The Alan Turing Institute

Celebrating 5 years with *The Turing Way*

Kirstie Whitaker
Pronouns: she/her

@TuringWay
https://doi.org/10.5281/zenodo.11098175

Personal experience of:

- File drawer effect
- Lack of reproducibility
- Imposter syndrome around coding
- Lonely
- Wasted time
- Hypocrisy of academia

Personal experience of:

- File drawer effect
- Lack of reproducibility
- Imposter syndrome around coding
- Lonely
- Wasted time
- Hypocrisy of academia

The Alan Turing Institute

The Turing Way

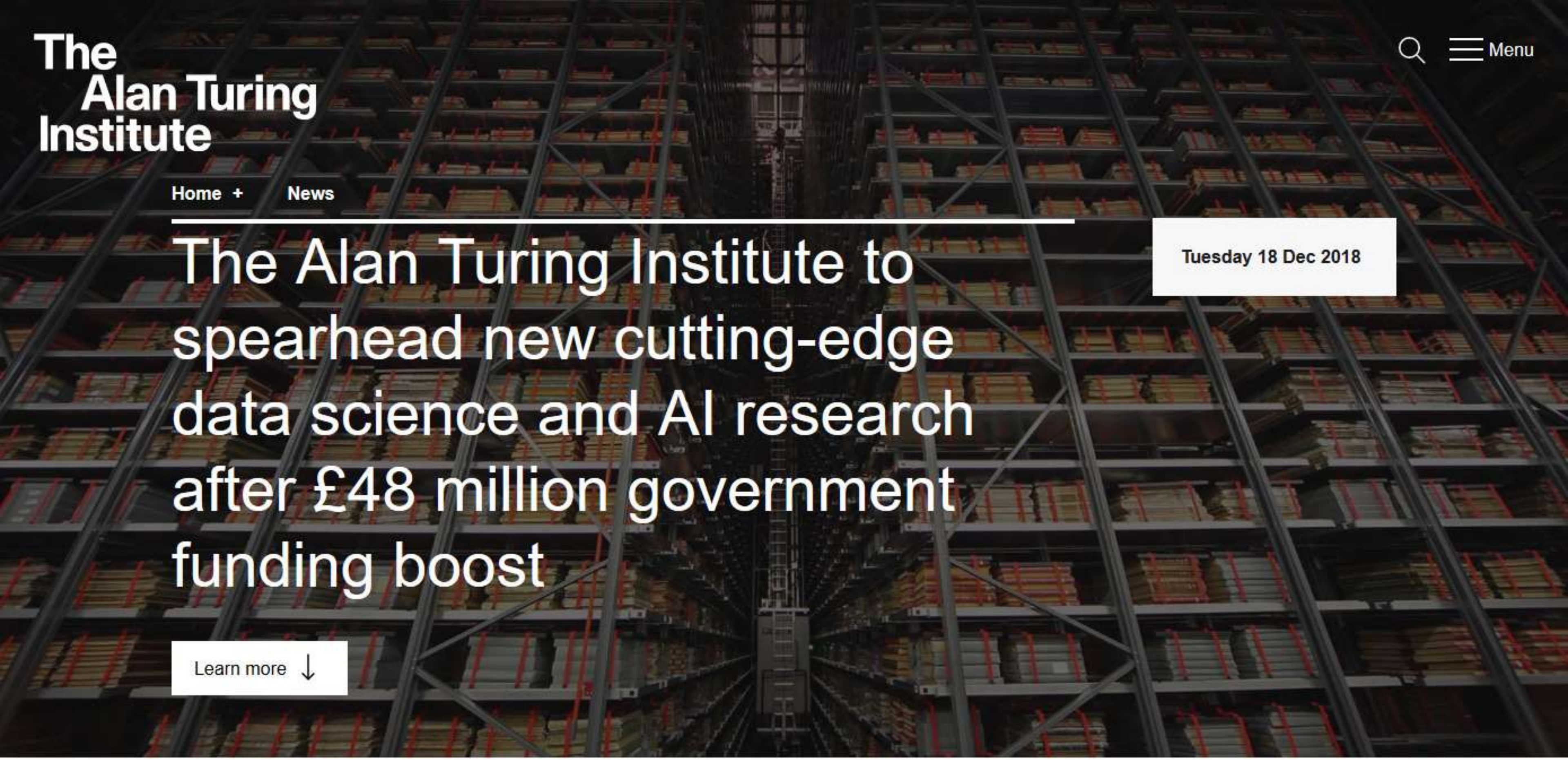
A handbook for reproducible research

Kirstie Whitaker

Collaborations Workshop Demo, 2 April 2019

Slides at https://doi.org/10.5281/zenodo.2621280

https://doi.org/10.5281/zenodo.11098175

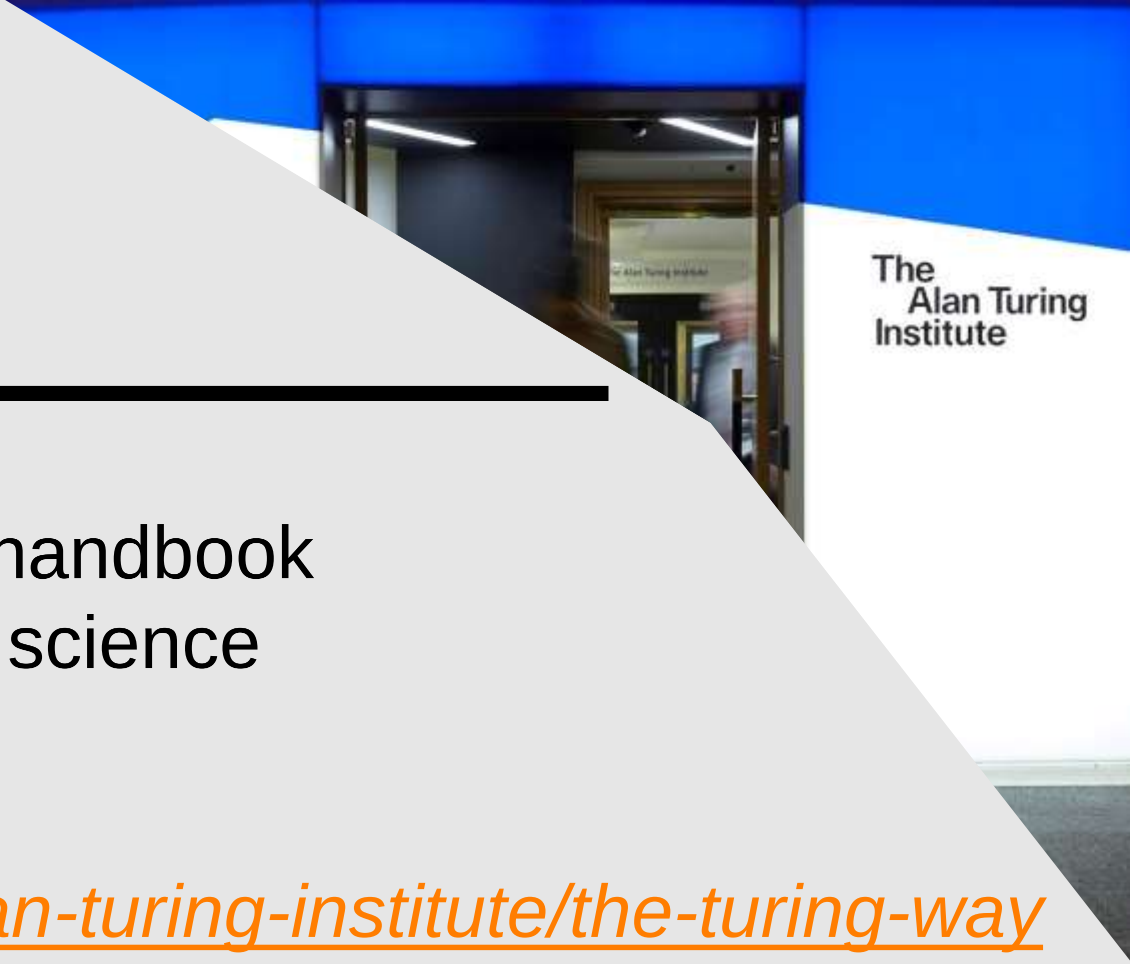


Tuesday 18 Dec 2018

The Alan Turing Institute to spearhead new cutting-edge data science and AI research after £48 million government funding boost

https://www.turing.ac.uk/news/alan-turing-institute-spearhead-new-cutting-edge-data-science-and-artificial-intelligence

https://doi.org/10.5281/zenodo.11098175



The Turing Way

A lightly opinionated handbook
for reproducible data science

https://github.com/alan-turing-institute/the-turing-way

https://doi.org/10.5281/zenodo.11098175

What does reproducible mean?

		Data	
		Same	Different
Analysis	Same	Reproducible	Replicable
	Different	Robust	Generalisable

https://dx.doi.org/10.6084/m9.figshare.7140050

https://doi.org/10.5281/zenodo.11098175

Why don’t people do this already?

Is not considered for promotion

Takes time

Publication bias towards novel findings

Barriers to reproducible research

Requires additional skills

Plead the 5th

Held to higher standards than others

Support additional users

https://dx.doi.org/10.6084/m9.figshare.7140050

https://doi.org/10.5281/zenodo.11098175

Why don’t people do this already?

Is not considered for promotion

Takes time

Publication bias towards novel findings

Barriers to reproducible research

Requires additional skills

Plead the 5th

Held to higher standards than others

Support additional users

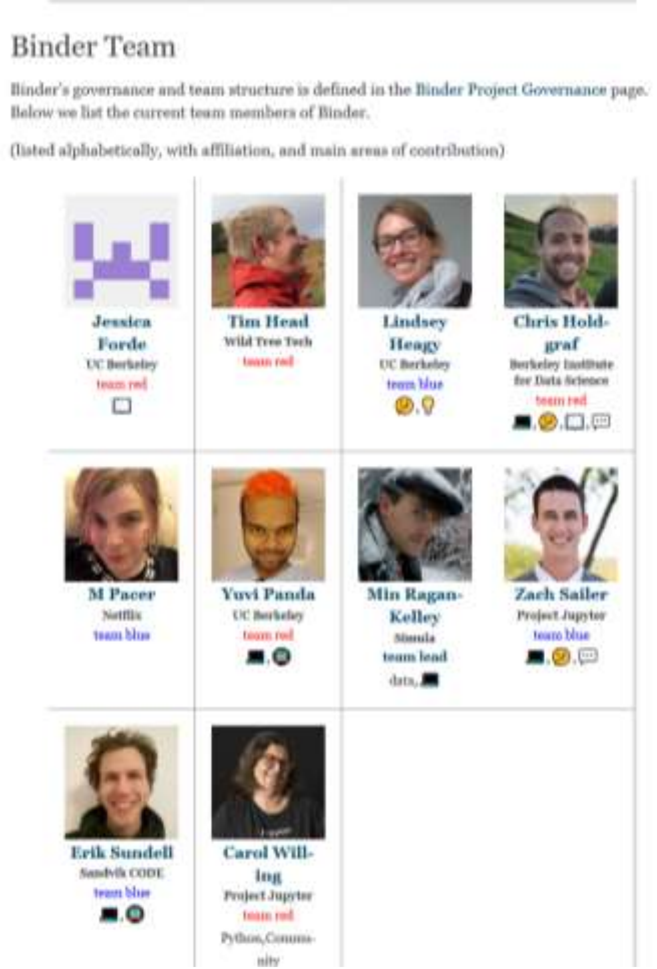
https://dx.doi.org/10.6084/m9.figshare.7140050

