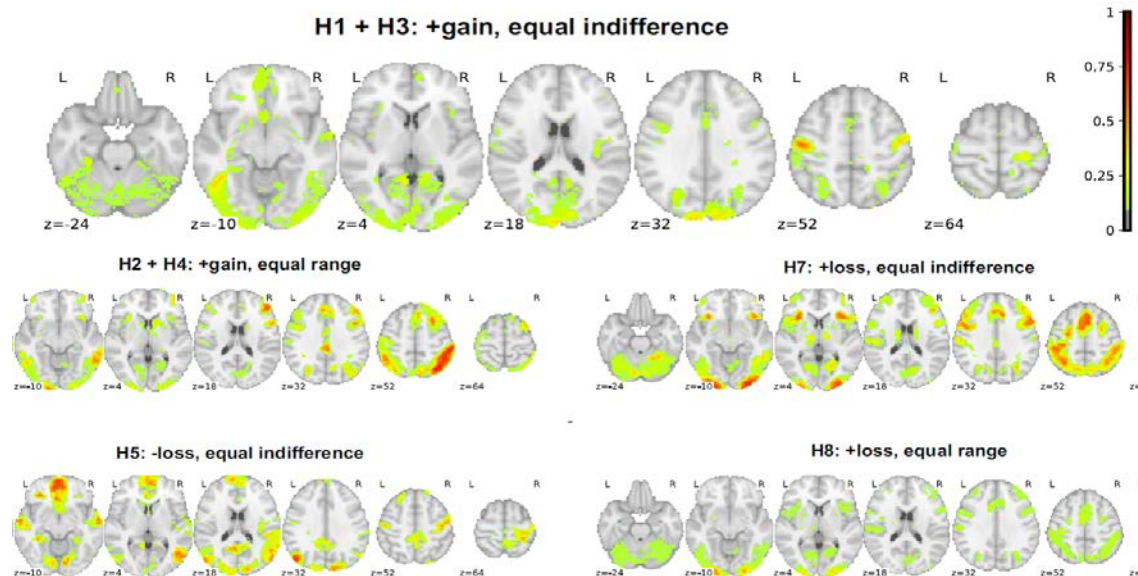
# Reproducibility Issues in Neuroimaging

## Setup

- **1** dataset
- **70** teams
- **9** hypotheses

## Findings

- Variability of results
- Analytical flexibility
- Optimism bias



[R. Botvinik-Nezer et al](#), "Variability in the analysis of a single neuroimaging dataset by many teams" *Nature* 2020

# Outline

- I. What is reproducibility?
- II. Computational reproducibility
- III. Larger overview

# Definitions

- Spectrum of concerns/terms
  - Terminologies [[Barba, 2018](#)]

		Data	
		Same	Different
Analysis	Same	Reproducible	Replicable
	Different	Robust	Generalisable

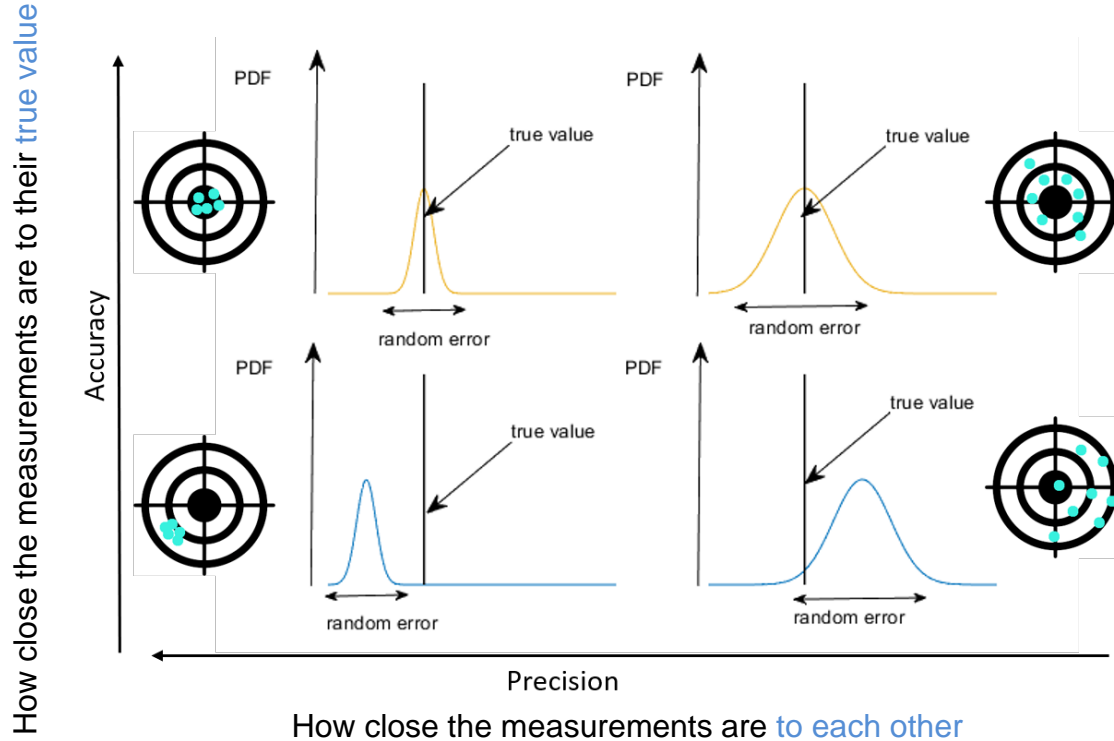
- Reproducible research
  - Authors provide all the necessary data and the computer codes to run the analysis again, re-creating the results

# What does « same result » mean?

- A result in particular, e.g. binary file
  - Bitwise reproducibility: checksum
  - Statistical reproducibility:  $p < 0,05$
  - Other specific metrics
- A published study, e.g. hypothesis testing
  - Figures, conclusions



# Accuracy versus precision



# Reproducibility categories

- Multiples categories
- [V Stodden and S Leonelli]
  - Computational reproducibility
  - Empirical reproducibility
  - Statistical reproducibility

## Exact reproducibility

Reproduction of strictly identical results as those of a previously published paper.

- *Example: reproducing classification accuracies using the exact same code, data and random seeds*

## Statistical reproducibility

Reproduction of the results of a study under statistically equivalent conditions. The results should be statistically compatible but not identical.

- *Example: reproducing a study using another sample of patients drawn from the same population or from a population with the same characteristics*

## Conceptual reproducibility

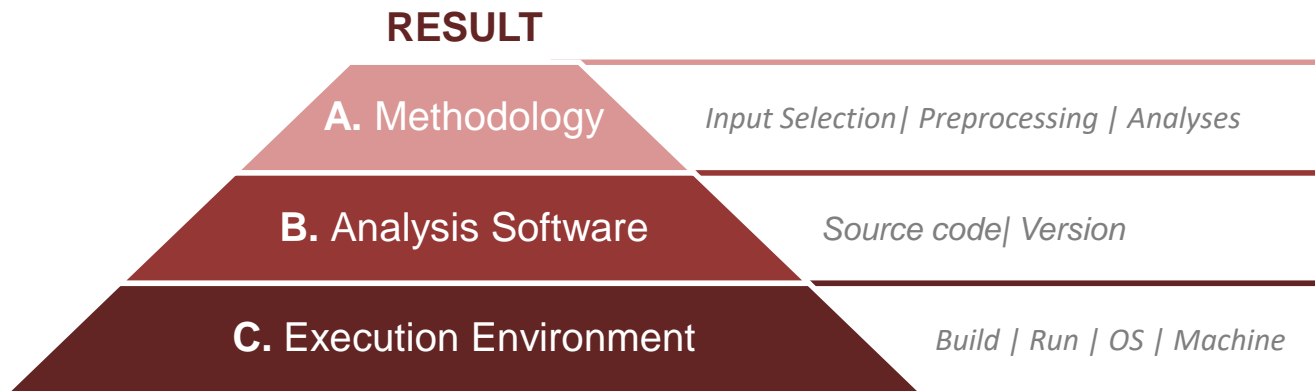
Reproduction of the results of a study under conceptually equivalent conditions. This includes generalizability studies.

