

SOFTWARE FACTORIES FOR REPRODUCIBLE RESEARCH IN BIG DATA/DL/AI

Arnaud Legrand



Artifact evaluation and ACM badges



Emerging Interest Group on <https://reproducibility.acm.org/>

Artifact evaluation and ACM badges



Emerging Interest Group on <https://reproducibility.acm.org/>

Major conferences

- ACM SIGMOD 2015-2019, ACM MM 2019-2020, ...
- Supercomputing: Artifact Description (AD) mandatory, Artifact Evaluation (AE) optional, Double blind vs. Open reviews
- NeurIPS, ICLR: open reviews, Joelle Pineau @ NeurIPS'18



Mindsets and practices are evolving: people care and make stuff available
But it's hard and tools are not mature yet

Why focus on AI/DL? Not only but it is **very active**, **empirical** and **computational** field

MAIN CHALLENGES

```
my_code --cfg=magical_param:0.94572 '*.dat' --output foo.csv
```

Tracking code version

- my_code is revision 21b95ecfa0911d6ca87668482b11ab9498edd8f3

MAIN CHALLENGES

```
my_code --cfg=magical_param:0.94572 '*.dat' --output foo.csv
```

Tracking code version

- my_code is revision 21b95ecfa0911d6ca87668482b11ab9498edd8f3

Tracking software environment

- my_code depends on a dozen of libraries, which depend on dozens of libraries
- my_code was compiled with clang 1:9.0-49.1 and -O3 -funroll-loops -fno-strict-aliasing -finline-functions ...

MAIN CHALLENGES

```
my_code --cfg=magical_param:0.94572 '*.dat' --output foo.csv
```

Tracking code version

- `my_code` is revision `21b95ecfa0911d6ca87668482b11ab9498edd8f3`

Tracking software environment

- `my_code` depends on a dozen of libraries, which depend on dozens of libraries
- `my_code` was compiled with `clang 1:9.0-49.1` and `-O3 -funroll-loops -fno-strict-aliasing -finline-functions ...`

Tracking parameters and data

- `*.dat`? Ooh, you ran this in `data/2091293-AJXQ37`?
- Wasn't `mymap.dat` updated since then?
- That was for `foo.csv`. What about `bar.csv`? Is it reproducible?

MAIN CHALLENGES

```
my_code --cfg=magical_param:0.94572 '*.dat' --output foo.csv
```

Tracking code version

- `my_code` is revision `21b95ecfa0911d6ca87668482b11ab9498edd8f3`

Tracking software environment

- `my_code` depends on a dozen of libraries, which depend on dozens of libraries
- `my_code` was compiled with `clang 1:9.0-49.1` and `-O3 -funroll-loops -fno-strict-aliasing -finline-functions ...`

Tracking parameters and data

- `*.dat`? Ooh, you ran this in `data/2091293-AJXQ37`?
- Wasn't `mymap.dat` updated since then?
- That was for `foo.csv`. What about `bar.csv`? Is it reproducible?

Tracking the process (on short/long term)

- Why did I run this? What did I learn from it? I remember doing this but when?

MAIN CHALLENGES

```
my_code --cfg=magical_param:0.94572 '*.dat' --output foo.csv
```

Tracking code version

- my_code is revision 21b95ecfa0911d6ca87668482b11ab9498edd8f3

Tracking software environment

- my_code depends on a dozen of libraries, which depend on dozens of libraries
- my_code was compiled with clang 1:9.0-49.1 and -O3 -funroll-loops -fno-strict-aliasing -finline-functions ...

Tracking parameters and data

- *.dat? Ooh, you ran this in data/2091293-AJXQ37?
- Wasn't mymap.dat updated since then?
- That was for foo.csv. What about bar.csv? Is it reproducible?

Tracking the process (on short/long term)

- Why did I run this? What did I learn from it? I remember doing this but when?

