Methods and Tools to Make Research Source Code Shareable and Reproducible

A Brief Overview
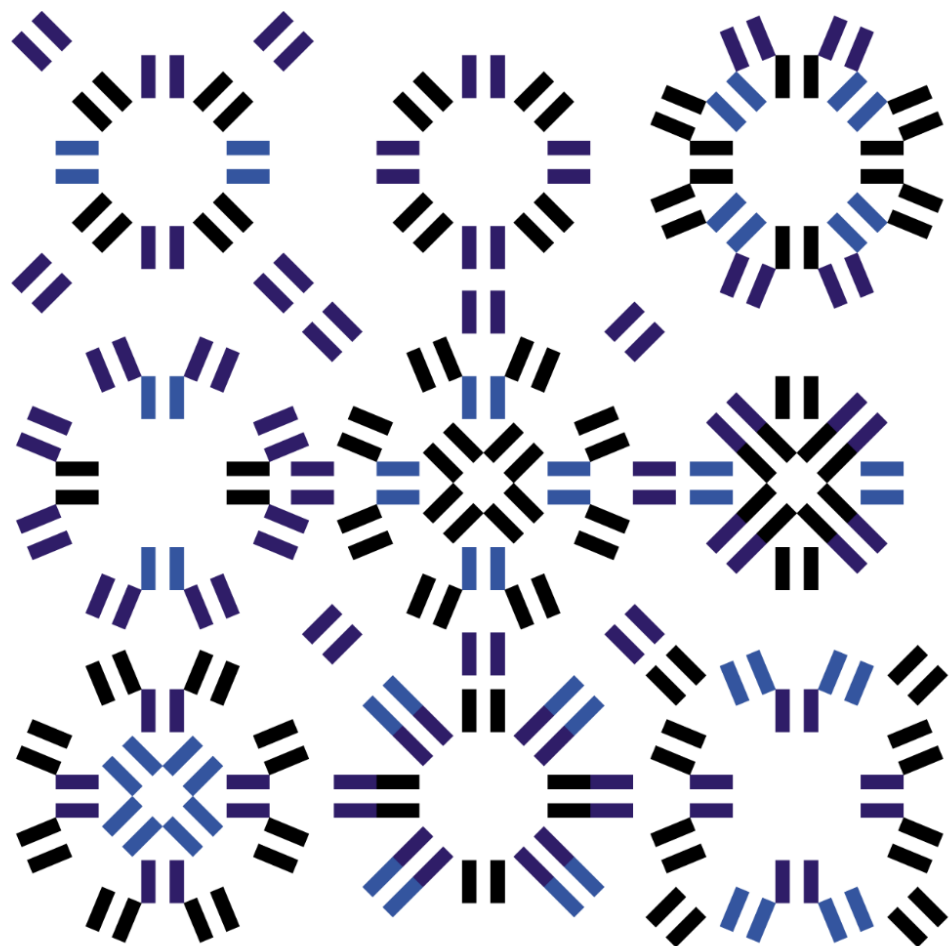
Damien Belvèze

damien.belveze@univ-rennes.fr

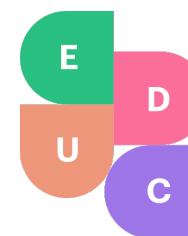Université de Rennes

Université de Rennes

2025-11-05

Methods and Tools to Make Research Source Code Shareable and Reproducible: A Brief Overview

Semaine du libre accès 2025
Université Paris Nanterre

E D U C
European Digital UniverCity
WIDE

Université de Rennes

1. Where scholars can seek help to make their research source code shareable and reproducible?

2. CODE methodology (Collaborate, Open, Document, Execute)

3. focus on the E (Execute = replicability)

Figure 1

CC-by:4.0 Damien Belvèze

Université de Rennes

1. Where scholars can seek help for their source code.

Figure 2

Université de Rennes

If we had an OSPO

PhD training :

- Developping and sharing best practices
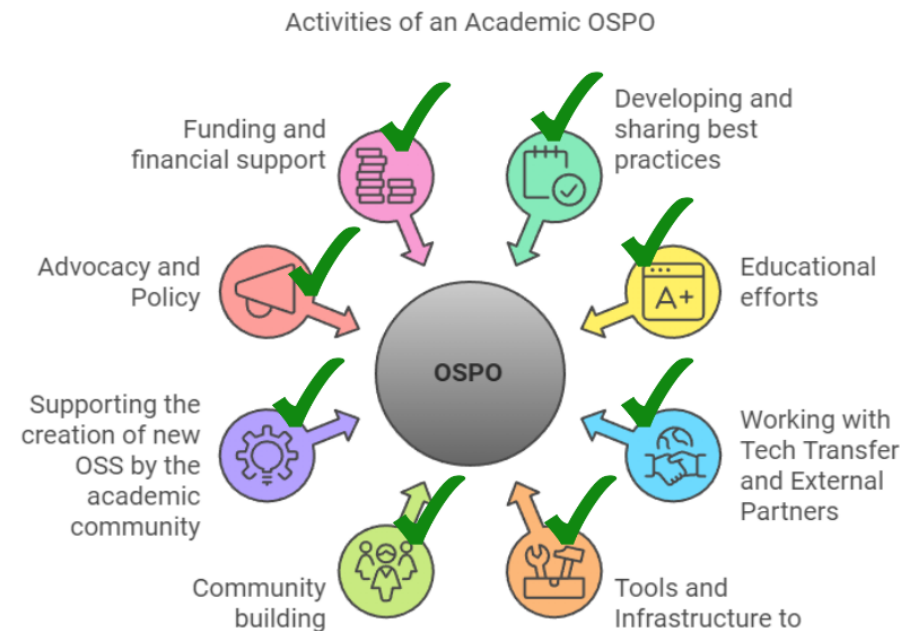
Metadata curation :

- how to cite, describe a software (SMP)

Links with other outputs and archiving :

