The Impact of Hardware Variability on Applications Packaged with Docker and Guix: a Case Study in Neuroimaging

Gaël Vila, Emmanuel Medernach, Inés Gonzalez, Axel Bonnet, Yohan Chatelain, Michaël Sdika, Tristan Glatard, Sorina Camarasu-Pop

The Impact of Hardware Variability on Applications Packaged with Docker and Guix: a Case Study in Neuroimaging

Gaël Vila*
Univ Lyon, INSA-Lyon, Université Claude Bernard Lyon 1, UJM-Saint Etienne, CNRS, Inserm, CREATIS UMR 5220, U1294
Villeurbanne, France
gael.vila@protonmail.com

Emmanuel Medernach*
IPHC, CNRS/IN2P3, Université de Strasbourg
Strasbourg, France
emmanuel.medernach@iphc.cnrs.fr

Inés Gonzalez Pepe
Department of Computer Science and Software Engineering, Concordia University
Montreal, Canada
i_gon@encs.concordia.ca

Axel Bonnet
Univ Lyon, INSA-Lyon, Université Claude Bernard Lyon 1, UJM-Saint Etienne, CNRS, Inserm, CREATIS UMR 5220, U1294
Villeurbanne, France
axel.bonnet@creatis.insa-lyon.fr

Yohan Chatelain
Department of Computer Science and Software Engineering, Concordia University
Montreal, Canada
yohanmichelbenoit.chatelain@concordia.ca

Michaël Sdika
Univ Lyon, INSA-Lyon, Université Claude Bernard Lyon 1, UJM-Saint Etienne, CNRS, Inserm, CREATIS UMR 5220, U1294
Villeurbanne, France
michael.sdika@creatis.insa-lyon.fr

Tristan Glatard
Department of Computer Science and Software Engineering, Concordia University
Montreal, Canada
tristan.glatard@concordia.ca

Sorina Camarasu-Pop
Univ Lyon, INSA-Lyon, Université Claude Bernard Lyon 1, UJM-Saint Etienne, CNRS, Inserm, CREATIS UMR 5220, U1294
Villeurbanne, France
sorina.pop@creatis.insa-lyon.fr

ABSTRACT

The reproducibility of neuroimaging analyses across computational environments has gained significant attention over the last few years. While software containerization solutions such as Docker and Singularity have been deployed to mask the effects of software-induced variability, variations in hardware architectures still impact neuroimaging results in an unclear way. We study the effect of hardware variability on linear registration results produced by the FSL FLIRT application, a widely-used software component in neuroimaging data analyses. Using the Grid'5000 infrastructure, we study the effect of nine different CPU models using two software packaging systems (Docker and Guix), and we compare the resulting hardware variability to numerical variability measured with random rounding. Results show that hardware, software, and numerical variability lead to perturbations of similar magnitudes — albeit uncorrelated — suggesting that these three types of variability act as independent sources of numerical noise with similar magnitude. Therefore, random rounding is a practical solution to measure the effect of numerical noise induced by hardware variability in this application. The effect of hardware perturbations on linear registration remains moderate, with average translation errors of 0.1 mm (maximum: 0.5 mm) and average rotation errors of 0.02 deg (maximum: 0.2 deg). Such variations might impact downstream analyses when linear registration is used as initialization step for other operations.

KEYWORDS

Reproducibility, Neuroimaging, CPU micro-architecture, Software packaging, Random rounding

ACM Reference Format:
Gaël Vila, Emmanuel Medernach, Inés Gonzalez Pepe, Axel Bonnet, Yohan Chatelain, Michaël Sdika, Tristan Glatard, and Sorina Camarasu-Pop. 2024. The Impact of Hardware Variability on Applications Packaged with Docker and Guix: a Case Study in Neuroimaging. In *ACM Conference on Reproducibility and Replicability (ACM REP '24), June 18–20, 2024, Rennes, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3641525.3663626

1 INTRODUCTION

The last few years have seen a growing awareness of reproducibility concerns in many areas of science. The neuroimaging research community has been particularly active on the subject, producing multiple studies and recommendations [2, 14, 15]. Reproducibility is contingent on multiple factors [11] including study design, data

*These authors have contributed equally to this work and share first authorship.

ACM REP '24, June 18–20, 2024, Rennes, France

ACM ISBN 979-8-4007-0530-4/24/06
https://doi.org/10.1145/3641525.3663626

acquisition and analysis approaches, software implementations and their dependencies, and execution hardware. Several studies concentrated on software factors. To cite a few examples from the neuroimaging domain, the work in [3] reported extensive differences in the results produced by different software packages in functional MRI analysis, the work in [5] evaluated the reproducibility of tumor segmentations produced by different versions of the same application, and the study in [7] evaluated the reproducibility of neuroimaging analyses across operating systems.

The main approach to circumvent software variability has been to containerize software using Docker or Singularity (now renamed Apptainer) [19][1]. Containers allow developers to package and run an application and its dependencies — including all configuration files and dependent libraries — in a portable environment. The discrepancies in software versions, configurations, and dependencies that often arise between different execution environments are thus minimized. A few platforms, including Neurodesk [19], BIDS apps [8] or Boutiques [6], rely on Docker and Singularity containers to reproduce neuroimaging results across execution environments. However, Docker and Singularity containers often lack transparency, in particular when their creation "recipe" is not available, resulting in software black boxes. In contrast, Guix [21] is a framework to build reproducible computational environments that accurately document the software building chain and its dependencies. Guix packages are defined in modules exportable as Docker containers or other types of archives.

Software containers do not control for hardware heterogeneity. The extent to which hardware — for instance CPU micro-architecture — impacts computational neuroimaging results is unclear. In [19], minor differences between results obtained in different environments were observed despite the use of Docker containers, which was attributed to hardware variability. In particular, heterogeneity between CPU instruction sets — such as support for Advanced Vector Extensions or AVX — may introduce numerical perturbations that could impact results similarly to software-induced numerical perturbations studied in [7] or [12].

Numerical perturbations impact neuroimaging applications differently depending on the numerical stability of their implementation. Monte-Carlo Arithmetic [16], and in particular random rounding, is a technique to investigate numerical stability experimentally in large and complex code bases. Previous studies showed that random rounding can accurately simulate operating system updates in neuroimaging data analyses [20]. However, it remains unclear whether random rounding could be a good model for numerical perturbations resulting from hardware updates.

This paper aims to: (1) evaluate the impact of hardware variability on results produced by a neuroimaging application, (2) compare and correlate hardware variability to software variability encountered in different software packages, and (3) compare and correlate hardware variability to numerical variability resulting from random rounding applied to elementary mathematical functions. We focus on the FSL FLIRT [10] application, a software tool frequently used as a building block in neuroimaging analyses. We execute FSL FLIRT on a representative dataset of 148 MRIs, comparing results obtained with nine different CPU models accessed in the Grid'5000 research infrastructure [1], using two packaging systems (Docker and Guix), and testing different compilation options.

[1]See also https://epcced.github.io/2020-12-08-Containers-Online

Table 1: Subset of Grid'5000 clusters used in our experiments

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
chifflot	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

2 MATERIALS AND METHODS

Figure 1 summarizes the execution environments compared in our experiments. The remainder of this section details each of these components more thoroughly.