Handle complex sequences and reuse results (leverage cloud/supercomputers)

MANY PROBLEMS, MANY PROTOTYPE SOLUTIONS

I can't be exhaustive here but feel free to help me getting a better view on all this by:

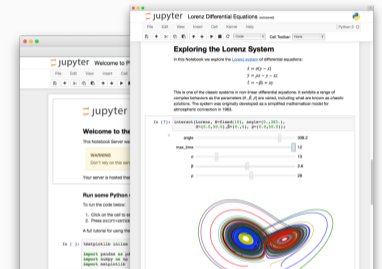
- Adding related projects in the pad
- Telling me about your experience on these tools if you ever tried them

<https://tinyurl.com/JDEVRR> → http://pads.univ-grenoble-alpes.fr/p/jdev_reproducible_ai

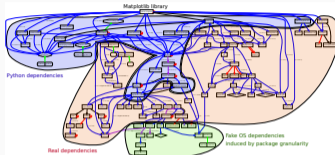
Or come discuss about all this and share news on <https://reproducible-research.inria.fr/>

EXISTING TOOLS AND STANDARDS

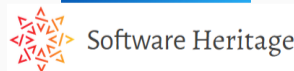
Notebooks and workflows



Software environments



Sharing platforms



Git has become the de facto standard and ultimate **code tracking** tool

- GitHub, BitBucket, GitLab, ... make it more human-friendly
- Note that **none** of these platforms is perennial (see Roberto's presentation on **SoftWare Heritage**)
- Git tracks a single software project: ~→ Git submodule 😊 or Git subtree 😞

Git has become the de facto standard and ultimate **code tracking** tool

- GitHub, BitBucket, GitLab, ... make it more human-friendly
- Note that **none** of these platforms is perennial (see Roberto's presentation on **SoftWare Heritage**)
- Git tracks a single software project: ~→ Git submodule 😊 or Git subtree 😞

Why not use **Git** for **tracking data**?

- Git was not designed to handle **large binary files** (e.g., databases, HDF5, video)
 - ~→ Git LFS (co-developed by GitHub and Atlassian around 2015) 😞 and Git Annex 😊
- Git was not designed with "**privacy**" or "**access control lists**" in mind
 - ~→ Cryptography support in Git Annex 😊

Binary based QEMU/VirtualBox/VMWare/...

LXC/Docker/Singularity/CharlieCloud/...

- Easy to use (e.g., share through DockerHub) although the first installation requires a root access
- Provides a common/standard software stack for a team
- No (or little) support for image reproducibility nor software inspection/modification

Binary based QEMU/VirtualBox/VMWare/...

LXC/Docker/Singularity/CharlieCloud/...

- Easy to use (e.g., share through DockerHub) although the first installation requires a root access
- Provides a common/standard software stack for a team
- No (or little) support for image reproducibility nor software inspection/modification

Source based Spack, Guix, Nix, ...

- Allows to rebuild a software stack in a controlled way (sources, compiling options)
- Use caches to save compiling time
- Time machine mechanism and export to docker/...

Binary based QEMU/VirtualBox/VMWare/... LXC/Docker/Singularity/CharlieCloud/...

- Easy to use (e.g., share through DockerHub) although the first installation requires a root access
- Provides a common/standard software stack for a team
- No (or little) support for image reproducibility nor software inspection/modification

Package Managers pip, conda, dpkg, ...

- Anaconda is OS agnostic and popular for data science
 - Reinstalling a basic environment even after a few months can reveal impossible
- Debian commits to reproducibility: snapshots (March 2005) and reproducible builds
 - You may freeze a Dockerfile by wisely freezing sources
 - Limited flexibility

Source based Spack, Guix, Nix, ...

- Allows to rebuild a software stack in a controlled way (sources, compiling options)
- Use caches to save compiling time
- Time machine mechanism and export to docker/...

