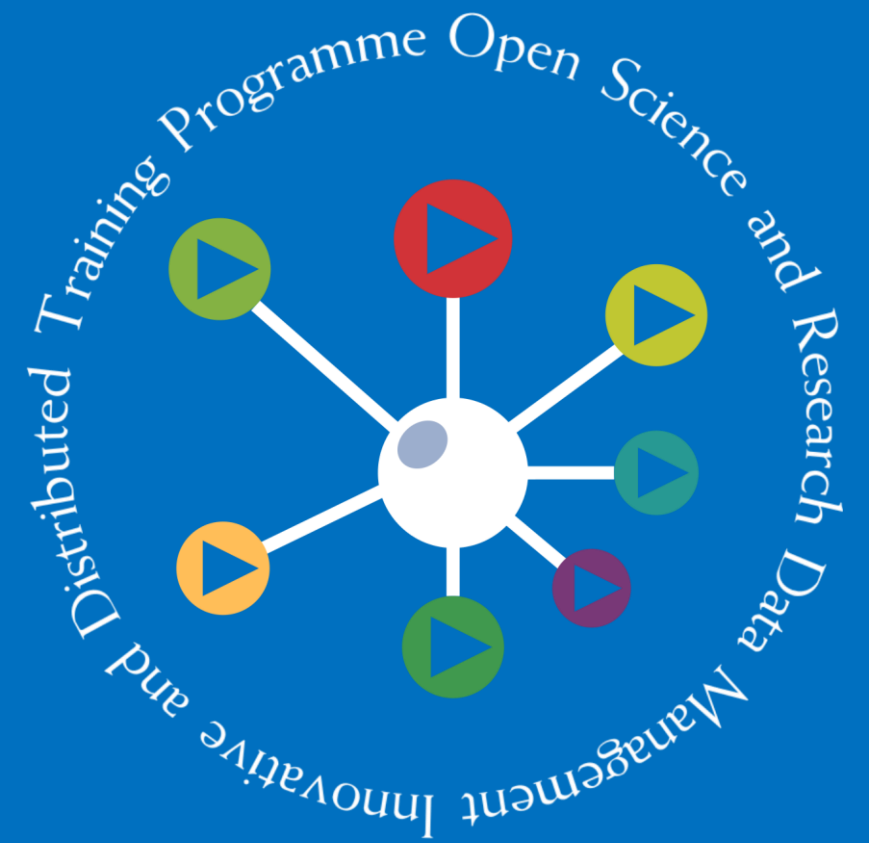


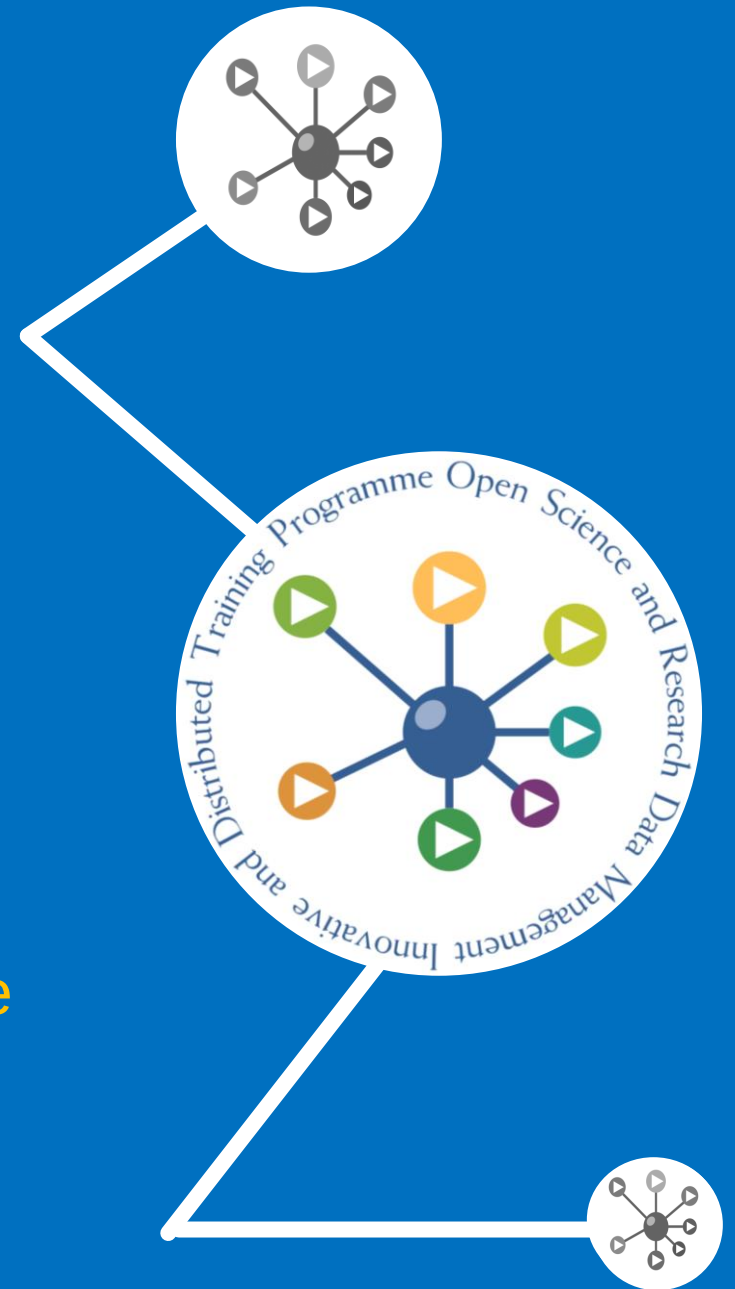
Reproducible Research and Data Analysis

University POLITEHNICA of Bucharest



1. Reproducible Research

Why do we need it, what are the challenges, how do we perform it?



Reproducible research



- ✿ **Replication** is fundamental in science, but only a small number of scientific studies can be replicated → the **replication crisis**
 - ✿ many attempted replications of well-known scientific studies have failed (in many different areas)
 - ✿ rates of paper retractions from top journals and conferences are increasing
- ✿ Replication with new independent data is expensive or methodologically impossible → we need **reproducible research** to assess the validity of scientific results
- ✿ Research is **reproducible** when we can reproduce the results of a scientific study with the original **data, code, and documentation**

Why perform reproducible research?