- HAL, Data repositories, Software Heritage

could be covered by others :

- licences (Innovation & Legal Departements)
- infrastructures (i.e Gitlab forge by IT Staff)
- Best development practices promoted by research

Activities of an Academic OSPO

Funding and financial support — Developing and sharing best practices — Educational efforts — Working with Tech Transfer and External Partners — Tools and Infrastructure to — Community building — Supporting the creation of new OSS by the academic community — Advocacy and Policy

Speaker notes

SMP stands for *Software Management Plan* ; since 2025 it has been possible to use the dmp.opidor template to write a Software Management Plan and link it to one's datasets Academic OSPOS are local coalitions of diverse stakeholders who work in the field of Free and Open Source Software (FOSS). These stakeholders include:

- University departments (innovation, legal, research teams in computational studies, IT staff)
- Library departments dedicated to Open Science
- Local private actors specializing in FOSS development

The first academic OSPO in France is located in Grenoble Alpes University and was funded in september 2025. At the University of Rennes, we are not yet organized as an OSPO, so we keep on trying to provide consistent advice to researchers who seek our help.

Université de Rennes

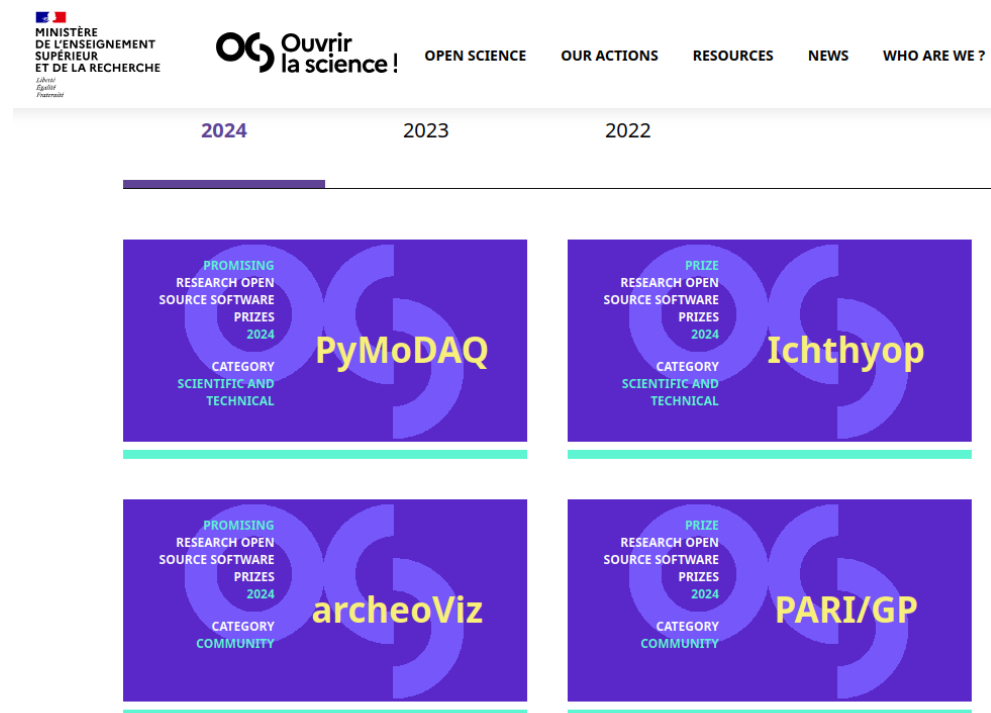# questions received show raising concerns on Software reproducibility :

- Do I have the right to reuse this piece of code in my application?

- which licence should I use to publish my code?

- How should I describe my software?

- How can I credit contributors in my code?

- How could I comply with Open Science principles while developing my software (and get a prize for it!)

Figure 3

Université de Rennes

questions received show raising concerns on Software reproducibility :

- Do I have the right to reuse this piece of code in my application?

- which licence should I use to publish my code?

- How should I describe my software?

- How can I credit contributors in my code? - **How could I comply with Open Science principles while developing my software (and get a** prize **for it!)**

2. CODE methodology

Figure 4

Code Beyond FAIR 📖 Rougier et al. (2025)

OPEN

- **Publish your source code on a public forge**   **mandatory**
- **Save your repository on dedicated archive**   **mandatory**
- **License your code with an open license**   **strongly recommended**
- Declare authorship & rightholders   recommended

DOCUMENT

- Choose meaningful names   recommended
- Comment code   recommended
- Provide examples, notebooks & tutorials   recommended
- Describe API   optional

EXECUTE

- List software & hardware dependencies   recommended
- Provide a computational environment   optional
- Implement a test suite   optional
- Show real-life usage example with expected results   optional

COLLABORATE

- Respond to issues & feature requests   optional
- Describe maintenance, features & support limits   optional
- Explain how to contribute   optional
- Build and animate a community   optional

Speaker notes

Summary of the CODE progressive road map organized in four blocks: Open, Document, Execute and Collaborate