- *Example: reproducing a study using a different sample of patients, affected by the same disorder, but with different socio-demographic characteristics and from different hospitals*



# Sources of Variability in Medical Imaging

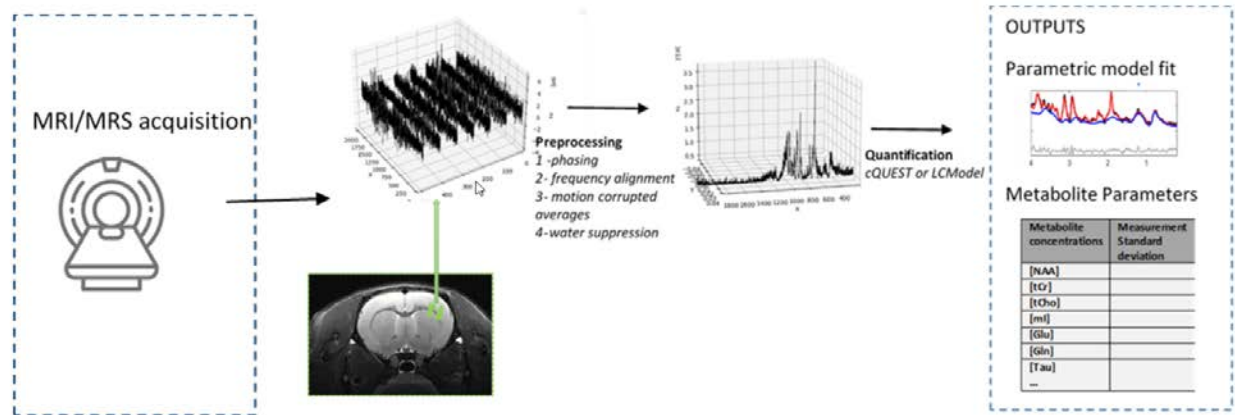
From a researcher's point of view



# Sources of Variability in Medical Imaging

## A. Research Methodology

Typical computation work  
in MR spectroscopy



### Data Selection

- Subjects, Measurement Devices,
- SNR

### Data Preprocessing

- Registration, filtering, -

### Data Analysis

- Model, constraints, -
- Statistics, machine learning, -

### Conclusions drawn

- Tables, graphs, -

# Sources of Variability in Medical Imaging

## B. Analysis Software

Distinct **implementations** / Distinct **versions**

<i>Research Domain</i>	<i>Available Software filling the same needs</i>				
<i>Functional MRI</i>					
<i>MR Spectroscopy</i>	 FSL- MRS				

# Sources of Variability in Medical Imaging

## C. The Execution Environment

### Software Building

- Static Libraries
- Compiler / Options



### Operating System

- Dynamic Libraries
- OS kernel



### Software Run

- Random Number Generation
- Numerical instabilities
- Parallel Computing

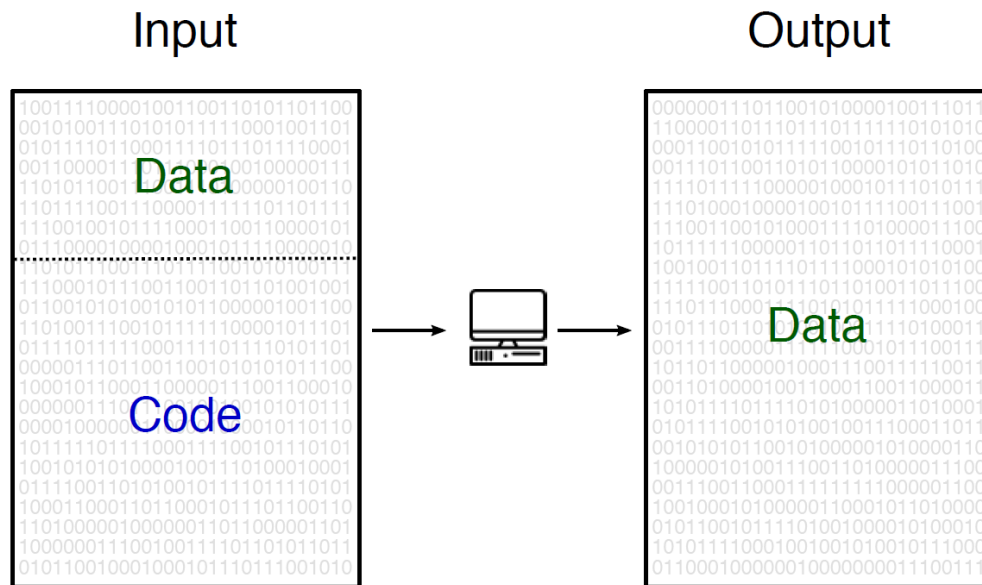
### Hardware

- Architectures
- Compounds

# Outline

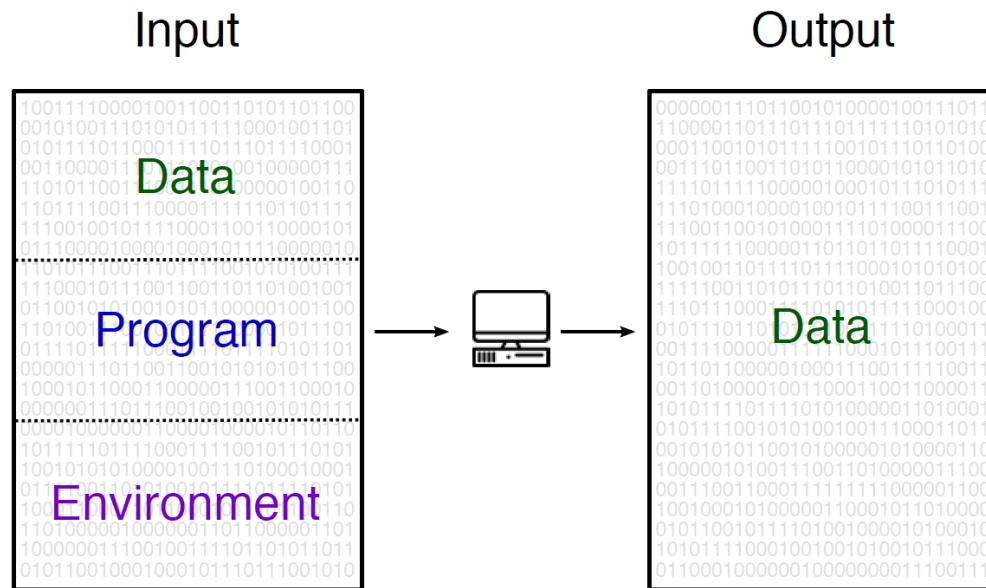
- I. What is reproducibility?
- II. Computational reproducibility**
- III. Larger overview