It benefits the researchers who perform it

-  they can repeat the same analysis multiple times with the same results
-  they can remember how and why they performed specific analyses
-  they can easily and rapidly modify analyses and figures
-  they can quickly reconfigure previous research tasks for use in new projects
-  reproducible research is a strong indicator of rigor, trustworthiness, and transparency in scientific research
-  reproducible research increases paper citation rates

Why perform reproducible research?



It benefits the research community

-  research is more accessible to more scientists → faster progress in methodological development and innovation
-  allows others to learn from a researcher's work
-  allows others to understand and reproduce a researcher's work
-  allows others to detect and avoid a researcher's mistakes

Challenges in reproducible research



Complexity

-  scientific research may require specialized (and often proprietary) knowledge and tools
-  some studies require models and techniques that involve domain knowledge
-  some analyses may need specific hardware or software configurations (high-performance computing, GPU clusters, licensed software, etc.)

How to address this?

-  refer to studies and papers that offer the required domain knowledge
-  annotate code with comments and explanations
-  provide documentation
-  use open software where possible

Challenges in reproducible research



Technological change

-  old tools become obsolete
-  small changes in software versions can lead to totally different results

How to address this?

-  use established tools
-  specify software versions used
-  run experiments in reproducible conditions → e.g., software containers

Challenges in reproducible research



Human error

-  we forget small details from the research process
-  we perform incomplete descriptions of the data collection or analysis
-  we don't collect or properly document data that we initially might consider unimportant
-  we lose or misplace data

How to address this?

-  record every detail of the data collection and analysis, as inconsequential or minor as it may seem
-  keep multiple copies of the data
-  keep raw copies of the data

Challenges in reproducible research



IPR

-  we don't share data and code because we are afraid that other researchers might use it incorrectly or unethically
-  we might not receive proper credit for others using our data
-  we might want to perform new analyses on our data and we are afraid that others might do this ahead of us

How to address this?

-  use specific data/code sharing tools that facilitate proper IPR management

Challenges in reproducible research



Data dredging

Also known as p-hacking, this involves repeatedly searching a dataset or trying alternative analyses until a 'significant' result is found.



Omitting null results

When scientists or journals decide not to publish studies unless results are statistically significant.



Underpowered study

Statistical power is the ability of an analysis to detect an effect, if the effect exists – an underpowered study is too small to reliably indicate whether or not an effect exists.

Issues



Errors

Technical errors may exist within a study, such as misidentified reagents or computational errors.



Underspecified methods

A study may be very robust, but its methods not shared with other scientists in enough detail, so others cannot precisely replicate it.



Weak experimental design

A study may have one or more methodological flaws that mean it is unlikely to produce reliable or valid results.

Stages of performing reproducible research



Before data analysis

-  always back up data (of all kinds, from all stages)
-  always replicate data in multiple locations and on multiple media
-  if you have physical data (papers, sheets, etc.), make digital copies
-  use portable, open and flat file formats where possible (e.g., txt or csv)
-  transform data into tidy formatting
 -  long format
 -  consistent data structure
 -  informative headers
-  store metadata along with the actual data
-  organize files in a logical structure and give them informative names
-  use version control

Stages of performing reproducible research



During data analysis

-  use scripts instead of visual tools
-  annotate code with comments
-  follow a consistent coding style
-  automate repetitive tasks
-  avoid using hard-coded values
-  have someone review your code
-  create a portable environment for running the analysis (e.g., software containers)
-  document all software packages, frameworks and libraries used, and their versions

Stages of performing reproducible research



After data analysis

-  generate figures and tables directly from code
-  automate data pre-processing, analysis and manuscript generation as “one-button” processes
-  increase access to publications by posting preprints
-  use data and code repositories for sharing (instead of personal websites)
-  create research compendiums → archives of data, code, software and products from a research project
-  in the published manuscript, offer explicit instructions regarding where to locate data, metadata and code

2. Research Code

Collaborative working with code: versioning, branching, metadata



Git - good practice for code management

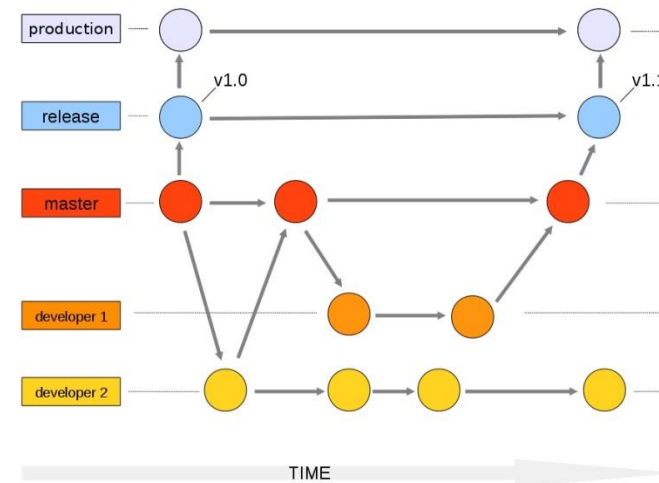


🌐 Versioning

- 🌐 Go back (to previous commits or versions)
- 🌐 Keep track of evolutions (diff)

🌐 Decentralized collaboration

- 🌐 Work in parallel (repository, branches)
- 🌐 Merge work with others



Git and GitHub



- Git is a **distributed version-control system** for tracking changes in source code during software development
 - Written to manage development of Linux
- GitHub is a **social network for software development**
 - Git repository hosting service + extra features
 - GitHub provides a web-based interface on top of Git
 - Collaboration features: access control, wikis, issues, projects, ...
 - Connected with **Zenodo**:
 - Get a DOI from a GitHub repo

How to create metadata?



Use the Dublin Core Element Schema (dces)

1. <http://dublincore.org/documents/dces/>
2. Fill up the 15 fields
 - Format: in a new text file in the repository. Enter one field per line, starting with the field name (e.g., «title: »)
 - Some thoughts:
 - Is it a good idea to include the institution in the «creators» or not?
 - How to describe the current branch and version?
 - What format should you use for the date?
 - Are there other ambiguous points?

