

# RSE AND RDM, A MATCH MADE IN HEAVEN

## CONNECTING THE DOTS FOR YOUR CREDIBILITY

International Open Access Week 2021 

28.10.2021 | OLIVER BERTUCH, CENTRAL LIBRARY

# AGENDA

- Introduction
- Reproducibility, Credibility, Provenance
- Tooling I-III
- Conclusions

Please write any questions into the chat or hold for later.

# INTRODUCTION

**Some basic understandings**

# RESEARCH SOFTWARE

my definition for this talk

*Any code written with intention of creating novel scientific knowledge.*

- Ranging from Excel formula to programs on exascale HPC clusters.
- Translating scientific methods and prior knowledge into machine language.
- In the following just called "software".

# RESEARCH SOFTWARE ENGINEERS

**Not my definition, but still...**

*Research Software Engineers are people who combine professional software expertise with an understanding of research.*

Quoted from <https://researchsoftware.org>

# ROLE OF SOFTWARE AND DATA

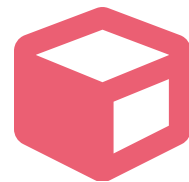
*As a rule, researchers make **all results available** as part of scientific/academic discourse. [...] Where possible and reasonable, this includes making the **research data**, materials and information on which the results are based, as well as the methods and **software** used, available and fully explaining the **work processes**.*

Quoted from  
"Guidelines for Safeguarding Good Research Practice"  
DFG Code of Conduct 2019

(There still is room for exceptions, embargos, NDAs, ...)

# SOFTWARE AS PART OF THE SCIENTIFIC METHOD

*Software*

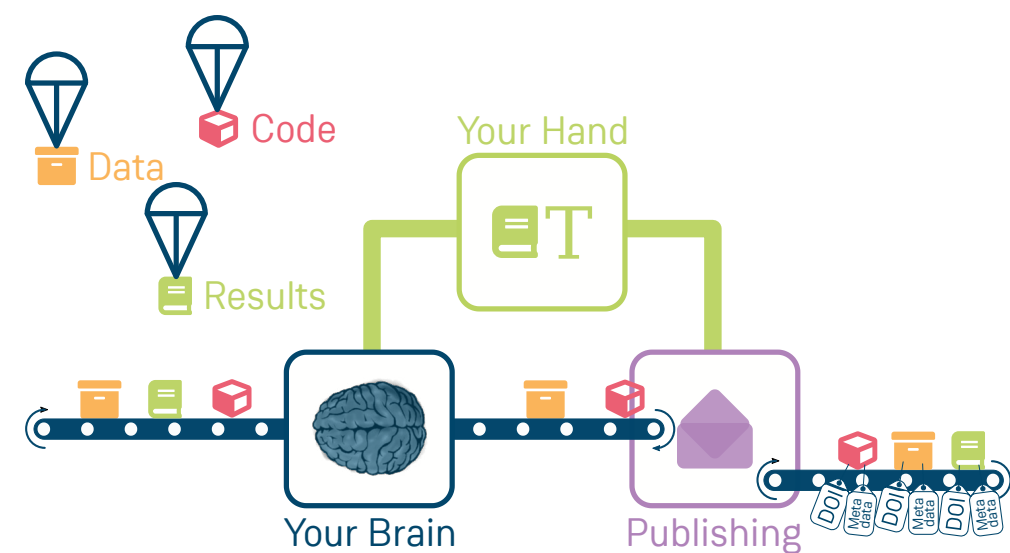


generates :: acquires :: analyses ::  
transforms :: processes :: visualizes  
:: stores :: moves :: sends :: ...

*Data*



# SCIENTIFIC WORKFLOW



Please note the term "workflow". We'll need it.



# REPRODUCIBILITY & PROVENANCE

Or: what could possibly go wrong?