# What is a « computation »?



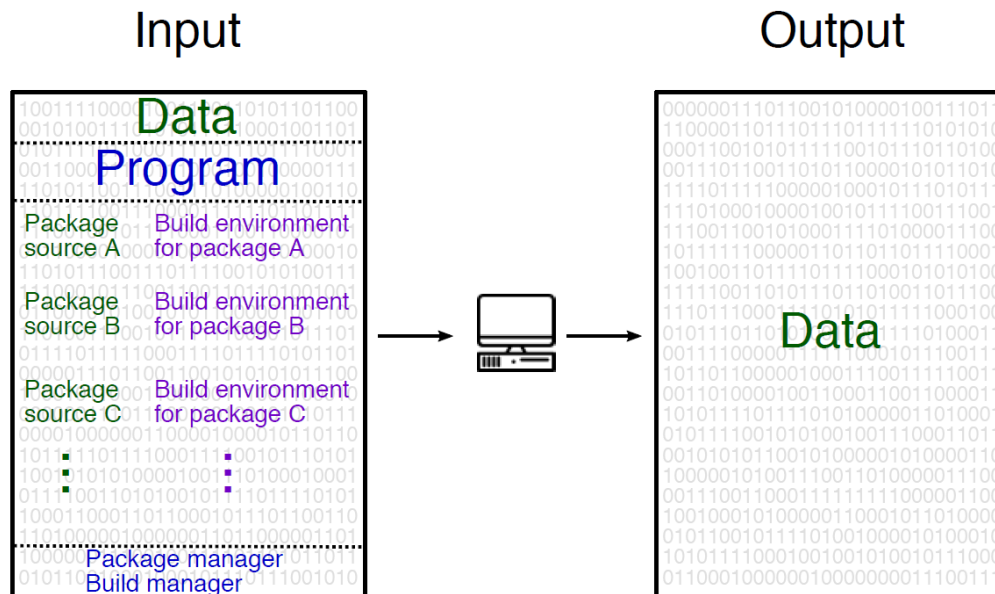
Credits: Konrad HINSEN

# What is a « computation »?



Credits: Konrad HINSEN

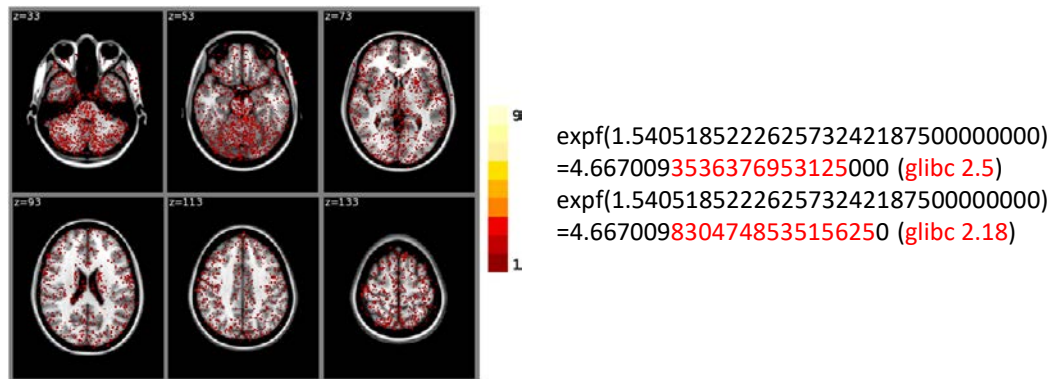
# What is a « computation »?



Credits: Konrad HINSEN



# Computational reproducibility



Same FSL version (5.0.6) and different versions of GNU/Linux

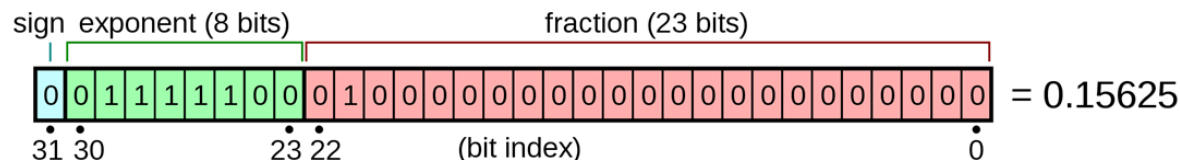
Sum of binarized differences between cortical tissue classifications obtained on cluster A (CentOS) and cluster B (Fedora) (FSL FAST, build 1,  $n = 150$  subjects). Credits: Tristan Glatard, <https://www.frontiersin.org/articles/10.3389/fninf.2015.00012/full>

## Main issues

- Numerical instability due to floating point arithmetic
- Software variability: dependencies and their evolution in time

# Floating point arithmetic

- Approximate real numbers within a limited precision



=> rounding errors

```

1 import numpy as np
2
3 # Large number
4 a = 1e16
5
6 # Slightly different large number
7 b = 1e16 + 1
8
9 # Expected difference
10 expected_difference = 1
11
12 # Actual difference due to floating-point arithmetic
13 actual_difference = b - a
14
15 print(f"Expected Difference: {expected_difference}")
16 print(f"Actual Difference: {actual_difference}")

```

```

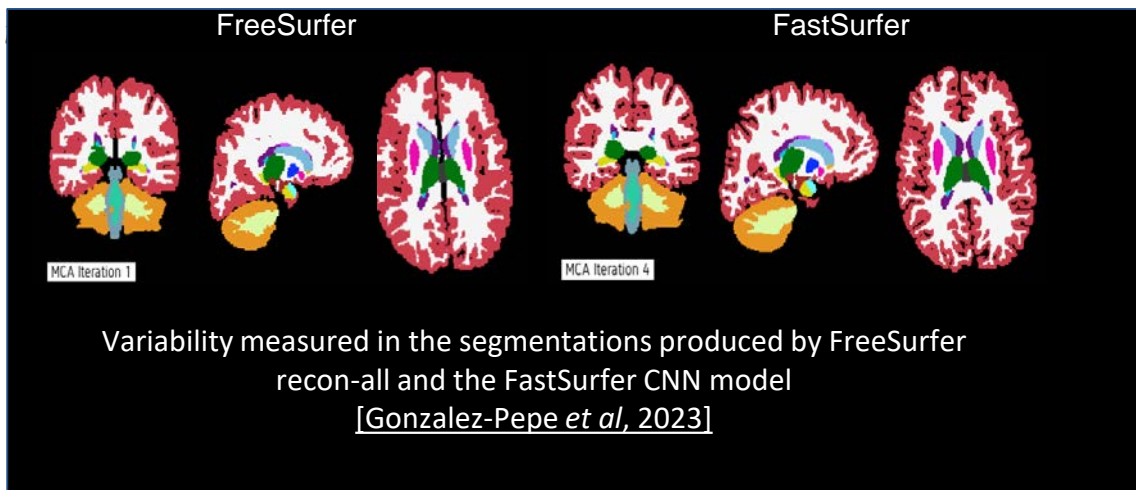
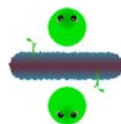
Expected difference : 1
Actual difference : 0.0

```

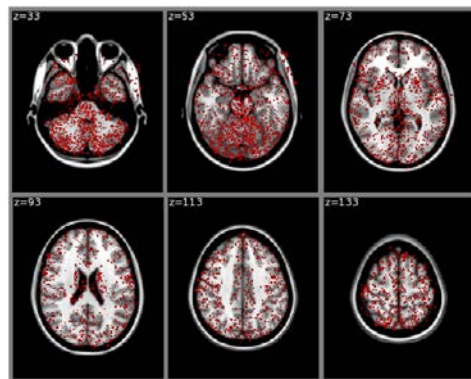
- Hardware and compiler optimizations

# Numerical instability and MCA

- Floating point arithmetic
  - Approximate real numbers within a limited precision => rounding errors
- Monte Carlo Arithmetic (MCA)
  - Noise injection into in floating-point operations:  $inexact(x) = x + 2^{e_x - t} \xi$
- Random Rounding (RR)
  - Perturbs the function output:
- MCA Implementations
  - [Verificarlo](#)
  - [Fuzzy Github repository](#)



# Computational reproducibility (again!)



$\exp(1.540518522262573242187500000000)$   
=4.6670093536376953125000 (glibc 2.5)  
 $\exp(1.540518522262573242187500000000)$   
=4.6670098304748535156250 (glibc 2.18)

## Main issues

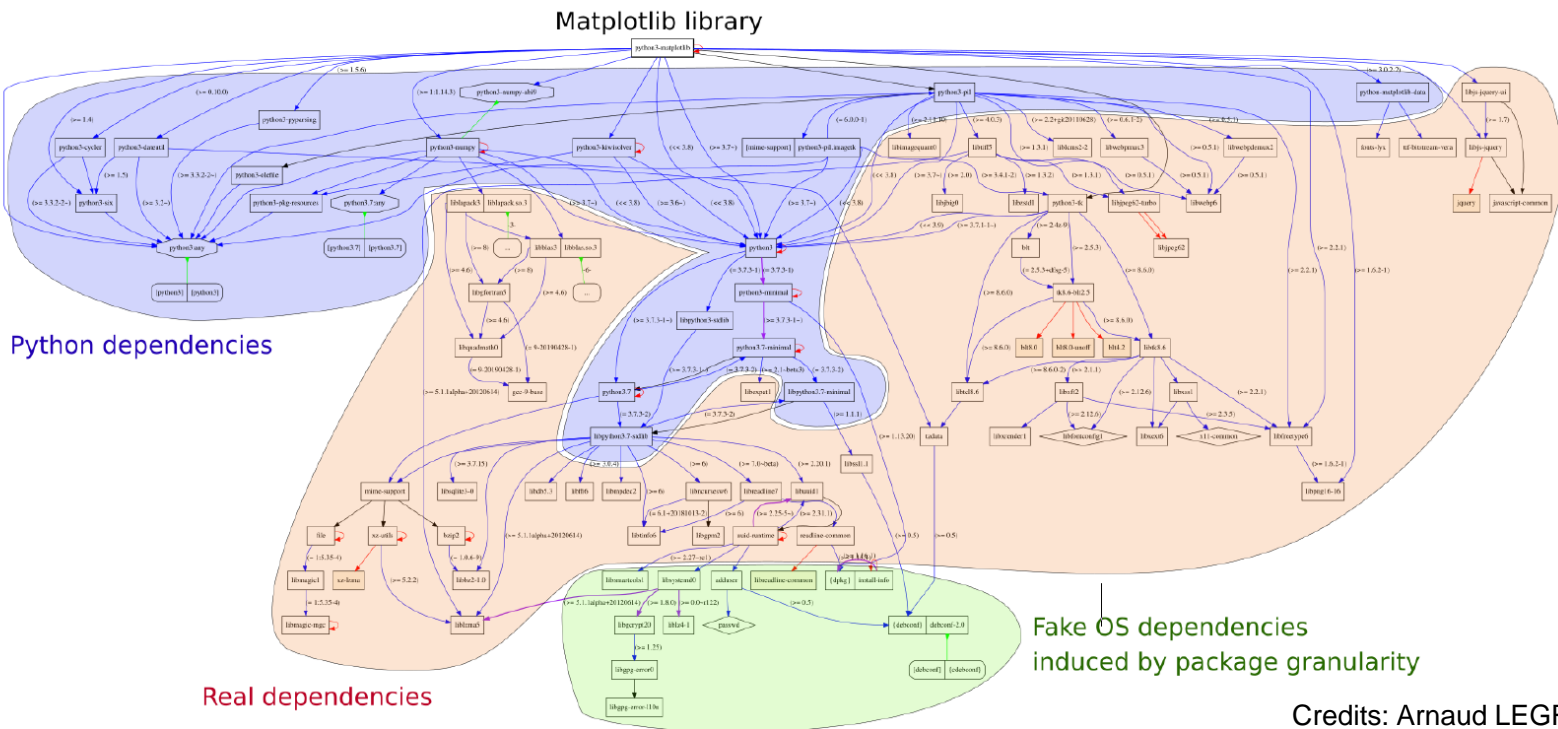
- Numerical instability due to floating point arithmetic
- **Software variability: dependencies and their evolution in time**