Citation metadata for Data Repositories. Key:
Statistical power and underpowered statistics...
www.refsmmat.com/statistics/power.html

Citation Metadata	Dublin Core ^a	Schema.org ^b	DataCite ^c	DATS ^d
Dataset Identifier	identifier	@id*	identifier	identifier
Title	title	name	title	title
Creator**	creator	author	creator	creator
Data repository or archive	publisher	publisher	publisher	publisher
Publication Date	date	datePublished	publicationYear	date
Version	<i>not available</i>	version	version	version
Type	type	type	resourceTypeGeneral	type

^aDublin Core Metadata Element Set (<https://dublincore.org/documents/dces/>);

^bDataset - Schema.org (<https://schema.org/Dataset>);

^cDataCite Metadata Working Group²¹;

^dGonzalez-Beltran & Rocca-Serra^{22,23};

* name of ID field depends on schema.org serialization format, it is **@id** for JSON-LD;

Metadata can use any standard



```
{
  "login": "torvalds",
  "id": 1024025,
  "avatar_url": "https://avatars.githubusercontent.com/u/1024025?v=3",
  "gravatar_id": "",
  "url": "https://api.github.com/users/torvalds",
  "html_url": "https://github.com/torvalds",
  "followers_url": "https://api.github.com/users/torvalds/followers",
  "following_url": "https://api.github.com/users/torvalds/following/other_user",
  "gists_url": "https://api.github.com/users/torvalds/gists/gist_id",
  "starred_url": "https://api.github.com/users/torvalds/starred/owner/repo",
  "subscriptions_url": "https://api.github.com/users/torvalds/subscriptions",
  "organizations_url": "https://api.github.com/users/torvalds/orgs",
  "repos_url": "https://api.github.com/users/torvalds/repos",
  "events_url": "https://api.github.com/users/torvalds/events/privacy",
  "received_events_url": "https://api.github.com/users/torvalds/received_events",
  "type": "User",
  "site_admin": false,
  "name": "Linus Torvalds",
  "company": "Linux Foundation",
  "blog": null,
  "location": "Portland, OR",
  "email": null,
  "hireable": null,
  "bio": null,
  "public_repos": 2,
  "public_gists": 0,
  "followers": 31456,
  "following": 0,
  "created_at": "2011-09-03T15:26:22Z",
  "updated_at": "2015-06-11T00:46:13Z",
}
```

Important discovery metadata for Data Repositories. Key:

Discovery Metadata	Dublin Core	Schema.org	DataCite	DATS
Description	description	description	description	dataType dimension Material...*
Keywords	subject	keywords	subject	keywords
License	license	license	rights	license
Related Dataset**	isPartOf isVersionOf references	isPartOf citation	relatedIdentifier	isPartOf
Related Publication***	bibliographicCitation	citation	relatedIdentifier	publication

Figure User meta information example. Display the meta information of a Github user by accessing the Github API with url=["https://api.github.com/users/Torvalds"](https://api.github.com/users/Torvalds)

- [Zenodo](https://zenodo.org/) is a public repository funded by OpenAire, CERN and the EU Horizon2020 program
 - 🌐 open to **all** fields of research
 - 🌐 open to **all** types of research data
 - 🌐 free service
 - 🌐 file size up to 50 Gigabyte
 - 🌐 integration with GitHub



Recent uploads

August 30, 2017 (v2) Working paper Open Access

View

Introducing Parsl: A Python Parallel Scripting Library

Babuji, Yadu; Brizius, Alison; Chard, Kyle; Foster, Ian; Katz, Daniel S.; Wilde, Michael; Wozniak, Justin

Researchers frequently rely on large-scale and domain-specific workflows to conduct their science. These workflows may integrate a variety of independent software functions and external applications. However, developing and executing such workflows can be difficult, requiring complex...

Uploaded on September 15, 2017

1 more version(s) exist for this record

<https://zenodo.org/>

Zenodo special features



DOI-Versioning:

-  edit/update data after publication
-  quote a specific version or all versions of a dataset

In combination with GitHub:

-  archiving of software for the evaluation of data
-  also with versioning and citable link/identifier

Zenodo now supports
DOI versioning!



[Read more](#) about it, in our
newest blog post.

<https://zenodo.org/>

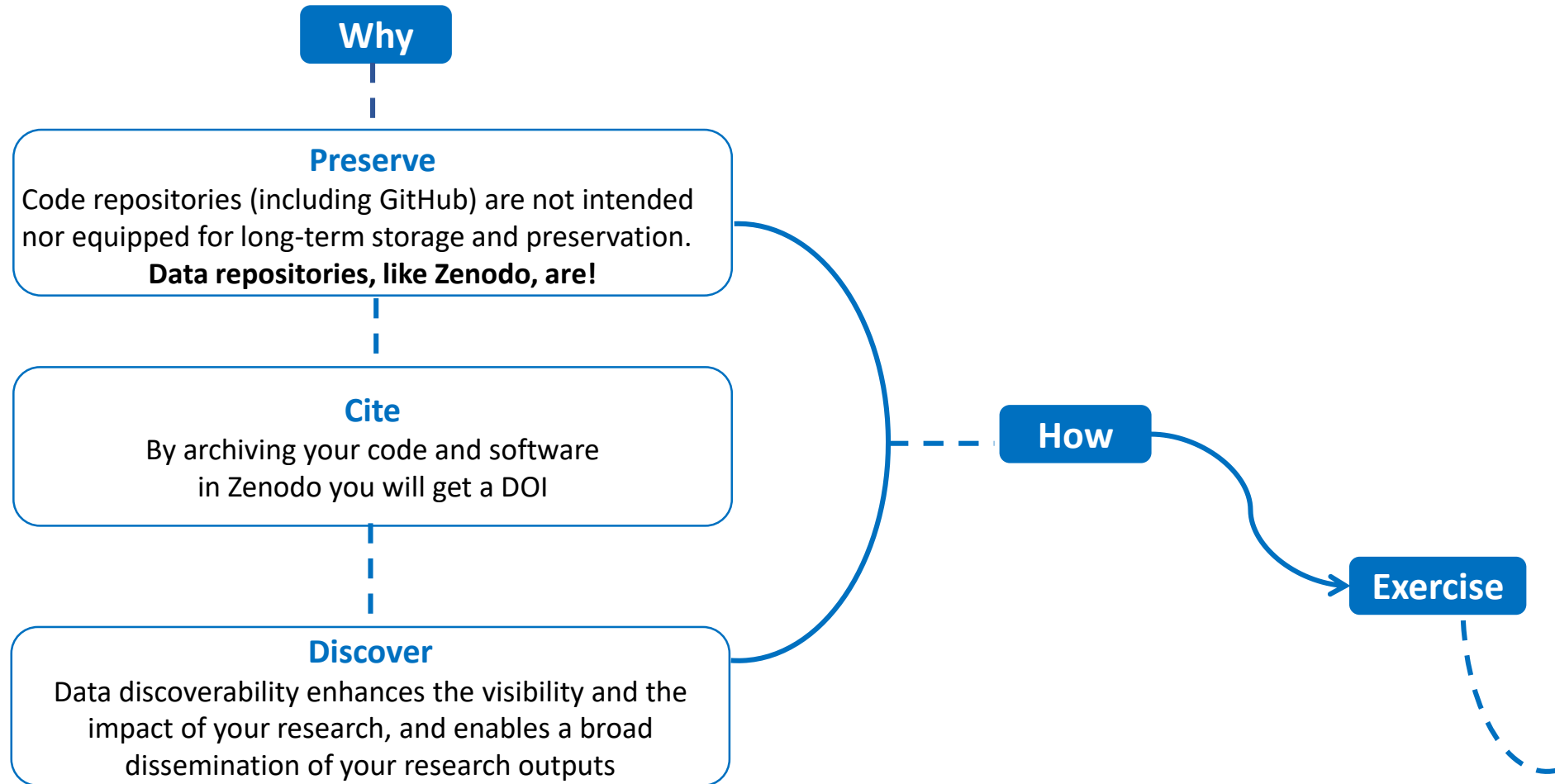
Using GitHub?



