Computational reproducibility with GNU Guix:
application to multiparametric cytometry

Simon Tournier[1] & Nicolas Vallet[2]

simon.tournier@u-paris.fr nicolas.vallet@inserm.fr

iPOP-UP, September 9th, 2021

[1]Research Eng. Sci. Computing (+little admin-sys)
[2]PhD student (and MD)

PARIS DIDEROT
Université de Paris

Science = Transparency
Scientific result = Experiment + Numerical treatment

Science at the numerical age:

1. Open Article — HAL, BioArxiv
2. Open Data — Zenodo
3. Open Source — GitHub, Software Heritage

and how to glue all that?

$$\begin{pmatrix} \text{reproductibility} & = & \text{verification} \\ \text{replicability} & = & \text{validation} \end{pmatrix}$$

R. Di Cosmo@Scibian2016 ~~(pdf)~~
K. Hinsen@Aramis2019 (pdf)

Science = Transparency

Scientific result = Experiment + Numerical treatment

Science at the numerical age:

1.	Open Article	HAL, BioArxiv
2.	Open Data	Zenodo
3.	Open Source	GitHub, Software Heritage
4.	Computational env.	**?**

and how to glue all that?

$$\begin{pmatrix} \text{reproductibility} & = & \text{verification} \\ \text{replicability} & = & \text{validation} \end{pmatrix}$$

R. Di Cosmo@Scibian2016 ~~(pdf)~~

K. Hinsen@Aramis2019 (pdf)

Redo (reproduce or replicate) a result?

		audit		opaque		depend?
result	⟵	**paper**	+	data	+	analysis
data	⟵	**protocol**	+	instruments	+	materials (reagent, etc.)
analysis	⟵	**script**	+	data	+	environment

- **audit** is the “easy” part
- opaque is generally the hard part

Redo (reproduce or replicate) a result?

		audit		opaque		depend?
	result	⟵ **paper**	+	data	+	analysis
	data	⟵ **protocol**	+	instruments	+	materials (reagent, etc.)
▷	analysis	⟵ **script**	+	data	+	environment

- **audit** is the “easy” part
- opaque is generally the hard part

Redo (reproduce or replicate) a result?

		audit		opaque		depend?
	result	⟵	**paper**	+ data	+	analysis
	data	⟵	**protocol**	+ instruments	+	materials (reagent, etc.)
▷	analysis	⟵	**script**	+ data	+	environment

- **audit** is the “easy” part
- opaque is generally the hard part
- how to evacuate depend? from the equations

Redo (reproduce or replicate) a result?

		audit		opaque		depend?
	result	⟵ **paper**	+	data	+	analysis
	data	⟵ **protocol**	+	instruments	+	materials (reagent, etc.)
▷	analysis	⟵ **script**	+	data	+	environment

- **audit** is the “easy” part
- opaque is generally the hard part
- how to evacuate depend? from the equations. . .
  . . . at least let speak about environment

Concrete scenarii about environment

- Alice used numerical tools `R@4.1.1`, `FlowCore@2.4` and `CATALYST@1.16.2`.
- Carole collaborates with Alice ...
  but also requires `R@4.0.4`, `FlowCore@2.0` for another project.
- Charlie upgrades their system then all is broken.
- Bob runs the same versions as Alice ...
  but does not get the same outputs.
- Dan tries to redo the analysis months (years?) later ...
  but hits the dependencies hell[1].

[1] at best :-)

Concrete scenarii about environment

- Alice used numerical tools `R@4.1.1`, `FlowCore@2.4` and `CATALYST@1.16.2`.
- Carole collaborates with Alice ...
  but also requires `R@4.0.4`, `FlowCore@2.0` for another project.
- Charlie upgrades their system then all is broken.
- Bob runs the same versions as Alice ...
  but does not get the same outputs.
- Dan tries to redo the analysis months (years?) later ...
  but hits the dependencies hell[1].

Solution(s)

1. **package manager**: `apt`, `yum`, etc.
2. **virtual environment**: `conda`, `modulefiles`, etc.
3. **container**: Docker, Singularity, etc.