Same FSL version (5.0.6) and different versions of GNU/Linux

Sum of binarized differences between cortical tissue classifications obtained on cluster A (CentOS) and cluster B (Fedora) (FSL FAST, build 1,  $n = 150$  subjects). Credits: Tristan Glatard, <https://www.frontiersin.org/articles/10.3389/fninf.2015.00012/full>

# Complex dependencies

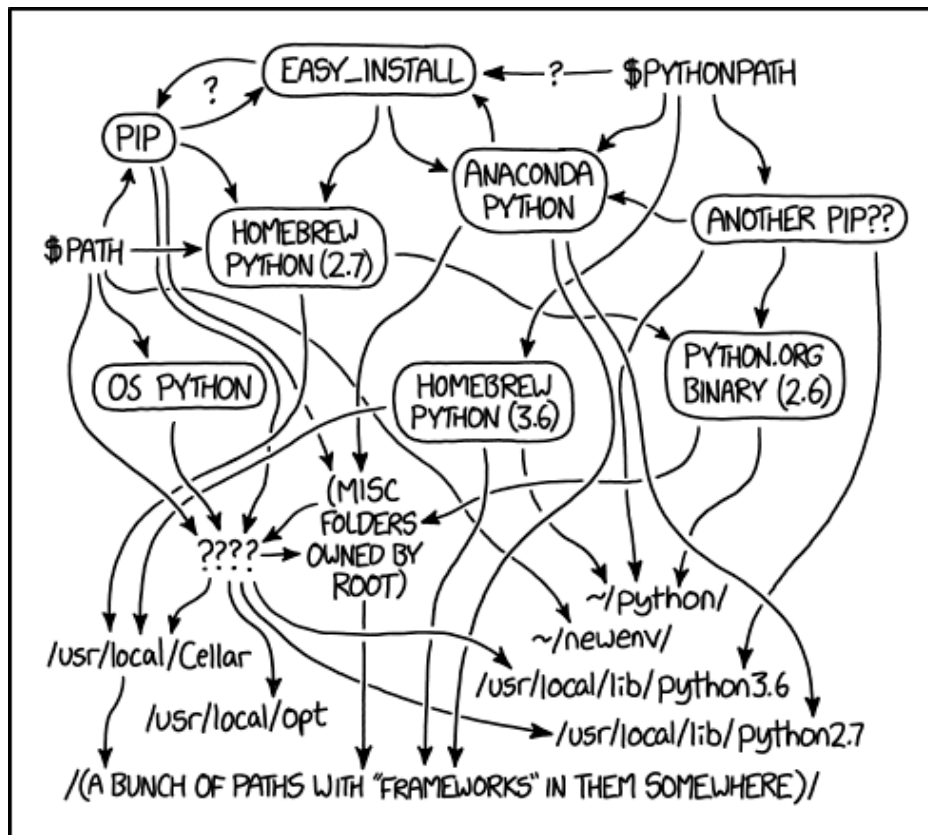
apt show python3-matplotlib



Credits: Arnaud LEGRAND

Environment complexity

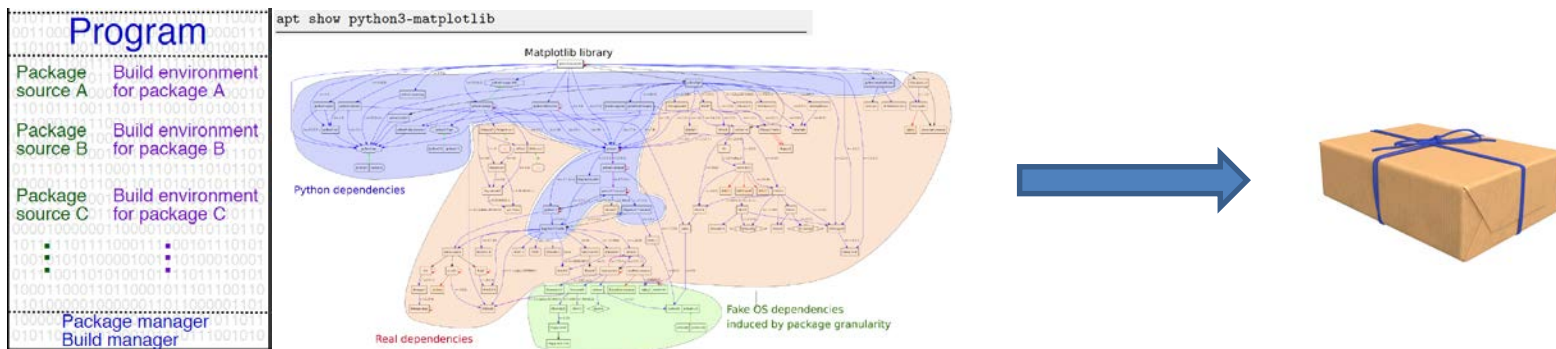
Source: <https://xkcd.com/1987>





MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

# Containerization

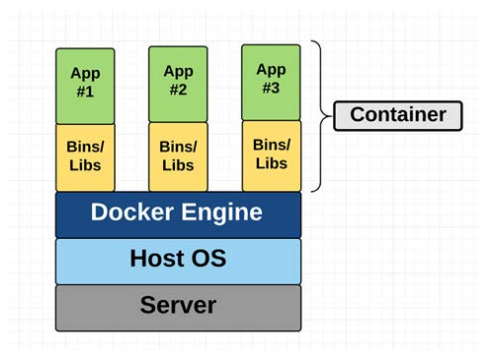
- What if we could package an application and its dependencies



- Containers  
  - Do that 😊

# What are containers?

- A container = an entire runtime environment
  - An application + all its dependencies, libraries and other binaries, and configuration files needed to run it, bundled into one package
- Containers and Virtual Machines (VMs) are similar in their goals
  - Isolate an application and its dependencies into a self-contained unit that can run anywhere





# More on containers

- Dockerfiles

- Image construction « recipe »

*# Use an official centos as a parent image*

from centos:latest (attention, “latest” not a good idea for reproducibility)

*# Install any needed packages*

RUN yum install -y epel-release git gcc make

RUN git clone git://git.creatis.insa-lyon.fr/demoSorina

*# Set the working directory to /demoSorina*

WORKDIR /demoSorina

RUN make

- Dockerhub : image sharing

- <https://hub.docker.com>

- Issues




- Containers can become black-boxes (lacking transparency)
- May be difficult to update/re-build