TRACKING THE EXPLORATION PROCESS WITH A NOTEBOOK

The *REPL* (Read–eval–print loop) vs. *Notebook* vs *IDE* debate

- In the beginning was the Mathematica (1988) and the Maple (1989) notebooks, which allow to tell a story (*literate programming*)

TRACKING THE EXPLORATION PROCESS WITH A NOTEBOOK

The REPL (Read-eval-print loop) vs. Notebook vs IDE debate

- In the beginning was the Mathematica (1988) and the Maple (1989) notebooks, which allow to tell a story (*literate programming*)
- Then IJulia, IPython, and IR merged into the Jupyter notebook 😊
 - The coolest kid on the block without ~~IDE/Structure~~, ~~Interactive collaboration/Versioning~~, ~~Software Environment Control~~, ~~Easy setup and use of Computing resources~~ !!! 😞

TRACKING THE EXPLORATION PROCESS WITH A NOTEBOOK

The REPL (Read-eval-print loop) vs. Notebook vs IDE debate

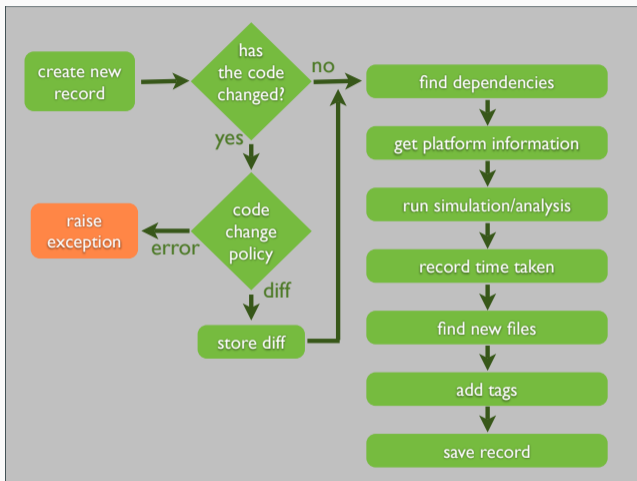
- In the beginning was the Mathematica (1988) and the Maple (1989) notebooks, which allow to tell a story (*literate programming*)
- Then IJulia, IPython, and IR merged into the Jupyter notebook 😊
 - ~~The coolest kid on the block without IDE/Structure, Interactive-collaboration/Versioning, Software-Environment-Control, Easy-setup-and-use-of-Computing-resources !!! 😞~~

Now we have:

- JupyterLab, Binder, JupyterHub
 - Guix-Jupyter
- IDE: Rstudio (not just R), Emacs, VSstudio (Jupyter~backend)
- CodeOcean *showroom, interactive*
- CoCalc/SAGE notebooks, Kaggle, Google Colab, DeepNote
real-time, versioning, custom environment
- fast.ai/nbdev *merge conflict, module export, test*
- Beaker, Count 😞 ???

Little support for computing resources: SoS Polyglot Notebook/Workflow System

RUNNING SIMULATIONS WITH SUMATRA (COMPUTATIONAL NEUROSCIENCE)



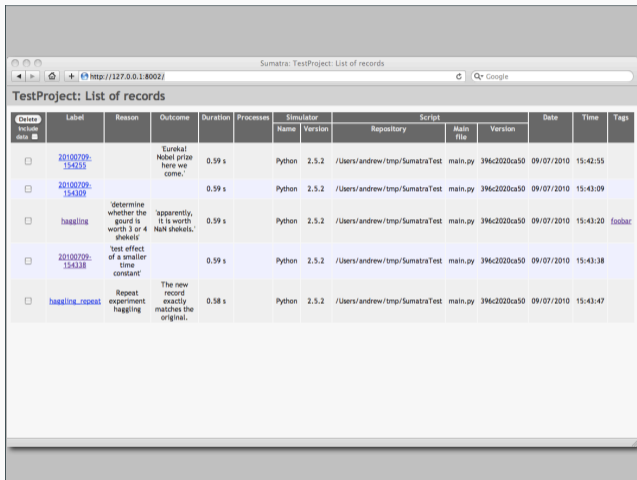
```
smt configure --executable=python \  
  --main=main.py \  
  --datapath /path/to/data  
  
smt run new.param --label=Sgamma \  
  --reason="Test a smaller gamma"  
  
smt comment 20110713-174949 \  
  "Eureka! Nobel prize here I come"  
smt tag "Figure 6"  
  
smt repeat Sgamma  
  
smtweb
```

Courtesy of Andrew Davison (AMP Workshop on RR)