} **Guix**

[1] at best :-)

Why is it complicated?

```
alice@laptop$ R -e '1+2'
> 1+2
[1] 3
```

Why is it complicated?

```
alice@laptop$ R -e '1+2'
> 1+2
[1] 3
```

What is the issue? **R is Open Source**
https://svn.r-project.org/R/trunk/src/main/**main.c**

Why is it complicated?

```
alice@laptop$ R -e '1+2'
> 1+2
[1] 3
```

What is the issue? **R is Open Source**
https://svn.r-project.org/R/trunk/src/main/**main.c**

```
alice@laptop$ file /usr/lib/R/bin/exec/R
/usr/lib/R/bin/exec/R: executable, dynamically linked, ...
```

Why is it complicated?

```
alice@laptop$ R -e '1+2'
> 1+2
[1] 3
```

What is the issue? **R is Open Source**
https://svn.r-project.org/R/trunk/src/main/**main.c**

```
alice@laptop$ file /usr/lib/R/bin/exec/R
/usr/lib/R/bin/exec/R: executable, dynamically linked, ...
```

Why is it complicated?

```
alice@laptop$ R -e '1+2'
> 1+2
[1] 3
```

What is the issue? **R is Open Source**
https://svn.r-project.org/R/trunk/src/main/**main.c**

```
alice@laptop$ file /usr/lib/R/bin/exec/R
/usr/lib/R/bin/exec/R: executable, dynamically linked, ...
```

Recipe to make an yogurt:

Yogurt $\longleftarrow$ Milk + Skyr[2]

[2] Icelandic strained yogurt

Why is it complicated?

```
alice@laptop$ R -e '1+2'
> 1+2
[1] 3
```

What is the issue? **R is Open Source**
https://svn.r-project.org/R/trunk/src/main/**main.c**

```
alice@laptop$ file /usr/lib/R/bin/exec/R
/usr/lib/R/bin/exec/R: executable, dynamically linked, ...
```

Recipe to make an ~~yogurt~~ executable:

Yogurt	⟵	Milk	+	Skyr[2]
executable		**source**		executable

[2] Icelandic strained yogurt

Why is it complicated?

```
alice@laptop$ R -e '1+2'
> 1+2
[1] 3
```

What is the issue? **R is Open Source**
https://svn.r-project.org/R/trunk/src/main/**main.c**

```
alice@laptop$ file /usr/lib/R/bin/exec/R
/usr/lib/R/bin/exec/R: executable, dynamically linked, ...
```

Recipe to make an ~~yogurt~~ executable:

Yogurt	⟵	Milk	+	Skyr[2]
executable		**source**		executable
R		`main.c`		C compiler

[2] Icelandic strained yogurt

Why is it complicated?

```
alice@laptop$ R -e '1+2'
> 1+2
[1] 3
```

What is the issue? **R is Open Source**
https://svn.r-project.org/R/trunk/src/main/**main.c**

```
alice@laptop$ file /usr/lib/R/bin/exec/R
/usr/lib/R/bin/exec/R: executable, dynamically linked, ...
```

Recipe to make an ~~yogurt~~ executable:

Yogurt	⟵	Milk	+	Skyr[2]
executable		**source**		
R		`main.c`		C compiler
				executable ⟵ **source** + executable

[2] Icelandic strained yogurt

Why is it complicated?

```
alice@laptop$ R -e '1+2'
> 1+2
[1] 3
```

What is the issue? **R is Open Source**
https://svn.r-project.org/R/trunk/src/main/**main.c**

```
alice@laptop$ file /usr/lib/R/bin/exec/R
/usr/lib/R/bin/exec/R: executable, dynamically linked, ...
```

Why is it complicated?

```
alice@laptop$ R -e '1+2'
> 1+2
[1] 3
```

What is the issue? **R is Open Source**

https://svn.r-project.org/R/trunk/src/main/**main.c**

```
alice@laptop$ file /usr/lib/R/bin/exec/R
/usr/lib/R/bin/exec/R: executable, dynamically linked, ...
```

```
alice@laptop$ ldd /usr/lib/R/bin/exec/R
    libblas.so.3 => /usr/lib/x86_64-linux-gnu/libblas.so.3
    libR.so => /usr/lib/libR.so
    libgfortran.so.5 => /usr/lib/x86_64-linux-gnu/libgfortran.so.5
    libquadmath.so.0 => /usr/lib/x86_64-linux-gnu/libquadmath.so.0
    ...
```