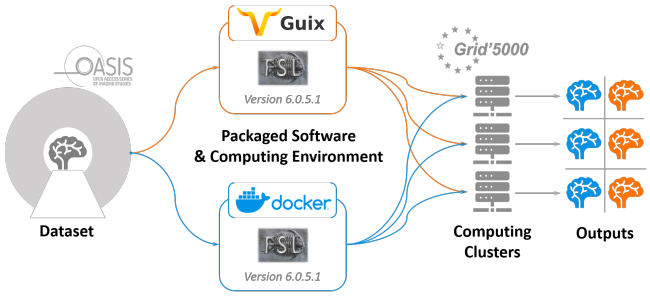

Figure 1: Summary of the Grid'5000 experiment

2.1 Computing Infrastructure

We used the Grid'5000 [1] research infrastructure for our experiments. Grid'5000 is a large-scale testbed deployed in France for experiment-driven research in all areas of computer science. Among others, it gives access to a wide spectrum of hardware[2], which is particularly interesting for this study. In our experiments, we used a subset of computing nodes with different CPU models summarized in Table 1 and using the Debian 11 operating system (OS) as it was the default Grid'5000 OS at the time we conducted our experiments. Grid'5000 also supports OS customization. Appendix A briefly describes how to access Grid'5000 nodes.

Table 1 lists the computing clusters and the associated CPU models involved in our experiments, with additional information on the CPU micro-architecture and Instruction Set Extension (ISE). Micro-architecture refers to the way that a given instruction set architecture (ISA) is implemented in a particular processor. An ISA describes the instruction set available on the processor. An ISE is a set of specific instructions that extends the ISA basis. It works as a

[2]https://www.grid5000.fr/w/Hardware

layer, a.k.a next generations of ISE support previous generations (AVX512 supports AVX2 that supports AVX that supports SSE4.2).

Advanced Vector Extensions (AVX, also known as Sandy Bridge New Instructions) are SIMD (Single Instruction, Multiple Data) extensions to the x86 instruction set architecture. They were proposed by Intel in 2008 and first supported with the Sandy Bridge processor. AVX-2 (also known as Haswell New Instructions) expands most integer commands to 256 bits and introduces new instructions. They were first supported with the Haswell processor in 2013. Finally, AVX-512 extends 256-bits AVX instructions to 512 bits.

The design and optimization of micro-architecture are critical to the development of modern CPUs, GPUs, and other types of processors, but they can also be a source of non-reproducibility.

Floating-point operations can yield different results on processors supporting different versions of SIMD extensions like AVX and AVX-2. AVX-2 introduced new instructions and capabilities not present in the original AVX, such as Fused Multiply-Add (FMA). FMA is a floating-point multiply–add operation performed in one step (fused operation), with a single rounding. FMA can speed up and improve the accuracy of computations that involve the accumulation of products. AVX-512, in its Foundation version, does not add new instructions introducing a different accuracy like FMA does. However, by extending the length of vectorized arithmetic operations, it may impact the results of vectorized operations, such as vectorized sum, because floating-point arithmetic lacks associativity.

The Grid'5000 research infrastructure was used to run FSL FLIRT experiments on the different CPU micro-architecture listed in Table 1. For random rounding experiments, we used the Narval cluster operated by Calcul Québec in the Digital Alliance of Canada which include AMD Rome 7502, AMD Rome 7532, and AMD Milan 7413 CPUs which support AVX-2. In addition to these computing resources, we also used the SCIGNE[3] cloud infrastructure for deploying our Guix and CVMFS servers.

2.2 Neuroimaging Application

The FMRIB Software Library[4] (FSL [9]) is a comprehensive library of analysis tools for FMRI, MRI and diffusion brain imaging data. FSL-FLIRT is a neuroimaging tool for affine brain registration. Affine brain registration consists of aligning an input object (in our case, a brain scan) with another one through rotation, translation, scaling, and shearing. In 3D, this results in a linear transformation with 12 degrees of freedom.

In this case, brain registration was performed using a subject image as input image and the T1-weighted MNI152 template brain with a voxel resolution of 1x1x1mm as reference image. In neuroimaging studies, linear registration to a template brain is commonly performed as an initialization step for spatial normalization which is a common pre-processing step for group analyses. The application was run with minimal configuration using the MNI152_T1_1mm.nii.gz template as detailed in appendix B.

[3] https://scigne.fr/
[4] https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/FSL

Table 2: Software versions used in Docker vs Guix

	FSL	gcc	libm	OpenBlas
Docker	6.0.5.1	4.8.5	2.27	0.3.3
Guix	6.0.5.1	9.5.0	2.33	0.3.20

2.3 Compilation and Packaging

Compilation affects the results of floating-point operations. Compilers may optimize floating-point calculations differently based on the available instruction set. For example, a compiler might use FMA instructions when they are available (as in AVX-2) but resort to separate multiplication and addition instructions on an AVX-only processor. The "march" (machine architecture) flag can help optimize software to run efficiently on specific hardware and is used in this study to evaluate its impact on the reproducibility of results.

We packaged the application with Docker and Guix using software versions reported in Table 2. For Docker, we used the FSL Docker image `vnmd/fsl_6.0.5.1` provided on Docker Hub by the NeuroDesk community organization. The image choice corresponds to that of a researcher targeting FSL version 6.0.5.1, but with no further requirements on the underlying environment and dependencies. The executable remained identical across experiments. The FLIRT binary was compiled with `gcc-4.8.5` using `-O3 -fexpensive-optimizations` optimization flags. The `-march` flag was not specified, resulting in the generation of assembly code that is portable across all existing `x86_64` architectures. We retrieved the gcc version by using the `readelf -p .comment flirt` command. The optimization flags were found in the `/opt/fsl-6.0.5.1/build.log` log file. FLIRT was linked with OpenBLAS v0.3.3 built for multiple targets with runtime detection of the target CPU. We installed Docker on the Grid'5000 nodes from package `docker-desktop-4.20.1-amd64.deb`, and we pulled the Docker image on each node at the beginning of our allocation. We processed the dataset nine times with this Docker container, once for every CPU model in Table 1.

For Guix, we built four different executables: one with the default compilation options, and three with compilation flags targeting specific micro-architectures (-march= sandybridge, haswell or skylake)[5]. We wrote FSL Guix modules to compile FSL and all its dependencies in a reproducible manner. In Guix, all packages are installed in separate directories. We adapted the compilation scripts to reference these paths. Package compilation with Guix was done using the `guix pack` command and using relocatable packages options (see appendix C).

Our Guix build server was a virtual machine deployed on a local cloud infrastructure (SCIGNE). We also installed a local CVMFS[6] server in a virtual machine. The packages were build on our Guix build server with gcc compiler version 9.5.0 and then deployed using CVMFS. After reserving a node on Grid'5000, we installed a CVMFS client to access files on our CVMFS server. This deployment strategy allowed the computing nodes to download only the files needed at runtime from the packages.

[5] https://gitlab.in2p3.fr/reprovip/reprovip-guix/-/tree/master/GUIX/FSL/compilation
[6] https://cernvm.cern.ch/fs/

We processed the dataset with Guix once for each of the four compilation configurations (one default and three targeting specific micro-architectures: sandybridge, haswell or skylake)) and for each of the nine CPU models in Table 1, resulting in 36 different runs. Additionally, the Docker and Guix experiments were run twice with the default compilation options to check that application runs were deterministic.