Just [Log in](#) with your GitHub
account and [click here](#) to start
preserving your repositories.

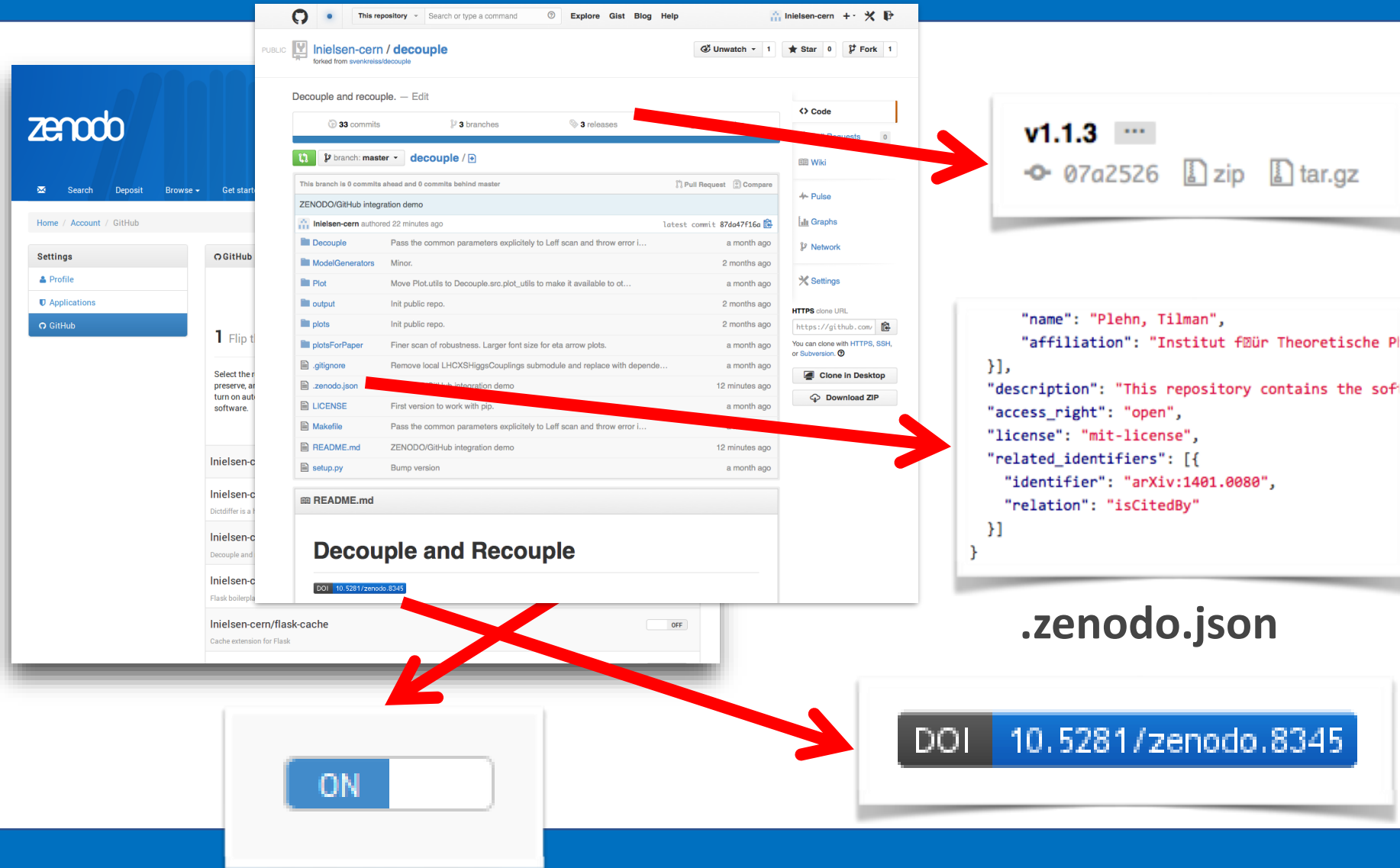
<https://zenodo.org/>

From GitHub to Zenodo



!!! Every (trusted) data repository helps you in making your data safely stored, easily citeable, and discoverable !!!

Zenodo-GitHub bridge



Publish GitHub repo in Zenodo

Exercise



When you write your software, you can **make the work you share on GitHub citable**:
archive one of your **GitHub repositories in Zenodo** and **assign it a DOI**

Preliminary
steps

Authorize application

Zenodo by @zenodo would like permission to
access your account

Review permissions

Personal user data
Email addresses (read-only)

Repository webhooks and services
Admin access

Organizations and teams
Read-only access

Authorize application

Zenodo

Software Preservation Made Simple!

[Visit application's website](#)

[Learn more about OAuth](#)

Publish GitHub repo in Zenodo

Exercise



[Upload](#)[Communities](#)

 ciprian.dobre@cs.pub.ro

[Home](#) / [Account](#) / [GitHub](#)

Settings

 Profile

 Change password

 Security

 Linked accounts

 Applications

 Shared links

 **GitHub**

GitHub Repositories

(updated 2 minutes ago)

Get started

1 Flip the switch

Select the repository you want to preserve, and toggle the switch below to turn on automatic preservation of your software.

☒ ON

2 Create a release

Go to GitHub and [create a release](#). Zenodo will automatically download a .zip-ball of each new release and register a DOI.

3 Get the badge

After your first release, a DOI badge that you can include in GitHub README will appear next to your repository below.