Université de Rennes

Open as public (=OPEN) 📖 Cosmo et al. (2020)
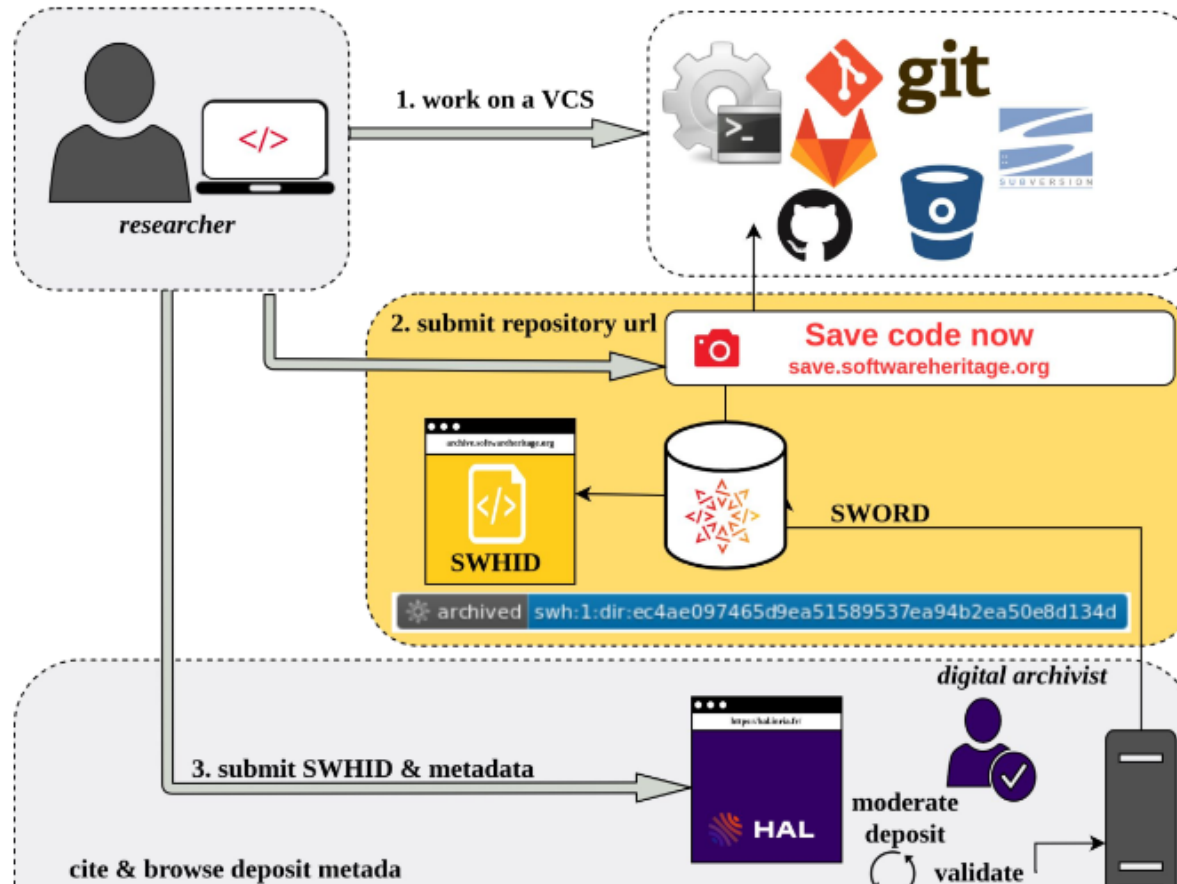
repositories on forges can be damaged or deleted by their owners, including by accident, and the forges themselves may be closed down, as it happened to two popular forges that were shut down in 2015 (Gitorious) and 2016 (Google code), invalidating all the URLs that link to them.

📖 Rougier et al. (2025)

CC-by:4.0 Damien Belvèze

⁂ Université de Rennes

Open as "open to contributions" (=COLLABORATE)

- Should I invite others to send issues?

- Should I invite others to send merge requests?

- How can I credit others for their contributions?

- What governance mechanisms protect the original intent of my software from being modified by others?

Documentation

README (capitals) file sorted on top of a list (in ASCII order) ▓ Abdelhafith (2015)

README file should contain at least:

goals and fonctions of the software environment (system and software stack required) installation and execution commands

Other information could be parsed elsewhere : contribution -> CONTRIBUTING
authors -> CITATION.cff)
versions notes -> CHANGELOG
global informations in a machine readable format (JSON) -> CODEMETA

```
[README.TXT is the DOC file for SPICE/SINC/SLIC]

This failsafe tape contains the circuit analysis programs:

        SPICE SINC and SLIC

described in the Applications Software Bulletin Volume 4.

requirements:

SPICE requires FORTRAN-10 version 4 because of  its  use  of
Right adjusted Holerith data.  Executes in about 47K.

SINC and SLIC should work with F40 and F10v1A but  have  not
been  tested with anything but F10 version 4.  SLIC requires
about 50K of core to execute.

FORTRAN-10 version 4 should be able to compile all three  in
less than 50K user core.

Note:  at one time there was a  hacked  up  version  of  F10
version  1  which  was  able  to compile SPICE.  It was only
guarenteed to compile the version of SPICE on this tape  and
had  many  severe  bugs.   With  this distribution I will no
longer make any attempt  to  distribute  that  compiler  nor
answer any questions on it.  -ADG

Performance:

Of the three programs, SPICE is generaly  reguarded  as  the
most  useful.   The  optimized version is roughly 1K smaller
and about 20% faster than the non-optimized version.

SLIC and SINC are about 10% faster when optimized.

Under 6.01 SPICE has  the  following  paging  profile  while
running the test data:

        phys page limit                 cpu seconds

        94 p                            16.2
        85 p                            22
        80 p - 45 p                     25
        40 p                            127
        10 p                            over 6 min.

I/O units

SPICE reads via 'DIALOG', writes SPCOUT.DAT
```

Université de Rennes

comment your code

comment it yourself, don't use a GenAI for that purpose

```python
In [12]: plt.figure(figsize = (10, 10))
         ax = plt.axes()

         plt.title("Sensor Positions and Ranges", fontsize = 18, y = 1.01)
         plt.xlabel("$x$ (cm)", fontsize = 16)
         plt.ylabel("$y$ (cm)", fontsize = 16)

         # Choose a display range that reflects the reference frame and encompasses
         all sensor positions
         plt.xlim(0, 900)
         plt.ylim(0, 900)

         for sensor in data:
             x, y, d = data[sensor]
             # Plot the sensor position
             ax.plot(x, y, 'go', ms = 8)
             # Annotate sensor positon with sensor's ID.
             # Offset the annotation from the marker so that they do not overlap.
             ax.annotate(sensor, (x + 10, y + 10), fontsize = 12)
             # Plot sensor's range
             circle = patches.Circle(
                 (x, y),
                 radius = d,
                 alpha = 0.2,
                 color = 'g',
                 linewidth = 5
```

Speaker notes

it's better to comment every function as you write it. These comments will be invaluable for researchers who are not developers by profession to understand better what the code actually does. It's also important for oneself later ; don't ask ChatGPT to comment it for you, you could lose control and over your source code and compromise its confidentiality. Some tools, like Doxygen, can automatically generate documentation for functions.

