This lecture will be recorded

064-0026-00L: COMPAS II

Introduction to Computational Methods for Digital Fabrication in Architecture

Today's goal

Course **feedback** & **reproducibility** in research

*https://*dfab.link/fs2022-feedback

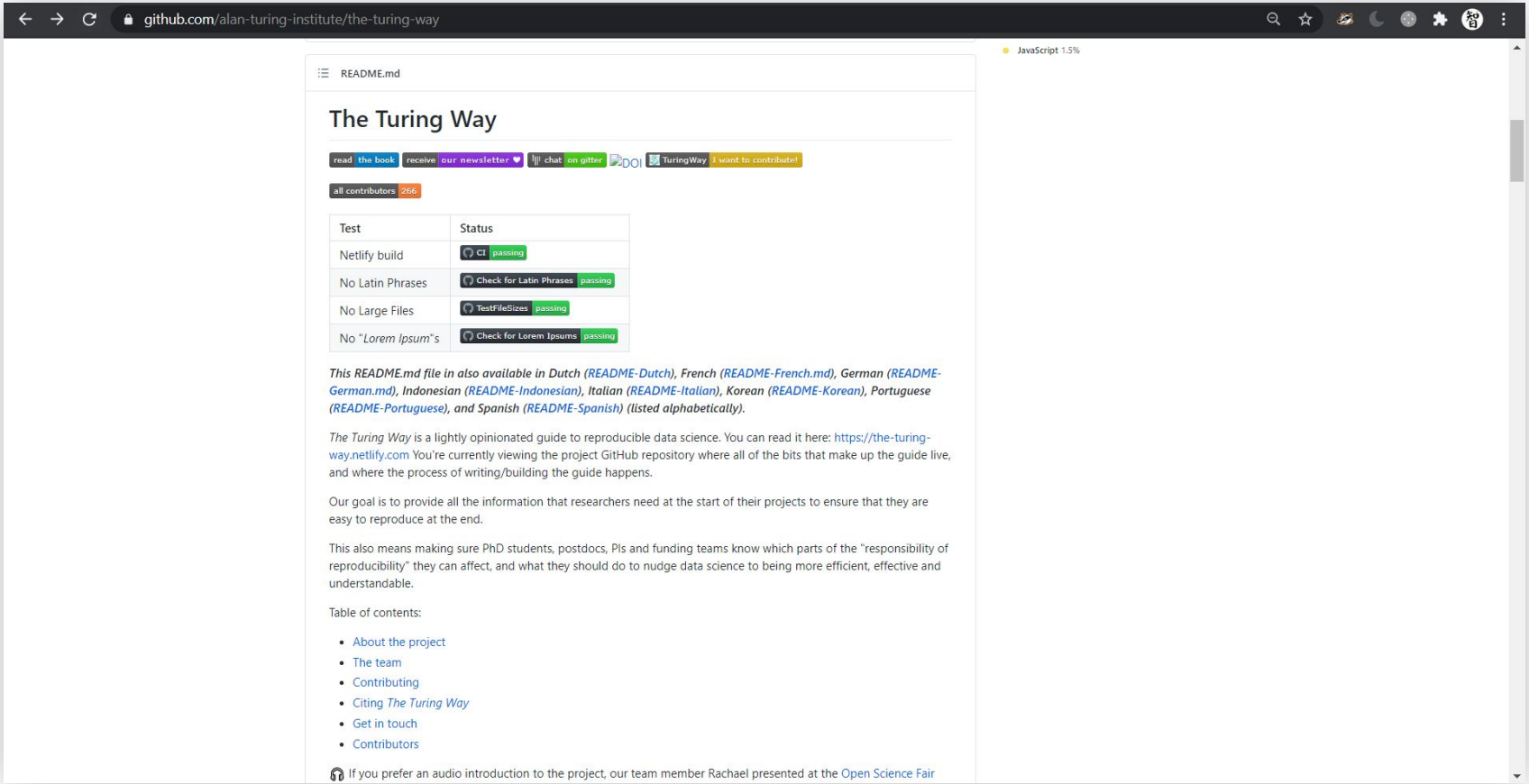