2.4 Dataset

The data used in this experiment consisted of 148 brain scans from 39 healthy subjects (31 subjects × 4 scans + 8 subjects × 3 scans) sampled from the OASIS-I dataset [13]. All images had the same voxel size ($1 \times 1 \times 1.25$mm) and dimensions ($256 \times 256 \times 128$ voxels).

The original scans in the OASIS-I dataset were available in a legacy `.hdr` format, which caused FSL to misinterpret the orientation information, a documented problem that requires manual correction[7]. Image orientation was therefore manually set to standard RAS (Right-Anterior-Superior) using the FSL commands `fslorient` and `fslswapdim`, after being converted to NIfTI format using `fslchfiletype`. Images were visually inspected for correct registration using `fsleyes`.

2.5 Random Rounding

Random rounding is a type of Monte Carlo Arithmetic [16], a stochastic arithmetic technique to empirically evaluate numerical stability by injecting noise into floating point operations and quantifying the resulting error at a given virtual precision. While Monte-Carlo Arithmetic provides different noise injection modes, we only used random rounding (RR). RR simulates rounding errors by applying the following perturbation to all floating-point operations of an application:

$$random\_rounding(x \circ y) = round(inexact(x \circ y))$$

where x and y are floating-point numbers, $\circ$ is an arithmetic operation, and *inexact* is a random perturbation defined at a given virtual precision:

$$inexact(x) = x + 2^{e_x - t}\xi$$

where e_x is the exponent in the floating-point representation of x, t is the virtual precision, and ξ is a random uniform variable of $(-\frac{1}{2}, \frac{1}{2})$. To measure numerical uncertainty, we applied a perturbation of 1 ulp (unit of least precision, a.k.a the spacing between two consecutive floating-point numbers), which corresponds to a virtual precision of $t = 24$ bits for single-precision and $t = 53$ bits for double-precision. We applied random rounding only to the elementary mathematical functions used in FSL FLIRT, as done in [20]. Elementary mathematical functions are widely used in linear registration, and their rounding is a main source of variability across computing environments. FSL FLIRT was instrumented through the Verificarlo [4] tool, a clang-based compiler. Verificarlo substitutes floating-point operations with a call to one of the various configurable floating-point models available. We used Verificarlo through "fuzzy libmath" v0.9.1 that we added to FSL Docker container `vnmd/fsl\_6.0.5.1`, converted to a Singularity image supported by the execution cluster, and executed through the Boutiques library. We executed 10 executions of random rounding for each of the 148 images processed with FSL FLIRT on the Narval cluster.

2.6 Reproducibility Measures

The outputs of FSL-FLIRT are (i) the registered brain image in NIfTI format (`.nii.gz`) resampled in the geometry of the template brain, and (ii) the transformation matrix in text format (`.mat`). Reproducibility analyses were conducted on both outputs on a local computer where all output files were copied from the Grid'5000 infrastructure.

We tested bitwise reproducibility by computing the MD5 sums of the output files. To facilitate the identification of experimental conditions producing identical results for all the 148 input images, we computed the checksum of the global output dataset, *i.e.*, of all 148 registered brain scans and transformation matrices. For each unique global output dataset we compared checksums on individual output files, *i.e.*, each registered brain scan and transformation matrix.

Transformation matrices were compared through their translation and rotation parameters. We extracted the translation vector and the rotation vector of Euler angles (roll, pitch, yaw). Given that the output affine transformation matrix can also include scaling and shearing, we removed these components and isolated the pure rotation part of the matrix using the polar decomposition. The matrix A was decomposed into $A = SR$ where S is a positive symmetric matrix and R is the best rotation approximating A. This decomposition was implemented using the singular decomposition (numpy.linalg.svd). Translation and rotation errors between transformations a and b were then computed using the Euclidean distance between translation or rotation vectors:

$$tr_{err} = \sqrt{\sum_{i=0}^{2}\left(t_i^a - t_i^b\right)^2} \quad \text{and} \quad rot_{err} = \sqrt{\sum_{i=0}^{2}\left(r_i^a - r_i^b\right)^2} \tag{1}$$

where (t_0, t_1, t_2) is the translation vector (expressed in *mm*) and (r_0, r_1, r_2) is the rotation vector of Euler angles (roll, pitch, yaw expressed in degrees).

Transformation matrices were also compared using the framewise displacement (FD) metric[8]. The FD metric is derived from the study of motion artefacts. It computes the displacement of a voxel located 50 mm from the center of the brain, i.e., approximately on the brain skull [18]. FD was computed between two transformations a and b using the following formula (from [17]):

$$FD_{a,b} = \sum_{i=0}^{2}|t_i^a - t_i^b| + 50 \cdot \frac{\pi}{180} \cdot \sum_{r=0}^{2}|r_i^a - r_i^b|,$$

where (t_0, t_1, t_2) is the translation vector (expressed in *mm*) and (r_0, r_1, r_2) is the rotation vector of Euler angles (roll, pitch, yaw expressed in degrees).

2.7 Availability of Code and Data

The scripts and notebooks used to generate our results are available at the following locations:

- FSL Docker image: hub.docker.com/r/vnmd/fsl_6.0.5.1

[7] https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/Orientation%20Explained

[8] https://wiki.cam.ac.uk/bmuwiki/FMRI#Framewise_Displacement

- FSL Guix modules and Grid'5000 scripts: gitlab.in2p3.fr/reprovip/reprovip-guix, also archived on Software Heritage
- Post-processing for the reproducibility analysis: gitlab.in2p3.fr/reprovipgroup/reprovip-notebooks-and-scripts, also archived on Software Heritage
- Dockerfile, Boutiques descriptor and analysis scripts used for random rounding: github.com/big-data-lab-team/fuzzy-linreg/tree/acm-rep-2024, also archived on Software Heritage.

We used FSL v6.0.5.1, fuzzy libmath v0.9.1, Docker v4.20.1 (package docker-desktop-4.20.1-amd64.deb), cvmfs v2.11.2 and Guix channels as defined in channels.scm[9].

The OASIS dataset used in our experiments is available at oasis-brains.org/#data.

The transformation matrix results (.mat files) are available at zenodo.org/records/10649569.

3 RESULTS

We define "hardware variability" as the output differences caused by using CPU micro-architectures with or without AVX2 support (Table 1), "software variability" the variability resulting from different compilation environments in Docker and Guix, mostly influenced by the FSL, gcc, libm, and OpenBlas versions (Table 2), and "numerical variability" the variability resulting from random rounding.

3.1 Hardware variability was measurable in Docker and Guix

Table 3 summarizes the results obtained across all CPU types. The global checksum identifies the whole output dataset, i.e. all the 148 registered brain scans and transformation matrices. Experiments were run on the 9 clusters listed in Table 1, but the reporting in Table 3 was simplified because no difference was observed among the results obtained between the clusters with the same micro-architecture.

The experiments associated with the first three enclosed row groups in Table 3 were executed twice. Each pair of identical executions produced the same global output dataset (same global checksum), confirming that FSL FLIRT results were deterministic and bitwise reproducible.