https://github.com/alan-turing-institute/the-turing-way/tree/master/workshops

https://doi.org/10.5281/zenodo.11098175

https://github.com/alan-turing-institute/the-turing-way/tree/master/workshops

https://doi.org/10.5281/zenodo.11098175



Binder Team

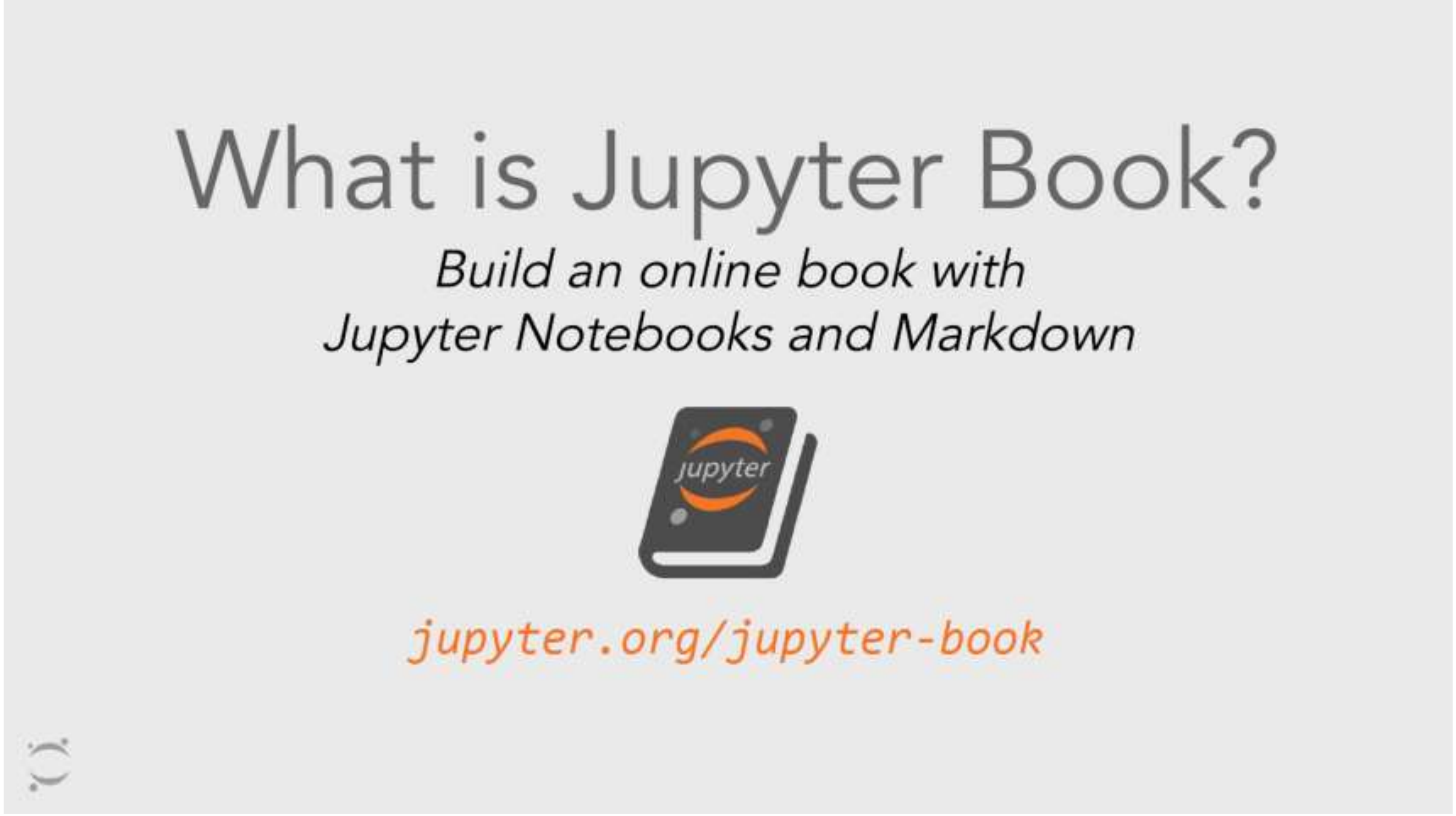
Binder's governance and team structure is defined in the Binder Project Governance page. Below we list the current team members of Binder.

(listed alphabetically, with affiliation, and main areas of contribution)

Jessica Forde UC Berkeley team red 📖	Tim Head Wild Tree Tech team red	Lindsey Heagy UC Berkeley team blue 🤔, 💡	Chris Holdgraf Berkeley Institute for Data Science team red 💻, 🤔, 📖, 💬
M Pacer Netflix team blue	Yuvi Panda UC Berkeley team red 💻, 🚇	Min Ragan-Kelley Simula team lead data, 💻	Zach Sailer Project Jupyter team blue 💻, 🤔, 💬
Erik Sundell Sandvik CODE team blue 💻, 🚇	Carol Willing Project Jupyter team red Python, Community		

https://github.com/alan-turing-institute/the-turing-way/tree/master/workshops

https://doi.org/10.5281/zenodo.11098175

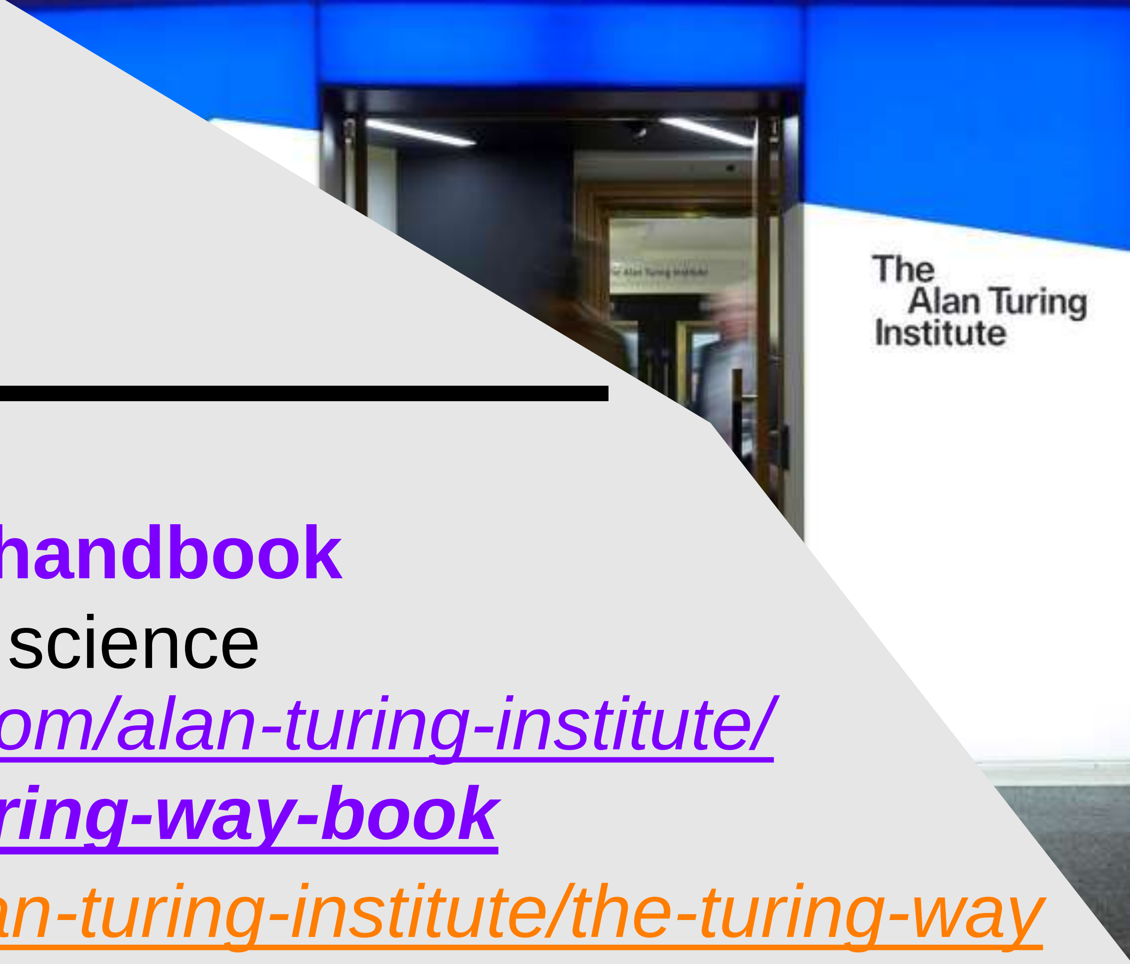


What is Jupyter Book?

Build an online book with Jupyter Notebooks and Markdown

jupyter.org/jupyter-book

https://speakerdeck.com/choldgraf/jupyter-book-interactive-books-running-in-the-cloud



The Turing Way

A lightly opinionated **handbook** for reproducible data science

https://github.com/alan-turing-institute/ ***the-turing-way-book***

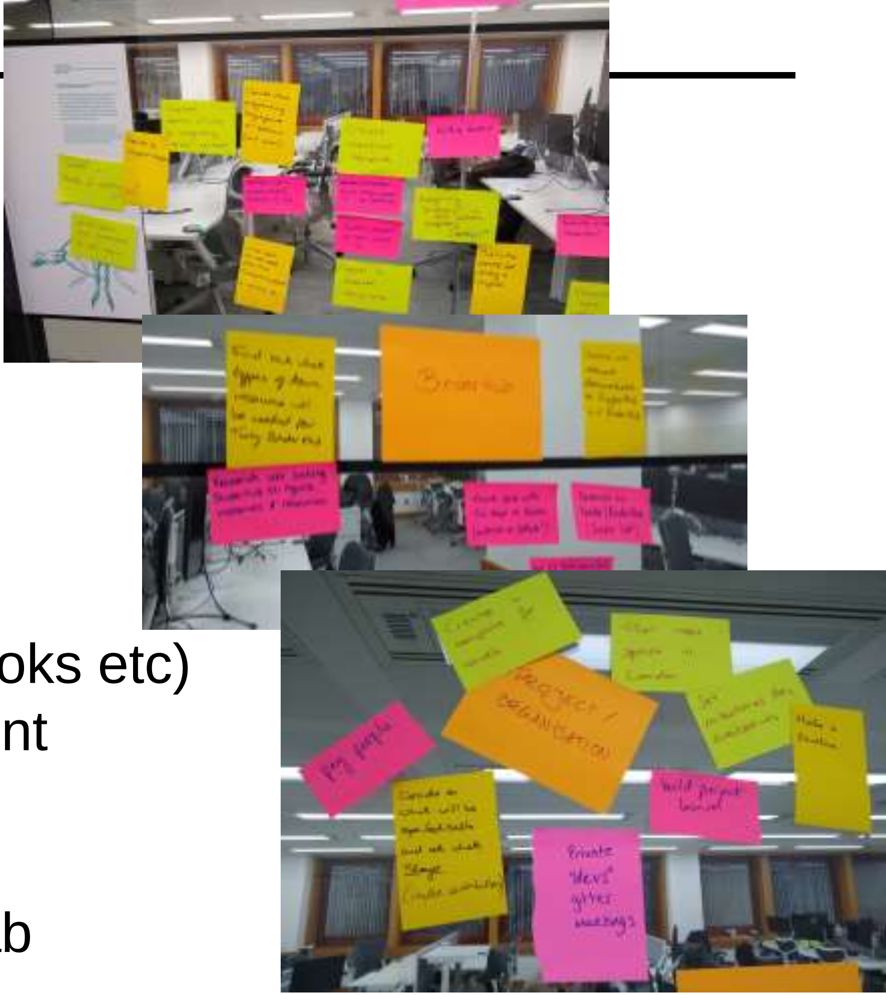
https://github.com/alan-turing-institute/the-turing-way

https://doi.org/10.5281/zenodo.11098175

Requires additional skills

Chapters will include:

- Research data management
- Open science
- Reproducibility
- Version control with git
- Your working environment (IDE, notebooks etc)
- Capturing your compute environment
- Testing for research
- Continuous integration
- Collaborating through GitHub/GitLab



https://github.com/alan-turing-institute/the-turing-way/blob/master/book_skeleton.md

https://doi.org/10.5281/zenodo.11098175

Requires additional skills

Chapters will include:

- Research data management
- Open research
- Reproducibility
- Version control with git
- Your working environment (IDE, notebooks etc)
- Capturing your compute environment
- Testing for research
- Continuous integration
- Collaborating through GitHub/GitLab

https://github.com/alan-turing-institute/the-turing-way/blob/master/book_skeleton.md

https://doi.org/10.5281/zenodo.11098175

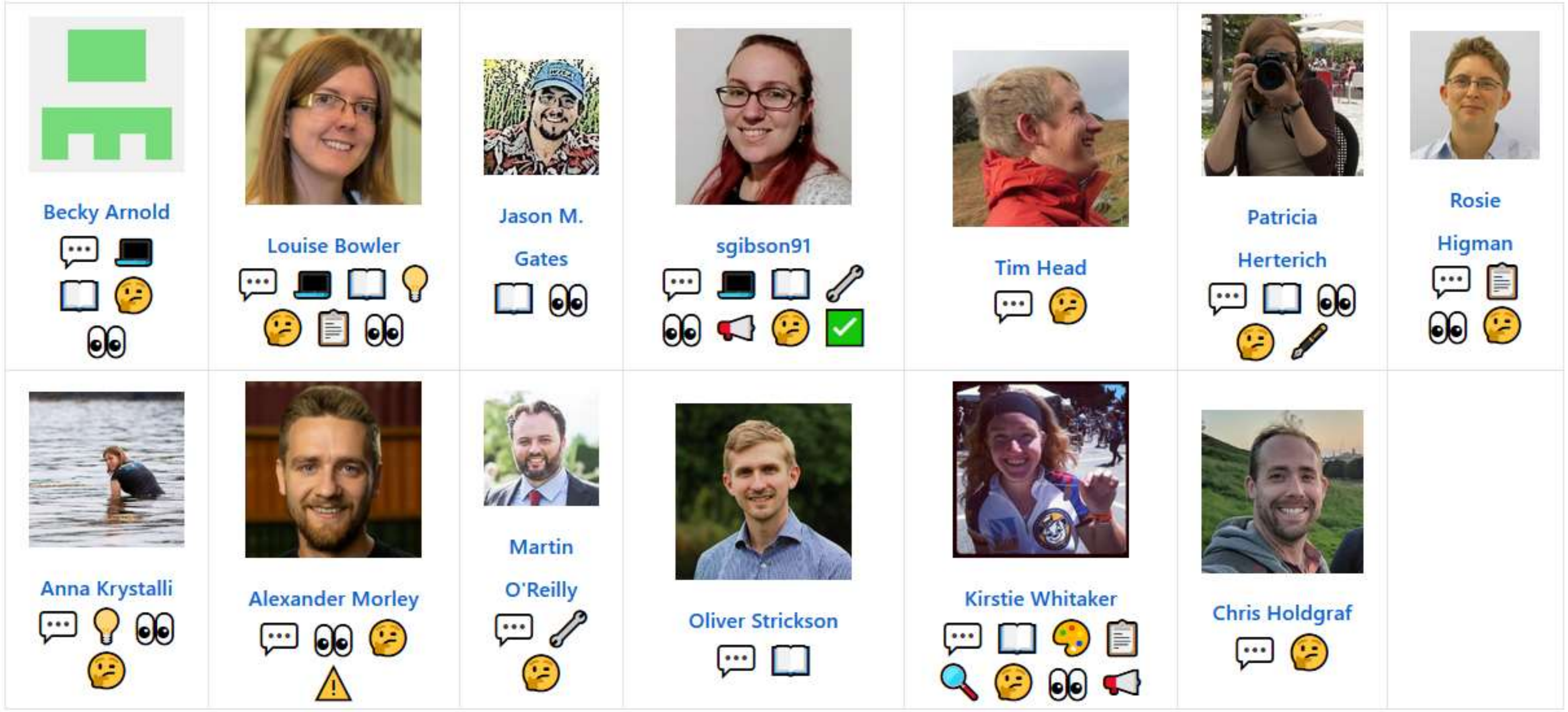
Built by a team….and you!

- Rachael Ainsworth
- Becky Arnold
- Louise Bowler
- Sarah Gibson
- Patricia Herterich
- Rosie Higman
- Anna Krystalli
- Alex Morley
- Martin O’Reilly
- . . .

https://github.com/alan-turing-institute/
the-turing-way/blob/master/contributors.md

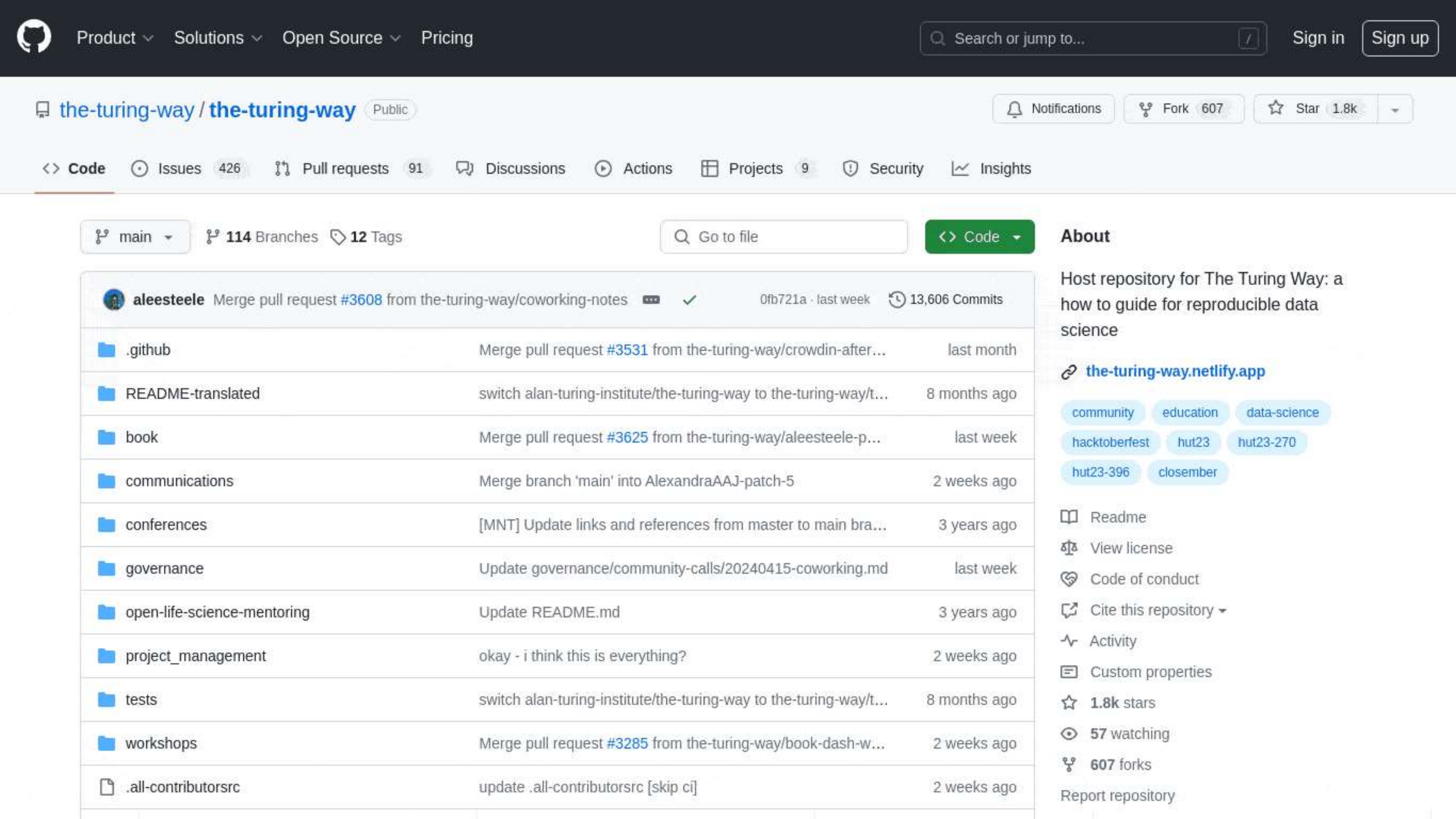
https://doi.org/10.5281/zenodo.11098175

The emoji key to celebrate our contributors



Becky Arnold 💬 💻 📖 🤔 👀	Louise Bowler 💬 💻 📖 💡 🤔 📋 👀	Jason M. Gates 📖 👀	sgibson91 💬 💻 📖 🔧 👀 📢 🤔 ✅	Tim Head 💬 🤔	Patricia Herterich 💬 📖 👀 🤔 🖋	Rosie Higman 💬 📋 👀 🤔
Anna Krystalli 💬 💡 👀 🤔	Alexander Morley 💬 👀 🤔 ⚠️	Martin O'Reilly 💬 🔧 🤔	Oliver Strickson 💬 📖	Kirstie Whitaker 💬 📖 🎨 📋 🔍 🤔 👀 📢	Chris Holdgraf 💬 🤔	

https://github.com/alan-turing-institute/the-turing-way/blob/master/README.md



Product Solutions Open Source Pricing

Search or jump to...