DOI

10.5281/zenodo.8475

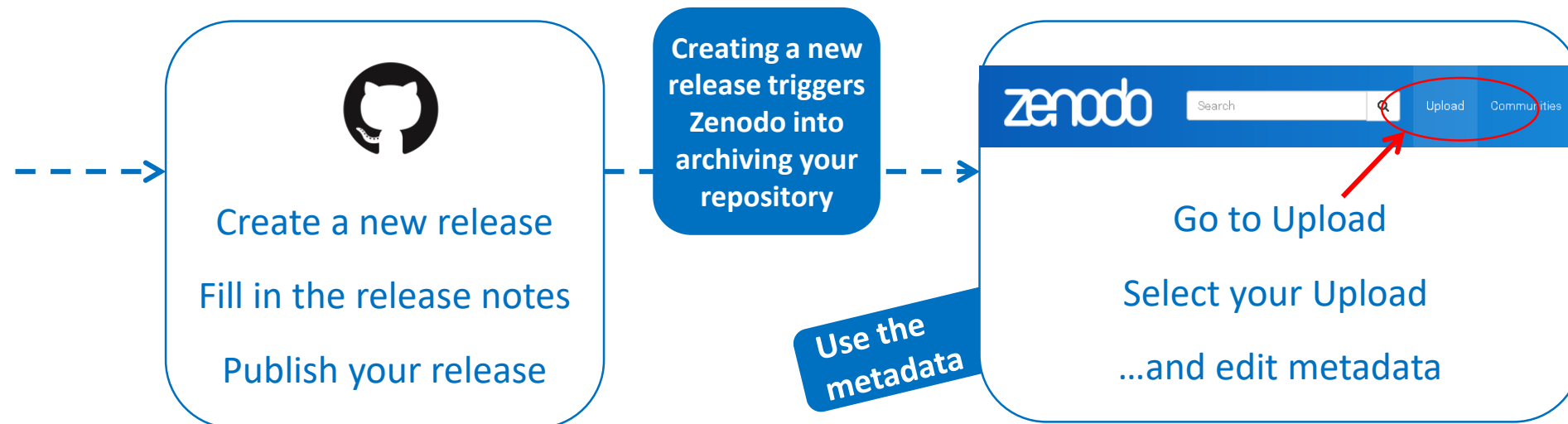
(example)



Repositories

Publish GitHub repo in Zenodo

Exercise



3. Reproducible Environments

Tools for obtaining a reproducible environment



Reproducible environments



- Every computer has its unique computational environment consisting of its operating system, installed software, versions of installed software packages, and other features that we will describe later.
- Suppose a research project is carried out on one computer but transferred to a different computer. In order for research to be reproducible, the computational environment that it was conducted in must be captured in such a way that others can replicate it.

```
a = 1
b = 5
print(a/b)
```

One divided by five is **0.2**, and this is what is printed if the script is run using Python 3. However, if a slightly older version of Python, such as Python 2, is used, the result printed is **0**. This is because integer division is applied to integers in Python 2, but (normal) division is applied to all types, including integers, in Python 3.

Capturing computational environments

What is reproduced?	Interaction style	
	Graphical	Command line
Software & versions	 binder	 CONDA
Entire system		 docker

Open notebooks



- 🌐 Electronic Lab Notebooks (ELNs) enable researchers to organize and store experimental procedures, protocols, plans, notes, data, and even unfiltered interpretations using their computer or mobile device
- 🌐 They are a digital analogue to the paper notebook most researchers keep
- 🌐 ELNs can offer several advantages over the traditional paper notebook in documenting research during the active phase of a project, including:
 - 🌐 searchability within and across notebooks
 - 🌐 secure storage with multiple redundancies
 - 🌐 remote access to notebooks
 - 🌐 the ability to easily share notebooks among team members and collaborators

Jupyter Notebooks

Reproducible Open Science



- The computational **tools** to solve a problem
 - Python, R, Julia, and wide ecosystem of libraries and tools for science
- An **interface** to facilitate coding/creating
 - Jupyter
- A way to **communicate** your work
 - notebooks
- Leverage on the EGI Cloud to **scale-up the resources**
- A way to **share** your work
 - GitHub, Zenodo or other similar repositories
- A way to **pack** it all for replication
 - Docker (used by Binder)
- A way to **persistently identify** it
 - DOIs (Digital Object Identifiers)