Why is it complicated?

```
alice@laptop$ R -e '1+2'
> 1+2
[1] 3
```

What is the issue? **R is Open Source**

https://svn.r-project.org/R/trunk/src/main/**main.c**

```
alice@laptop$ file /usr/lib/R/bin/exec/R
/usr/lib/R/bin/exec/R: executable, dynamically linked, ...
```

```
alice@laptop$ ldd /usr/lib/R/bin/exec/R
    libblas.so.3 => /usr/lib/x86_64-linux-gnu/libblas.so.3
    libR.so => /usr/lib/libR.so
    libgfortran.so.5 => /usr/lib/x86_64-linux-gnu/libgfortran.so.5
    libquadmath.so.0 => /usr/lib/x86_64-linux-gnu/libquadmath.so.0
    ...
```

Why is it complicated?

```
alice@laptop$ R -e '1+2'
> 1+2
[1] 3
```

What is the issue? **R is Open Source**

https://svn.r-project.org/R/trunk/src/main/**main.c**

```
alice@laptop$ file /usr/lib/R/bin/exec/R
/usr/lib/R/bin/exec/R: executable, dynamically linked, ...
```

K. Hinsen@Aramis2019 (video)

Blog: Reproducible computations with Guix (link)

Is it really so complicated?

- Wait! **Package managers** (`apt`, `yum`, `conda`, `brew`, etc.) do the job for us.
- Yes, but...

Is it really so complicated?

- Wait! **Package managers** (`apt`, `yum`, `conda`, `brew`, etc.) do the job for us.
- Yes, but...

Yogurt	⟵	Milk	+	Skyr	+	Utensils
executable		**source**		executable		links(executable too)
R		`main.c`		C compiler		Deps. (libblas, etc.)

Is it really so complicated?

- Wait! **Package managers** (`apt`, `yum`, `conda`, `brew`, etc.) do the job for us.
- Yes, but...

Yogurt	⟵	Milk	+	Skyr	+	Utensils
executable		**source**		executable		links(executable too)
R		`main.c`		C compiler		Deps. (libblas, etc.)

R@4.1.1	⟵	source**@4.1.1**	+	executable@A	+	links@C

Is it really so complicated?

- Wait! **Package managers** (`apt`, `yum`, `conda`, `brew`, etc.) do the job for us.
- Yes, but...

Yogurt	⟵	Milk	+	Skyr	+	Utensils
executable		**source**		executable		links(executable too)
R		`main.c`		C compiler		Deps. (libblas, etc.)

R@4.1.1	⟵	source**@4.1.1**	+	executable@A	+	links@C
R@4.1.1	⟵	source**@4.1.1**	+	executable@B	+	links@D

Is it really so complicated?

- Wait! **Package managers** (`apt`, `yum`, `conda`, `brew`, etc.) do the job for us.
- Yes, but...

Yogurt	⟵	Milk	+	Skyr	+	Utensils
executable		**source**		executable		links(executable too)
R		`main.c`		C compiler		Deps. (libblas, etc.)

R@4.1.1	⟵	source**@4.1.1**	+	executable@A	+	links@C
||?						
R@4.1.1	⟵	source**@4.1.1**	+	executable@B	+	links@D

Is it really so complicated?

- Wait! **Package managers** (`apt`, `yum`, `conda`, `brew`, etc.) do the job for us.
- Yes, but...

Yogurt	⟵	Milk	+	Skyr	+	Utensils
executable		**source**		executable		links(executable too)
R		`main.c`		C compiler		Deps. (libblas, etc.)

R@4.1.1	⟵	source**@4.1.1**	+	executable@A	+	links@C
||?						
R@4.1.1	⟵	source**@4.1.1**	+	executable@B	+	links@D

- Too many combinations to have the same environment on different machines
  when something is going wrong, it is hard to say from where it comes

  is it the environment? is it the analysis? something else?

Is it really so complicated?