Sign in Sign up

the-turing-way / **the-turing-way** Public

Notifications Fork 607 Star 1.8k

Code Issues 426 Pull requests 91 Discussions Actions Projects 9 Security Insights

main 114 Branches 12 Tags Go to file Code

aleesteele Merge pull request #3608 from the-turing-way/coworking-notes 0fb721a · last week 13,606 Commits

.github	Merge pull request #3531 from the-turing-way/crowdin-after…	last month
README-translated	switch alan-turing-institute/the-turing-way to the-turing-way/t…	8 months ago
book	Merge pull request #3625 from the-turing-way/aleesteele-p…	last week
communications	Merge branch 'main' into AlexandraAAJ-patch-5	2 weeks ago
conferences	[MNT] Update links and references from master to main bra…	3 years ago
governance	Update governance/community-calls/20240415-coworking.md	last week
open-life-science-mentoring	Update README.md	3 years ago
project_management	okay - i think this is everything?	2 weeks ago
tests	switch alan-turing-institute/the-turing-way to the-turing-way/t…	8 months ago
workshops	Merge pull request #3285 from the-turing-way/book-dash-w…	2 weeks ago
.all-contributorsrc	update .all-contributorsrc [skip ci]	2 weeks ago

About

Host repository for The Turing Way: a how to guide for reproducible data science

the-turing-way.netlify.app

community education data-science hacktoberfest hut23 hut23-270 hut23-396 closember

Readme
View license
Code of conduct
Cite this repository
Activity
Custom properties
1.8k stars
57 watching
607 forks
Report repository

Localisation & Translation

Batool Almarzouq, Andrea Tapia Sanchez, Melissa Black

Research Infrastructure Roles

Arielle Bennett, Esther Plomp, Emma Karoune (Skills Policy Award)

Training and outreach

Regular volunteer and experts from the community

Infrastructure Maintainers

Sarah Gibson, Jim Madge, Danny Garside, Brigitta Sipőcz

Accessibility

Liz Hare, Laurel Ascenzi, Alexandra Araujo Alvarez

Practitioners Hub

Cross-sector engagement

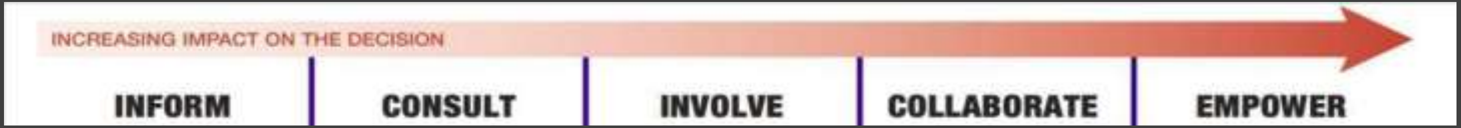
Different PathWAYs in *The Turing Way*

Environmental Data Science

Alejandro Coca, Anne Fouilloux

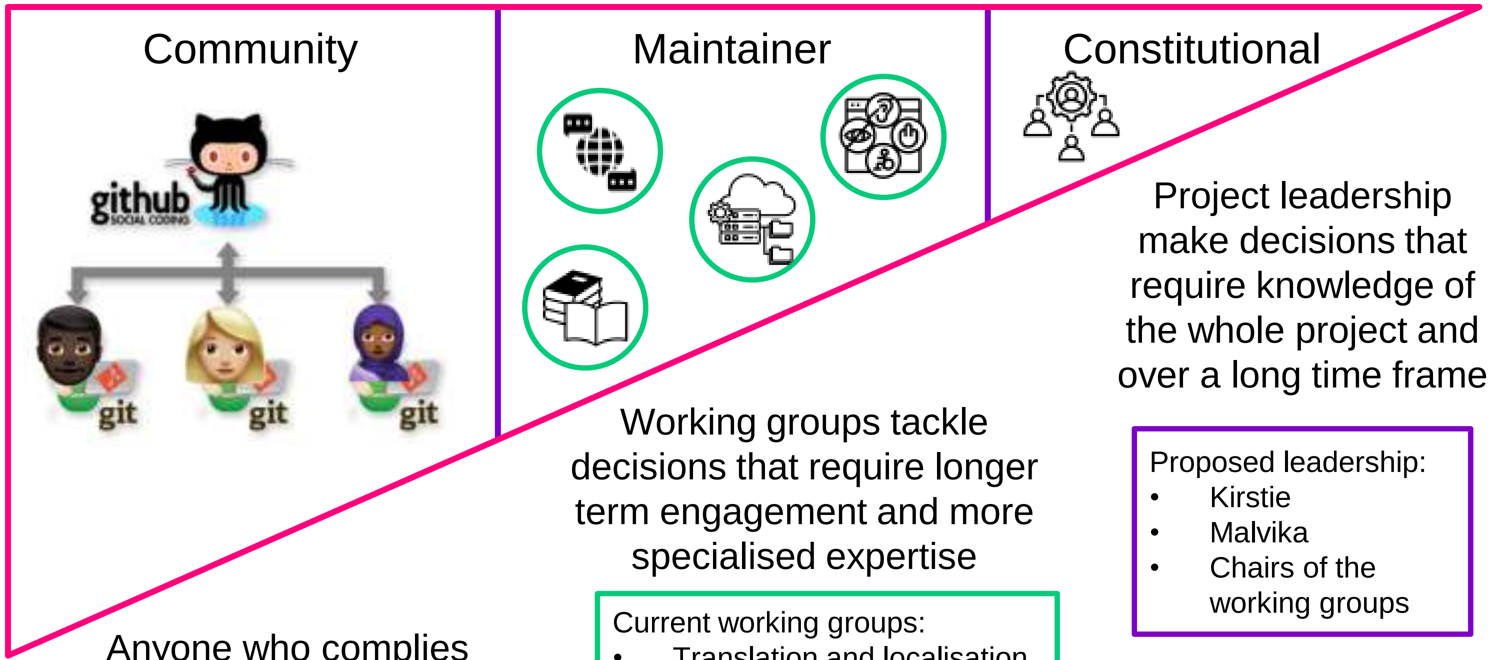
Book Dash and Community Events

Team and term-based roles.



Spectrum of Public Participation – Organizing Engagement. https://organizingengagement.org/models/spectrum-of-public-participation

https://doi.org/10.5281/zenodo.11098175



Project leadership make decisions that require knowledge of the whole project and over a long time frame

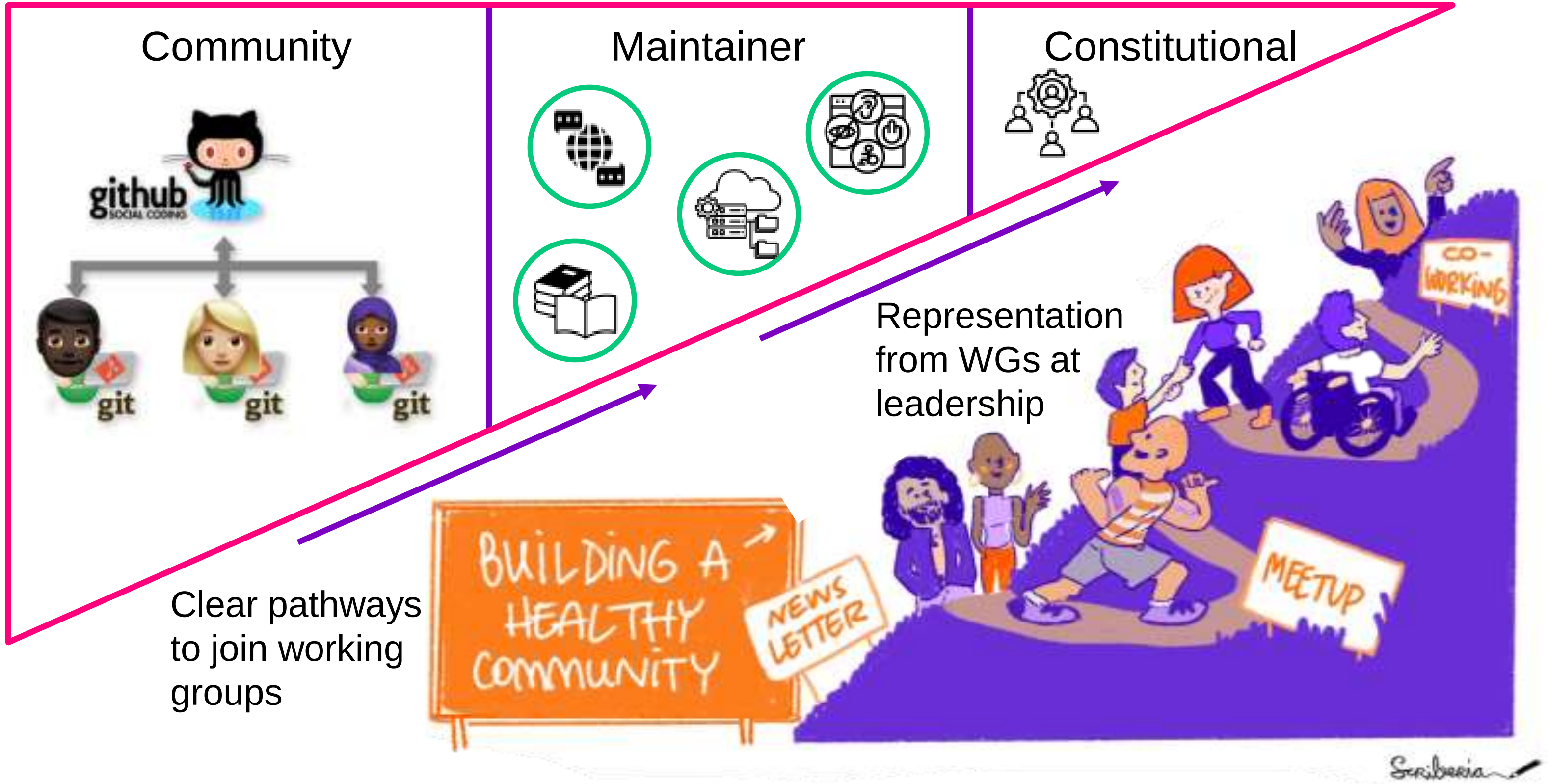
Working groups tackle decisions that require longer term engagement and more specialised expertise

Anyone who complies with the project's code of conduct can participate

Current working groups:
- Translation and localisation
- Accessibility
- Infrastructure
- Book Dash

Proposed leadership:
- Kirstie
- Malvika
- Chairs of the working groups

https://doi.org/10.5281/zenodo.11098175


Community
Maintainer
Constitutional
github
git
git
git
Representation from WGs at leadership
Clear pathways to join working groups
BUILDING A HEALTHY COMMUNITY
NEWS LETTER
MEETUP
CO-WORKING

Malvika Sharan

“No one can change research culture on their own. Scaling our community by empowering YOU to participate is how we will change the world.”

https://doi.org/10.5281/zenodo.10069243
https://doi.org/10.15131/shef.data.24162249.v1