Store records in a DB (duration, OS, args), environment, file content, concurrent executions

PyPet extends Sumatra with Parameter Sweep (Fork-Join, store trajectories in HDF5)

RUNNING SIMULATIONS WITH SUMATRA (COMPUTATIONAL NEUROSCIENCE)



The screenshot shows a web browser window titled "Sumatra: TestProject: List of records" with the URL "http://127.0.0.1:8002/". The page displays a table of simulation records with the following columns: Delete, Label, Reason, Outcome, Duration, Processes, Simulator (Name, Version), Script (Repository, Main file, Version), Date, Time, and Tags. The table contains five rows of data.

Delete	Label	Reason	Outcome	Duration	Processes	Simulator		Script			Date	Time	Tags
						Name	Version	Repository	Main file	Version			
<input type="checkbox"/>	20100709-154255		Eureka! Nobel prize here we come.	0.59 s		Python	2.5.2	/Users/andrew/tmp/SumatraTest	main.py	396c2020ca50	09/07/2010	15:42:55	
<input type="checkbox"/>	20100709-154309			0.59 s		Python	2.5.2	/Users/andrew/tmp/SumatraTest	main.py	396c2020ca50	09/07/2010	15:43:09	
<input type="checkbox"/>	haggling	determine whether the gourd is worth 3 or 4 shekels.	'apparently, it is worth Nak's shekels.'	0.59 s		Python	2.5.2	/Users/andrew/tmp/SumatraTest	main.py	396c2020ca50	09/07/2010	15:43:20	foobar
<input type="checkbox"/>	20100709-154338	'test effect of a smaller time constant'		0.59 s		Python	2.5.2	/Users/andrew/tmp/SumatraTest	main.py	396c2020ca50	09/07/2010	15:43:38	
<input type="checkbox"/>	haggling_repeat	Repeat experiment haggling	The new record exactly matches the original.	0.58 s		Python	2.5.2	/Users/andrew/tmp/SumatraTest	main.py	396c2020ca50	09/07/2010	15:43:47	

```
smt configure --executable=python \  
--main=main.py \  
--datapath /path/to/data  
  
smt run new.param --label=Sgamma \  
--reason="Test a smaller gamma"  
  
smt comment 20110713-174949 \  
"Eureka! Nobel prize here I come"  
smt tag "Figure 6"  
  
smt repeat Sgamma  
  
smtweb
```

Courtesy of Andrew Davison (AMP Workshop on RR)

Store records in a DB (duration, OS, args), environment, file content, concurrent executions

PyPet extends Sumatra with Parameter Sweep (Fork-Join, store trajectories in HDF5)

TRACKING DATA WITH DATALAD (AN OTHER NEUROSCIENCE PROJECT)

Builds on `git annex` and `git submodule` plus small JSON metadata

```
datalad create myfirstrepo # datalad clone/update  
datalad save
```

- 3rd-party integration (owncloud, S3, figshare, GitHub/Lab)

```
datalad create-sibling-github  
datalad export-to-figshare
```

TRACKING DATA WITH DATALAD (AN OTHER NEUROSCIENCE PROJECT)

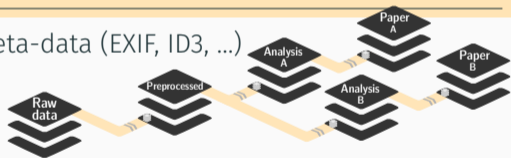
Builds on `git annex` and `git submodule` plus small JSON metadata

```
datalad create myfirstrepo # datalad clone/update
datalad save
```

- 3rd-party integration (owncloud, S3, figshare, GitHub/Lab)

```
datalad create-sibling-github
datalad export-to-figshare
```

- data set combination/linkage, handles meta-data (EXIF, ID3, ...)