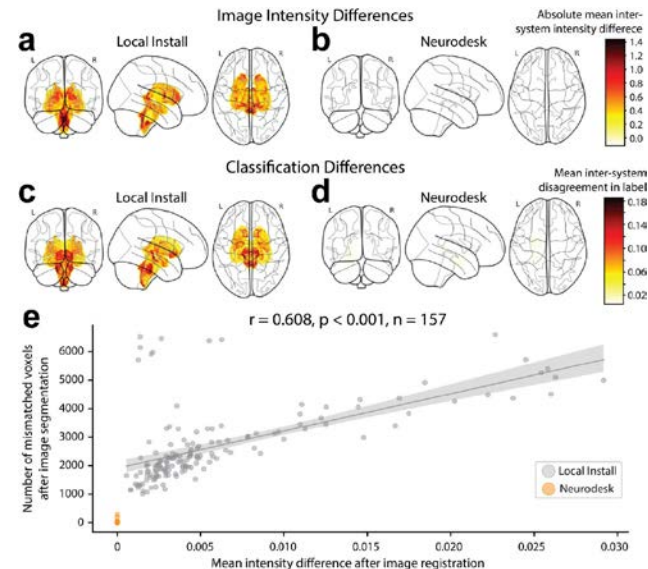
# Guix

- Functional package manager
- Allows to build reproducible computational environments
- Captures the whole computational environment, controls the complete recursive stack and is able to redeploy anytime
  - <https://guix.gnu.org>
  - Scheme language
  - <https://www.nature.com/articles/s41597-022-01720-9>
- Guix packages are defined in modules exportable as containers



# Computational reproducibility

- Main causes
  - Software dependencies and their evolution over time
  - Numerical instability due to floating point arithmetic
- Containerization  
  - Package and run an application and its dependencies
- Guix 
  - Functional package manager
  - Reproducible computational environments

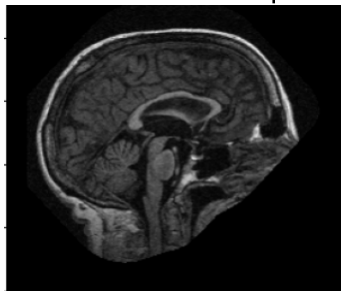


What about **hardware heterogeneity**?

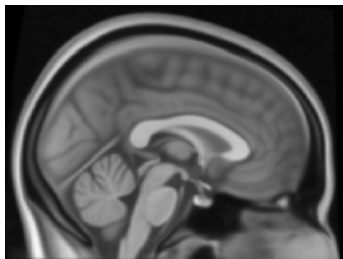
[A.I. Renton et al](#), *Neurodesk*  
*Nature* 2024

# The Impact Of Hardware Variability

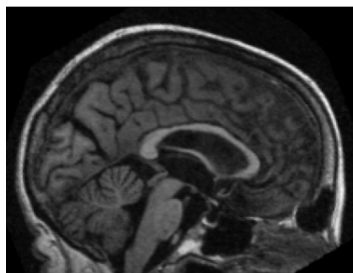
- The Impact Of Hardware Variability On Applications Packaged With Docker And Guix: A Case Study In Neuroimaging
  - [ACM REP'24](#) Best Paper award 😊
  - <https://hal.science/hal-04480308v2>
- Objectives
  - Evaluate the **impact of hardware variability**
  - Compare and correlate hardware variability to
    - **Software variability** encountered in different software packages
    - **Numerical variability** resulting from MCA RR



original image



reference



registered image

# FSL FLIRT

- FMRIB Software Library ([FSL](#))
  - Library of analysis tools for FMRI, MRI and diffusion brain imaging data
- FLIRT: FMRIB's Linear Image Registration Tool
  - Affine brain registration: align a brain scan with another one through rotation, translation, scaling and shearing
- FLIRT outputs
  - Registered brain image in NifTI format (.nii.gz)
  - Transformation matrix in text format (.mat)

```
1.129633431    0.009161432163   -0.002279976965   -2.097511242
-0.004720817456  1.028899087     0.3437343964     -50.46994368
0.01111236612  -0.413128704     1.142416095      -28.62331337
0              0                0                1
```

Example of transformation matrix (.mat file)



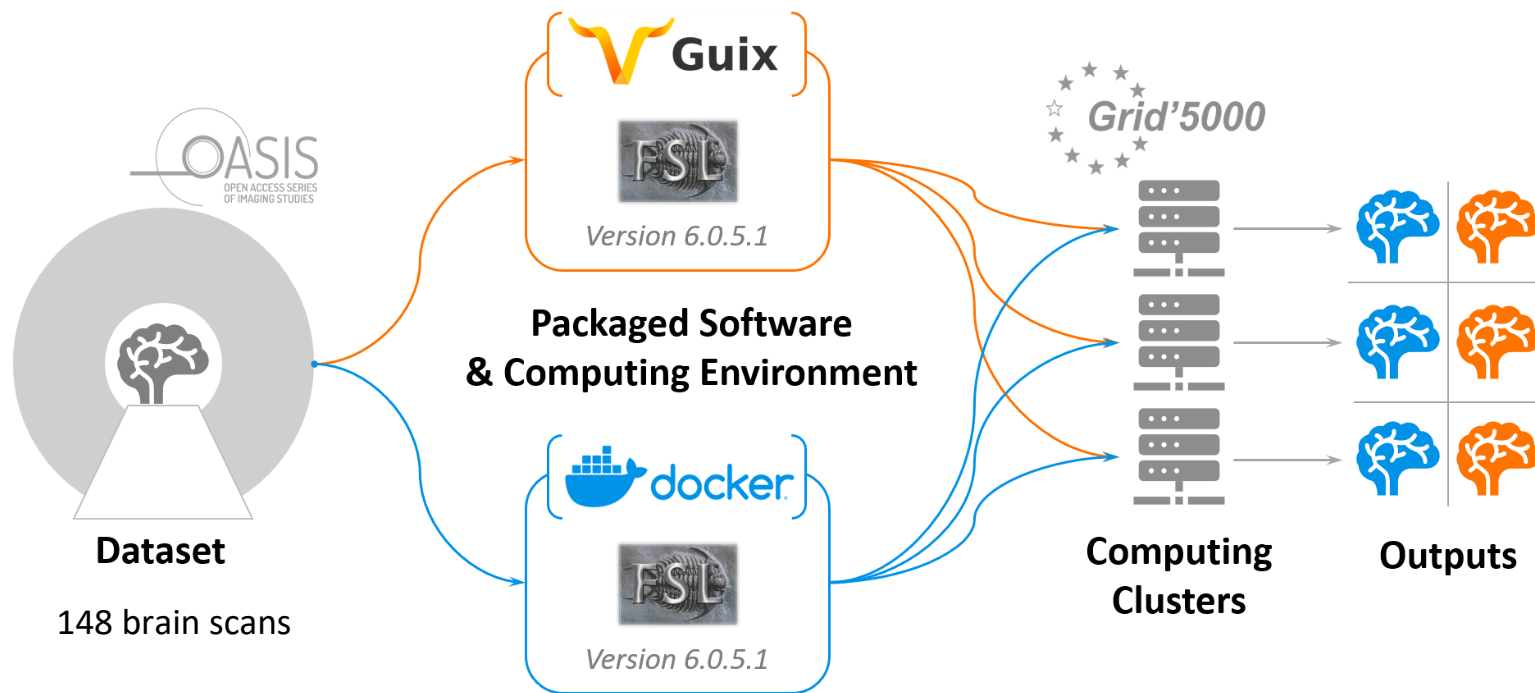
# Grid'5000 research infrastructure

- Large-scale testbed for experiment-driven research in computer science
- Access to a wide spectrum of hardware

Cluster	CPU	Model	Micro-arch	ISE
uvb	Intel	Xeon X5670	Westmere	SSE4.2
hercule	Intel	Xeon E5-2620	Sandy Bridge	AVX
taurus	Intel	Xeon E5-2630	Sandy Bridge	AVX
parasilo	Intel	Xeon E5-2630 v3	Haswell	AVX2
nova	Intel	Xeon E5-2620 v4	Broadwell	AVX2
chiffnot	Intel	Xeon Gold 6126	Skylake	AVX-512
chiclet	AMD	EPYC 7301	Zen	AVX2
neowise	AMD	EPYC 7642	Zen 2	AVX2
abacus21	AMD	EPYC 7F72	Zen 2	AVX2