@TuringWay
https://doi.org/10.5281/zenodo.11098175

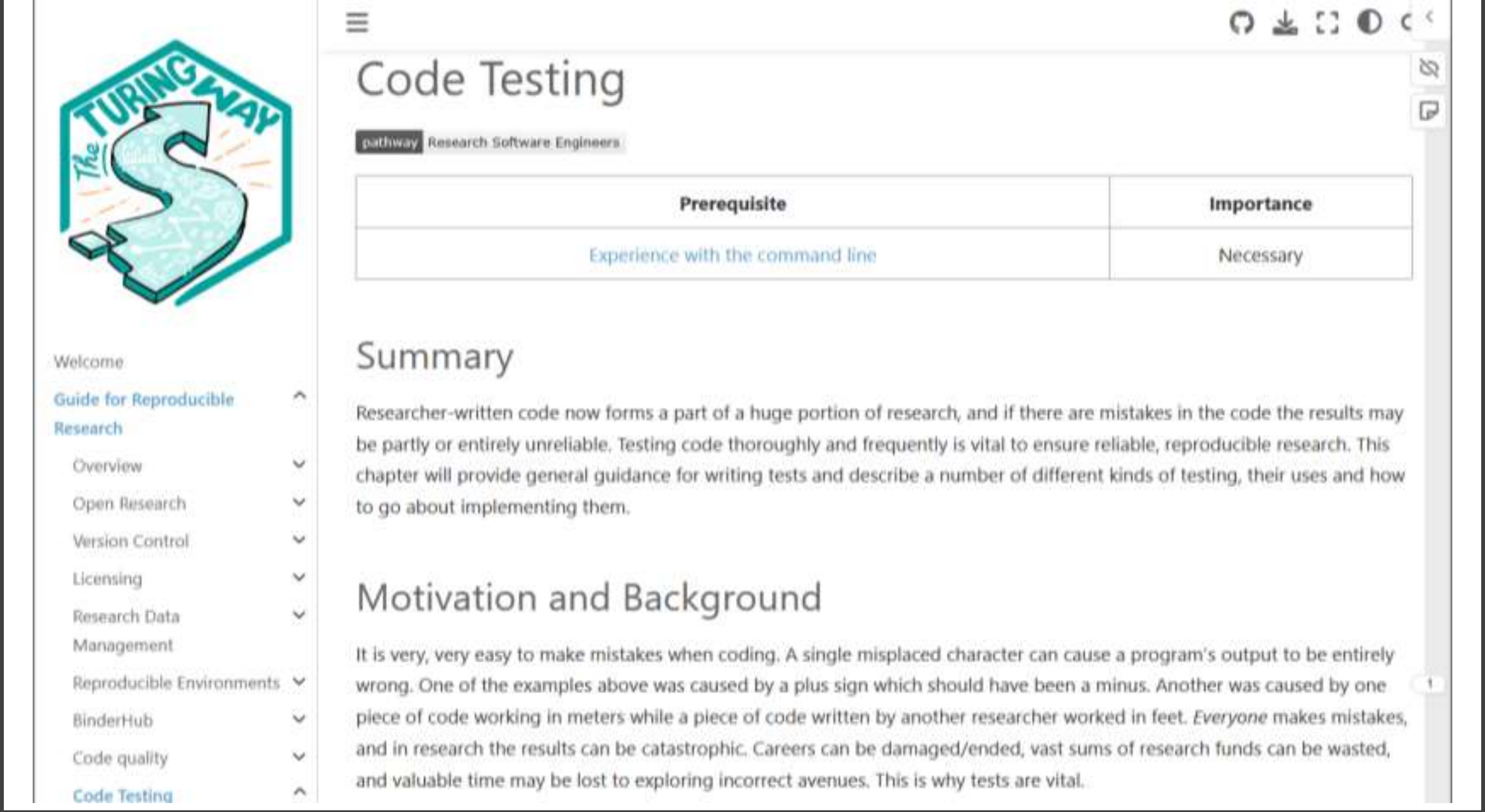
The Alan Turing Institute

The Turing Way: Software Testing

Kirstie Whitaker
Pronouns: she/her

@TuringWay

https://doi.org/10.5281/zenodo.11098175



Code Testing

pathway Research Software Engineers

Prerequisite	Importance
Experience with the command line	Necessary

Summary

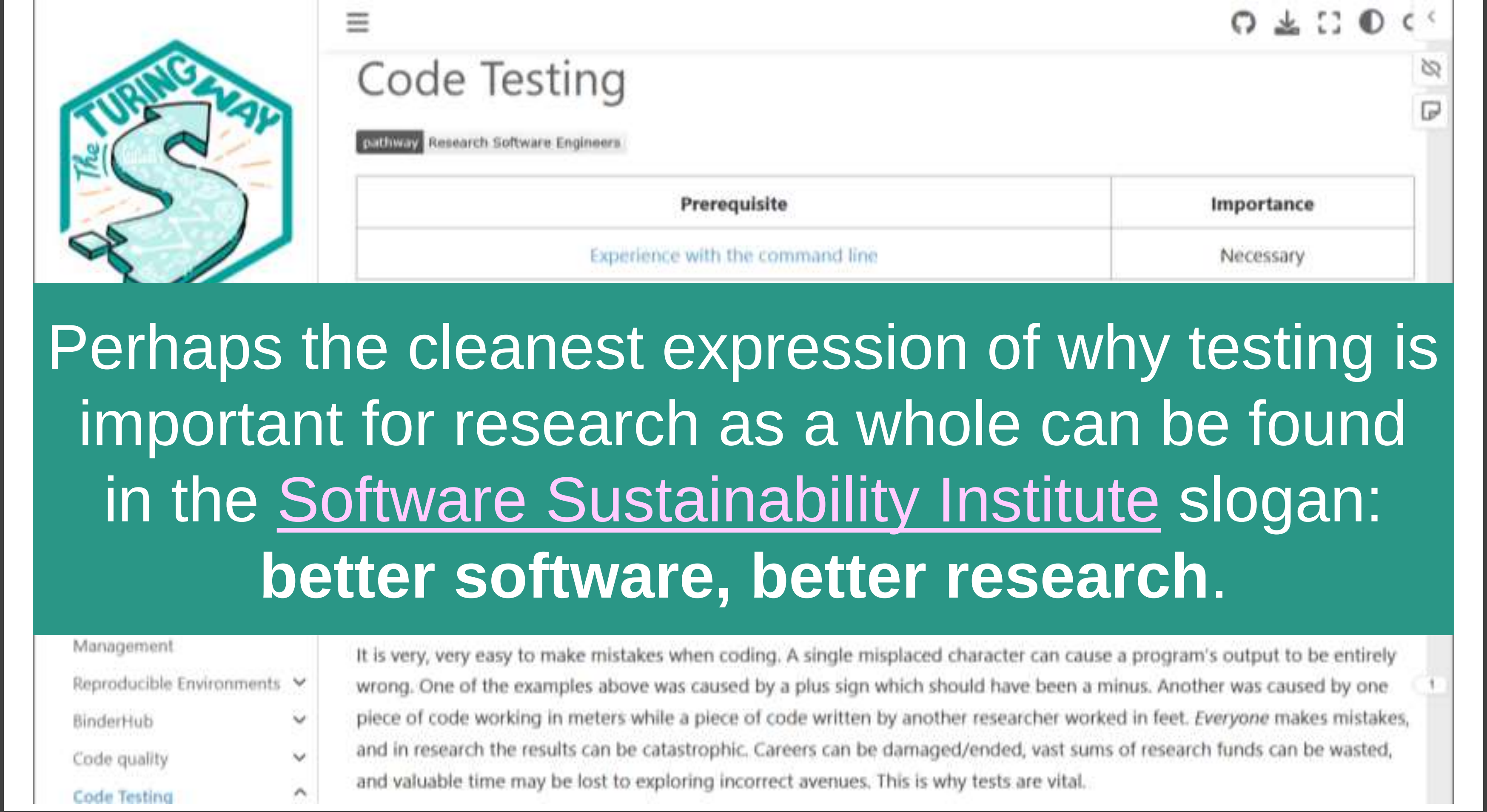
Researcher-written code now forms a part of a huge portion of research, and if there are mistakes in the code the results may be partly or entirely unreliable. Testing code thoroughly and frequently is vital to ensure reliable, reproducible research. This chapter will provide general guidance for writing tests and describe a number of different kinds of testing, their uses and how to go about implementing them.

Motivation and Background

It is very, very easy to make mistakes when coding. A single misplaced character can cause a program's output to be entirely wrong. One of the examples above was caused by a plus sign which should have been a minus. Another was caused by one piece of code working in meters while a piece of code written by another researcher worked in feet. *Everyone* makes mistakes, and in research the results can be catastrophic. Careers can be damaged/ended, vast sums of research funds can be wasted, and valuable time may be lost to exploring incorrect avenues. This is why tests are vital.

https://the-turing-way.netlify.app/index.html
https://github.com/the-turing-way/the-turing-way

@TuringWay
https://doi.org/10.5281/zenodo.11098175



Code Testing

pathway Research Software Engineers

Prerequisite	Importance
Experience with the command line	Necessary

Perhaps the cleanest expression of why testing is important for research as a whole can be found in the Software Sustainability Institute slogan: **better software, better research**.

It is very, very easy to make mistakes when coding. A single misplaced character can cause a program's output to be entirely wrong. One of the examples above was caused by a plus sign which should have been a minus. Another was caused by one piece of code working in meters while a piece of code written by another researcher worked in feet. *Everyone* makes mistakes, and in research the results can be catastrophic. Careers can be damaged/ended, vast sums of research funds can be wasted, and valuable time may be lost to exploring incorrect avenues. This is why tests are vital.

Is your code doing what you think it's doing?

```
Assert.AreEqual(
    GetTimeOfDay(),
    "Morning" );
```

https://the-turing-way.netlify.com/reproducible-research/testing.html
https://www.toptal.com/qa/how-to-write-testable-code-and-why-it-matters

@TuringWay
https://doi.org/10.5281/zenodo.11098175

Is your code doing what you think it's doing?

https://the-turing-way.netlify.com/reproducible-research/testing.html
https://www.toptal.com/qa/how-to-write-testable-code-and-why-it-matters

Is your code doing what you think it's doing?

```
Assert.AreEqual(
    GetTimeOfDay(),
      "Morning" )
```

```
Assert.AreEqual(
    GetTimeOfDay(),
    "Morning" );
```

https://the-turing-way.netlify.com/reproducible-research/testing.html
https://www.toptal.com/qa/how-to-write-testable-code-and-why-it-matters

@TuringWay
https://doi.org/10.5281/zenodo.11098175

Is your code doing what you think it's doing?

```
Assert.AreEqual(
    GetTimeOfDay(),
        "Morning" )
```

https://the-turing-way.netlify.com/reproducible-research/testing.html
https://www.toptal.com/qa/how-to-write-testable-code-and-why-it-matters https://doi.org/10.5281/zenodo.11098175