Each packaging method (Docker or Guix) led to a different global checksum for two micro-architecture subsets, resulting in a total of four different global checksums. For the Docker experiments we used the same Docker image with a pre-built FSL binary (see section 2.3 for compilation options). Differences in results were due to differences in micro-architectures, divided in two categories. The first category (Intel Sandy Bridge and Westmere) does not provide support for AVX-2, while the second category (Intel Haswell, Broadwell, Skylake, AMD Zen, Zen 2) does, which suggests that differences in results were due to AVX-2 support. Moreover, Skylake's AVX-512 support did not influence the outcomeswhich underscores AVX2's significance in the observed variations.

For the Guix experiments, we generated four different packages, corresponding to four different FSL FLIRT executables. The first one was built with the default compiler options and was executed successfully on all Grid'5000 clusters listed in Table 1, yielding bit-wise reproducible results on all types of micro-architectures (third enclosed row group in Figure 3). This implies OpenBLAS was compiled for a generic `x86_64` architecture, disabling dynamic architecture support. We then created modified packages for different micro-architectures by adding specific `-march=` flags.

The Guix modified package built using the Sandy Bridge flag was, understandably, unable to run on the Westmere nodes, while the one using Haswell or Skylake flags was unable to run on Sandy Bridge or Westmere. We retrieve the same two micro-architectures categories leading to different results as with Docker. We note that a Guix package always gave the same results on all the nodes on which it was able to run. Using the default compiler options (`-march=x86_64 -O3`) allows to (i) execute the application on all nodes and (ii) obtain bit-wise reproducible results. However, it renders inaccessible the latest micro-architecture optimizations.

3.2 Variability was data-dependent

Among the four sets of outputs, three of them shared a few identical (bit-wise reproducible) results, while the fourth one (`Guix-75e`) shared no common result with the three others. The intersections between the three overlapping sets are summarized in Figure 2. We note that groups `Guix-b48` and `Docker-03f` (corresponding to the group of microarchitectures without AVX-2 support) count 43 (i.e., more than a third) identical results. Visual inspection of the inputs producing bit-wise reproducible results provided no insight to explain this behaviour. Previous works [5] have already shown that variability in segmentation outcomes was dependent on the input data. Figure 3 plots the framewise displacement across subjects, showing important variability across subjects. The fact that reproducibility results are data-dependent reiterates the necessity to conduct such experiments on large image databases.

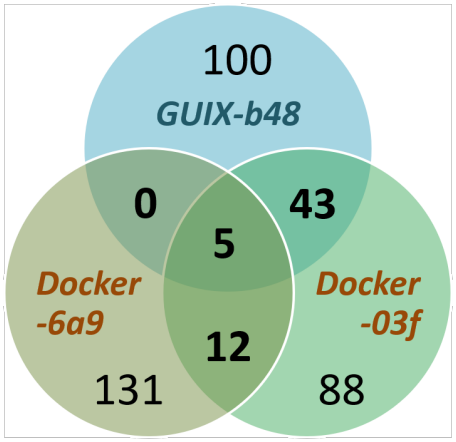

Figure 2: Intersections between result sets (matrix files) obtained on Grid'5000 for three of the four experiments. The fourth result set is disjoint.

[9] https://gitlab.in2p3.fr/reprovip/reprovip-guix/-/blob/master/GUIX/FSL/compilation/v0.1/channels.scm?ref_type=heads

Packaging	**Compilation flags** (-march=)	**Microarchitecture of the execution node**	ISE	**Global checksum**
Docker	x86_64	Intel Westmere	SSE4.2	**03f**8688da59b02bc55922c9dca322fe2
Docker	x86_64	Intel Sandy Bridge	AVX	**03f**8688da59b02bc55922c9dca322fe2
Docker	x86_64	Intel Haswell	AVX-2	**6a9**2985a9f458557cd62fb3eb0f0cebf
Docker	x86_64	Intel Broadwell	AVX-2	**6a9**2985a9f458557cd62fb3eb0f0cebf
Docker	x86_64	AMD Zen	AVX-2	**6a9**2985a9f458557cd62fb3eb0f0cebf
Docker	x86_64	AMD Zen2	AVX-2	**6a9**2985a9f458557cd62fb3eb0f0cebf
Docker	x86_64	Intel Skylake	AVX-512	**6a9**2985a9f458557cd62fb3eb0f0cebf
Guix	x86_64	Intel Westmere	SSE4.2	**b48**2bbfec28a23013196942185754132
Guix	x86_64	Intel Sandy Bridge	AVX	**b48**2bbfec28a23013196942185754132
Guix	x86_64	Intel Haswell	AVX-2	**b48**2bbfec28a23013196942185754132
Guix	x86_64	Intel Broadwell	AVX-2	**b48**2bbfec28a23013196942185754132
Guix	x86_64	AMD Zen	AVX-2	**b48**2bbfec28a23013196942185754132
Guix	x86_64	AMD Zen2	AVX-2	**b48**2bbfec28a23013196942185754132
Guix	x86_64	Intel Skylake	AVX-512	**b48**2bbfec28a23013196942185754132
Guix	sandybridge	Intel Westmere	SSE4.2	flirt execution not possible (incompatibility)
Guix	sandybridge	Intel Sandy Bridge	AVX	**b48**2bbfec28a23013196942185754132
Guix	sandybridge	Intel Haswell	AVX-2	**b48**2bbfec28a23013196942185754132
Guix	sandybridge	Intel Broadwell	AVX-2	**b48**2bbfec28a23013196942185754132
Guix	sandybridge	AMD Zen	AVX-2	**b48**2bbfec28a23013196942185754132
Guix	sandybridge	AMD Zen2	AVX-2	**b48**2bbfec28a23013196942185754132
Guix	sandybridge	Intel Skylake	AVX-512	**b48**2bbfec28a23013196942185754132
Guix	haswell	Intel Westmere	SSE4.2	flirt execution not possible (incompatibility)
Guix	haswell	Intel Sandy Bridge	AVX	flirt execution not possible (incompatibility)
Guix	haswell	Intel Haswell	AVX-2	**75e**c7e47fe5e2bd35aa6903398b47dcf
Guix	haswell	Intel Broadwell	AVX-2	**75e**c7e47fe5e2bd35aa6903398b47dcf
Guix	haswell	AMD Zen	AVX-2	**75e**c7e47fe5e2bd35aa6903398b47dcf
Guix	haswell	AMD Zen 2	AVX-2	**75e**c7e47fe5e2bd35aa6903398b47dcf
Guix	haswell	Intel Skylake	AVX-512	**75e**c7e47fe5e2bd35aa6903398b47dcf
Guix	skylake	Intel Westmere	SSE4.2	flirt execution not possible (incompatibility)
Guix	skylake	Intel Sandy Bridge	AVX	flirt execution not possible (incompatibility)
Guix	skylake	Intel Haswell	AVX-2	**75e**c7e47fe5e2bd35aa6903398b47dcf
Guix	skylake	Intel Broadwell	AVX-2	**75e**c7e47fe5e2bd35aa6903398b47dcf
Guix	skylake	AMD Zen	AVX-2	**75e**c7e47fe5e2bd35aa6903398b47dcf
Guix	skylake	AMD Zen 2	AVX-2	**75e**c7e47fe5e2bd35aa6903398b47dcf
Guix	skylake	Intel Skylake	AVX-512	**75e**c7e47fe5e2bd35aa6903398b47dcf

Table 3: Checksums obtained with different packaging methods and compilation flags on different microarchitectures

3.3 Effect of all perturbations remained moderate

Figure 4 quantifies the translation and rotation errors (Equation 1) within the four sets of results taken two by two. For all types of perturbations, translation errors remained under 0.6 mm, and rotation errors remained under 0.2 deg. While such differences remain moderate, they could impact downstream analyses initialized with linear registration.

