

Reproducibility with hpc

Becky Arnold & Anna Krystalli

Attribution 4.0 International (CC BY 4.0)



Our bit of the project

Open source textbook (Becky)

- What is reproducibility?
- Version control
- Open research
- Testing
- Reproducible computational environments
- Continuous integration

Checklists (Becky and Anna)

The Turing Way

1. Reproducibility
2. Open Research
3. Version Control
4. Reproducible Environments
5. Testing
6. Continuous Integration
7. Research Data Management

Powered by Jupyter Book

← TOGGLE SIDEBAR

Reproducible environments

Prerequisites / recommended skill level

Prerequisite	Importance	Notes
Experience with the command line	Necessary	Experience with downloading software via the command line is particularly useful
Version control	Helpful	Experience using git and GitHub are helpful for the section on Binder

A tutorial on working via the command line can be found [here](#).

Recommended skill level: intermediate-advanced.

Table of contents

- [Summary](#)
 - [What is a computational environment?](#)
- [How this will help you/why this is useful](#)
- [Summary of ways to capture computational environments](#)
 - [Package management systems outline](#)
 - [Binder outline](#)
 - [Virtual machines outline](#)
 - [Containers outline](#)

ON THIS PAGE

PREREQUISITES / RECOMMENDED SKILL LEVEL

TABLE OF CONTENTS

SUMMARY

WHAT IS A COMPUTATIONAL ENVIRONMENT?

HOW THIS WILL HELP YOU/ WHY THIS IS USEFUL

SUMMARY OF WAYS TO CAPTURE COMPUTATIONAL ENVIRONMENTS

PACKAGE MANAGEMENT SYSTEMS

BINDER

VIRTUAL MACHINES

CONTAINERS

PACKAGE MANAGEMENT SYSTEMS

WHAT DOES CONDA DO?

INSTALLING CONDA

MAKING AND USING ENVIRONMENTS

DEACTIVATING AND

DELETING ENVIRONMENTS

INSTALLING AND REMOVING

PACKAGES WITHIN AN

ENVIRONMENT

EXPORTING AND

REPRODUCING

COMPUTATIONAL

ENVIRONMENTS

Make use of Git

- Make your project version controlled by initialising a Git repository in its directory using `git init`
- Add and commit all your files to the repository using `git add .` then `git commit`
- Continue to add and commit changes as your project progresses. Stage the changes in specific files to be committed with `git add filename`, and add messages to your commits
 - Each commit should make one simple change
 - No generated files committed
 - Commit messages are meaningful, with a ~50 character summary at the top
 - Commit messages are in the present tense and imperative
- Develop new features on their own branches, which you can create via `git checkout -b branch_name` and switch between via `git checkout branch_name`
 - Branches have informative names
 - Master branch is kept clean
 - Each branch has a single purpose and only changes related to that purpose are made on it
- Once features are complete merge their branches into the master branch by switching to the feature branch and running `git merge master`
 - Merge other's changes into your work frequently
 - When dealing with merge conflicts make sure you fully understand both versions before trying to resolve them

Our bit of the project

Open source textbook (Becky)

- What is reproducibility?
- **Version control**
- Open research
- **Testing**
- **Reproducible computational environments**
- Continuous integration

Checklists (Becky and Anna)

The Turing Way

1. Reproducibility
2. Open Research
3. Version Control
4. Reproducible Environments
5. Testing
6. Continuous Integration
7. Research Data Management

Powered by Jupyter Book

← TOGGLE SIDEBAR

Reproducible environments

Prerequisites / recommended skill level

Prerequisite	Importance	Notes
Experience with the command line	Necessary	Experience with downloading software via the command line is particularly useful
Version control	Helpful	Experience using git and GitHub are helpful for the section on Binder

A tutorial on working via the command line can be found [here](#).

Recommended skill level: intermediate-advanced.

Table of contents

- Summary
 - What is a computational environment?
- How this will help you/why this is useful
- Summary of ways to capture computational environments
 - Package management systems outline
 - Binder outline
 - Virtual machines outline
 - Containers outline

ON THIS PAGE

PREREQUISITES / RECOMMENDED SKILL LEVEL

TABLE OF CONTENTS

SUMMARY

WHAT IS A COMPUTATIONAL ENVIRONMENT?