# REPRODUCIBILITY

*Reproducibility is a major principle underpinning the scientific method. For the findings of a study to be reproducible means that results obtained [...] should be achieved again with a high degree of reliability when the study is replicated.*

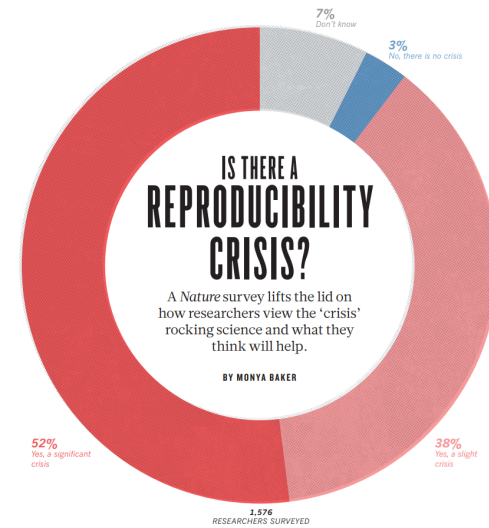
Quoted from Wikipedia

Hint: sophisticated definitions and disambiguation may be found within Turing 2020

No big deal software and data are a foundation of most results, right?

(Uh-oh...)

# CRISIS? WHAT CRISIS?



- ~90% of 1.500 scientists say there is...
- See survey [Nature 2016](#)
- Started in social and medical sciences
- Counter measures still ongoing

# CRISIS? WHAT CRISIS?

Things become brittle, when ...

... there is a "lack of strong basic theories and a tradition of controlled experiments". Peng 2016

... you blindly rely on blackboxes to draw conclusions (aka AI/ML).  
Menske 2020

... publish-or-perish hits you.  
(pressure for high volumes and high ranked journals, not publishing "bad" results, #IchBinHannah)  
Camerer 2018, Heumüller 2020,  
JUnQ Journal

... peer review fails to detect wrong data, software bugs or even fraud.  
Remember cold fusion? Don't feel too safe, natural sciences.

# CREDIBILITY REVOLUTION

This is not about bad people.

Nor doing a bad job.

Let's make this sound more positive.

This is about trust and credibility.

(See [Vazire 2018](#))



Reproducibility  
Crisis?

Credibility  
Revolution!

# EXAMPLE 1/2

## A trust issue

You train an AI model with some training data. Good!

How do you ensure your model can be trusted?

- What happens when trained again with the same data?
- What happens when the model is applied to new data?
- What if you didn't notice your selection bias for your training dataset?
- What if you share your model, but no one can actually verify it because you didn't share data or code?
- What if you did, but didn't communicate this very well?

See example with a breast cancer AI model: [Haibe-Kains 2020](#)

What if you would have referenced proper data and code publications?

# EXAMPLE 2/2

## Not caring about environments et al

You wrote a superb software and it "works on my machine". No tests written.

How do you ensure results produced with it can be trusted?

- What if you introduced a bug, but results aren't suspicious?
- What if you change your software and introduce a bug, but results aren't suspicious?
- What if a new version does not replicate old results? Good? Bad?
- What if your colleague cannot run your code but you can?
- What if you use specialized or buggy hardware?



*Environments matter. (Tests, too...)*

# KEEPING TRACK OF THINGS

We need a triplet: (*Data, Software, Environment*).

Let's call it:

*Provenance is a record that describes the people, institutions, entities, and activities, involved in producing, influencing, or delivering a piece of data or a thing.*

Quoted from W3C PROV Specifications - A Quick Introduction



# TOOLING I

Choose your fighter (1P/2P)



Spreadsheets



Research IDEs



Pipelines



Attributes

+++	++	+	Interactive Data Exploring	-	--	---
---	--	-	Research Software Engineering	+	++	+++
---	--	-	Reproducibility & Provenance	+	++	+++

# SPREADSHEETS

**Prone for chaos, but a nice interactive tool**

## **Pros:**

- Interactive data exploring
- No programming skills
- Easy to share
- Perfect to view and edit tabular data files

## **Cons:**

- Hard to test
- Hard to debug
- Hard to track changes
- Hard to reproduce
- Hard to preserve & archive
- Hard to extend & program
- Tight coupling of data and code
- Potential legal issues
- Version compatibility
- Interesting bug history (genes renamed!)