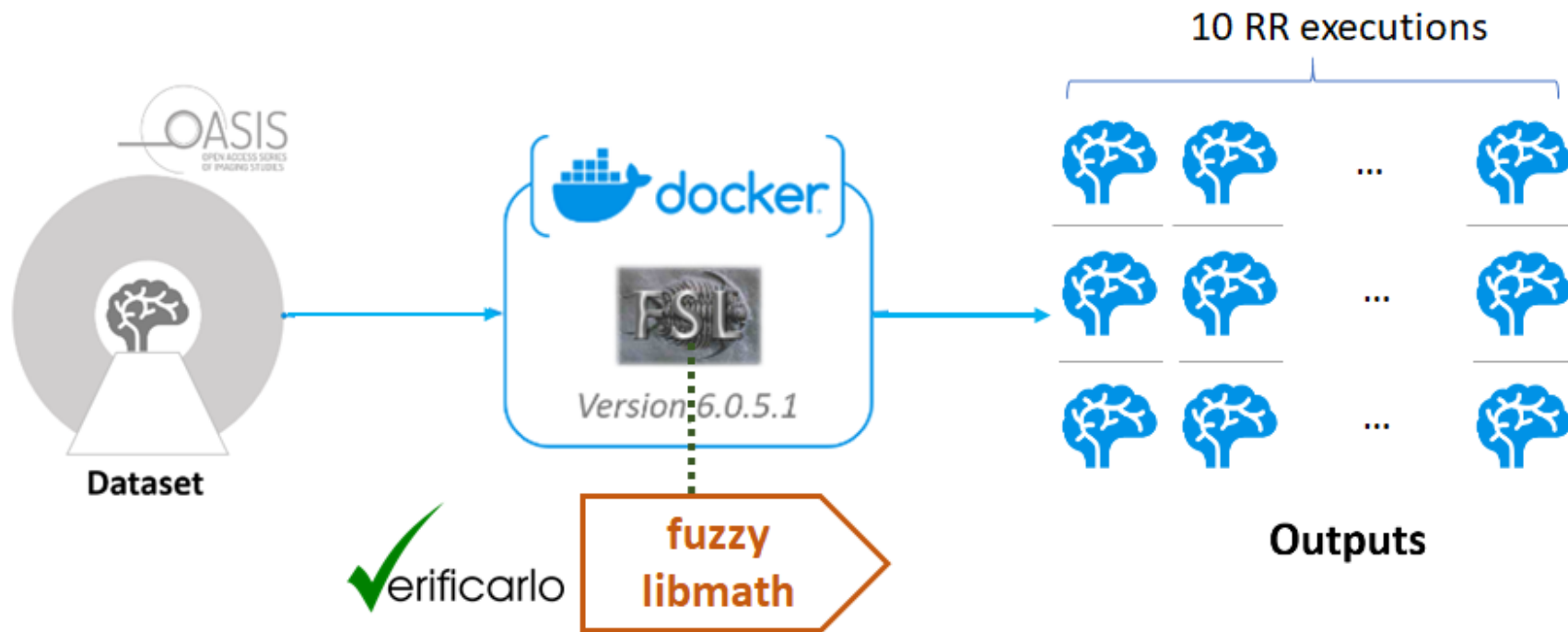
← Fused Multiply-Add (FMA)

# Overview of experiments on Grid'5000



(4 Guix + 1 Docker executables) x 9 clusters = 45 experiments

# Overview of MCA experiments





# Hardware variability

- Comparison of global checksums
  - tarball of the 148 results for each one of the 45 experiments

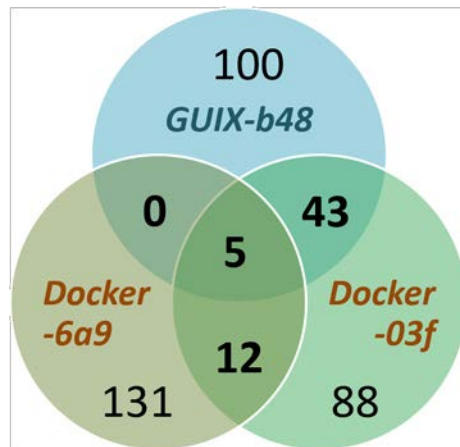
Deployment	Compilation flags (-march=)	Microarchitecture of the execution node	ISE	Global checksum
Docker	x86_64	Intel Westmere, Sandy Bridge	SSE4.2, AVX	03f...
Docker	x86_64	Intel Haswell, Broadwell, Skylake, AMD Zen, Zen 2	AVX-2	6a9...
Guix	x86_64	All	SSE4.2, AVX, AVX-2	b48...
Guix	sandybridge	Intel Sandy Bridge	AVX	b48...
Guix	haswell or skylake	Intel Haswell, Broadwell, Skylake, AMD Zen, Zen 2	AVX-2	75e...
Guix	sandybridge	Intel Westmere	SSE4.2	incompatibility
Guix	haswell or skylake	Intel Westmere, Sandy Bridge	SSE4.2, AVX	incompatibility

Four different global checksums

Two micro-architecture subsets: with and without AVX-2

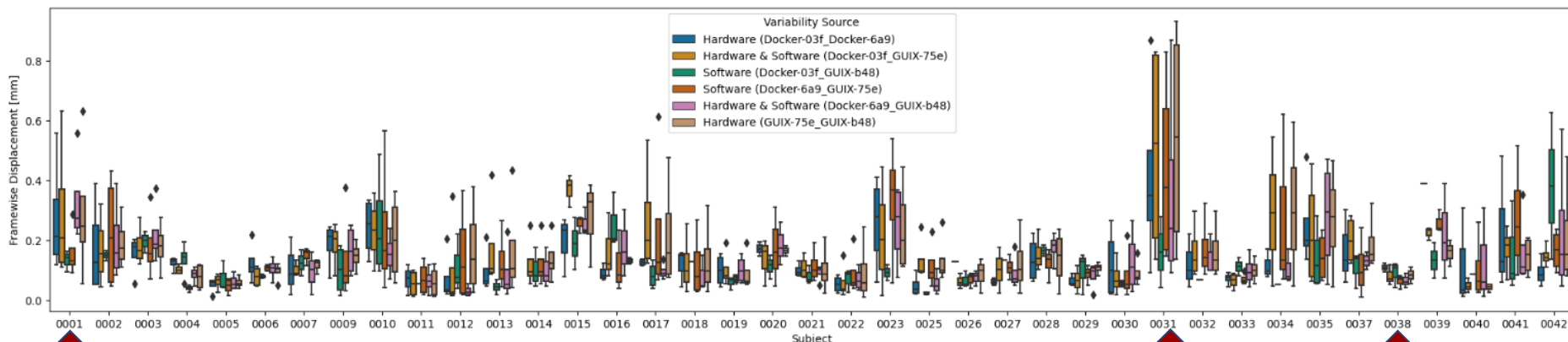
# Variability depends on input data

- Comparison of the 148 individual results among the four sets of results
- Three of the four sets share a few identical results



Intersections between result sets (individual matrix files) for three of the four experiments.

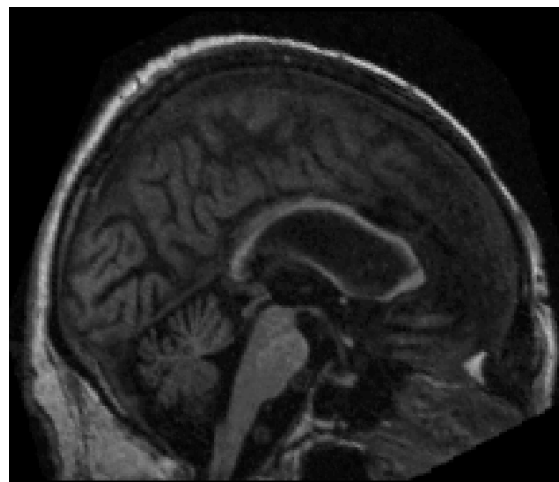
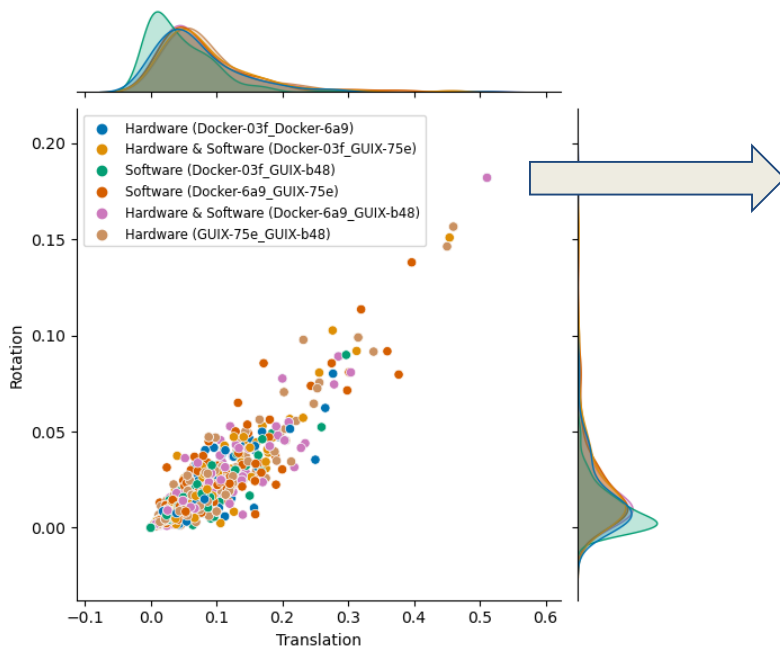
# Variability across subjects