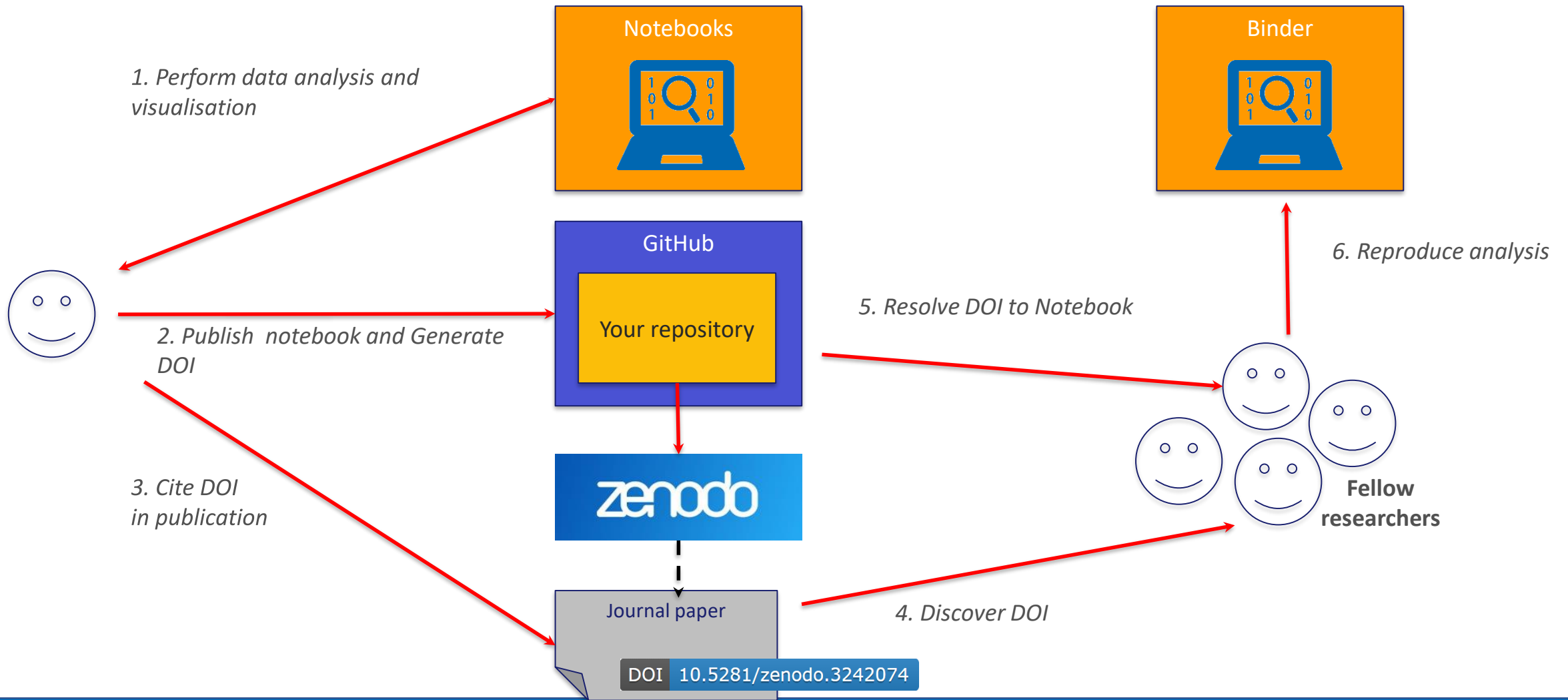


binder



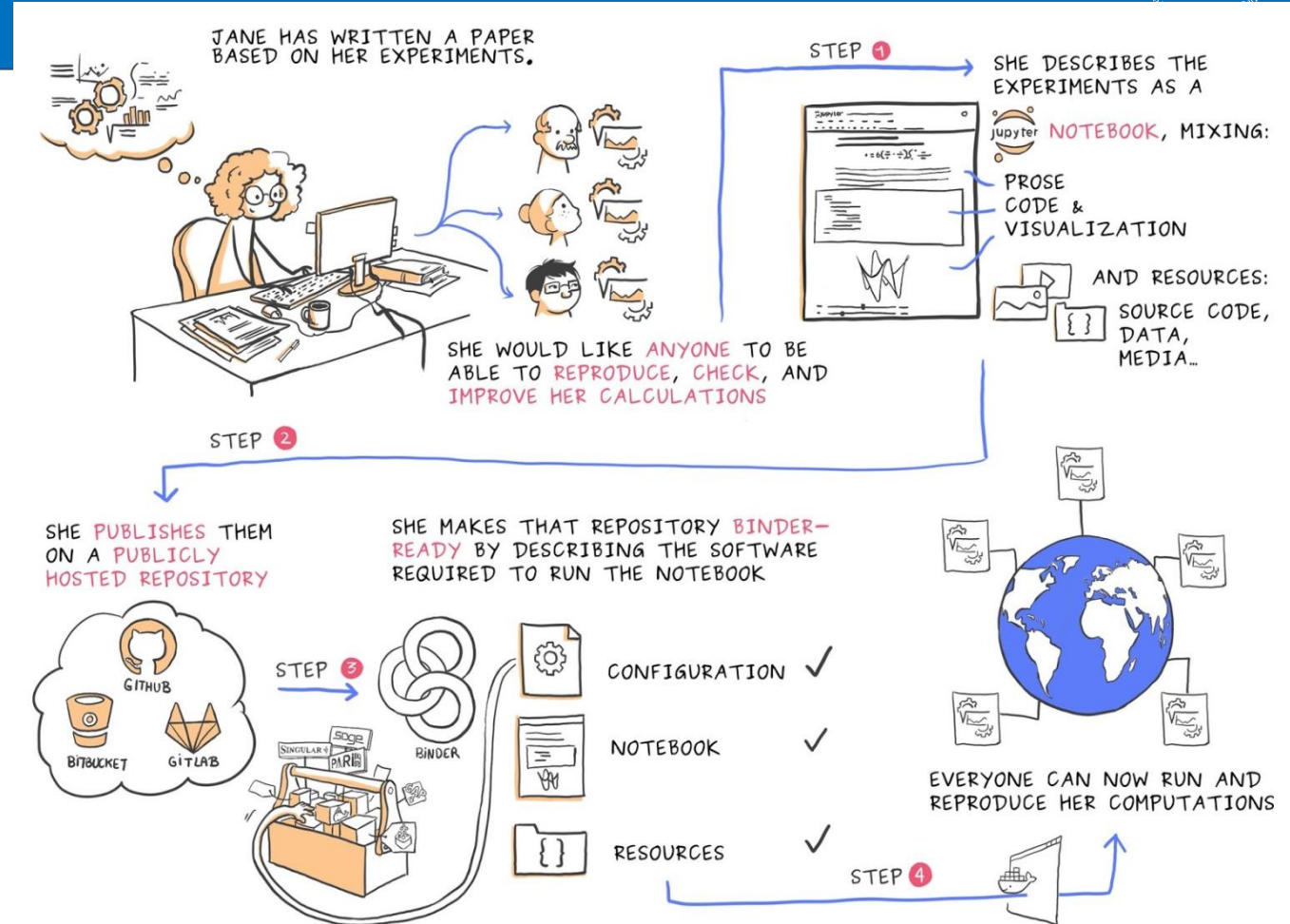
zenodo

Implement Open Science



Binder

- An open-source web application to turn repositories into interactive notebooks
- It uses modern technology in cloud orchestration (Kubernetes), interactive computing (Jupyter), scientific computing (the open-science ecosystem)



Credit: Juliette Taka <https://twitter.com/JulietteTaka>

What does Binder do?



GitHub

Pulls code from
repository



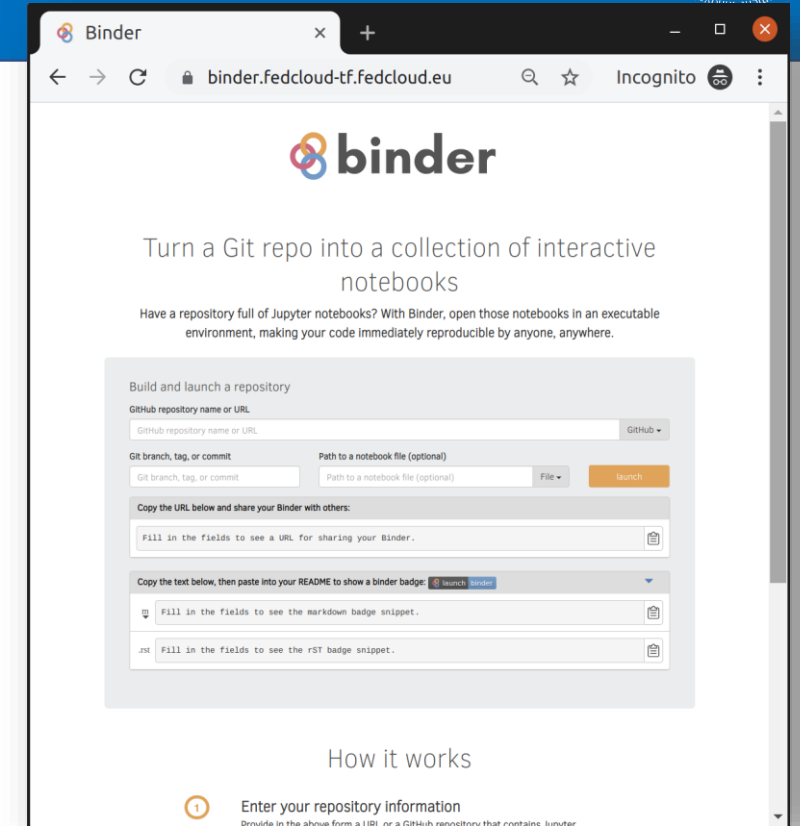
repo2docker: Creates
reproducible containers
from repositories