Nest modular datasets to create a linked hierarchy of datasets,
and enable recursive operations throughout the hierarchy

TRACKING DATA WITH DATALAD (AN OTHER NEUROSCIENCE PROJECT)

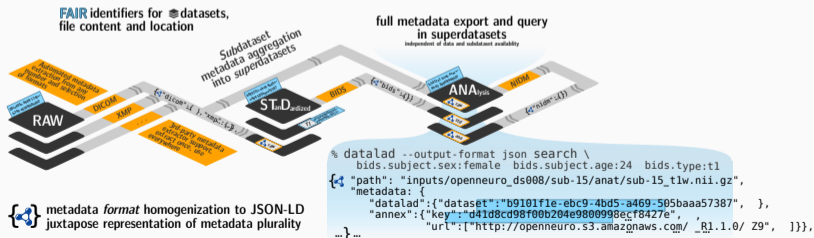
Builds on `git annex` and `git submodule` plus small JSON metadata

```
datalad create myfirstrepo # datalad clone/update  
datalad save
```

- 3rd-party integration (owncloud, S3, figshare, GitHub/Lab)

```
datalad create-sibling-github  
datalad export-to-figshare
```

- data set combination/linkage, handles meta-data (EXIF, ID3, ...)



TRACKING DATA WITH DATALAD (AN OTHER NEUROSCIENCE PROJECT)

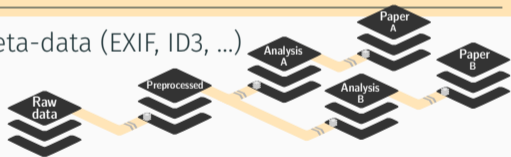
Builds on `git annex` and `git submodule` plus small JSON metadata

```
datalad create myfirstrepo # datalad clone/update
datalad save
```

- 3rd-party integration (owncloud, S3, figshare, GitHub/Lab)

```
datalad create-sibling-github
datalad export-to-figshare
```

- data set combination/linkage, handles meta-data (EXIF, ID3, ...)



- Basic *provenance and reproducibility* support

Nest modular datasets to create a linked hierarchy of datasets, and enable recursive operations throughout the hierarchy

```
datalad run -m "create a list" "bash src/list_titles.sh > data/lst.tsv"
datalad rerun eee1356bb7e8f921174e404c6df6aadcc1f158f0
```

- Extension for running in containers

DATA VERSION CONTROL WITH **DVC** (MACHINE LEARNING)

- **git annex** but special `.dvc` files with information about local/remote storage

```
dvc add data/data.xml
git commit data/data.xml.dvc -m "Dataset updates"
dvc push

git checkout
dvc checkout # Get the content
```

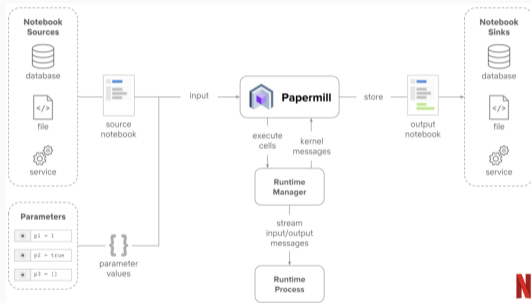
- **snakemake** but Data Pipelines through a `dvc.yaml` task description

```
dvc run -n prepare \
    -p prepare.seed,prepare.split \
    -d src/prepare.py -d data/data.xml \
    -o data/prepared \
    python src/prepare.py data/data.xml
dvc repro
```

- A basic Workflow Management System with little support for running pipelines in parallel

At Netflix, we've put substantial effort into adopting notebooks as an integrated development platform.

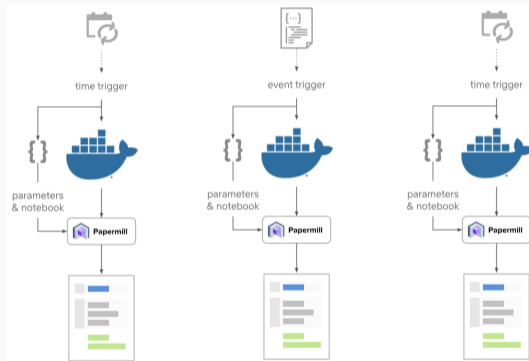
- Data scientist, data/reliability engineer, machine learning engineer
- Notebooks as functions with **papermill** (Parameterized Notebooks)



DATA ANALYTICS AT NETFLIX

At Netflix, we've put substantial effort into adopting notebooks as an integrated development platform.