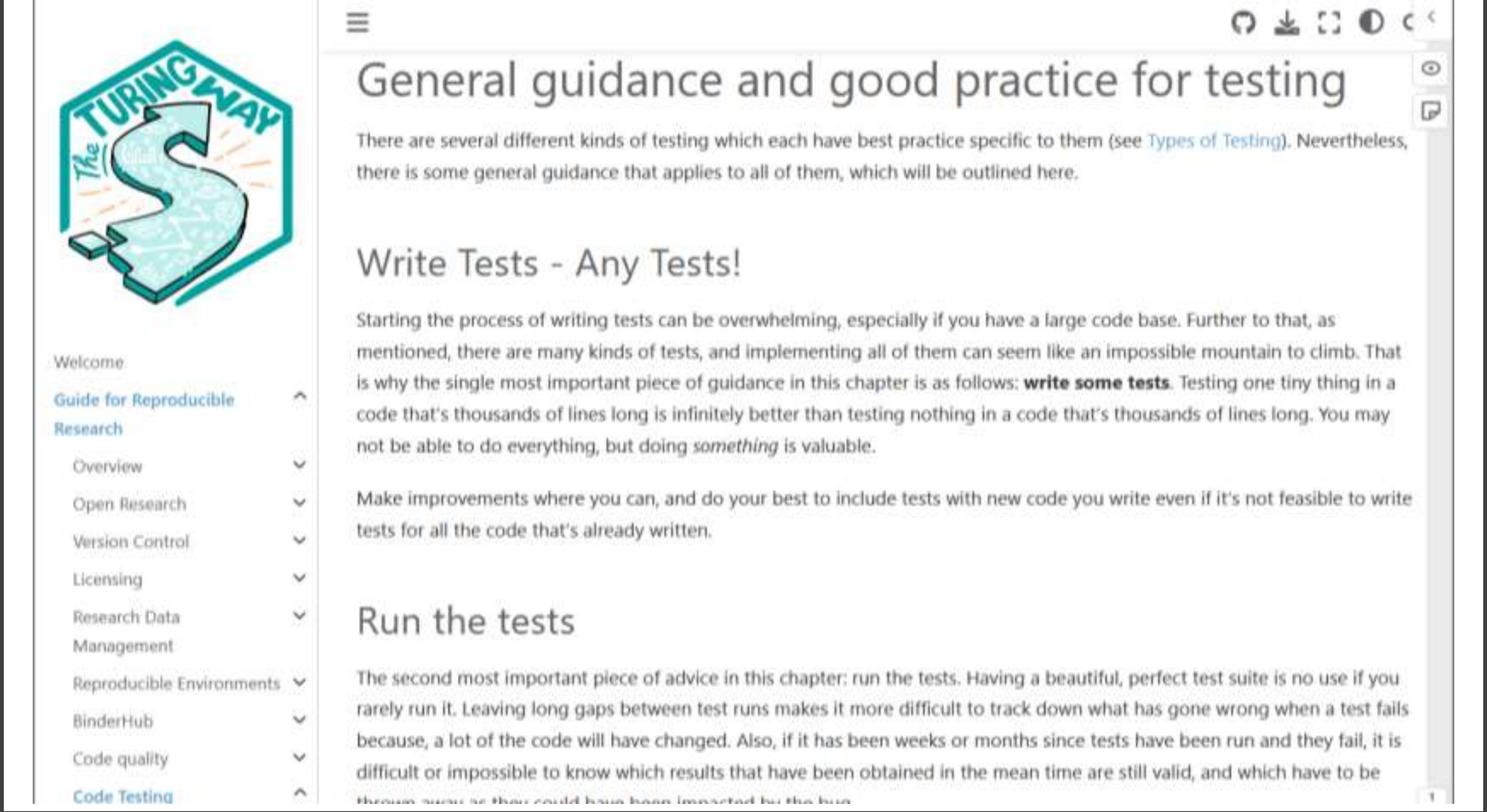
@TuringWay

Is your code doing what you think it's doing?

Testing sub-chapters on:
- Smoke
- Unit
- Integration
- System
- Acceptance and regression
- Runtime

And continuous integration…

https://the-turing-way.netlify.com/reproducible-research/testing.html
https://www.toptal.com/qa/how-to-write-testable-code-and-why-it-matters



General guidance and good practice for testing

There are several different kinds of testing which each have best practice specific to them (see Types of Testing). Nevertheless, there is some general guidance that applies to all of them, which will be outlined here.

Write Tests - Any Tests!

Starting the process of writing tests can be overwhelming, especially if you have a large code base. Further to that, as mentioned, there are many kinds of tests, and implementing all of them can seem like an impossible mountain to climb. That is why the single most important piece of guidance in this chapter is as follows: **write some tests**. Testing one tiny thing in a code that's thousands of lines long is infinitely better than testing nothing in a code that's thousands of lines long. You may not be able to do everything, but doing *something* is valuable.

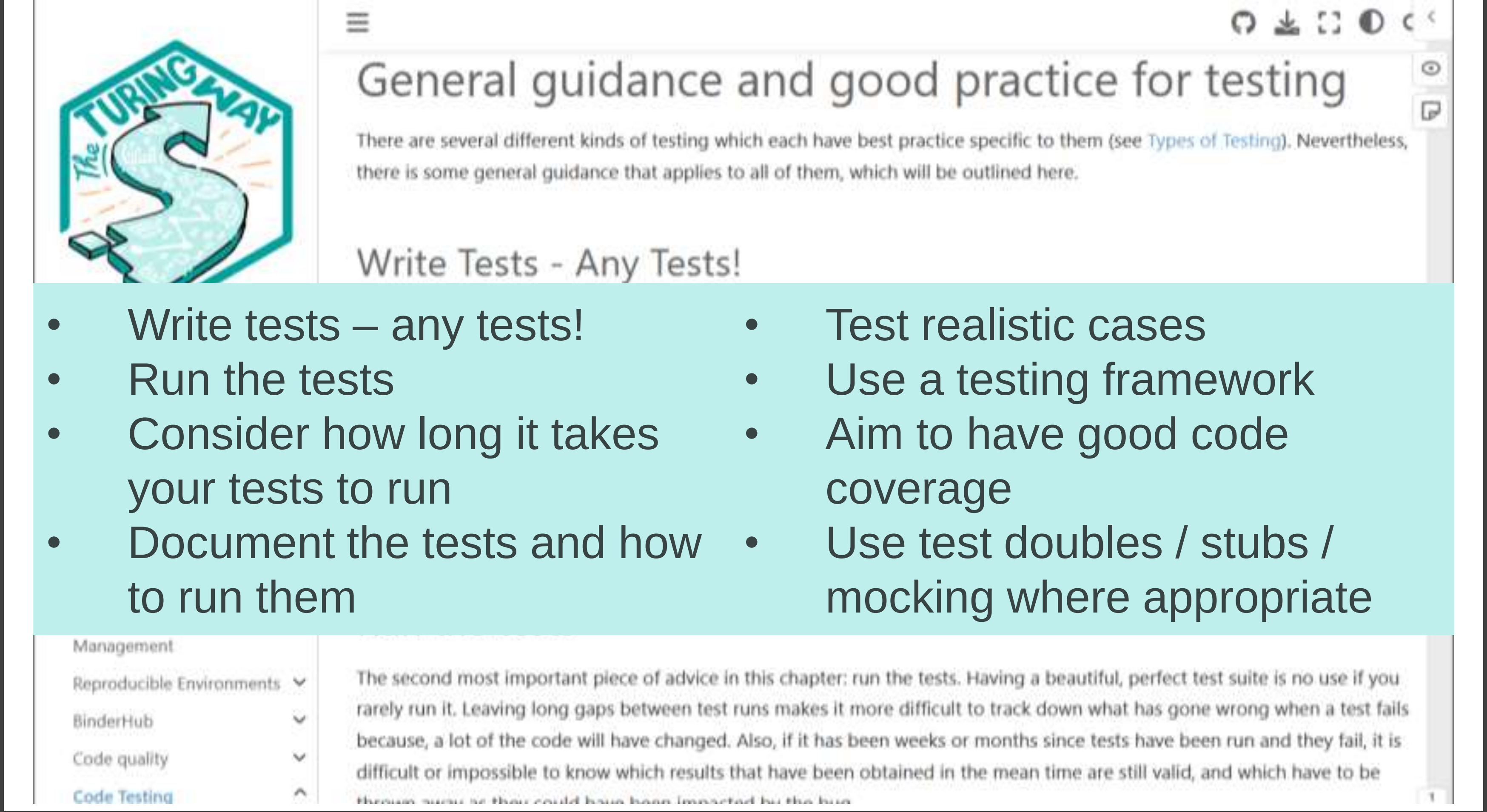
Make improvements where you can, and do your best to include tests with new code you write even if it's not feasible to write tests for all the code that's already written.

Run the tests

The second most important piece of advice in this chapter: run the tests. Having a beautiful, perfect test suite is no use if you rarely run it. Leaving long gaps between test runs makes it more difficult to track down what has gone wrong when a test fails because, a lot of the code will have changed. Also, if it has been weeks or months since tests have been run and they fail, it is difficult or impossible to know which results that have been obtained in the mean time are still valid, and which have to be thrown away as they could have been impacted by the bug.

https://the-turing-way.netlify.app/index.html
https://github.com/the-turing-way/the-turing-way

@TuringWay
https://doi.org/10.5281/zenodo.11098175



General guidance and good practice for testing

There are several different kinds of testing which each have best practice specific to them (see Types of Testing). Nevertheless, there is some general guidance that applies to all of them, which will be outlined here.

Write Tests - Any Tests!