Figure 5 illustrates the visual differences in results for the two outputs with the largest differences in translation and rotation (pink dot in the upper right-hand side of Figure 4) along with corresponding inputs (subject 31, scan 2). The corresponding animated gif is available online[10].

[10] https://raw.githubusercontent.com/big-data-lab-team/fuzzy-linreg/main/anim.gif

3.4 Hardware, software, and numerical variabilities were of comparable magnitude

No significant differences were observed between numerical and hardware variability (Figures 6 and 7), except between the numerical variability and the Docker experiments for rotation error (Figure 7) where the average difference between rotation errors was of 0.02 deg. Significant but small differences were observed between software variability and hardware variability, and between software variability and numerical variability. Overall, numerical, hardware, and software variability were of comparable magnitude.

3.5 Hardware, software, and numerical variabilities were uncorrelated

Correlation tests were conducted between numerical, hardware, and software perturbation types, comparing rotation and translation errors using the Spearman correlation coefficient as implemented

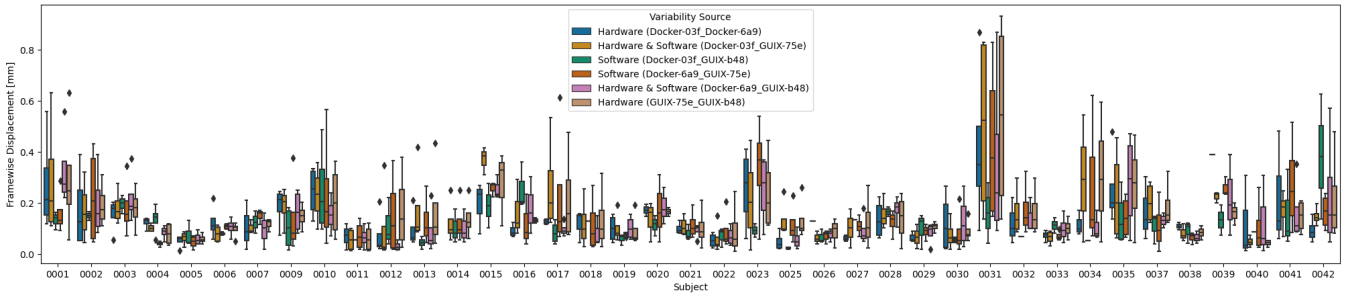

Figure 3: Framewise displacement across subjects

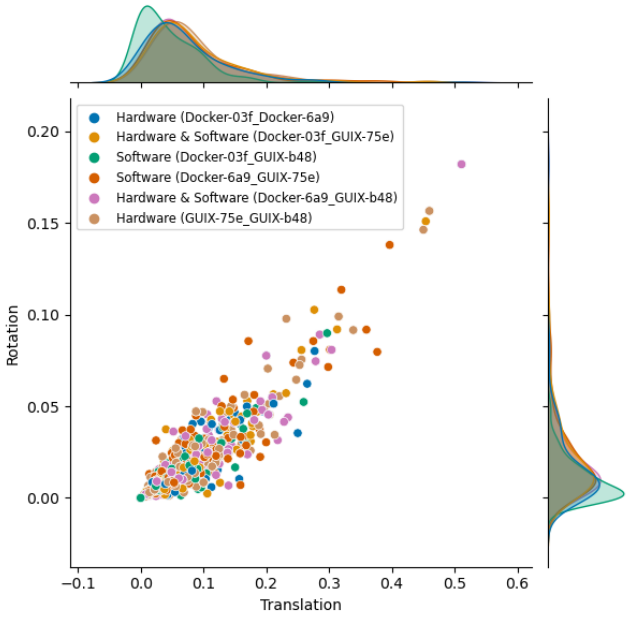

Figure 4: Distributions of rotation and translation differences in the transformation matrix results ('.mat' result files). Differences are computed for the six result pairs corresponding to hardware, software or (hardware & software) variability.

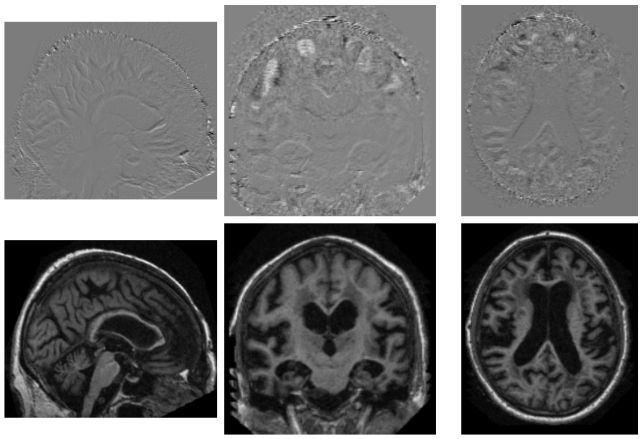

Figure 5: Sagittal, coronal and axial views of differences in results along with the corresponding input file. Differences on the top line are computed between outputs belonging to groups `Docker-6a9` and `Guix-b48` and representing the pair of outputs with the largest difference in translation and rotation. Input slices on the second line correspond to subject 31, scan 2.

in Scipy's `stats` package (Table 4 and 5). No significant correlation was found between hardware variability and numerical variability, or between hardware variability and software variability. It means that for a given subject, hardware variability cannot be reliably predicted from numerical or software variability. Unsurprisingly, a moderate but significant correlation was observed between hardware variability in the Docker experiments and hardware variability in the Guix experiments, for both translation and rotation errors.

4 DISCUSSION

We studied the impact of hardware variability on the reproducibility of results of a neuroimaging application (FSL FLIRT) deployed on Grid'5000 clusters using Docker and Guix. We then used random rounding to empirically evaluate the numerical stability of the application, which we compared to the variability of results from the first experiments. Results show that hardware, software and numerical variability lead to variations in results of similar magnitudes but

Table 4: Correlations between hardware, software and numerical variability measured in translation vectors. Hardware (Docker): Docker-03f vs Docker-6a9. Hardware (Guix): Guix-b48 vs Guix-75e. Software (Docker vs Guix): Docker-03f vs Guix-b48. *p < 0.05, Bonferroni corrected.