HOW THIS WILL HELP YOU/ WHY THIS IS USEFUL

SUMMARY OF WAYS TO CAPTURE COMPUTATIONAL ENVIRONMENTS

PACKAGE MANAGEMENT SYSTEMS

BINDER

VIRTUAL MACHINES

CONTAINERS

PACKAGE MANAGEMENT SYSTEMS

WHAT DOES CONDA DO?

INSTALLING CONDA

MAKING AND USING ENVIRONMENTS

DEACTIVATING AND DELETING ENVIRONMENTS

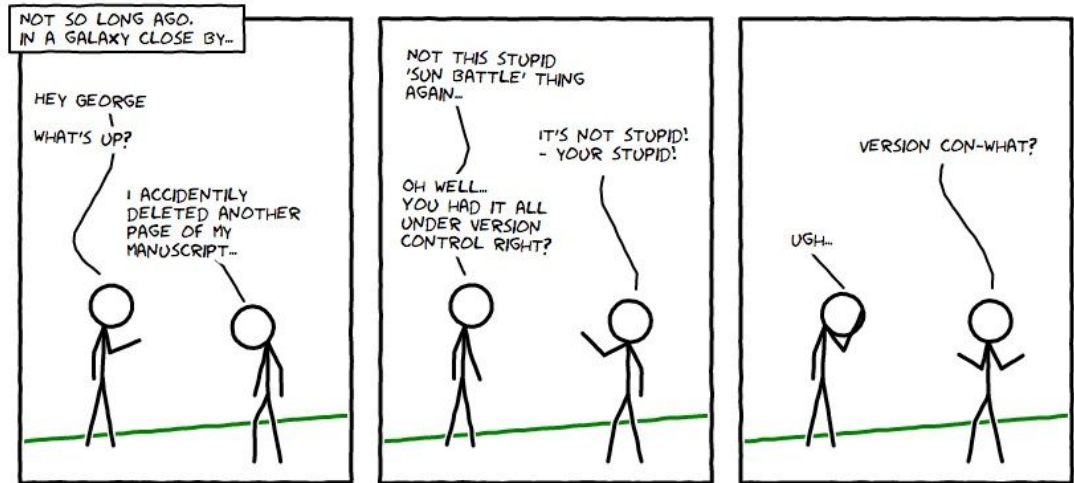
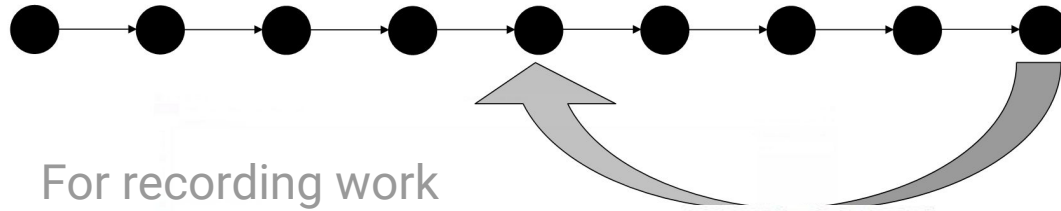
INSTALLING AND REMOVING PACKAGES WITHIN AN ENVIRONMENT

EXPORTING AND REPRODUCING COMPUTATIONAL ENVIRONMENTS

Make use of Git

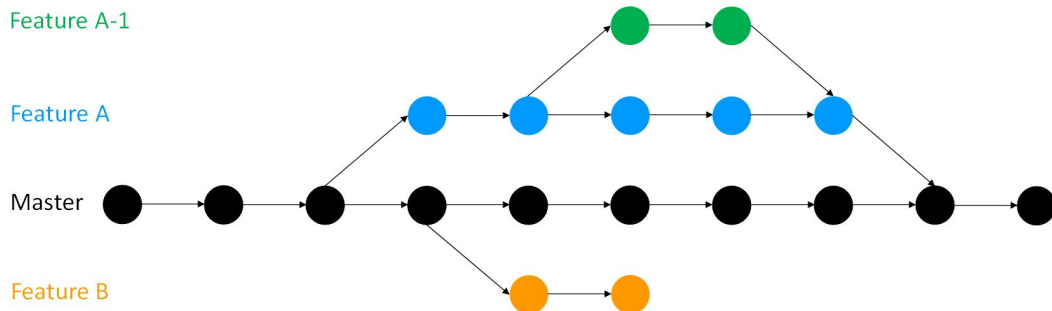
- Make your project version controlled by initialising a Git repository in its directory using `git init`
- Add and commit all your files to the repository using `git add .` then `git commit`
- Continue to add and commit changes as your project progresses. Stage the changes in specific files to be committed with `git add filename`, and add messages to your commits
 - Each commit should make one simple change
 - No generated files committed
 - Commit messages are meaningful, with a ~50 character summary at the top
 - Commit messages are in the present tense and imperative
- Develop new features on their own branches, which you can create via `git checkout -b branch_name` and switch between via `git checkout branch_name`
 - Branches have informative names
 - Master branch is kept clean
 - Each branch has a single purpose and only changes related to that purpose are made on it
- Once features are complete merge their branches into the master branch by switching to the feature branch and running `git merge master`
 - Merge other's changes into your work frequently
 - When dealing with merge conflicts make sure you fully understand both versions before trying to resolve them