Advancing computational research

Reproducibility & Upstreaming

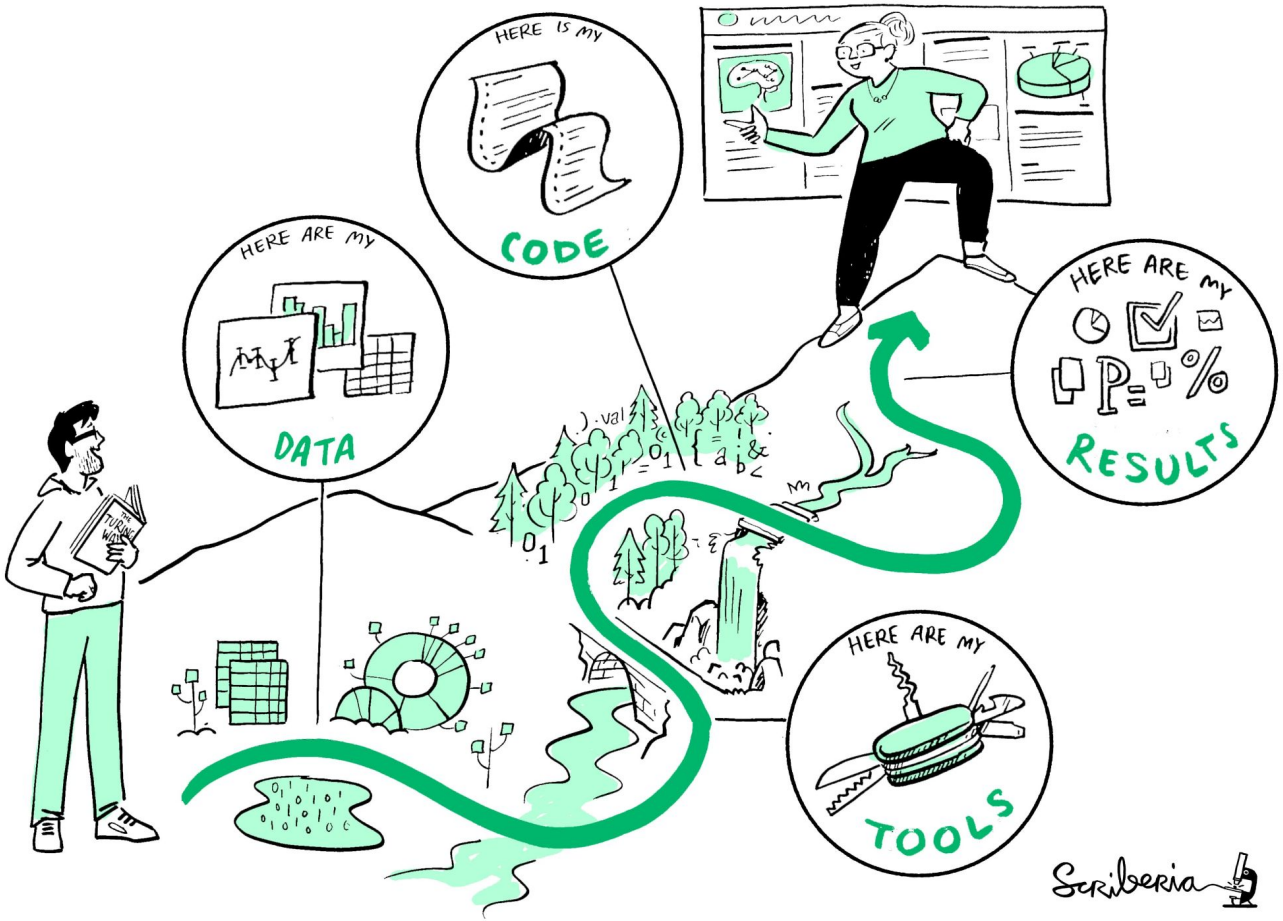
A definition of reproducible research

Work that can be **independently recreated** from the **same data** and the **same code** that the original team used.