- Wait! **Package managers** (`apt`, `yum`, `conda`, `brew`, etc.) do the job for us.
- Yes, but… e.g., **Conda is not enough**

Yogurt	⟵	Milk	+	Skyr	+	Utensils
executable		**source**		executable		links(executable too)
R		`main.c`		C compiler		Deps. (libblas, etc.)

R@4.1.1	⟵	source**@4.1.1**	+	executable@A	+	links@C
||?						
R@4.1.1	⟵	source**@4.1.1**	+	executable@B	+	links@D

- Too many combinations to have the same environment on different machines
  when something is going wrong, it is hard to say from where it comes

is it the environment? is it the analysis? something else?

Conclusion: Tools allowing to capture the environment (= **@4.1.1**, @A, @C)

- Alice says she uses R@4.1.1

```
alice@laptop$ install r@4.1.1
alice@laptop$ R --version
R version 4.1.1 (2021-08-10) -- "Kick Things"
```

- Alice says she uses R@4.1.1

```
alice@laptop$ install r@4.1.1
alice@laptop$ R --version
R version 4.1.1 (2021-08-10) -- "Kick Things"
```

- Bob runs this same version @4.1.1

```
 bob@cluster$ install r@4.1.1
 bob@cluster$ R --version
R version 4.1.1 (2021-08-10) -- "Kick Things"
```

- Alice says she uses R@4.1.1

```
alice@laptop$ install r@4.1.1
alice@laptop$ R --version
R version 4.1.1 (2021-08-10) -- "Kick Things"
```

- Bob runs this same version @4.1.1

```
 bob@cluster$ install r@4.1.1
 bob@cluster$ R --version
R version 4.1.1 (2021-08-10) -- "Kick Things"
```

- but Bob does not get the same result (e.g., precision, performance, other)

- Alice says she uses R@4.1.1

```
alice@laptop$ install r@4.1.1
alice@laptop$ R --version
R version 4.1.1 (2021-08-10) -- "Kick Things"
```

- Bob runs this same version @4.1.1

```
 bob@cluster$ install r@4.1.1
 bob@cluster$ R --version
R version 4.1.1 (2021-08-10) -- "Kick Things"
```

- but Bob does not get the same result (e.g., precision, performance, other)
- **why? what is different?** both uses R**@4.1.1**

- Alice says she uses R@4.1.1

```
alice@laptop$ install r@4.1.1 # gcc-toolchain@7.5.0
alice@laptop$ R --version
R version 4.1.1 (2021-08-10) -- "Kick Things"
```

- Bob runs this same version @4.1.1

```
bob@cluster$ install r@4.1.1 # gcc-toolchain@9.4.0
bob@cluster$ R --version
R version 4.1.1 (2021-08-10) -- "Kick Things"
```

- but Bob does not get the same result (e.g., precision, performance, other)
- **why? what is different?** both uses R**@4.1.1**
  - `$ diff with-{7.5.0,9.4.0}/lib/R/bin/exec/R`
    `Binary files differ`
  - some links (`libquadmath.so`) are at different versions

- Alice says she uses R@4.1.1

```
alice@laptop$ install r@4.1.1 # gcc-toolchain@7.5.0
alice@laptop$ R --version
R version 4.1.1 (2021-08-10) -- "Kick Things"
```