Université de Rennes

3. A focus on replicability and execution
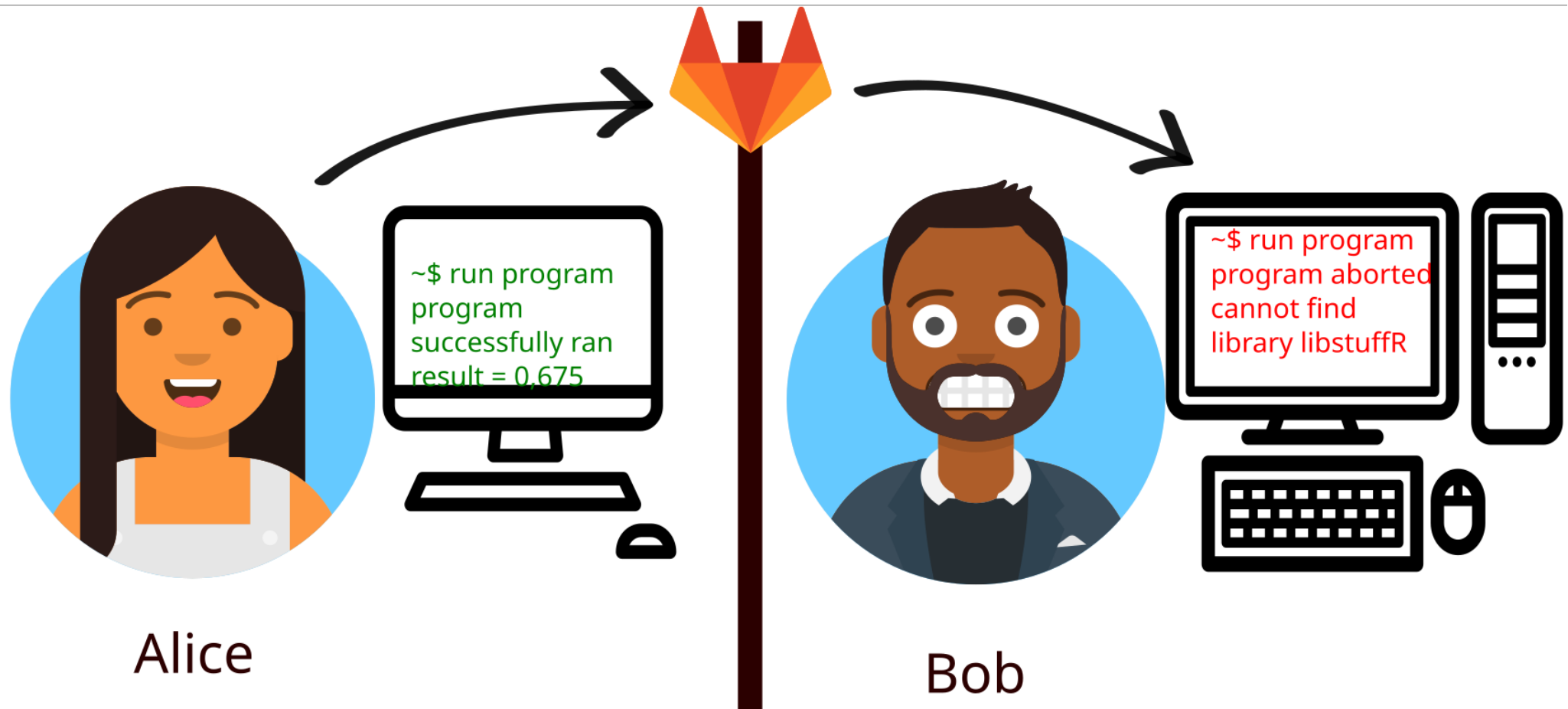
Figure 6

Université
de Rennes

replicability vs reproducibility

- **reproducibility** = same software but other data

- **replicability** = same software, same data (we only consider test data)

CC-by:4.0 Damien Belvèze

Université de Rennes

missing dependencies

Alice

Bob

~$ run program
program
successfully ran
result = 0,675

~$ run program
program aborted
cannot find
library libstuffR

Speaker notes

The code written and correctly run on Alice's computer does not work - or does not work the same way on Bob's computer, since Bob's does not have the same software stack used by Alice on his computer. Clearly there's a missing dependency on Bob's computer. When he tries to install it, he may fail because other dependencies are missing. Bob is propelled in what is called sometimes the 'Dependency Hell'