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*

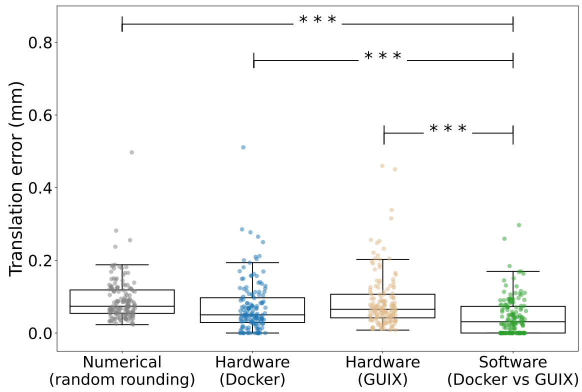

Figure 6: Comparison between translation errors measured for numerical, hardware, and software variability, for each subject. Numerical variability: average error across n=10 RR repetitions. Hardware (Docker): Docker-03f vs Docker-6a9. Hardware (Guix): Guix-b48 vs Guix-75e. Software (Docker vs Guix): Docker-03f vs Guix-b48. *p < 0.05; ***p < 0.01. p-values were obtained with a two-sample t-test and Bonferroni corrected.

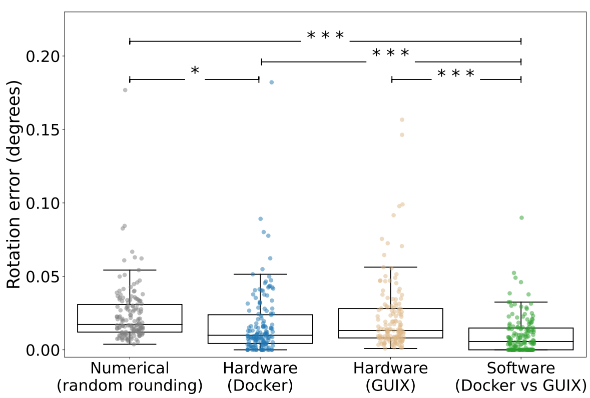

Figure 7: Comparison between rotation errors measured for numerical, hardware, and software variability, for each subject. Numerical variability: average error across n=10 RR repetitions. Hardware (Docker): Docker-03f vs Docker-6a9. Hardware (Guix): Guix-b48 vs Guix-75e. Software (Docker vs Guix): Docker-03f vs Guix-b48. *p < 0.05; ***p < 0.01: p-values were obtained with a two-sample t-test and Bonferroni corrected.

uncorrelated with each other. Variations remained moderate but might impact downstream analyses given that linear registration is often used as an initialization step to other operations.

Table 5: Correlations between hardware, software and numerical variability mesured in rotation vectors. Hardware (Docker): Docker-03f vs Docker-6a9. Hardware (Guix): Guix-b48 vs Guix-75e. Software (Docker vs Guix): Docker-03f vs Guix-b48. *p < 0.05, Bonferroni corrected.

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*

Docker and Guix, the two packaging solutions used in our experiments are known to mitigate software variability. From a computational bit-wise reproducibility point of view, experiments conducted in this study show that the two packaging solutions lead to similar conclusions: results are bit-wise reproducible when using the same packaged FLIRT executable on equivalent micro-architectures. We note however that, despite using the same version of the FLIRT source code, the two solutions yielded different outputs for most of the input files (but not for all). This is due to the software variability resulting from different compilation environments, mostly influenced by the gcc, libm and OpenBlas versions. The Docker image is a black box providing little or no information on how the executable was built (both on the compilation process, and on software dependency stack). In contrast, the Guix solution enforces full transparency on both compiling and runtime environments, requesting for the description and availability of all dependencies. The Guix package is thus more complex to produce than a Docker image, but, once available, variations can be easily built by modifying compiler options or by using other versions of dependent packages.

Regarding the compilation options, we only studied the impact of the "march" (machine architecture) flag, directly connected to the hardware aspects in the study. The "march" option can help optimizing software to run efficiently on specific hardware, but it's important to be aware of the portability issues it may engender. In our experiments, using "haswell" or "skylake" options prevented the FLIRT execution on Sandy Bridge and Westmere architectures. Other compilation options (not studied here) are also known to impact reproducibility. As an example, different optimization levels (-O0, -O1, -O2, -O3, -Os) can lead to different execution paths in the code, potentially causing variations in results, especially in floating-point computations.

In our study, results suggest that differences in FSF FLIRT outputs related to hardware variability are due to AVX-2 support. Further work is needed for a finer analysis of the differences observed.

Random rounding had already been shown to accurately simulate the effect of operating system updates in a similar neuroimaging application [20]. The present results show that RR introduces perturbations of a similar magnitude when compared to executions on

different micro-architectures, which establishes RR as a practical method to simulate both hardware and operating system updates.

The lack of correlation observed between hardware and numerical variability indicates that both types of variability are different in nature and essentially act as independent sources of numerical noise with similar magnitude. This observation seems reasonable given that AVX-2 instructions involved in hardware variability and elementary numerical functions involved in numerical variability impact different parts of the application.

5 FUNDING

This work was supported by the French ANR through the ReproVIP project (ANR-21-CE45-0024-01). This work was also supported by the Canada Research Chairs program.

ACKNOWLEDGMENTS

Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see www.grid5000.fr). The authors acknowledge the support of the Grid'5000 and Guix teams, as well as the support and resources provided by France Grilles and the IPHC Computing team. This work was performed within the framework of the LABEX PRIMES (ANR-11-LABX-0063). Random rounding experiments were executed on the Narval cluster operated by Calcul Québec in the Digital Alliance of Canada.

REFERENCES

[1] Daniel Balouek, Alexandra Carpen Amarie, Ghislain Charrier, Frédéric Desprez, Emmanuel Jeannot, Emmanuel Jeanvoine, Adrien Lèbre, David Margery, Nicolas Niclausse, Lucas Nussbaum, Olivier Richard, Christian Pérez, Flavien Quesnel, Cyril Rohr, and Luc Sarzyniec. 2013. Adding Virtualization Capabilities to the Grid'5000 Testbed. In *Cloud Computing and Services Science*, Ivan I. Ivanov, Marten van Sinderen, Frank Leymann, and Tony Shan (Eds.). Communications in Computer and Information Science, Vol. 367. Springer International Publishing, 3–20. https://doi.org/10.1007/978-3-319-04519-1_1