Frame-wise displacement across subjects

=> importance of using large image databases

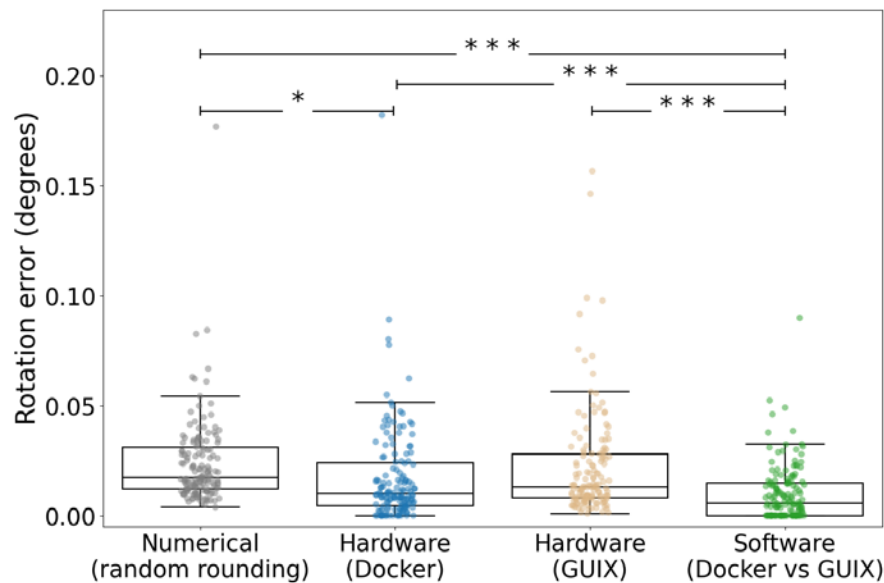
# Effects on the registration



Distributions of rotation and translation differences in the transformation matrix results (‘.mat’ result files)

Differences between outputs (belonging to groups Docker-6a9 and Guix-b48) with the largest difference in translation and rotation (subject 31, scan 2)

# Variabilities of comparable magnitude



Comparison between rotation errors for numerical, hardware, and software variability, for each subject

# Correlation

Variability 1	Variability 2	Spearman correlation
Hardware (Docker)	Numerical (RR)	0.04
Hardware (Guix)	Numerical (RR)	0.11
Software (Docker vs Guix)	Numerical (RR)	-0.11
Software (Docker vs Guix)	Hardware (Guix)	0.20
Software (Docker vs Guix)	Hardware (Docker)	0.11
Hardware (Guix)	Hardware (Docker)	0.41*

Correlations in translation vectors

Variability 1	Variability 2	Spearman correlation
Hardware (Docker)	Numerical (RR)	0.00
Hardware (Guix)	Numerical (RR)	0.01
Software (Docker vs Guix)	Numerical (RR)	-0.12
Software (Docker vs Guix)	Hardware (Guix)	0.22
Software (Docker vs Guix)	Hardware (Docker)	0.08
Hardware (Guix)	Hardware (Docker)	0.36*

Correlations in rotation vectors

# Paper conclusions

- Hardware, software and numerical variability lead to variations
  - of similar magnitudes but
  - uncorrelated with each other
- RR introduces perturbations of similar magnitude
  - Practical method to simulate both hardware and OS updates
- Variations remained moderate but might impact downstream analyses
- Both packaging solutions (Docker and Guix) produced
  - Each one bit-wise reproducible results when using the same packaged FLIRT executable on equivalent micro-architectures
  - Different outputs from one another due to the software variability

# Discussion

- Packaging solutions
  - Docker image: little or no information on how the executable was built
  - Guix: full transparency on both compiling and runtime environments
- Compilation options
  - Only studied the impact of the “march” flag, directly related to hardware
  - Other compilation options (e.g. optimization levels) are also known to impact reproducibility
- In our case hardware variability was due to AVX-2 support
  - Further work is needed for a finer analysis of the differences observed



# ReproVIP

- Ongoing ANR JCJC project
  - Coordinator: [sorina.pop@creatis.insa-lyon.fr](mailto:sorina.pop@creatis.insa-lyon.fr)
  - Partners: CREATIS, IPHC, Concordia University
- Main objectives
  - Evaluate and improve the reproducibility of scientific results: same result when the code is executed with the same set of inputs
  - Provide an integrated, end to end solution, allowing to launch reproducible executions in a transparent manner
  - Evaluate the proposed methods and tools on two studies
    - Optimization of the MRI acquisition protocol
    - Optimization of a processing pipeline for brain cancer prediction

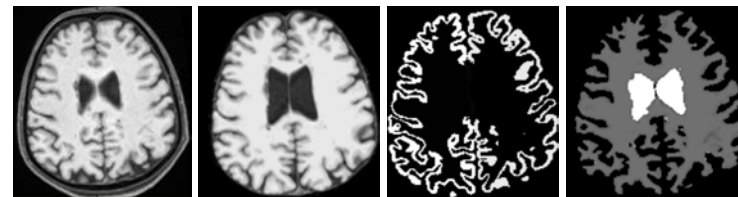
**CREATIS**

# The Virtual Imaging Platform (VIP)

- Scientific applications as a Service
  - More than 20 applications publicly available
  - <https://vip.creatis.insa-lyon.fr/home.html>
- Transparent access to computing resources
  - 40 CPU years (EGI biomed VO) used in 2022
- Large community
  - More than 1500 registered users
  - 75 publications since 2011
- Open and reproducible science
  - [Zenodo](#), DOIs, Containers, Boutiques



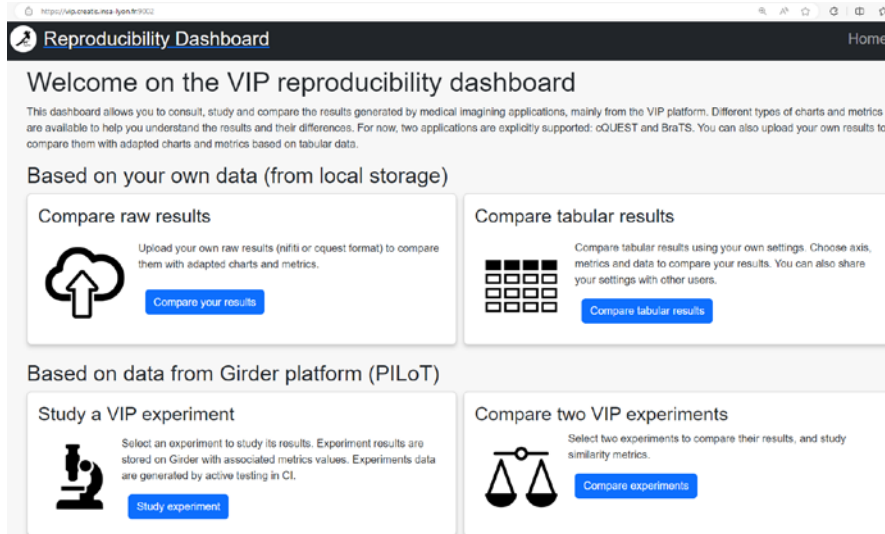
CREATIS



Example of white/grey matter brain segmentation with [Freesurfer](#) on VIP  
Credits : Berardino Barile and Dominique Sappey-Marini r, Creatis

# Integrated end to end solution

- VIP portal
  - Applications as a service
  - Execution sharing (Zenodo)
- Automation
  - Jupyter Notebooks (templates)
  - Python client, REST API
- Reproducibility Dashboard
  - <https://vip.creatis.insa-lyon.fr:9002>
- Continuous Integration (CI)
- Integration with storage platforms
  - Girder, Shanoir



The screenshot shows a web browser window displaying the "Reproducibility Dashboard". The page title is "Reproducibility Dashboard" and the URL is "https://vip.creatis.insa-lyon.fr:9002". The dashboard is titled "Welcome on the VIP reproducibility dashboard" and provides instructions on how to use the platform. It is divided into two main sections: "Based on your own data (from local storage)" and "Based on data from Girder platform (PILoT)".