Université
de Rennes

Language specific dependencies

cranly dependence tree for package names with
"leaflet"
CRAN database version
lun., 20 oct. 2025, 15:35
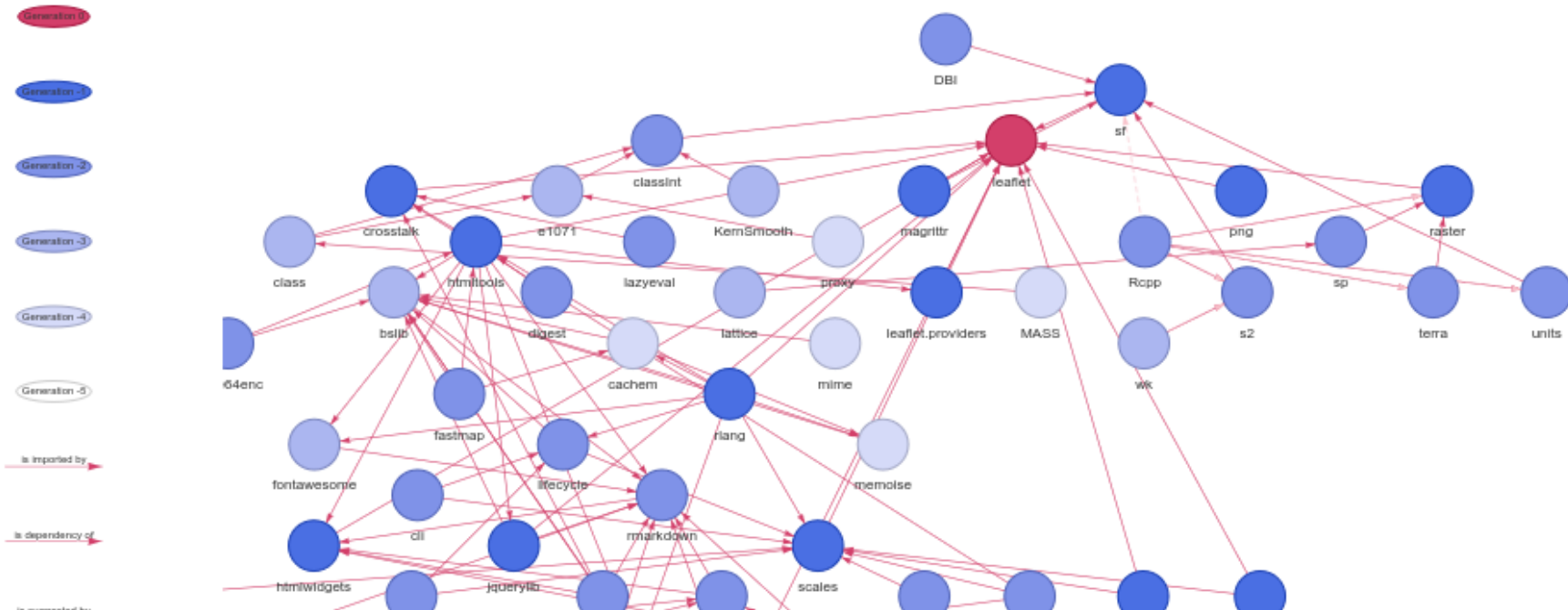
Speaker notes

In the following figure, we have reproduced the five-levels graph of internal dependencies within CRAN concerning the Leaflet dependency. The Leaflet package, which allows data to be represented on map backgrounds, relies, among other things, on the raster and sf libraries, which are well known to cartographers who use R. Sf itself works using the s2 and dbi libraries. It should be noted that other components necessary for the Leaflet package to function in R are not shown here because they are not part of CRAN. This is the case, for example, with the gdal and geos system dependencies, which are also commonly used for mapping. We will come back to this later.

Université de Rennes

virtual environments with Python


```
        e@pr030165:~/Bureau/python_code$ python3 -m venv .venv
        e@pr030165:~/Bureau/python_code$ source .venv/bin/activate
(.venv)    e@pr030165:~/Bureau/python_code$ pip install pypandoc
Collecting pypandoc
  Using cached pypandoc-1.15-py3-none-any.whl.metadata (16 kB)
Using cached pypandoc-1.15-py3-none-any.whl (21 kB)
Installing collected packages: pypandoc
Successfully installed pypandoc-1.15
(.venv)    e@pr030165:~/Bureau/python_code$ pip freeze > requirements.txt
(.venv)    e@pr030165:~/Bureau/python_code$ cat requirements.txt
pypandoc==1.15
(.venv)    e@pr030165:~/Bureau/python_code$
```

Speaker notes

this image shows the simple commands you have to use, once you have virtualenv installed on your computer to create a virtual environment for Python. Here's an example where we load a pandoc version for Python in a new virtual environment. The command `pip freeze` is used to freeze a specific version of this package, the one just loaded, in file called requirements.txt It's not recommended to share whole virtual environment within a github or gitlab repository, because the binaries which the requirements.txt refers to are far too heavy ; we only need to share the requirements.txt file. On their computer, people who need to run this script will only have to install command for dependencies mentioned in requireemnts.txt : `pip -r install requirements.txt`