The Turing Way, https://github.com/alan-turing-institute/the-turing-way

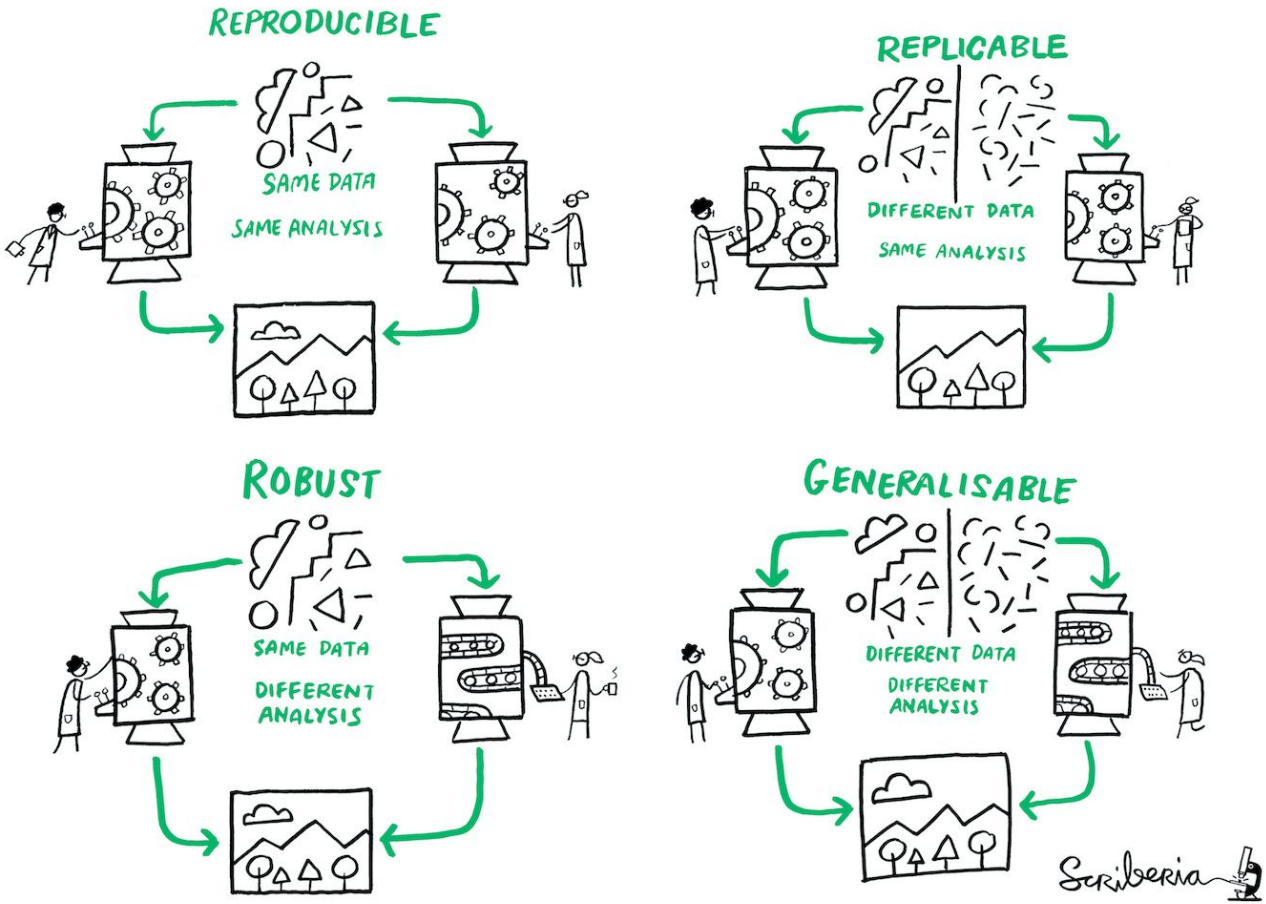


The Turing Way, https://github.com/alan-turing-institute/the-turing-way

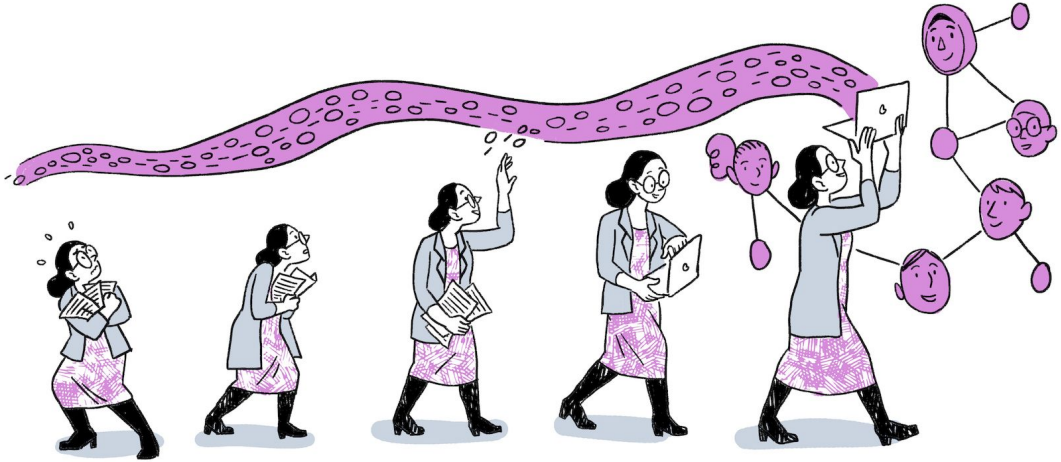


The Turing Way project illustration by Scriberia. Zenodo. http://doi.org/10.5281/zenodo.3332807

ETH zürich