Version control for Research I



Version control for Research II

For managing collaboration



"FINAL".doc



FINAL.doc!



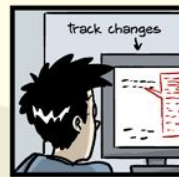
FINAL_rev.2.doc



FINAL_rev.6.COMMENTS.doc



FINAL_rev.8.comments5.
CORRECTIONS.doc



FINAL_rev.18.comments7.
corrections9.MORE.30.doc



FINAL_rev.22.comments49.
corrections.10.##\$%WHYDID
ICOMETOGRADSCHOOL?????.doc

JORGE CHAN © 2012

Git & GitHub are a great way to sync with HPC

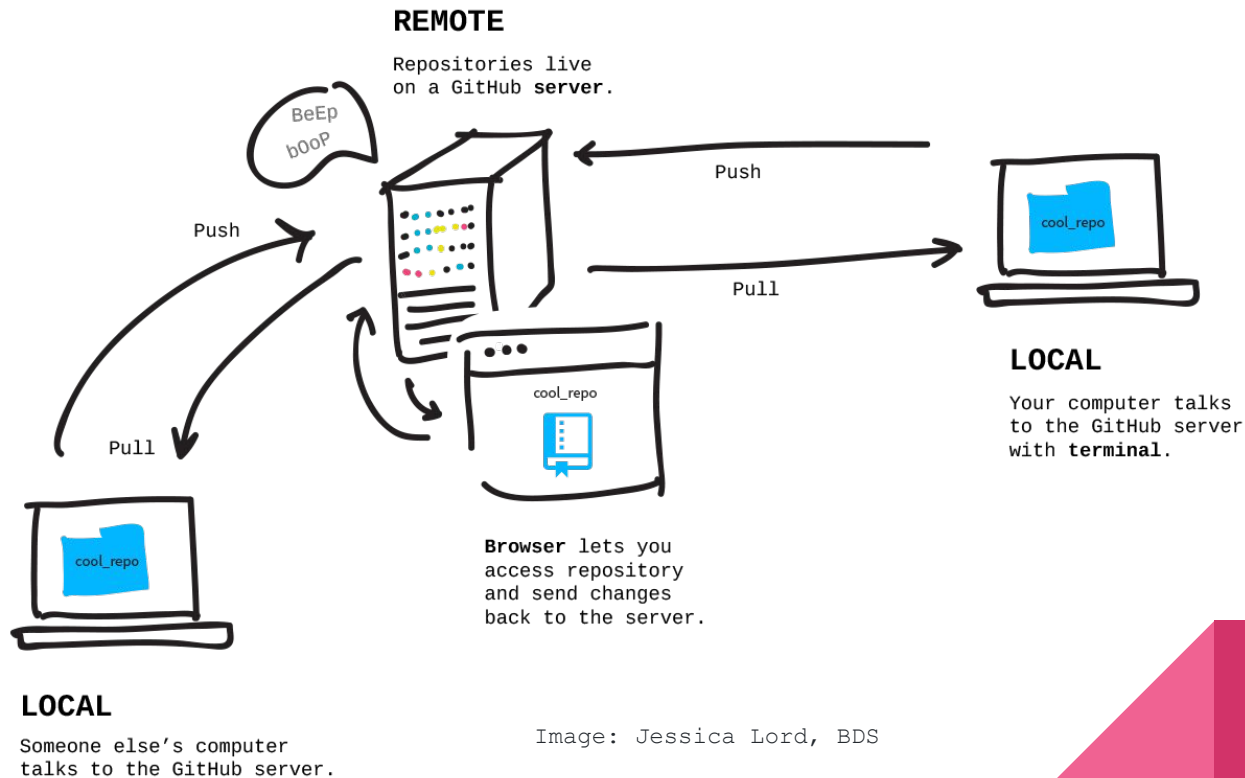


Image: Jessica Lord, BDS

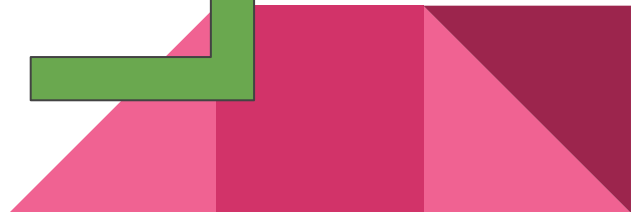
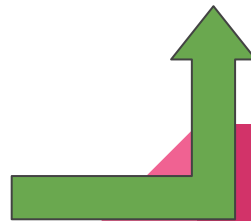
Version control for tracking provenance

Lots of files to keep track of:

- Input files
- Code
- Make files
- Config files

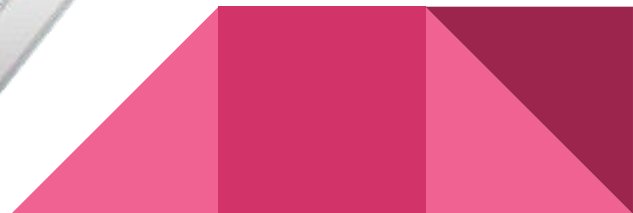
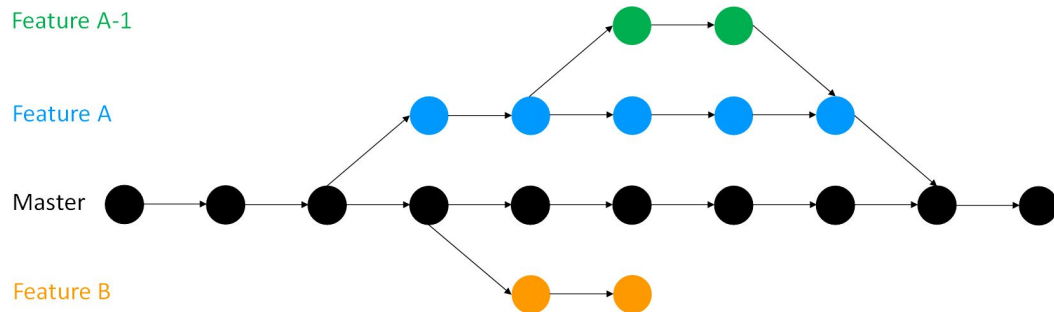
Changes to these changes the results

- Overwritten files
- Can't remember what they are = wasted hpc time



Easier to provide support

Easier viewing and access to code



Reproducible computational environments

Write on one computer, run on another

Transfer problems

- Installation issues (no admin privileges)
- Values/paths “baked in”

May not be able to run. If you can run it results may differ

E.g. different compilers/package versions

Defining environment important for reproducibility