Université de Rennes

virtual environments with R

Figure 8

```
1  renv::init() # open a virtual environment, captured packages versions are s
2  renv::status() # make sure that a new package has not been loaded without h
3  renv::screenshot() # capture all packages loaded in R (not only those menti
4  renv::restore() # restore all packages captured from other's person project
```
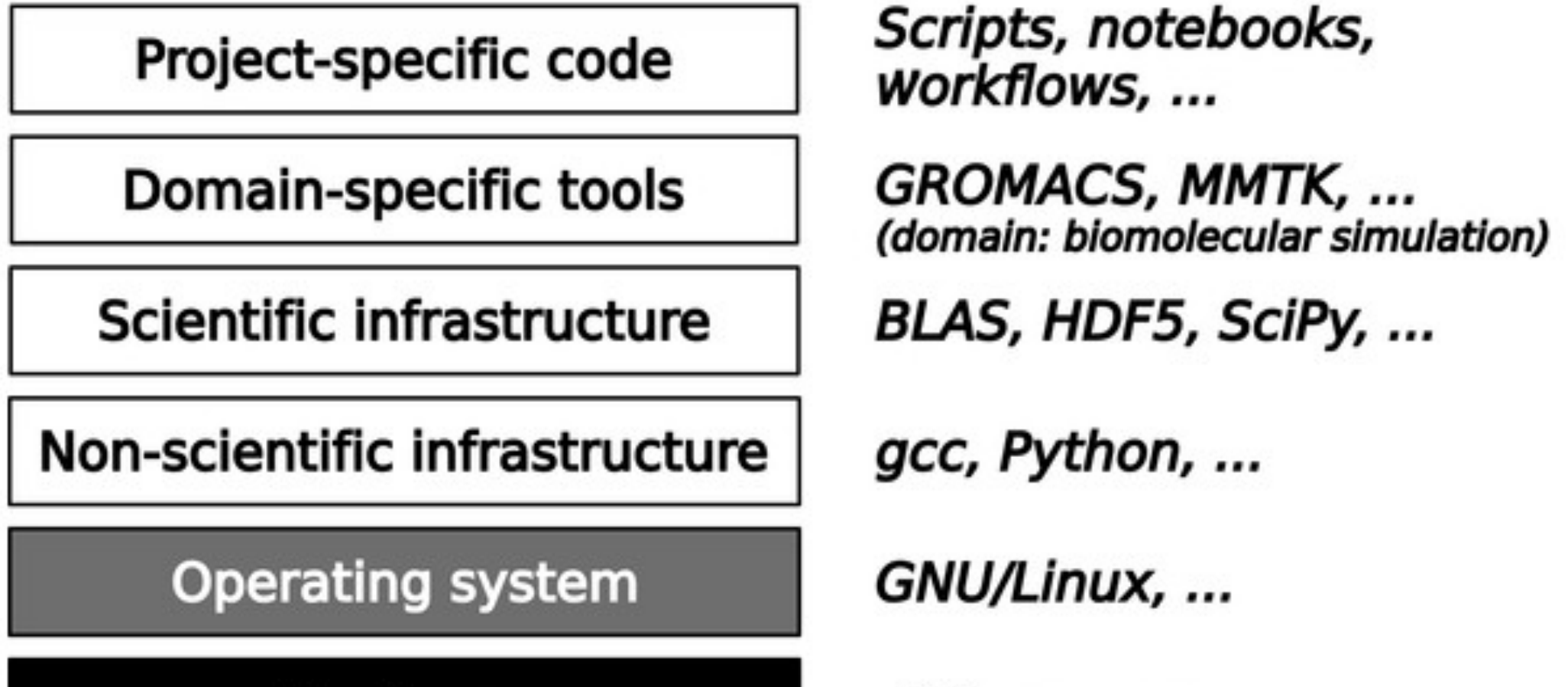
📓 *Package Renv* (n.d.)

Université de Rennes

beyond virtual environments and software dependencies 📓
Hinsen (2018)

Project-specific code	*Scripts, notebooks, workflows, ...*
Domain-specific tools	*GROMACS, MMTK, ...* *(domain: biomolecular simulation)*
Scientific infrastructure	*BLAS, HDF5, SciPy, ...*
Non-scientific infrastructure	*gcc, Python, ...*
Operating system	*GNU/Linux, ...*

Speaker notes

Guix language cannot cover the more profond layers of the pile : neither the operating system, nor the processor's architecture That is why we need in some cases to associate Guix with containerization tool such as Docker

Université de Rennes

- Guix package manager allows to **manage on the same computer different versions of the same package**

- Guix package is agnostic : R and Python are covered (better coverage for R than for Python)

- Guix declarative nature makes simpler to **define exactly which version of a package was used** for a given source code

- **Roll back** is always possible (get back to the previous version used of a software)

Guix

Speaker notes

Guix, like Nix,; comes as a package manager and an operating system. The package manager can be used on other GNU/Linux operating systems. Guix (written in a Domain Specific Language : Guile Scheme) is **functional** : same inputs (configuration files) and same channels (binaries sources) will always be processed to have the same outputs and any change can be rolled back to a previous configuration if something went wrong. This makes configurations perfectly repeatable on one machine as well as on several machines. Each single configuration with a specific set of dependencies (each dependency served in a specific version) contitutes a **profile**. This profile co-exists with previous profiles (combination of the same packages but with different versions or combination or different packages).

Université de Rennes

different profiles, different machines, same confirations
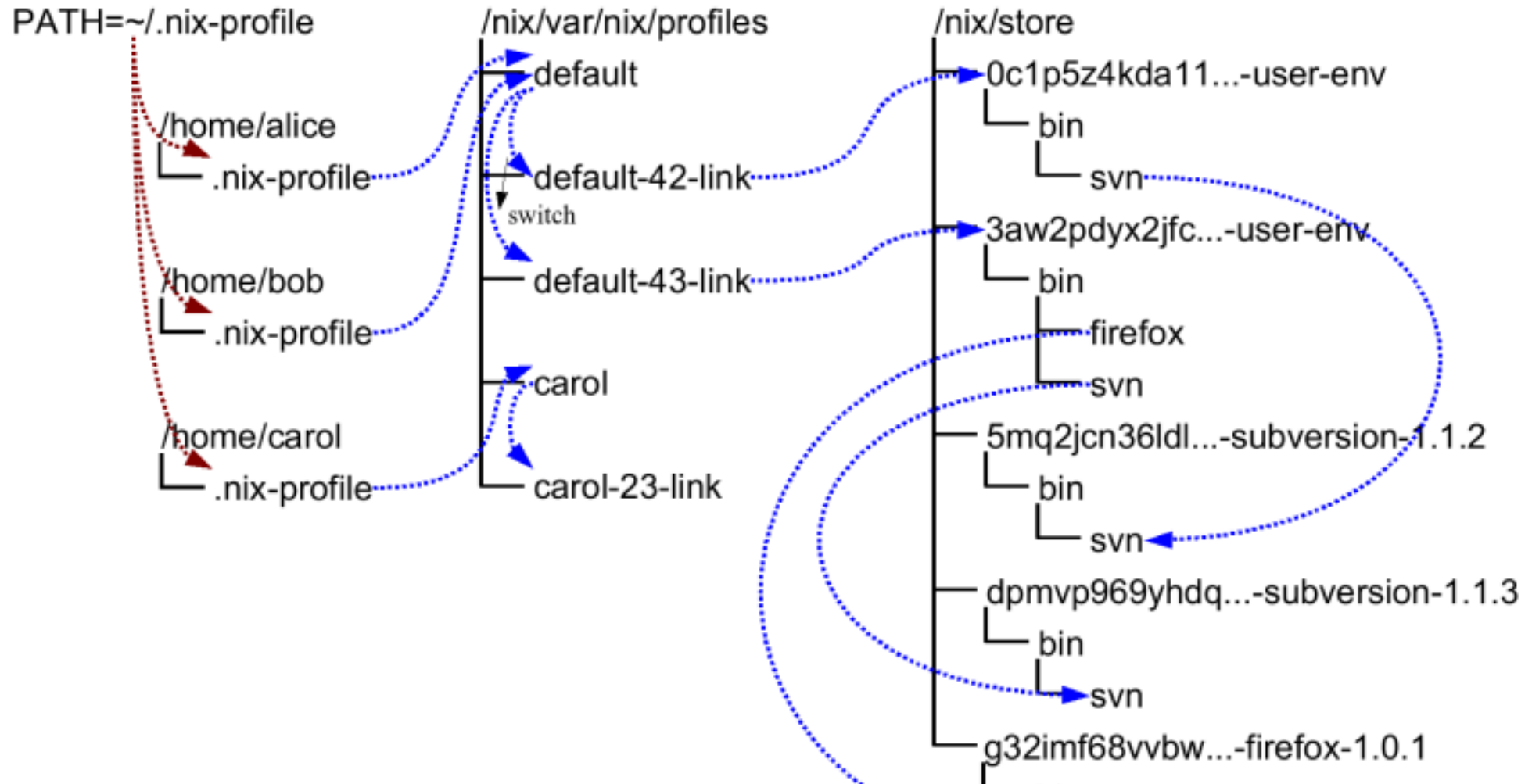
CC-by:4.0 Damien Belvèze