[2] Rotem Botvinik-Nezer, Felix Holzmeister, Colin F. Camerer, Anna Dreber, Juergen Huber, Magnus Johannesson, Michael Kirchler, Roni Iwanir, Jeanette A. Mumford, R. Alison Adcock, Paolo Avesani, Blazej M. Baczkowski, Aahana Bajracharya, Leah Bakst, Sheryl Ball, Marco Barilari, Nadège Bault, Derek Beaton, Julia Beitner, Roland G. Benoit, Ruud M.W.J. Berkers, Jamil P. Bhanji, Bharat B. Biswal, Sebastian Bobadilla-Suarez, Tiago Bortolini, Katherine L. Bottenhorn, Alexander Bowring, Senne Braem, Hayley R. Brooks, Emily G. Brudner, Cristian B. Calderon, Julia A. Camilleri, Jaime J. Castrellon, Luca Cecchetti, Edna C. Cieslik, Zachary J. Cole, Olivier Collignon, Robert W. Cox, William A. Cunningham, Stefan Czoschke, Kamalaker Dadi, Charles P. Davis, Alberto De Luca, Mauricio R. Delgado, Lysia Demetriou, Jeffrey B. Dennison, Xin Di, Erin W. Dickie, Ekaterina Dobryakova, Claire L. Donnat, Juergen Dukart, Niall W. Duncan, Joke Durnez, Amr Eed, Simon B. Eickhoff, Andrew Erhart, Laura Fontanesi, G. Matthew Fricke, Shiguang Fu, Adriana Galván, Remi Gau, Sarah Genon, Tristan Glatard, Enrico Glerean, Jelle J. Goeman, Sergej A. E. Golowin, Carlos González-García, Krzysztof J. Gorgolewski, Cheryl L. Grady, Mikella A. Green, João F. Guassi Moreira, Olivia Guest, Shabnam Hakimi, J. Paul Hamilton, Roeland Hancock, Giacomo Handjaras, Bronson B. Harry, Colin Hawco, Peer Herholz, Gabrielle Herman, Stephan Heunis, Felix Hoffstaedter, Jeremy Hogeveen, Susan Holmes, Chuan-Peng Hu, Scott A. Huettel, Matthew E. Hughes, Vittorio Iacovella, Alexandru D. Iordan, Peder M. Isager, Ayse I. Isik, Andrew Jahn, Matthew R. Johnson, Tom Johnstone, Michael J. E. Joseph, Anthony C. Juliano, Joseph W. Kable, Michalis Kassinopoulos, Cemal Koba, Xiang-Zhen Kong, Timothy R. Koscik, Nuri Erkut Kucukboyaci, Brice A. Kuhl, Sebastian Kupek, Angela R. Laird, Claus Lamm, Robert Langner, Nina Lauharatanahirun, Hongmi Lee, Sangil Lee, Alexander Leemans, Andrea Leo, Elise Lesage, Flora Li, Monica Y.C. Li, Phui Cheng Lim, Evan N. Lintz, Schuyler W. Liphardt, Annabel B. Losecaat Vermeer, Bradley C. Love, Michael L. Mack, Norberto Malpica, Theo Marins, Camille Maumet, Kelsey McDonald, Joseph T. McGuire, Helena Melero, Adriana S. Méndez Leal, Benjamin Meyer, Kristin N. Meyer, Glad Mihai, Georgios D. Mitsis, Jorge Moll, Dylan M. Nielson, Gustav Nilsonne, Michael P. Notter, Emanuele Olivetti, Adrian I. Onicas, Paolo Papale, Kaustubh R. Patil, Jonathan E. Peelle, Alexandre Pérez, Doris Pischedda, Jean-Baptiste Poline, Yanina Prystauka, Shruti Ray, Patricia A. Reuter-Lorenz, Richard C. Reynolds, Emiliano Ricciardi, Jenny R. Rieck, Anais M. Rodriguez-Thompson, Anthony Romyn, Taylor Salo, Gregory R. Samanez-Larkin, Emilio Sanz-Morales, Margaret L. Schlichting, Douglas H. Schultz, Qiang Shen, Margaret A. Sheridan, Jennifer A. Silvers, Kenny Skagerlund, Alec Smith, David V. Smith, Peter Sokol-Hessner, Simon R. Steinkamp, Sarah M. Tashjian, Bertrand Thirion, John N. Thorp, Gustav Tinghög, Loreen Tisdall, Steven H. Tompson, Claudio Toro-Serey, Juan Jesus Torre Tresols, Leonardo Tozzi, Vuong Truong, Luca Turella, Anna E. van 't Veer, Tom Verguts, Jean M. Vettel, Sagana Vijayarajah, Khoi Vo, Matthew B. Wall, Wouter D. Weeda, Susanne Weis, David J. White, David Wisniewski, Alba Xifra-Porxas, Emily A. Yearling, Sangsuk Yoon, Rui Yuan, Kenneth S.L. Yuen, Lei Zhang, Xu Zhang, Joshua E. Zosky, Thomas E. Nichols, Russell A. Poldrack, and Tom Schonberg. 2020. Variability in the Analysis of a Single Neuroimaging Dataset by Many Teams. *Nature* 582, 7810 (June 2020), 84–88. https://doi.org/10.1038/s41586-020-2314-9

[3] Alexander Bowring, Camille Maumet, and Thomas E Nichols. 2019. Exploring the impact of analysis software on task fMRI results. *Human brain mapping* 40, 11 (2019), 3362–3384.

[4] C. Denis, P. De Oliveira Castro, and E. Petit. 2016. Verificarlo: Checking Floating Point Accuracy through Monte Carlo Arithmetic. In *2016 IEEE 23nd Symposium on Computer Arithmetic (ARITH)*. IEEE Computer Society, Los Alamitos, CA, USA, 55–62. https://doi.org/10.1109/ARITH.2016.31

[5] Morgane Des Ligneris, Axel Bonnet, Yohan Chatelain, Tristan Glatard, Michaël Sdika, Gaël Vila, Valentine Wargnier-Dauchelle, Sorina Pop, and Carole Frindel. 2023. REPRODUCIBILITY OF TUMOR SEGMENTATION OUTCOMES WITH A DEEP LEARNING MODEL. In *International Symposium on Biomedical Imaging (ISBI)*. Cartagena de Indias, Colombia. https://hal.science/hal-04006057

[6] Tristan Glatard, Gregory Kiar, Tristan Aumentado-Armstrong, Natacha Beck, Pierre Bellec, Rémi Bernard, Axel Bonnet, Shawn T Brown, Sorina Camarasu-Pop, Frédéric Cervenansky, et al. 2018. Boutiques: a flexible framework to integrate command-line applications in computing platforms. *GigaScience* 7, 5 (2018), giy016.

[7] T. Glatard, L. Lewis, R. Ferreira da Silva, R. Adalat, N. Beck, C. Lepage, P. Rioux, M.-E. Rousseau, T. Sherif, E. Deelman, N. Khalili-Mahani, and A. Evans. 2015. Reproducibility of neuroimaging analyses across operating systems. *Frontiers in Neuroinformatics* (2015), 1–14. https://doi.org/10.3389/fninf.2015.00012

[8] Krzysztof J Gorgolewski, Fidel Alfaro-Almagro, Tibor Auer, Pierre Bellec, Mihai Capotă, M Mallar Chakravarty, Nathan W Churchill, Alexander Li Cohen, R Cameron Craddock, Gabriel A Devenyi, et al. 2017. BIDS apps: Improving ease of use, accessibility, and reproducibility of neuroimaging data analysis methods. *PLoS computational biology* 13, 3 (2017), e1005209.

[9] Mark Jenkinson, Christian F Beckmann, Timothy EJ Behrens, Mark W Woolrich, and Stephen M Smith. 2012. Fsl. *Neuroimage* 62, 2 (2012), 782–790.

[10] Mark Jenkinson and Stephen Smith. 2001. A global optimisation method for robust affine registration of brain images. *Medical image analysis* 5, 2 (2001), 143–156.