Management

Reproducible Environments

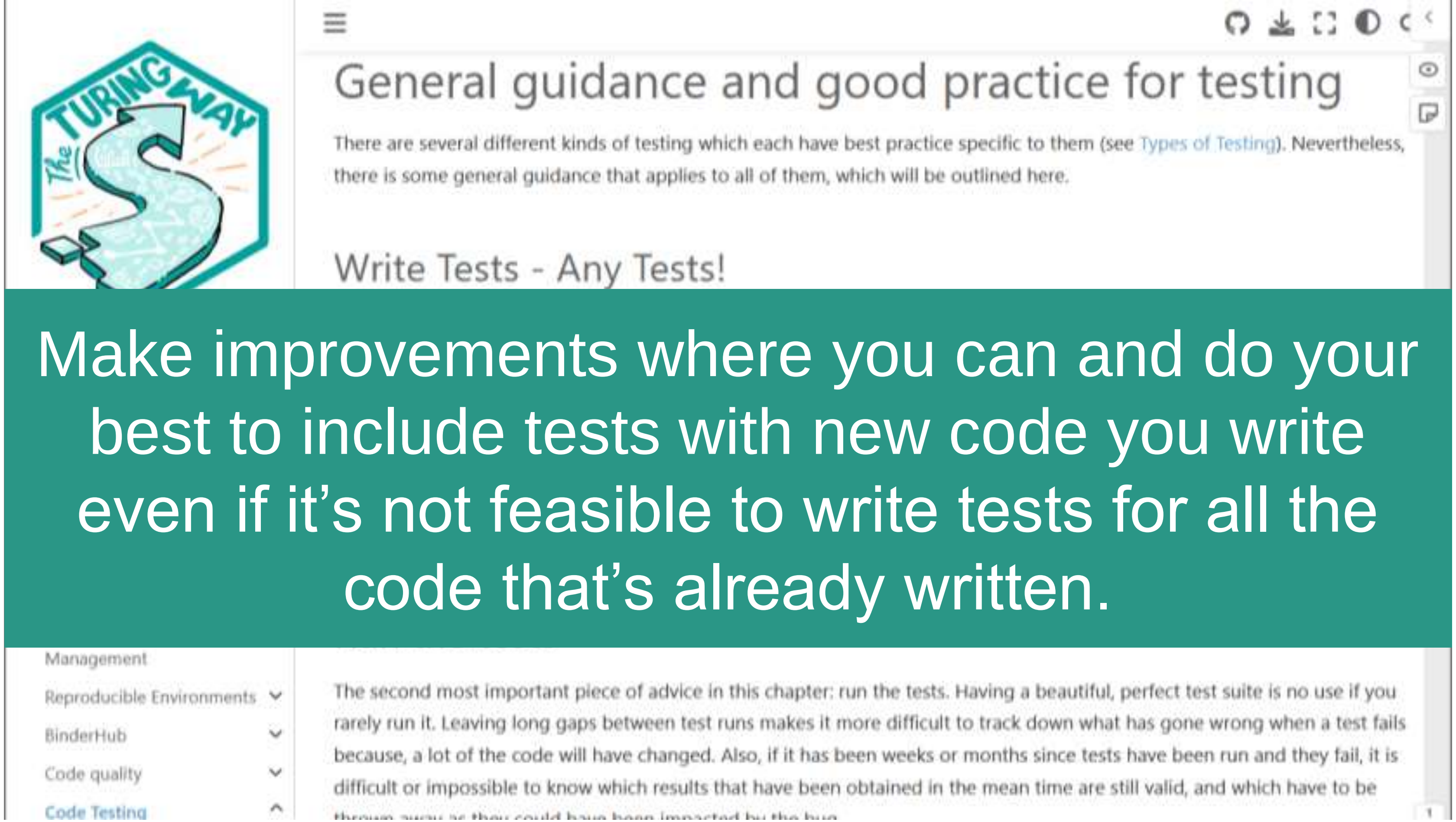
BinderHub

Code quality

Code Testing

The second most important piece of advice in this chapter: run the tests. Having a beautiful, perfect test suite is no use if you rarely run it. Leaving long gaps between test runs makes it more difficult to track down what has gone wrong when a test fails because, a lot of the code will have changed. Also, if it has been weeks or months since tests have been run and they fail, it is difficult or impossible to know which results that have been obtained in the mean time are still valid, and which have to be thrown away as they could have been impacted by the bug.

- Write tests – any tests!
- Run the tests
- Consider how long it takes your tests to run
- Document the tests and how to run them
- Test realistic cases
- Use a testing framework
- Aim to have good code coverage
- Use test doubles / stubs / mocking where appropriate



General guidance and good practice for testing

There are several different kinds of testing which each have best practice specific to them (see Types of Testing). Nevertheless, there is some general guidance that applies to all of them, which will be outlined here.

Write Tests - Any Tests!

Management

Reproducible Environments

BinderHub

Code quality

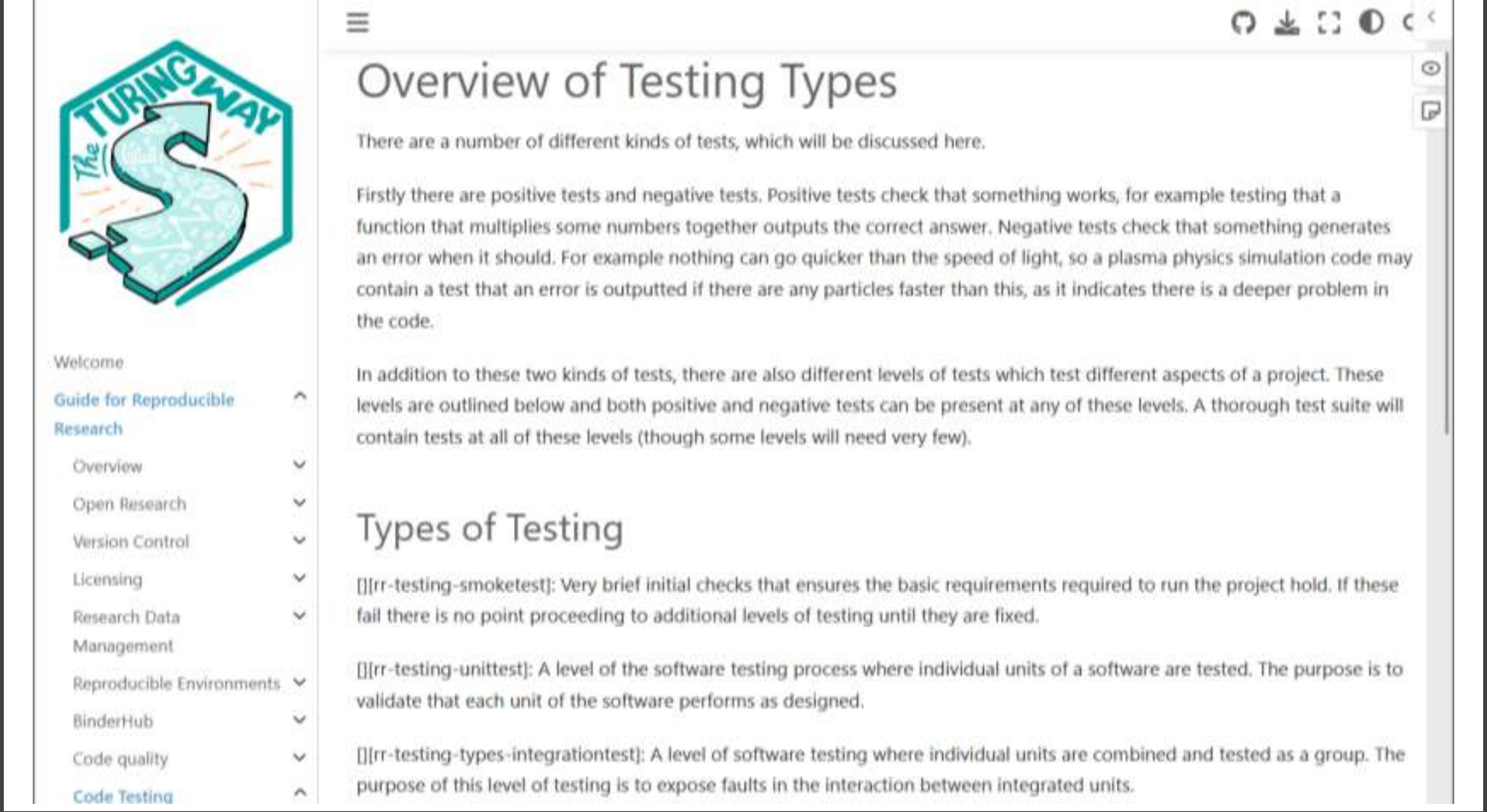
Code Testing

The second most important piece of advice in this chapter: run the tests. Having a beautiful, perfect test suite is no use if you rarely run it. Leaving long gaps between test runs makes it more difficult to track down what has gone wrong when a test fails because, a lot of the code will have changed. Also, if it has been weeks or months since tests have been run and they fail, it is difficult or impossible to know which results that have been obtained in the mean time are still valid, and which have to be thrown away as they could have been impacted by the bug.

Make improvements where you can and do your best to include tests with new code you write even if it’s not feasible to write tests for all the code that’s already written.

https://the-turing-way.netlify.app/index.html
https://github.com/the-turing-way/the-turing-way

@TuringWay
https://doi.org/10.5281/zenodo.11098175



Overview of Testing Types

There are a number of different kinds of tests, which will be discussed here.

Firstly there are positive tests and negative tests. Positive tests check that something works, for example testing that a function that multiplies some numbers together outputs the correct answer. Negative tests check that something generates an error when it should. For example nothing can go quicker than the speed of light, so a plasma physics simulation code may contain a test that an error is outputted if there are any particles faster than this, as it indicates there is a deeper problem in the code.

In addition to these two kinds of tests, there are also different levels of tests which test different aspects of a project. These levels are outlined below and both positive and negative tests can be present at any of these levels. A thorough test suite will contain tests at all of these levels (though some levels will need very few).

Types of Testing

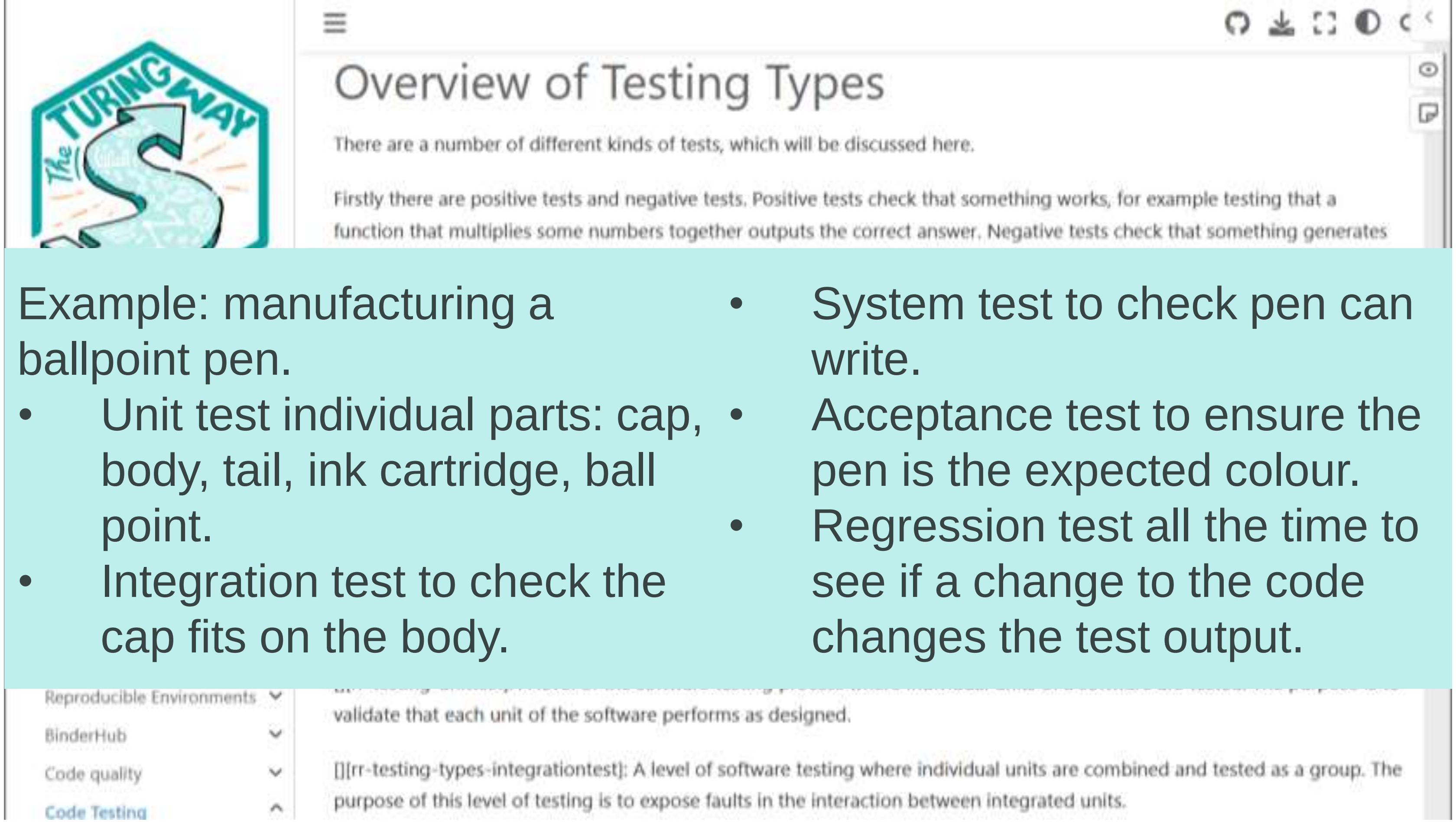
[][rr-testing-smoketest]: Very brief initial checks that ensures the basic requirements required to run the project hold. If these fail there is no point proceeding to additional levels of testing until they are fixed.