The Turing Way project illustration by Scriberia. Original version on Zenodo. http://doi.org/10.5281/zenodo.3695300.



EVOLVING TOWARDS AN ERA OF OPEN RESEARCH

Scriberia

The Turing Way project illustration by Scriberia. Original version on Zenodo. http://doi.org/10.5281/zenodo.3695300.

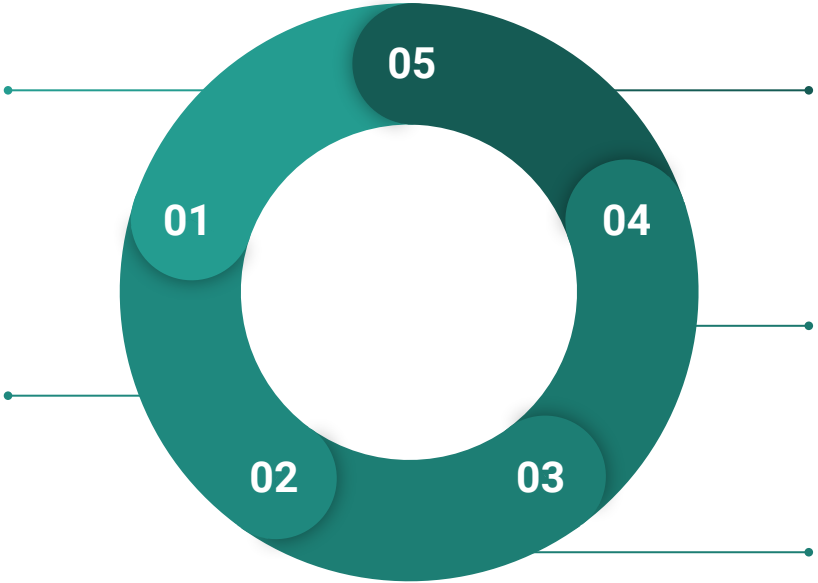
Elements of open research

Open Data

Documenting and sharing research data openly for re-use.