[11] David N. Kennedy, Sanu A. Abraham, Julianna F. Bates, Albert Crowley, Satrajit Ghosh, Tom Gillespie, Mathias Goncalves, Jeffrey S. Grethe, Yaroslav O. Halchenko, Michael Hanke, Christian Haselgrove, Steven M. Hodge, Dorota Jarecka, Jakub Kaczmarzyk, David B. Keator, Kyle Meyer, Maryann E. Martone, Smruti Padhy, Jean-Baptiste Poline, Nina Preuss, Troy Sincomb, and Matt Travers. 2019. Everything Matters: The ReproNim Perspective on Reproducible Neuroimaging. *Frontiers in Neuroinformatics* 13 (2019), 1. https://doi.org/10.3389/fninf.2019.00001

[12] Gregory Kiar, Pablo de Oliveira Castro, Pierre Rioux, Eric Petit, Shawn T. Brown, Alan C. Evans, and Tristan Glatard. 2020. Comparing Perturbation Models for Evaluating Stability of Neuroimaging Pipelines. arXiv:1908.10922 [q-bio.NC]

[13] Daniel S. Marcus, Tracy H. Wang, Jamie Parker, John G. Csernansky, John C. Morris, and Randy L. Buckner. 2007. Open Access Series of Imaging Studies (OASIS): Cross-sectional MRI Data in Young, Middle Aged, Nondemented, and Demented Older Adults. *Journal of Cognitive Neuroscience* 19, 9 (09 2007), 1498–1507. https://doi.org/10.1162/jocn.2007.19.9.1498 arXiv:https://direct.mit.edu/jocn/article-pdf/19/9/1498/1936514/jocn.2007.19.9.1498.pdf

[14] Thomas E Nichols, Samir Das, FAU Eickhoff, Alan C Evans, Tristan Glatard, Michael Hanke, Nikolaus Kriegeskorte, Michael P Milham, Jean-Baptiste Poldrackand Poline, Bertrand Proal, Erikaand Thirion, and B T Thomas Van Essenand White, Tonyaand Yeo. 2017. Best practices in data analysis and sharing in neuroimaging using MRI. *Nature neuroscience* (2017), 299–303. https://doi.org/10.1038/nn.4500

[15] Guiomar Niso, Rotem Botvinik-Nezer, Stefan Appelhoff, Alejandro De La Vega, Oscar Esteban, Joset A. Etzel, Karolina Finc, Melanie Ganz, Rémi Gau, Yaroslav O. Halchenko, Peer Herholz, Agah Karakuzu, David B. Keator, Christopher J. Markiewicz, Camille Maumet, Cyril R. Pernet, Franco Pestilli, Nazek Queder, Tina Schmitt, Weronika Sójka, Adina S. Wagner, Kirstie J. Whitaker, and Jochem W. Rieger. 2022. Open and reproducible neuroimaging: From study inception to publication. *NeuroImage* 263 (2022), 119623. https://doi.org/10.1016/j.neuroimage.2022.119623

[16] Douglass Stott Parker. 1997. *Monte Carlo Arithmetic: Exploiting Randomness in Floating-Point Arithmetic.* University of California (Los Angeles). Computer Science Department.

[17] Ameera X. Patel, Prantik Kundu, Mikail Rubinov, P. Simon Jones, Petra E. Vértes, Karen D. Ersche, John Suckling, and Edward T. Bullmore. 2014. A Wavelet Method for Modeling and Despiking Motion Artifacts from Resting-State fMRI Time Series. *NeuroImage* 95 (July 2014), 287–304. https://doi.org/10.1016/j.neuroimage.2014.03.012

[18] Jonathan D Power, Kelly A Barnes, Abraham Z Snyder, Bradley L Schlaggar, and Steven E Petersen. 2012. Spurious but Systematic Correlations in Functional Connectivity MRI Networks Arise from Subject Motion. *Neuroimage* 59, 3 (Feb. 2012), 2142–2154. https://doi.org/10.1016/j.neuroimage.2011.10.018

[19] Angela I. Renton, Thanh Thuy Dao, David F. Abbott, Saskia Bollmann, Megan E. J. Campbell, Jeryn Chang, Thomas G. Close, Korbinian Eckstein, Gary F. Egan, Stefanie Evas, Kelly G. Garner, Marta I. Garrido, Anthony J. Hannan, Renzo Huber, Tom Johnstone, Jakub R. Kaczmarzyk, Lars Kasper, Levin Kuhlmann, Kexin Lou, Paris Lyons, Jason B. Mattingley, Akshaiy Narayanan, Franco Pestilli, Aina Puce, Fernanda L. Ribeiro, Nigel C. Rogasch, Thomas B Shaw, Paul F. Sowman, Gershon Spitz, Ashley Stewart, Ryan P. Sullivan, David J. White, Xincheng Ye, Judy D. Zhu, Aswin Narayanan, and Steffen Bollmann. 2022. Neurodesk: An accessible, flexible, and portable data analysis environment for reproducible neuroimaging. *bioRxiv* (2022). https://doi.org/10.1101/2022.12.23.521691 arXiv:https://www.biorxiv.org/content/early/2022/12/23/2022.12.23.521691.full.pdf

[20] Ali Salari, Yohan Chatelain, Gregory Kiar, and Tristan Glatard. 2021. Accurate simulation of operating system updates in neuroimaging using Monte-Carlo arithmetic. In *Uncertainty for Safe Utilization of Machine Learning in Medical Imaging, and Perinatal Imaging, Placental and Preterm Image Analysis: 3rd International Workshop, UNSURE 2021, and 6th International Workshop, PIPPI 2021, Held in Conjunction with MICCAI 2021, Strasbourg, France, October 1, 2021, Proceedings 3*. Springer, 14–23.

[21] Nicolas Vallet, David Michonneau, and Simon Tournier. [n. d.]. Toward practical transparent verifiable and long-term reproducible research using Guix. *Scientific Data* 9 ([n. d.]). https://doi.org/10.1038/s41597-022-01720-9

A APPENDIX A: GRID'5000 OARSUB

Grid'5000 nodes can be accessed through different site frontends. Once connected on a site frontend, users can reserve resources (nodes) using the OAR batch scheduler. Listing 1 gives an example of how to ask for an interactive reservation of one node from the hercule cluster for 4 hours during day time with the OAR scheduler.

Listing 1: Grid'5000 example of an interactive node reservation from the `hercule` cluster for 4 hours with OAR scheduler

```
oarsub -I -l host=1,walltime=4:00 -t day -p hercule
```

B APPENDIX B: FSL COMMAND LINE

The application was run with minimal configuration using the command shown in Listing 2, where `$FILE` was the path to the input brain scan.

Listing 2: FSL command line

```
flirt -in $FILE\
      -ref $FSLDIR/data/standard/MNI152_T1_1mm.nii.gz\
      -omat $FILE.mat -o $FILE.nii.gz
```

C APPENDIX C: GUIX RELOCATABLE PACKAGE

Guix allows to build archives that can be used on any machine that does not have Guix, so that one can run the exact same binaries on different machines. Listing 3 gives the command line to create a relocatable package.

Listing 3: Guix comand line to create a relocatable package

```
guix pack --tune=skylake -RR -f tarball -S /sbin=sbin
 -S /share=share -S /lib=lib -S /config=config
-S /etc=etc -S /var=var -S /extras=extras
-S /nvvm=nvvm -S /libexec=libexec -S /src=src
-S /doc=doc -S /python=python -S /lib64=lib64
-S /bin=bin -S /refdoc=refdoc -S /data=data
-S /include=include -S /tcl=tcl
--save-provenance
--manifest=${MANIFEST}/manifest:skylake.scm
```