No



Reproduc^{ing} computational environments

Package management systems

- Conda
- Create and delete environments
- No admin privileges needed
- Export/import environments



Containers

- Package a computer
- Singularity (no admin)



Why testing matters

Reproducibility: “How do we know this code works?”

Everyone makes mistakes.

Can major consequences for projects and careers

NEWS OF THE WEEK | SCIENTIFIC PUBLISHING

A Scientist's Nightmare: Software Problem Leads to Five Retractions

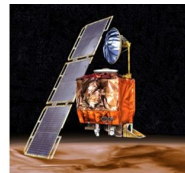
Greg Miller

✦ See all authors and affiliations

Science 22 Dec 2006:
Vol. 314, Issue 5807, pp. 1856-1857
DOI: 10.1126/science.314.5807.1856

LISA ERDSMAN 11.10.10 7:00 AM

NOV. 10, 1999: METRIC MATH MISTAKE MUFFED MARS METEOROLOGY MISSION



The **\$125 million satellite** was supposed to be the first weather observer on another world. But as it approached the red planet to slip into a stable orbit Sept. 23, the [orbiter vanished](#). Scientists realized quickly it was gone for good. “It was pretty clear that morning, within half-an-hour, that the spacecraft had more or less **hit the top of the atmosphere and burned up**,” recalled NASA engineer Richard Cook, who was project manager for Mars exploration projects at the time.