Open Source Software

Documenting research code and routines, and making them freely accessible and available.



Open Notebooks

An emerging practice, documenting and sharing the experimental process of trial and error.

Open Access

Making all published outputs freely accessible for maximum use and impact.

Open Hardware

Documenting designs, materials, and other relevant information related to hardware, and making them freely accessible and available.

Open Source Software

- Put your code in a **freely accessible repository**.
- Include a **license** granting others the right to use, copy and modify your work.
- Include a **README** file containing useful information about a project such as what it is, how to use/install it and how to run any tests.
- If you want others to collaborate on the project include **contribution guidelines**.

Open Data

- Ensure your data is in a simple, **standard format** or formats which is machine and human-readable.
- Check, reformat or create **metadata to clearly describe what the data is**, how it was collected, and any associated strengths/weaknesses to someone that finds it.
- Identify a relevant, easily **discoverable** repository or repositories to host your data, and upload it there.
- Assign your data a **persistent identifier** such as a DOI.

Open Hardware

- Make detailed **documentation and designs** for any hardware you develop **openly available**.
- Include a **license** granting others the right to use, copy and modify your work.
- Include a **README** file containing useful information about a project (for example, what it is and the materials used).

Open Access

- Publish your research in an **open-access journal**.
- Store a copy or **preprint** of your work in a freely accessible public repository.

Open Notebook

- Keep notes in an **Electronic Lab Notebook**.
- Make your notebooks **publicly accessible** online.

Licenses

Spectrum of OSS licenses

Strong copyleft

eg. GPL

Free

Weak copyleft

eg. LGPL

Free

Permissive

eg. MIT, BSD, Apache

Open

Proprietary

Non-free, closed

choosealicense.com

Data

The Turing Way project illustration by Scriberia. Original version on Zenodo. http://doi.org/10.5281/zenodo.3695300

Software

Code management

Reproducible environments

- Tools for reproducible environments:
  - Package managers (conda/pip)
  - Containers (Docker)
- Use and rely on semantic versioning: semver.org
  - MAJOR.MINOR.PATCH
  - e.g. 1.5.0
- Describe your dependencies meticulously

Code quality

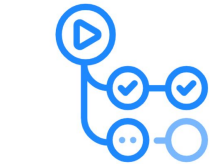
- **Always optimize for readability**
- Follow naming & style conventions (PEP8)
- Naming is hard, take your time
- Encapsulate complexity: solve it just once
- Don’t reinvent the wheel, reuse encapsulated code
- Less code is always more
- **Leverage tools to assist your coding (linters, formaters, static analyzers, etc)**

Code testing

- There are lots of test types
  - Unit tests
  - Integration tests
  - Performance tests
- But at least, write unit tests. Please, pretty please! 😉
- Test algorithms, not UI
- If you fix a bug, write a test to verify and make sure you don’t break it again.
- Run them automatically