how it works

Any time we need a package, we make `guix install <package>`from the shell or use a different channel than the main one (for instance CRAN for R packages) ; then the command will be `guix import <repository> -r <package>`

```
 1  guix import cran -r wikidataR > manifest.scm
 2  guix import cran -r leaflet > manifest.scm
 3  # imports wikidataR and Leaflet from the CRAN channel which extends guix ch
 4
 5  # in the manifest.scm file, hashs will identify specific versions of each p
 6
 7  #1l0slppa61hmzkqj9wmplb9ldsyvg61igsd9dlv5yx06k2by77xg for Leaflet v.2.2.3
 8  #120833b7zyq1rhmn9c8iv0j6br60af7gbn5lc4dil55qhh2lp9rx for wikidataR v.2.3.3
 9
10  #to run later the war_cemeteries Rscript with the specific versions of Leaf
11  guix shell -m manifest.scm -- R -e 'war_cemeteries.R'
```
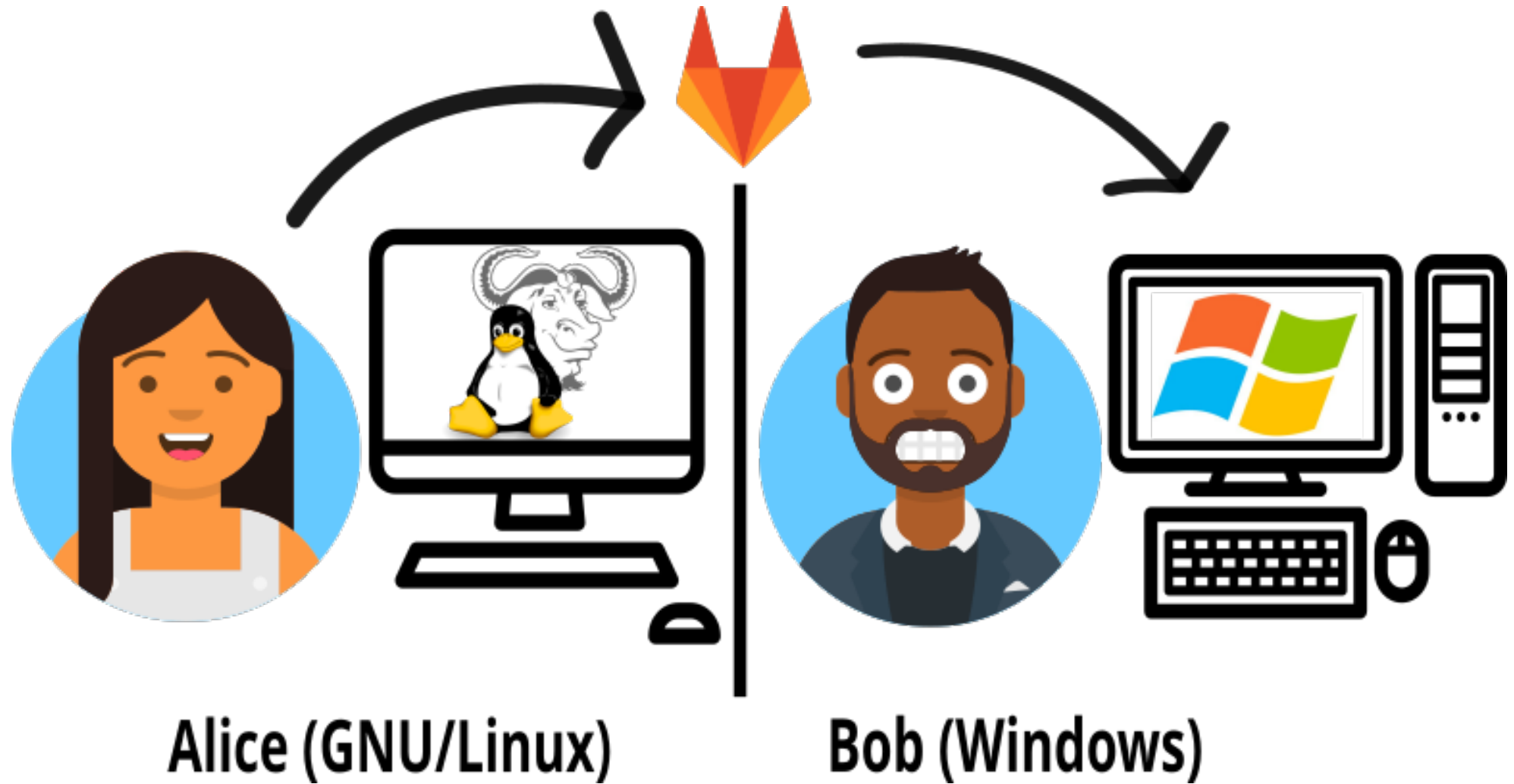
Université de Rennes