# TOOLING II

Choose your fighter (1P/2P)



Spreadsheets



Research IDEs



Pipelines



Attributes

+++	++	+	Interactive Data Exploring	-	--	---
---	--	-	Research Software Engineering	+	++	+++
---	--	-	Reproducibility & Provenance	+	++	+++

# PROPRIETARY RESEARCH IDES

Examples: Origin, MatLab, IDL, Stata, ...

## Pros:

- Decoupling of data + code
- Still interactive
- At least minor coding skills
- Many use ASCII files for code
- Some provide test frameworks and version control integration
- Share and reuse possible

## Cons:

- Usual SE chaos hazard
- Many "walled gardens"
- Huge cost factor
- Extension packages for convenience at extra cost
- License requirement impedes sharing & reuse
- Usage of extensions makes sharing hard
- Troublesome to archive

# OPEN SOURCE RESEARCH IDES

Examples: Jupyter Notebooks, RStudio, Octave, ...

## Pros:

- **Free Open Source Software**
- Decoupling of data + code
- Still interactive
- At least minor coding skills
- ASCII files for code
- Test frameworks and version control integration possible
- **Sharing is easy**, reuse possible
- **Easy to archive**

## Cons:

- Usual SE chaos hazard
- Cumbersome for complex or production grade projects
- Minimal distance to full-fledged coding ecosystem
- Notebooks = Junk Food? [\[1\]](#), [\[2\]](#)

# RIAAS

## Research IDEs as a Service

FOSS Research IDEs mostly browser-based

Researcher convenience is key

Some proprietary tools (i. e. MatLab, Stata) provide integrations

Many local, institutional cloud offerings (near to big datasets!)

## Paid external offerings

Examples: [CodeOcean](#), [WholeTale](#) and [MyBinder](#)

Some provide fire-and-forget archive depositing

Beware of vendor lock-in effects and legal issues!

# TOOLING III

Choose your fighter (1P/2P)



Spreadsheets



Research IDEs



Pipelines

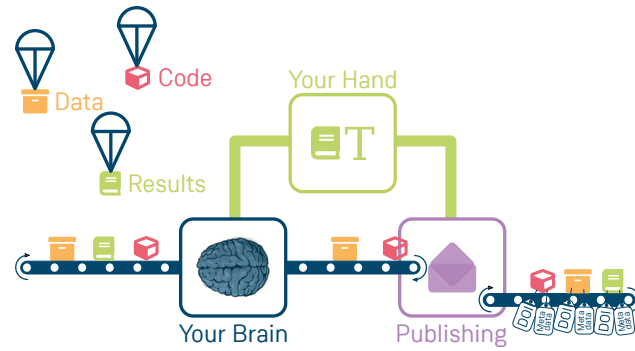


Attributes

+++	++	+	Interactive Data Exploring	-	--	---
---	--	-	Research Software Engineering	+	++	+++
---	--	-	Reproducibility & Provenance	+	++	+++

# PIPELINES

## Remember workflows and provenance?



- What if you ...
  - run the same code for different data?
  - run the same code for different parameters?
  - chain multiple code & data steps?
  - transform a step result into 3 new steps?
  - build complex graphs of data flows?
  - involve different environments?
  - need your colleague to reproduce all this?
- Can you trust your results?
- Can you trust such beasts by someone else?
- Uh-oh. Again.



# FORMALIZE PIPELINES

## Workflows to the rescue

Doing these things manually is very error prone.




Integrate all the existing software codes, not rewrite as Notebooks.