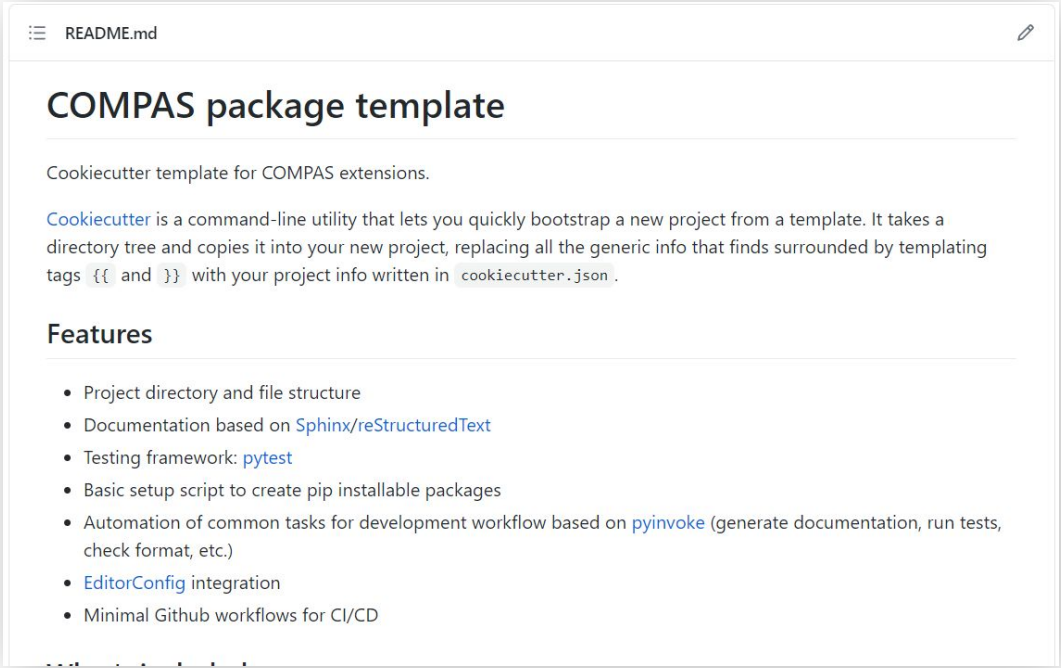
Continuous Integration

- Automate all the thingz!
  - Testing
  - Building/compiling
  - Publishing
  - Style / consistency checks
  - Anything repetitive!
- Makes sure you discover regressions when they occur



GitHub Actions

COMPAS Templates



README.md

COMPAS package template

Cookiecutter template for COMPAS extensions.

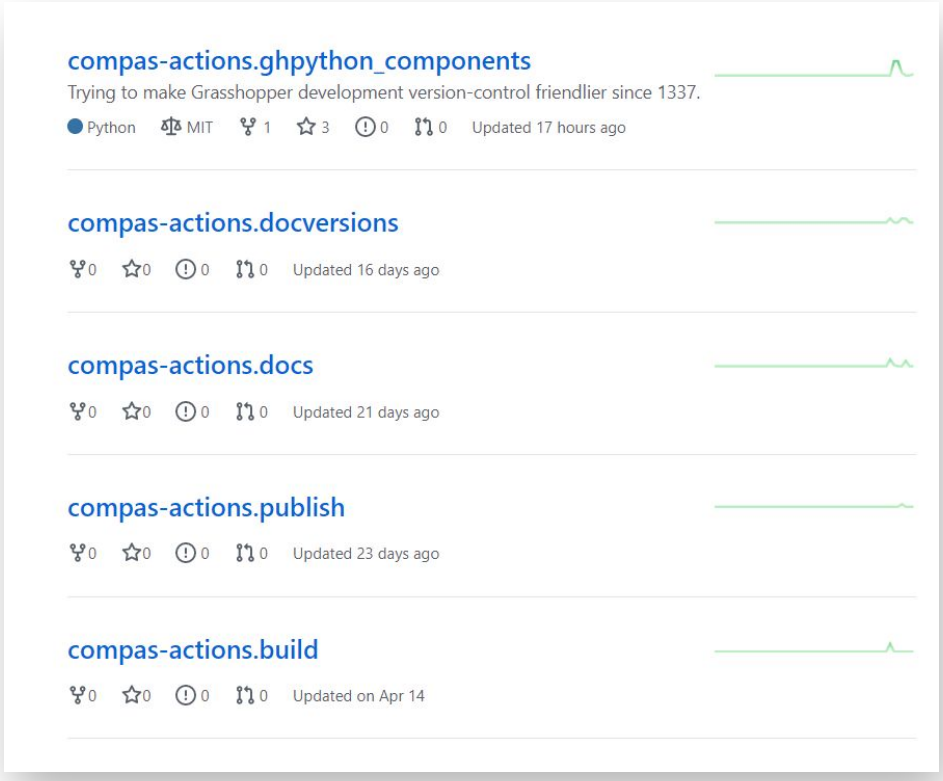
Cookiecutter is a command-line utility that lets you quickly bootstrap a new project from a template. It takes a directory tree and copies it into your new project, replacing all the generic info that finds surrounded by templating tags `{{` and `}}` with your project info written in `cookiecutter.json`.