**Based on your own data (from local storage)**

- Compare raw results**: Upload your own raw results (nifti or cquest format) to compare them with adapted charts and metrics. Includes a cloud upload icon and a "Compare your results" button.
- Compare tabular results**: Compare tabular results using your own settings. Choose axis, metrics and data to compare your results. You can also share your settings with other users. Includes a grid icon and a "Compare tabular results" button.

**Based on data from Girder platform (PILoT)**

- Study a VIP experiment**: Select an experiment to study its results. Experiment results are stored on Girder with associated metrics values. Experiments data are generated by active testing in CI. Includes a microscope icon and a "Study experiment" button.
- Compare two VIP experiments**: Select two experiments to compare their results, and study similarity metrics. Includes a scales icon and a "Compare experiments" button.

ReproVIP reproducibility dashboard

# Outline

- I. What is reproducibility?
- II. Computational reproducibility
- III. Larger overview**

# Ultimate goal

		Data	
		Same	Different
Analysis	Same	Reproducible	Replicable
	Different	Robust	Generalisable

# Software tools

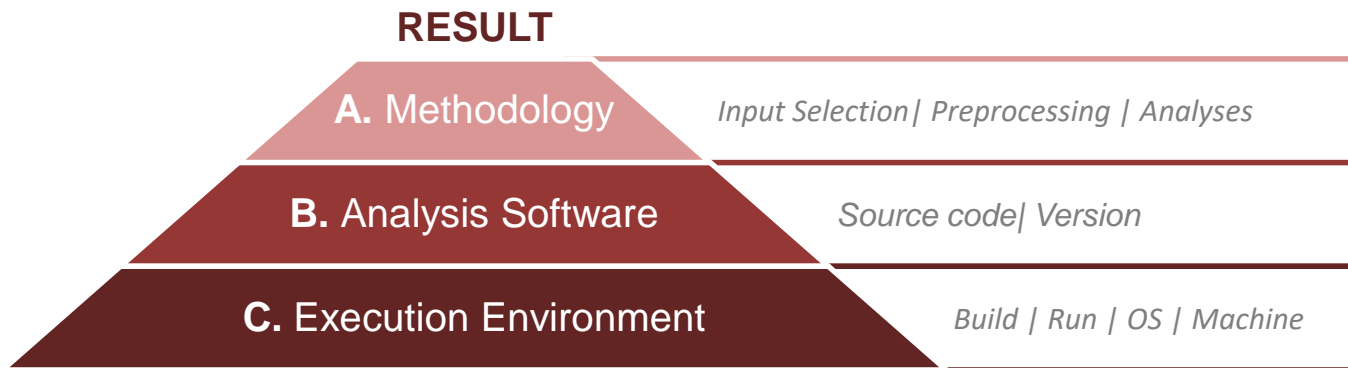
- Multiple layers
  - Custom scripts or notebooks implementing study design and analysis
  - Core image processing methods such as segmentation and registration
  - Direct software dependencies (optimization toolboxes, data manipulation libraries)
  - Contingent dependencies (elementary mathematical functions, compilers)

# Good practices

- Software development and sharing (whenever possible)
  - Use git to manage and share code
  - Clear licensing
  - Proper documentation
  - Code formatting standards
  - Use permanent identifiers (Digital Object Identifiers or DOIs) for software releases
- Deployment
  - Publish versioned software packages (PyPI or directly on GitHub)
  - Release software container images (Docker)
  - Create Guix packages

# Analysis

- Anything that can be described in a scientific paper, including the methods and algorithms used to produce such results





# Good practices

- Reproducibility checklists
  - <https://miccai2021.org/files/downloads/MICCAI2021-Reproducibility-Checklist.pdf>
  - <http://www.cs.mcgill.ca/~jpineau/ReproducibilityChecklist.pdf>
- Technical solutions for re-runnable analyses
  - Workflows: NiPype, Snakemake, Nextflow
  - Jupyter Notebooks
  - Platforms such as Neurolibre: <https://neurolibre.org>
  - Journals such as IPOL: <https://www.ipol.im/>

# Data

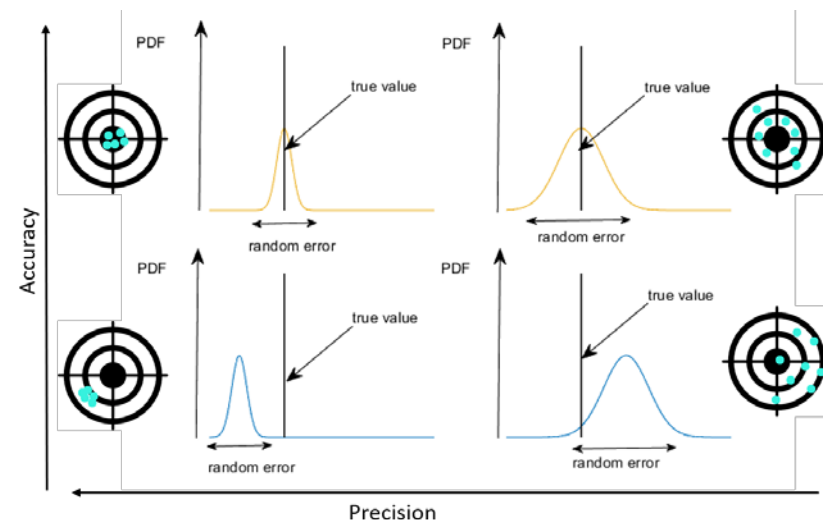
- Multiple changes from their acquisition to their final processing
- When data access and sharing is not possible, ensure proper description and documentation
- Use of data management platforms
- Follow **FAIR** principles
  - Findable. Metadata and data should be easy to find
  - Accessible. Possibly including authentication and authorisation.
  - Interoperable. Formal, accessible and shared language
  - Reusable. Metadata and data should be well-described
- Data Management Plans (DMP)
  - DMP OPIDoR
- Data quality

# Transparency

- The one practice that can be universally commended is the transparent and complete reporting of all facets of a study, allowing a critical reader to evaluate the work and fully understand its strengths and limitations
  - [[Nichols et al, 2017](#)]
- Guidelines
  - Document choices and analyses
  - Use version control systems, such as Git
  - Share code and data whenever possible
- Challenges
  - Ethical and legal problems

# Validation

- Continuous (never ending) process
  - Evolving software
  - New databases
- Guidelines
  - Define clear validation objectives
  - Define/use formalised and transparent validation procedures
  - Use standardized open datasets



# Sum-up on computational reproducibility

- Containers help mitigate the extent of environment-introduced variability
  - May lose trace of the build environment
- Reproducible builds with NIX, GUIX
- Parallelization or hardware may still lead to different results
- In the long term, software and infrastructures cannot be frozen
- Variability sources need to be taken into account, evaluated and addressed

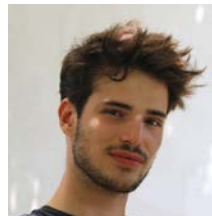
# Take-home messages

- Computational reproducibility
  - Challenging and often over-looked
  - Various, possibly complex solutions
- Reproducible and generalisable software solutions
  - Computational reproducibility is only a small aspect of a larger issue
  - Transparency and validation are also essential

# Additional info

- French network
  - <http://www.recherche-reproductible.fr>
- Fun Mooc
  - <https://www.fun-mooc.fr/en/courses/reproducible-research-methodological-principles-transparent-scie/>
  - <https://www.fun-mooc.fr/en/courses/reproducible-research-ii-practices-and-tools-for-managing-comput/>
- Reproducibility tutorials
  - <https://www.creatis.insa-lyon.fr/miccai2023> (Hands-on material)
  - <https://miccai2023-reproducibility-tutorial.github.io/>
- The turing way
  - <https://the-turing-way.netlify.app/index.html>
- French book « Vers une recherche reproductible »
  - <https://hal.science/hal-02144142>

# Acknowledgements

**CREATIS**ReproVIP **anr**<sup>®</sup>





**Thank you for your attention!  
Questions?**

ReproVIP **anr**<sup>®</sup>