Let's create "data-driven pipelines" and take back control:

1. Standardize environments and reuse.
2. Tests, tests, tests. Did I mention tests?
3. Use workflow DSLs to express your jobs and steps.
4. Track data, software, environments (provenance) and combine with workflows.

# WORKFLOWS

## Examples for Domain Specific Languages (DSLs)

-  NextFlow
-  Common Workflow Language
-  SnakeMake
- Galaxy, KNIME, Jupyter, shell scripts, ...

## Share and reuse them

- Via platforms, data repositories, publications, ...
- Community specific hubs like [Dockstore](#)
- [EOSC WorkflowHub](#) (via ro-crate)

# TWO EXAMPLES FOR PROVENANCE MODELING

## W3C PROV Model (since 2013)

- <https://www.w3.org/TR/prov-overview>
- Roots in machine-actionable **Semantic Web**
- Data model with different representations as Knowledge Graph, XML, JSON, etc
- **Tool implementations**
- Adoption seems stalled?

## RO-Crate (since 2020)

- <https://www.researchobject.org>
- Includes support for workflows since 1.1
- JSON-LD graph based representation of research objects and relations
- Much more lightweight compared to PROV
- Adoption rising

# PIPELINES & WORKFLOWS

## Pros:

- Loose coupling of data & code
- Reuse existing codes
- Easy to preserve & archive
- Easy to reproduce
- Easy to share & reuse
- Self-Documenting
- No Junk Food

## Cons:

- Verbose
- Needs more & new skills
- Steep Learning Curve
- Ecosystem not yet grown up

# CONCLUSIONS

1. Times they are a-changin'
2. Homework
3. You're not alone

# TIMES THEY ARE A-CHANGIN'

(But there is no free lunch)

*The landscape of workflow systems for scientific applications is notoriously convoluted with hundreds of seemingly equivalent workflow systems, many isolated research claims, and a **steep learning curve**.*

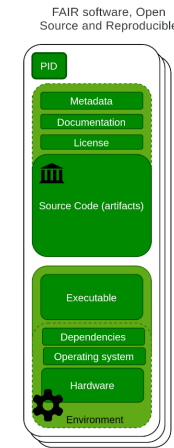
Quoted from Da Silva 2021

1. Changing mindsets  
(no more Excel engineering and data chaos)
2. Changing work habits  
(publish data and code, create workflows)
3. Enabling people  
(provide support and education, foster open science culture)
4. Adding resources  
(change the system)

# HOMEWORK

## Items to achieve before reproducible workflows can take off

- Open data formats as default "Save as" target
- Open source as the reasonable default choice
- Docs writing is joyfull part of research life
- Researchers sleep at night by having software tests by default
- Data and software publications are default to enable citation and referencing (see also [Katz 2021](#))



"The holy grail"  
from [Katz 2021](#)

*Open Science means Open Access, Open Data and Open Source*

# ADRESSING CHAOS HAZARD BY ENABLING PEOPLE

It's about time for Research Software Engineering!



- Enable researchers to write better code and use best practices
- Enable software engineers to understand science
- Achieve together!



# YOU ARE NOT ALONE

## Reproducibility



ReproHack Hub



German Reproducibility  
Network



Turing Way Project

November, 16th 2021: 1st German Reproducibility Day

**RSE @ FZJ and beyond**

#rse RocketChat @ FZJ, HIFIS Software Services, de-RSE e. V.

# THANK YOU FOR YOUR ATTENTION!

\$ whoami



Oliver Bertuch  
Central Library


\$ reachout

✉ o.bertuch@fz-juelich.de  
🐦 @poi\_ki\_lo\_therm  
🐙 @poikilotherm

\$ Is /workplaces

Research Data Management 🐦 FZJ\_RDM  
+  
**HE**lmholtz **RI**ch **ME**tadata **S**oftware  
Publication @ HMC

\$ attribution

Slides licensed under ,  
Most icons by Font Awesome  
Logos are non-CC material