[][rr-testing-unittest]: A level of the software testing process where individual units of a software are tested. The purpose is to validate that each unit of the software performs as designed.

[][rr-testing-types-integrationtest]: A level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units.

https://the-turing-way.netlify.app/index.html
https://github.com/the-turing-way/the-turing-way

@TuringWay
https://doi.org/10.5281/zenodo.11098175



Overview of Testing Types

There are a number of different kinds of tests, which will be discussed here.

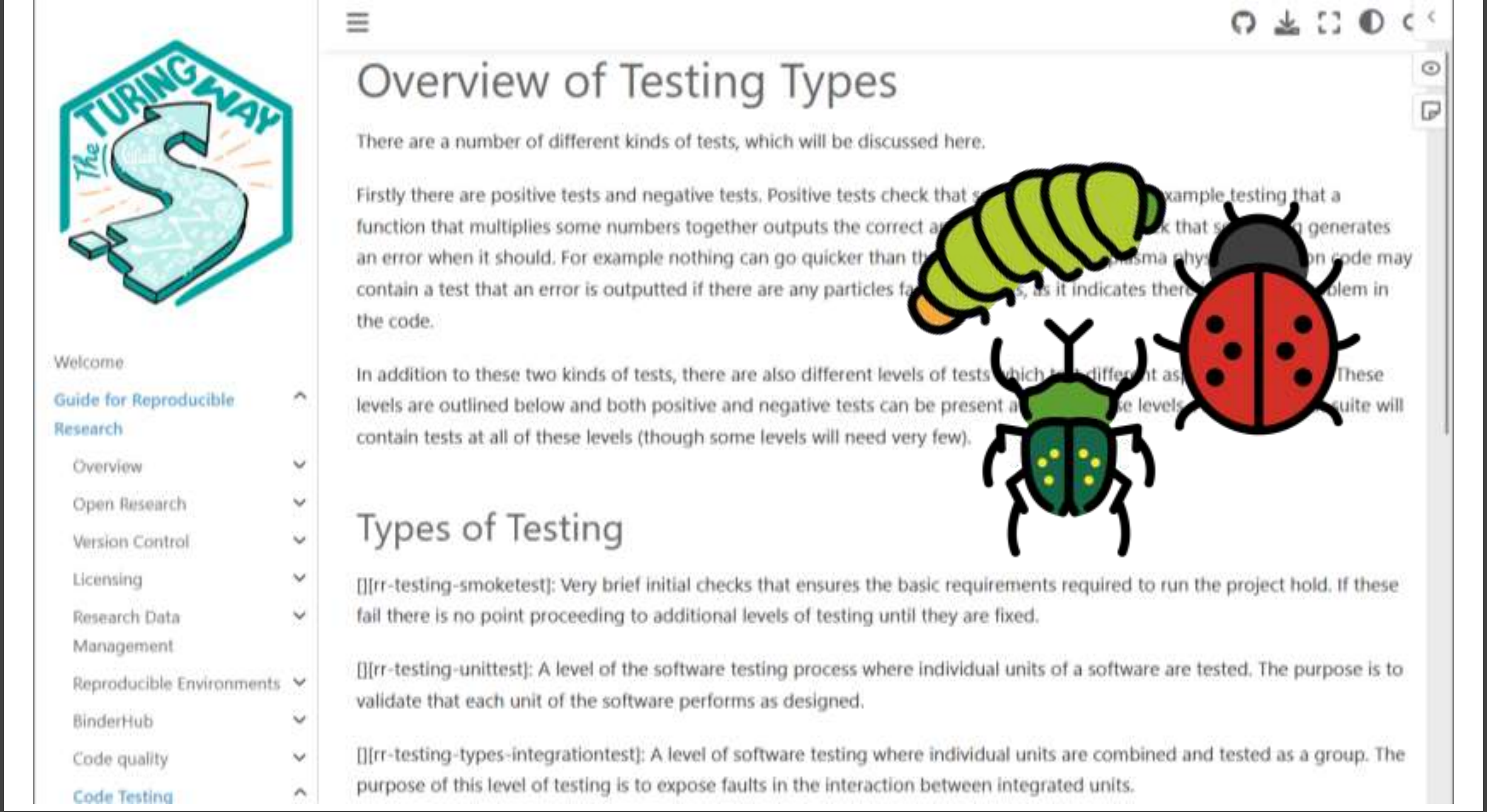
Firstly there are positive tests and negative tests. Positive tests check that something works, for example testing that a function that multiplies some numbers together outputs the correct answer. Negative tests check that something generates

validate that each unit of the software performs as designed.

[][rr-testing-types-integrationtest]: A level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units.

Example: manufacturing a ballpoint pen.

- Unit test individual parts: cap, body, tail, ink cartridge, ball point.
- Integration test to check the cap fits on the body.
- System test to check pen can write.
- Acceptance test to ensure the pen is the expected colour.
- Regression test all the time to see if a change to the code changes the test output.



Welcome

Guide for Reproducible Research

- Overview
- Open Research
- Version Control
- Licensing
- Research Data Management
- Reproducible Environments
- BinderHub
- Code quality
- Code Testing

Overview of Testing Types

There are a number of different kinds of tests, which will be discussed here.

Firstly there are positive tests and negative tests. Positive tests check that s xample testing that a function that multiplies some numbers together outputs the correct a k that s g generates an error when it should. For example nothing can go quicker than th plasma phys on code may contain a test that an error is outputted if there are any particles fa s, as it indicates there blem in the code.

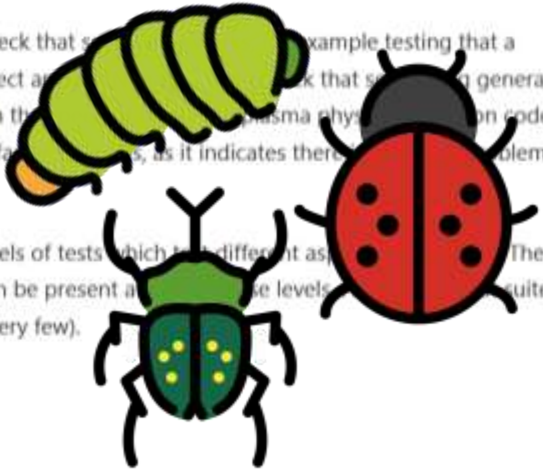
In addition to these two kinds of tests, there are also different levels of tests hich t differ nt as These levels are outlined below and both positive and negative tests can be present a e levels uite will contain tests at all of these levels (though some levels will need very few).

Types of Testing

[][rr-testing-smoketest]: Very brief initial checks that ensures the basic requirements required to run the project hold. If these fail there is no point proceeding to additional levels of testing until they are fixed.

[][rr-testing-unittest]: A level of the software testing process where individual units of a software are tested. The purpose is to validate that each unit of the software performs as designed.

[][rr-testing-types-integrationtest]: A level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units.

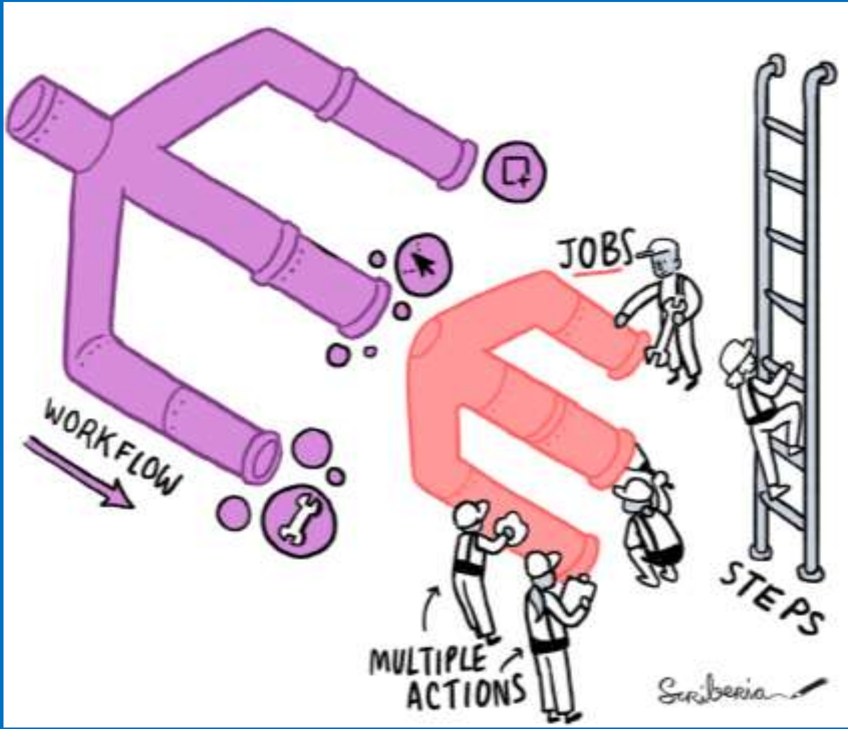


https://the-turing-way.netlify.app/index.html
https://github.com/the-turing-way/the-turing-way

@TuringWay
https://doi.org/10.5281/zenodo.11098175

Reproducible and explainable results

- Code your data processing, analysis & visualisations. Share protocols for manual steps.
- Test to catch changes (planned and unplanned).
- Keep a human in the loop to track how analyses behave under different circumstances.



https://the-turing-way.netlify.app/reproducible-research/testing
https://the-turing-way.netlify.com/reproducible-research/ci.html

@TuringWay
https://doi.org/10.5281/zenodo.11098175

Please join us in The Turing Way!

- 🔖 Bookmark the start page: https://the-turing-way.start.page
- 📚 Read the guides: https://book.the-turing-way.org
- 📝 Contribute on GitHub: https://github.com/the-turing-way/the-turing-way
- 💌 Subscribe to our newsletter: https://buttondown.email/turingway
- 👋 Chat with us on slack: https://tinyurl.com/jointuringwayslack
- ☕ Join a collaboration cafe: https://annuel2.framapad.org/p/ttw-collaboration-cafe
  - Next one: 15 May 2024, 15:00-17:00 BST / 14:00-16:00 UTC
- 🙋 Join an onboarding call: https://annuel2.framapad.org/p/ttw-onboarding
- 🎨 Use The Turing Way Scriberia images (CC-BY): https://doi.org/10.5281/zenodo.3332807

This work was supported by The UKRI Strategic Priorities Fund under the EPSRC Grant EP/T001569/1, particularly the "Tools, Practices and Systems" theme within that grant, and by The Alan Turing Institute under the EPSRC grant EP/N510129/1.

@TuringWay

https://the-turing-way.netlify.com

https://doi.org/10.5281/zenodo.11098175