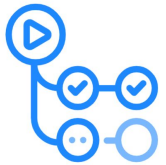
Features

- Project directory and file structure
- Documentation based on Sphinx/reStructuredText
- Testing framework: pytest
- Basic setup script to create pip installable packages
- Automation of common tasks for development workflow based on pyinvoke (generate documentation, run tests, check format, etc.)
- EditorConfig integration
- Minimal Github workflows for CI/CD

New extension/package: https://github.com/compas-dev/tpl-extension

COMPAS Actions

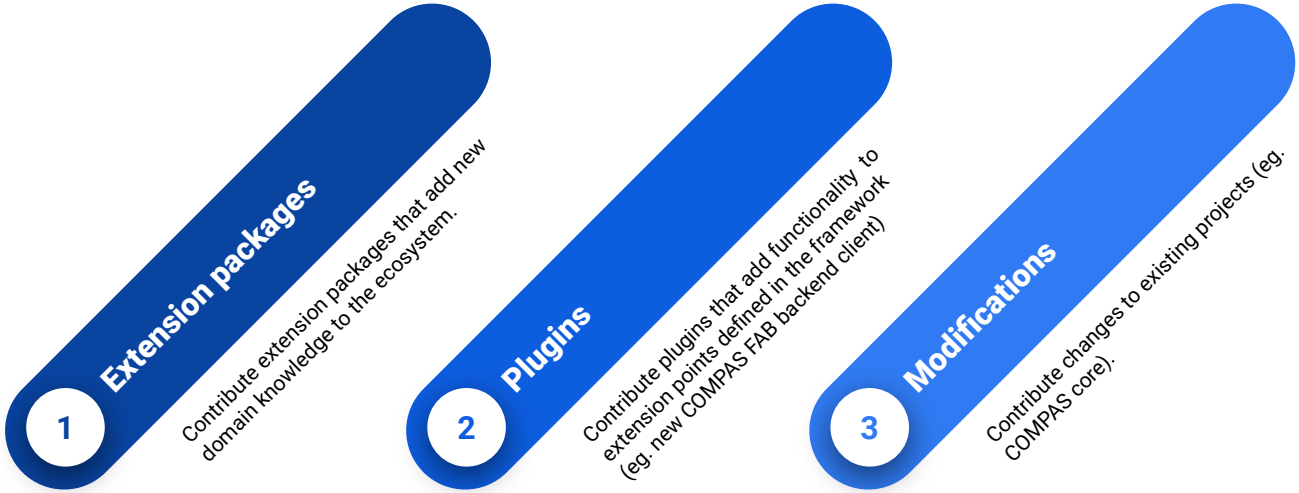




GitHub Actions

...is the process of **contributing** private research source code **back to the open source sphere**.

Contribution types in COMPAS ecosystem



Upstream process

Identify

Identify components in your research that can be upstreamed

Principles

Ensure adherence to framework's principles:

- Python-based coding
- CAD-agnostic coding
- Code style & conventions: naming, structure, protocols.

Organization

Establish organization & mgmt best practices:

- Version control
- Documentation
- Dev & release processes

Extract

Extract upstream components

- Modifications: pull request
- Extensions: new repository

Publish

Publish project if extension package, or stay involved in upstream integration otherwise

Sustainability

Ensure sustainable maintenance and ownership is in place.

More resources

The Turing Way
https://the-turing-way.netlify.app

Awesome Reproducible Research (and other awesome-* lists)
https://github.com/leipzig/awesome-reproducible-research

Thanks!