Figure 10

Containers are usefull to reproduce software on different systems (AMD or ARM, GNUX/Linux or Windows)

Alice (GNU/Linux)

Bob (Windows)

Université de Rennes

Docker without Guix

```
1   FROM rocker/r-ver:4
2   RUN echo 'options(repos = c(CRAN = "https://cloud.r-project.org"))' >>"${R_HOME}/etc/Rprofile.site"
3   RUN apt-get update && apt-get install -y \
4       software-properties-common \
5       libudunits2-dev \
6       libgdal-dev \
7       libgeos-dev \
8       libproj-dev \
9       libsqlite3-dev \
10      pandoc \
11      && apt-get clean \
12      && rm -rf /var/lib/apt/lists/*
13
14  RUN R -e "install.packages('renv', repos = c(CRAN = 'https://cloud.r-project.org'))"
15  WORKDIR /code_r
16  COPY . .
17  RUN R -e "renv::restore()"
```

Speaker notes

In order to package in a container our source code and the software stack needed to make it run correctly, we need to write a Dockerfile (some kind of recipe) which will mention:

- a base image for the containerization (same process as for a continuous integration) ; here we use Rocker (r-ver) which is a r-base image baded on Ubuntu 20.4.
- then we load in Ubuntu some system dependencies that are needed to make the R packages work (GDAL, GEOS, etc.)
- then we load in R-base the package Renv
- and from the Renv.lock we load in R-base all the needed R-packages
- finaly we execute the source code which should generate a html page in the container

Université
de Rennes

Docker with Guix 📓 Tournier (2024)

You don't need either a Dockerfile nor a specific image like Ubuntu or Rocker to build the image. A single command and the manifest.scm will do the job for you.

```
1  guix pack -f docker -m manifest.scm
2
3  #this command packages the script and the dependencies included in manifest
4
5  docker load
```

as a conlusion

Findability : HAL

Accessibility : Software Heritage

Interoperability : virtual environments, containers, replicable packages managers

Reusability : Licence, Documentation, tests

Data Without Software Are Just Numbers

📓 Davenport et al. (2020)

Université de Rennes

figures

figure	source et crédits
Figure 1	cover of a video shot by Issaquah Library https://www.youtube.com/watch?v=TOjBXuJeuRk
Figure 6	Library as a R package : made with ChatGPT with the following prompt : "fill an hexagone (the common shape for R packages logos) with a drawing of a uni library"

Université de Rennes

figure	source et crédits
Figure 3	ARDoISE Data Hub's official logo (ARDoISE is The Rennes data hub)
Figure 5	University of Buffalo https://www.math.buffalo.edu/~badzioch/MTH337/Report_guide/report9.html
Figure 7	made with R package Cranly
Figure 9	from https://linuxfr.org/news/nix-1-7-nixpkgs-et-nixos-14-04-guix-0-6

Université de Rennes

figure	source et crédits
Figure 10	image made with Avataars by Damien Belvèze, CC-by ; original idea by Candice Savonen
Figure 11	see https://gitlab.huma-num.fr/dbelveze/guide_replicability_practice/-/blob/main/code_r/Dockerfile?ref_type=heads

Université de Rennes

software used for this presentation

```
[1] "Quarto version: 1.6.40"

R version 4.5.2 (2025-10-31)
Platform: x86_64-pc-linux-gnu
Running under: Ubuntu 24.04.3 LTS

Matrix products: default
BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.12.0
LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.12.0  LAPACK version
3.12.0

locale:
 [1] LC_CTYPE=fr_FR.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=fr_FR.UTF-8        LC_COLLATE=fr_FR.UTF-8
 [5] LC_MONETARY=fr_FR.UTF-8    LC_MESSAGES=fr_FR.UTF-8
 [7] LC_PAPER=fr_FR.UTF-8       LC_NAME=C
```

Université de Rennes

References

Abdelhafith, O. (2015). README.md: History and Components. In *Medium*.

Cosmo, R. di, Gruenpeter, M., Marmol, B., Monteil, A., Romary, L., & Sadowska, J. (2020). Curated archiving of research software artifacts: Lessons learned from the french open archive HAL. *International Journal of Digital Curation*, *15*(1), 16–16. https://doi.org/10.2218/ijdc.v15i1.698

Davenport, J. H., Grant, J., & Jones, C. M. (2020). Data Without Software Are Just Numbers. *Data Science Journal*, *19*(1). https://doi.org/10.5334/dsj-2020-003

Hinsen, K. (2018). Verifiability in computer-aided research: The role of digital scientific notations at the human-computer interface. *PeerJ Computer Science*, *4*, e158. https://doi.org/10.7717/peerj-cs.158

Louvet, V. (2025). *Official opening of the UGA OSPO*. UGA.

Package renv : Présentation et retour d'expérience. (n.d.). Retrieved May 13, 2025, from https://elisemaigne.pages.mia.inra.fr/2021_package_renv/presentation.html#41

Rougier, N., Di Cosme, R., Hinsen, C., Maurice, C., Le Berre, D., Monat, R., Louvet, V., Jullien, N., Granger, S., & Maumet, C. (2025). *Code beyond FAIR*. https://inria.hal.science/hal-04930405v1

Tournier, S. (2024, December). (Re)déploiement de conteneurs et machines virtuelles avec guix. *JRES (Journées réseaux de l'enseignement Et de La Recherche)* 2024

Université de Rennes