- Bob runs this same version @4.1.1

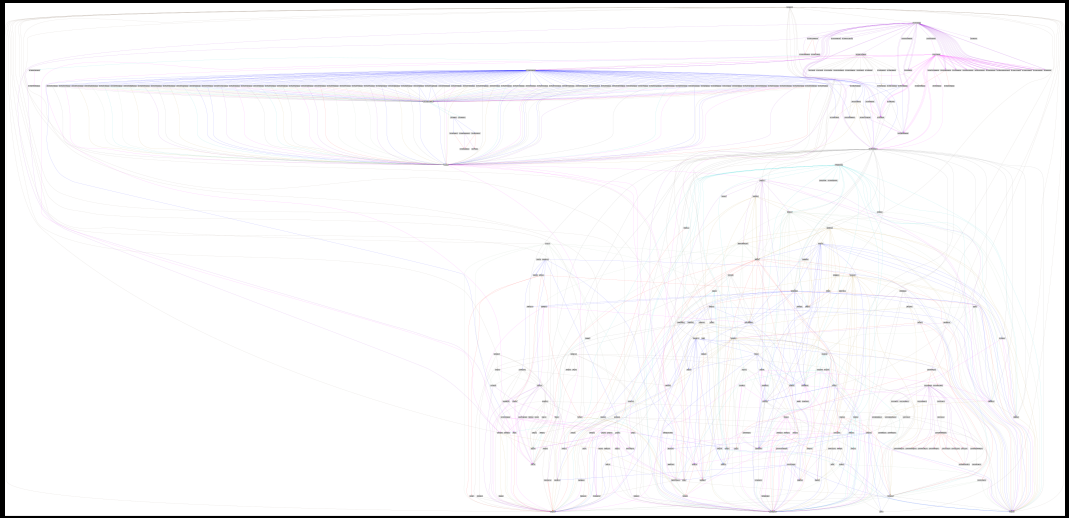
```
bob@cluster$ install r@4.1.1 # gcc-toolchain@9.4.0
bob@cluster$ R --version
R version 4.1.1 (2021-08-10) -- "Kick Things"
```

- but Bob does not get the same result (e.g., precision, performance, other)
- **why? what is different?** both uses R**@4.1.1**
  - `$ diff with-{7.5.0,9.4.0}/lib/R/bin/exec/R`
    `Binary files differ`
  - some links (`libquadmath.so`) are at different versions

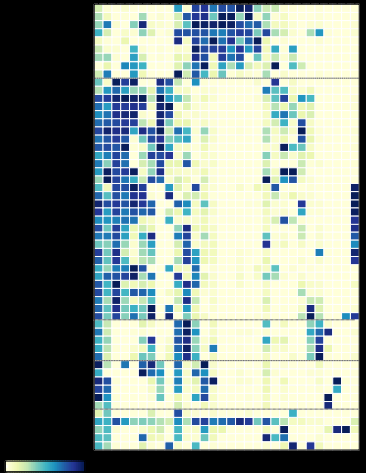
Conclusion: Version **@4.1.1** of **source** is not enough for reproducing

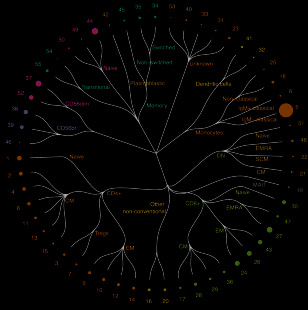
Reproducibility requires to capture the environment

Concretely, just R requires a lot!



Usual (soon?) published result





- Cytometry CyTOF: 43 parameters
- zillions of events (cells)
- FlowJo (cleaning)
- R plus 19 packages from BioConductor

How to redo this (soon?) published result

Assuming nothing is lost and all is transparent

		audit		opaque		depend?
result	⟵	**paper**	+	data	+	analysis
data	⟵	**protocol**	+	instruments	+	materials
analysis	⟵	**script**	+	data	+	environment

environment = collection of packages

package ⟵ **source** + executable (+ others packages)

How to redo this (soon?) published result

Assuming nothing is lost and all is transparent

		audit		opaque		depend?
result	⟵	**paper**	+	data	+	analysis
data	⟵	**protocol**	+	instruments	+	materials
analysis	⟵	**script**	+	data	+	environment

environment = collection of packages

package ⟵ **source** + executable (+ others packages)

Alice says she uses `R@4.1.1` with `CATALYST@1.16.2` from Bioconductor means:

How to redo this (soon?) published result

Assuming nothing is lost and all is transparent

		audit		opaque		depend?
result	⟵	**paper**	+	data	+	analysis
data	⟵	**protocol**	+	instruments	+	materials
analysis	⟵	**script**	+	data	+	environment

environment = collection of packages

package ⟵ **source** + executable (+ others packages)

Alice says she uses `R@4.1.1` with `CATALYST@1.16.2` from Bioconductor means:

- `R`: 40 input packages and 40 build packages (but 408 closure packages)
- `CATALYST`: 27 input packages and 48 build packages

(but 907 closure packages)

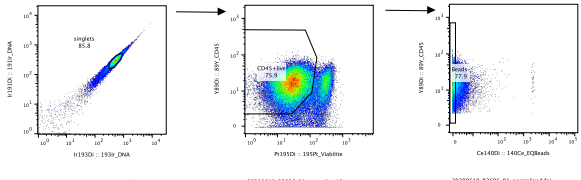
Redo data is complicated

CyTOF pipeline before downstream analyses

CyTOF → 'raw file' → normalization → Manual gating FlowJO → 'clean file' with only living cells

R package 'premessa'

remove
non gaussian events
doublets
unstable flowstream events
debris
normalization beads
dead cells



Redo analysis: use Guix!