- Data scientist, data/reliability engineer, machine learning engineer
- Notebooks as functions with **papermill** (Parameterized Notebooks)
- Run in **docker environments**
- **MESON**: a home made workflow orchestration/scheduler (100,000+ jobs per day)
 - Store results in S3 by default
 - Event/Time Trigger, Wait-for
 - Dashboard like usage to generate daily reports



Probably needed with so many people but I've no idea of how they survive such a **complexity** level

SUPER INTEGRATED APPROACHES: DEEPLIT

The screenshot displays the DeepKit web interface for an experiment titled "#272 Experiment". The interface is divided into several sections:

- Left Panel:** A sidebar with navigation options like "Cluster", "Leaf", "google-cloud-gpu", "instance-k80", "instance-p100", "google-cloud-small", "Projects", "pytorch", "keras", "deepkit-python-sdk", and "keras-gan".
- Experiment Overview:** Shows the experiment is "RUNNING". It includes a progress bar for "EPOCH 37/50" and "STEP 91/291". A table lists "Elapsed" and "Remaining" time, and "Seconds/Epoch" and "samples" processed.
- Configuration:** Shows the configuration file path: "examples/keras-ctar16/deepkit.yml".
- Debug View:** A central area showing a computational graph with layers like "input_conv2d_1_input", "Conv2D", "MaxPool", "Dropout", "Flatten", "Dense", "Dropout", and "output_dense_2_softmax". Each layer has associated metrics and visualizations.
- Right Panel:** A "scope" section for "sequential_conv2d_2" with a "Stop watching" button. It lists various parameters such as "Parameters: 1744", "Type: scope", "Sub-type: Conv2D", "Reconstruable: true", "Scope: sequential_conv2d_1", "activation: relu", "activity_regularizer: NoneType", "bias_initializer: NoneType", "bias_regularizer: zeros", "bias_regularizer: NoneType", "data_format: channels_last", "dilation_rate: (1, 1)", "dtype: float32", "filters: 16", "input_shape: None, 30, 30, 12", "kernel_constraint: NoneType", "kernel_initializer: VarianceScaling", "kernel_regularizer: NoneType", "kernel_size: (3, 3)", "non_trainable_weights: 0", "padding: valid", "rank: 2", "strides: 1", and "strides: (1, 1)".

- Python and Neural Network centric
- Execute, track and ML experiments
- Layer debugger for Keras2 and Pytorch
- Side by side file and metrics diff

<https://deepkit.ai/assets/images/deepkit-v2020.mp4>

LORD, HAVE MERCY!

- **PolyAxon**:
 - Integrates with Keras, TensorFlow, SciKitLearn, ...
 - Tracks key model metrics, hyperparams, visualizations, artifacts and resources,
 - Version control code and data
 - Kubernetes
- **Pachyderm** (Version-controlled data science)
 - Versioned data without ~~git~~: a centralized location (no conflict, allows removal,...)
 - Pipelines and distributed computer: Kubernetes/KubeFlow
- **Kono**
- ...

CRAFTWORK "VS." SOFTWARE FACTORIES

The industrialization of scientific research (Konrad Hinsén's Blog, 2019)

The underlying cause for the *reproducibility crisis* is the *ongoing industrialization of scientific research*.

Most software was written for in-lab use, and not even made available to others. Only a small number of basic, *standardized*, and widely used tools, such as compilers, were already industrial products. *Most data were likewise not shared* outside the research group [..]

Science is intrinsically a bottom-up process, whereas *management* is about *top-down organization*.

Towards Long-term and Archivable Reproducibility (software-collapse)

Reproducible workflow solutions commonly use high-level technologies that were *popular when they were created*, providing an immediate solution which is *unlikely to be sustainable in the long term*. (Python 2 vs. Python 3)

The cost of staying up to date within this rapidly-evolving landscape is high.

↪ *No dependency beyond a POSIX-compatible operating system, no administrator privileges, no network connection and storage primarily in plain text*