jupyterhub: Generates
user sessions that serve
these containers



Kubernetes: manages
the computing
infrastructure



EGI Binder:
Provides an interface to create, share,
and share the sessions

Compute resources



- ✿ BinderHub is a cloud-based technology that can launch a repository of code (from GitHub, GitLab, and others) in a browser window such that the code can be executed and interacted with
 - ✿ a unique URL is generated allowing the interactive code to be easily shared
- ✿ The purpose of these Binder instances is to promote reproducibility in research projects by encouraging researchers to document their software dependencies and produce fun, interactive environments!
- ✿ Binder, as a user interface, is useful for reproducibility because the code needs to be version controlled and the computational environment needs to be documented in order to benefit from the functionality of Binder
- ✿ Each change to the code repository forces a new build of the Binder instance → this acts as a proxy for continuous integration of the computational environment as the Binder instance will break if the configuration file is not updated
- ✿ BinderHub relies on different tools and resources in order to create and launch the Binder instances

Reproduce your analysis with Binder



Build and launch a repository

Zenodo DOI (10.5281/zenodo.3242074)

10.5281/zenodo.3452485

Zenodo DOI ▾

Git branch, tag, or commit

Git branch, tag, or commit

Path to a notebook file (optional)

Path to a notebook file (optional)

File ▾

launch

Copy the URL below and share your Binder with others:

`https://binder.fedcloud-tf.fedcloud.eu/v2/zenodo/10.5281/zenodo.3452485/`



Copy the text below, then paste into your README to show a binder badge:

 launch  binder ▾

`[![Binder](https://binder.fedcloud-tf.fedcloud.eu/badge_logo.svg)](https://binder.fedcloud-tf.fedcloud.eu/v2/zenodo/10.5281/zenodo.3452485/)`



`.. image:: https://binder.fedcloud-tf.fedcloud.eu/badge_logo.svg
:target: https://binder.fedcloud-tf.fedcloud.eu/v2/zenodo/10.5281/zenodo.3452485/`



Waiting

Building

Pushing

WARNING:

Registration of the DOI can take up to 10 minutes
Binder has to wait until registration is finished.

To check whether the DOI is resolved, please use the
website: <https://dx.doi.org/>