- Alice lists the packages with the file `requirements.scm`

```
(specifications->manifest
  (list
    "r"
    "r-catalyst"
    "r-flowcore"))
```

Redo analysis: use Guix!

- Alice lists the packages with the file `requirements.scm`
- Alice describe her current state

```
alice@laptop$ guix describe
Generation 65 Aug 16 2021 20:42:44 (current)
  guix a9eb969
    repository URL: https://git.savannah.gnu.org/git/guix.git
    branch: master
    commit: a9eb969bb63fc421e5cb2a699ef1f1e7c5cbd1b4
alice@laptop$ guix describe -f channels > alice-state.scm
```

Redo analysis: use Guix!

- Alice lists the packages with the file `requirements.scm`
- Alice describe her current state

```
alice@laptop$ guix describe
Generation 65 Aug 16 2021 20:42:44 (current)
  guix a9eb969
    repository URL: https://git.savannah.gnu.org/git/guix.git
    branch: master
    commit: a9eb969bb63fc421e5cb2a699ef1f1e7c5cbd1b4
alice@laptop$ guix describe -f channels > alice-state.scm
```

- Bob recreates the exact same environment as Alice

```
bob@cluster$ guix time-machine -C alice-state.scm \
             -- package -m requirements.scm
```

Redo analysis: use Guix!

- Alice lists the packages with the file `requirements.scm`
- Alice describe her current state

```
alice@laptop$ guix describe
Generation 65 Aug 16 2021 20:42:44 (current)
  guix a9eb969
    repository URL: https://git.savannah.gnu.org/git/guix.git
    branch: master
    commit: a9eb969bb63fc421e5cb2a699ef1f1e7c5cbd1b4
alice@laptop$ guix describe -f channels > alice-state.scm
```

- Bob recreates the exact same environment as Alice

```
bob@cluster$ guix time-machine -C alice-state.scm \
               -- package -m requirements.scm
```

Conclusion: All executable composing the environment are captured by the state.

Generate containers (Docker, Singularity)

- `guix time-machine` allows to rebuild the same environment from **source**

Generate containers (Docker, Singularity)

- `guix time-machine` allows to rebuild the same environment from **source**
- Bob runs analysis on another infrastructure

```
bob@cluster$ guix time-machine -C alice-state.scm
               -- pack -f docker -m requirements.txt
bob@IFB$ docker load < image.tar.gz
bob@IFB$ docker run ...
```

Generate containers (Docker, Singularity)

- `guix time-machine` allows to rebuild the same environment from **source**
- Bob runs analysis on another infrastructure

```
bob@cluster$ guix time-machine -C alice-state.scm
               -- pack -f docker -m requirements.txt
bob@IFB$ docker load < image.tar.gz
bob@IFB$ docker run ...
```

This Docker image disappears

Generate containers (Docker, Singularity)

- `guix time-machine` allows to rebuild the same environment from **source**
- Bob runs analysis on another infrastructure

```
bob@cluster$ guix time-machine -C alice-state.scm
             -- pack -f docker -m requirements.txt
bob@IFB$ docker load < image.tar.gz
bob@IFB$ docker run ...
```

This Docker image disappears

- Later, Dan rebuilds this same Docker image

```
dan@laptop$ guix time-machine -C alice-state.scm
            -- pack -f docker -m requirements.txt
```

Generate containers (Docker, Singularity)

- `guix time-machine` allows to rebuild the same environment from **source**
- Bob runs analysis on another infrastructure

```
bob@cluster$ guix time-machine -C alice-state.scm
             -- pack -f docker -m requirements.txt
bob@IFB$ docker load < image.tar.gz
bob@IFB$ docker run ...
```