Create a LaunchBinder button in your repo



Stop My Server

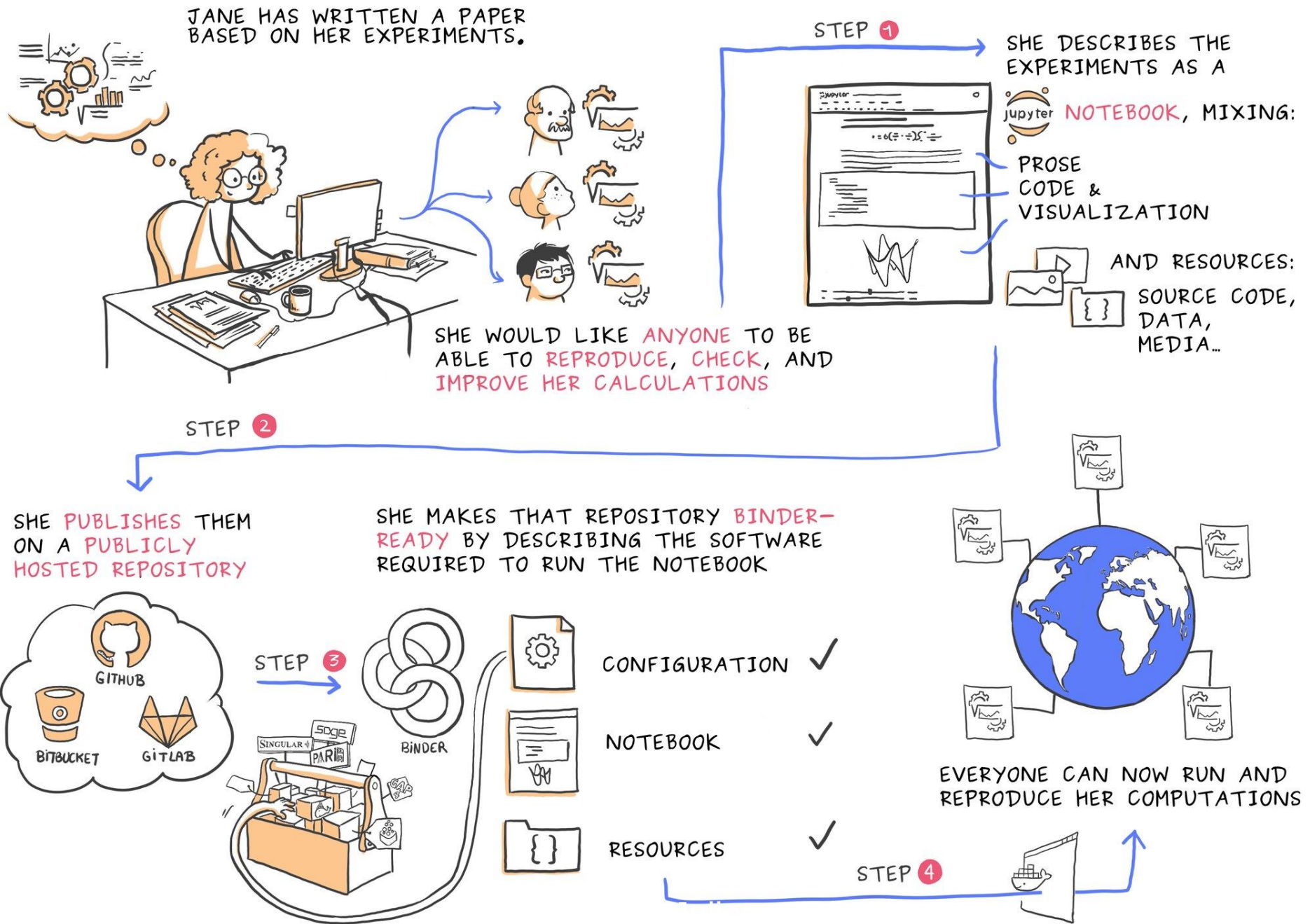
My Server

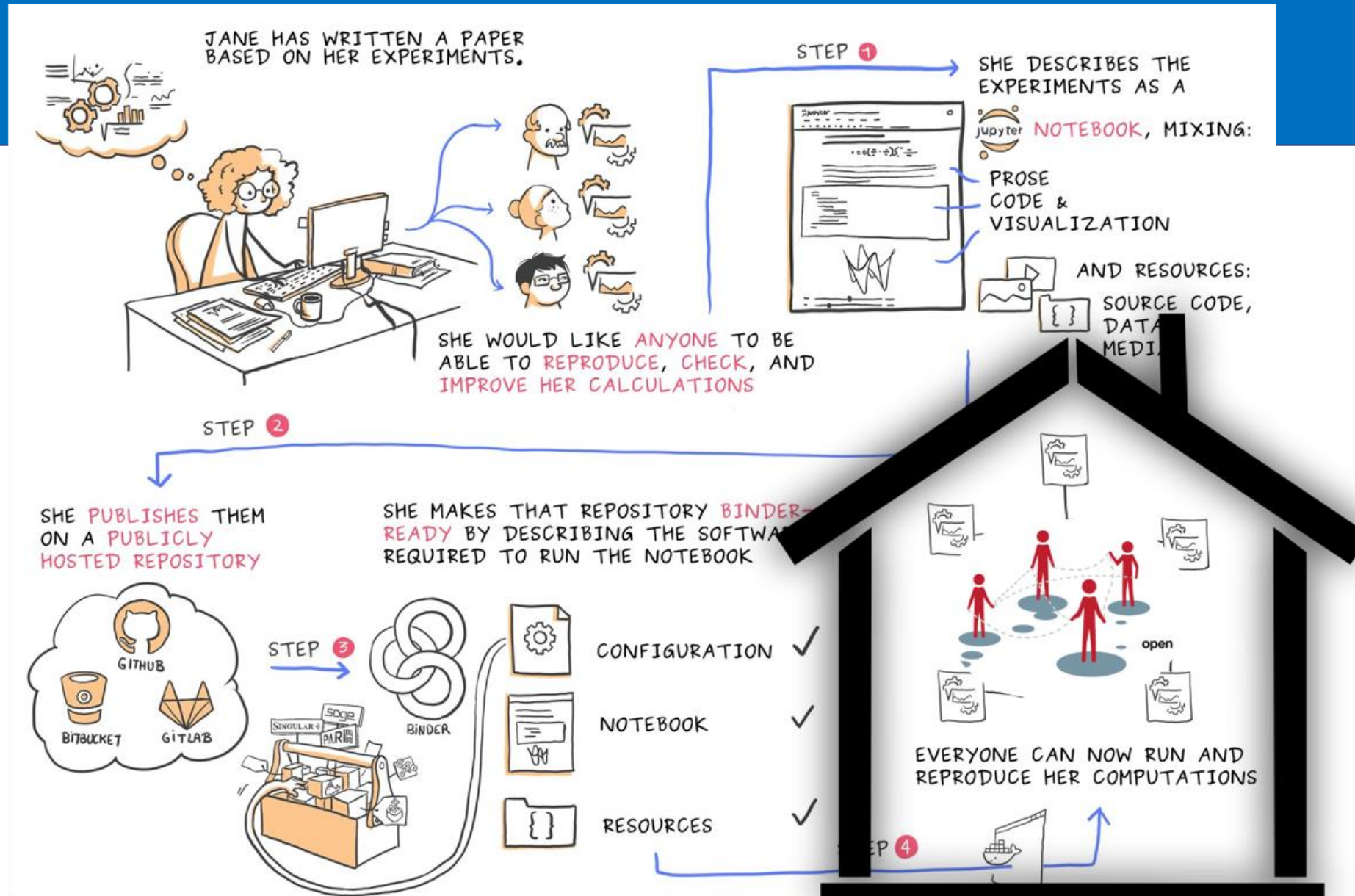
Go to <https://cs3.fedcloud-tf.fedcloud.eu/hub/home>
and stop your server before retry!

What happens when we click a Binder link?



1. The link to the repository is resolved by BinderHub
2. BinderHub searches for a Docker image relating to the provided reference (for example, git commit hash, branch or tag)
3. **If a Docker image is not found**, BinderHub requests resources from the Kubernetes cluster to run *repo2docker* to do the following:
 1. fetch the repository
 2. build a Docker image containing the software requested in the configuration file
 3. push that image to the Docker registry
4. BinderHub sends the Docker image to JupyterHub
5. JupyterHub requests resources from the Kubernetes cluster to serve the Docker image
6. JupyterHub connects the user's browser to the running Docker environment
7. JupyterHub monitors the container for activity and destroys it after a period of inactivity





Bibliography and further reading



- 🌐 <https://open-science-training-handbook.gitbook.io/book/open-science-basics/reproducible-research-and-data-analysis>
- 🌐 <https://www.annualreviews.org/doi/10.1146/annurev-publhealth-012420-105110>
- 🌐 <https://esajournals.onlinelibrary.wiley.com/doi/full/10.1002/bes2.1801>
- 🌐 Ioannidis, John PA. "Why most published research findings are false." *PLoS medicine* 2, no. 8 (2005): e124.
- 🌐 Schooler, Jonathan W. "Metascience could rescue the 'replication crisis'." *Nature* 515, no. 7525 (2014): 9-9.
- 🌐 Essawy, Bakinam T., Jonathan L. Goodall, Daniel Voce, Mohamed M. Morsy, Jeffrey M. Sadler, Young Don Choi, David G. Tarboton, and Tanu Malik. "A taxonomy for reproducible and replicable research in environmental modelling." *Environmental Modelling & Software* 134 (2020): 104753.
- 🌐 Peng, Roger D. "Reproducible research in computational science." *Science* 334, no. 6060 (2011): 1226-1227.

THANK YOU!



Follow us