This Docker image disappears

- Later, Dan rebuilds this same Docker image

```
dan@laptop$ guix time-machine -C alice-state.scm
            -- pack -f docker -m requirements.txt
```

- Dan runs on another infrastructure using the exact same tools as Alice

```
dan@AWS$ docker load < image.tar.gz
dabn@AWS$ docker run ...
```

Generate containers (Docker, Singularity)

- `guix time-machine` allows to rebuild the same environment from **source**
- Bob runs analysis on another infrastructure

```
bob@cluster$ guix time-machine -C alice-state.scm
             -- pack -f docker -m requirements.txt
bob@IFB$ docker load < image.tar.gz
bob@IFB$ docker run ...
```

This Docker image disappears

- Later, Dan rebuilds this same Docker image

```
dan@laptop$ guix time-machine -C alice-state.scm
            -- pack -f docker -m requirements.txt
```

- Dan runs on another infrastructure using the exact same tools as Alice

```
dan@AWS$ docker load < image.tar.gz
dabn@AWS$ docker run ...
```

(Lack real examples to ensure this scenario fully works ;-)

Long-term with Software Heritage

What happens if the package on GitHub disappears?

```
dan@workstation$ guix install hi
fatal: could not read Username for 'https://github.com': No such device
or address
Trying content-addressed mirror at berlin.guix.gnu.org...
Trying to download from Software Heritage...
SWH: found revision e1eefd033b8a2c4c81babc6fde08ebb116c6abb8
at 'https://archive.softwareheritage.org/api/1/directory/c3e538ed2de412...
```

The **source** of `hi` were at ~~https://github.com/zimoun/hello-example~~.

(See Software Heritage ambassador Pierre Poulain :-)

(Package defined here.)

Guix in a nutshell

Tools helping in managing complexity

1. package manager: functional and transactional paradigm
2. environment manager: switch between projects
3. generate container: Docker, Singularity
4. provide tools (Scheme library) to extend for user specificities

(fine) Control over the dependency chain

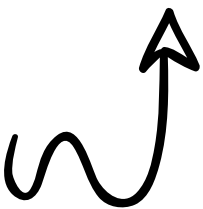
- ⋆⋆ Binary reproductibility (at least to track it)
- ⋆⋆ Bootstrap (new yogurt ← milk + old yogurt)

Similar to other package manager

```
# Alice
guix install foo@1.2 bar@3.4 baz@5.6       # -m requirements.scm
guix describe -f channels > alice-conf.scm

# Carole
guix install --profile=./with-alice foo@1.2 bar@3.4 baz@5.6
guix install foo@7.8 bar@9.0

# Charlie
guix package --roll-back                   # generations

# Bob
guix pull --channels=alice-conf.scm
guix pack -f docker ...                    # -f squashfs

# Dan
guix time-machine -C alice-conf.scm -- install foo@1.2 bar@3.4 baz@5.6
```

The ideal target

Software Heritage



The ReScience Journal

Clusters using Guix

GriCAD	**Grenoble**:	72-node	(1000+ cores)
CCIPL	**Nantes**:	230-node	(4000+ cores)
PlaFRIM Inria	**Bordeaux**:	120-node	(3000+ cores)
Max Delbrück Center	**Berlin**:	250-node	+ workstations
UMC Utrecht	:	68-node	(1,000+ cores)
(Univ. Paris ?)	:	?9-node?	+ 2 workstations

https://hpc.guix.info

simon.tournier@u-paris.fr nicolas.vallet@inserm.fr

https://hpc.guix.info

help-guix@gnu.org https://guix.gnu.org

Container is a smoothie

- Container = a binary box that stores (captures) all the executables
- How to generate one?

  For instance, `Dockerfile` based on some Linux distribution
- What happens if this binary container is lost? Can I rebuild it?

  The answer is no, you cannot!

  Well, require a lot of hard and low-level work.

  A. Legrand@GuixHPC2021 (video)(pdf)

Conclusion 1: Container is **useful** to move binaries from one place to another

Conclusion 2: Container is **not the solution** for the Reproducibility Crisis

(= run the same analysis on different machines at different moments)