Reproducibility: “How do we know this code works?”

Mistakes on HPC

Everyone makes mistakes.



Mike Croucher

@walkingrandomly

Follow



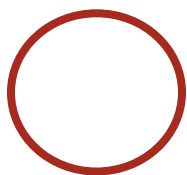
I burned 10 years of CPU time on a condor cluster computing the exact same Monte Carlo simulation because I didn't handle random seeds correctly.

Replying to [@walkingrandomly](#)

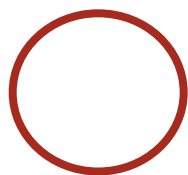
I spent two weeks of supercomputer time, across 200 cores, and forgot to save my results to disk.

On hpc mistakes can be costly in both time & resources

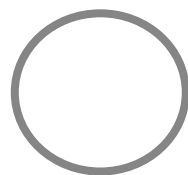
Testing & HPC



Testing suite



Better local code



Better code on the cluster!

Running tests on the cluster:

- + Can flag problems early on
- + Can aid trouble-shooting

Testing in the Turing way

General good practice

Tips e.g. for testing stochastic code

Types of tests

- Uses
- How to implement them
- Type-specific good practice
- Type-specific tips



Come talk to us!

Booth over lunch- come chat!

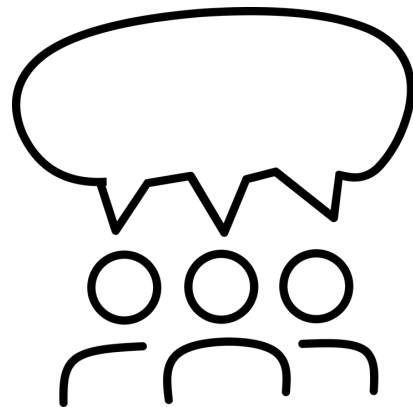
Groundwork for chapter on reproducibility with HPC

HackMD: <https://bit.ly/2CGhH1K>

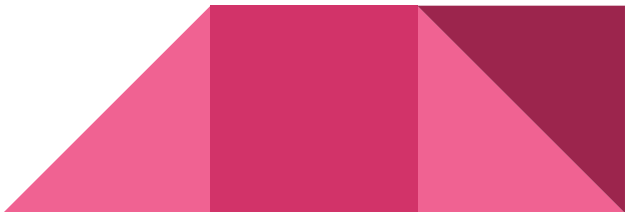
Want to hear experiences/horror stories/advice: <https://bit.ly/2CL9CsD>

Project repository: <https://github.com/alan-turing-institute/the-turing-way>

Book: <https://the-turing-way.netlify.com>



Thank you for listening



#ColabW19 Project: a library of markdown checklists that can be incorporated as GitHub issue templates into your own projects.



Set up GitHub Authentication

Set up GitHub authentication through a Personal Access Token

Get started

Set up GitHub Authentication in R

Set up GitHub authentication in R through a Personal Access Token

Get started

<https://github.com/annakrystalli/IUCNextractR/issues/1>

Set up GitHub Authentication in R #1

Edit New issue

Open annakrystalli opened this issue 4 days ago · 0 comments



annakry... commented 4 days ago • edited -

Owner + @ ...

Set up GitHub Access in R/Rstudio

Set up GitHub authentication through a Personal Access Token in R

- Create New personal access token (PAT)
- Store PAT as Environment Variable

Guidance

Create Personal Access Token

Personal access tokens function like ordinary [OAuth access tokens](#). They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

You can set PAT authentication through R using package [usethis](#) and is easier through RStudio. To install the package run:

```
install.packages("usethis")
```

To create a new PAT use:

```
browse_github_pat()
```

This launches a browser and navigates to the GitHub url for creating new Personal Access Token. The url prepopulates the PAT specification with:

- A description for the token: `R:GITHUB_PAT`
- The scope of the authorisation. This defines the access for personal token and defaults to full access to repos and gists only. You can amend this if required. [Read more about OAuth scopes](#)

Click Generate token.

Then copy the token to your clipboard. For security

Assignees

No one—
assign yourself

Labels

None yet

Projects

None yet

Milestone

No milestone

Notifications

Unsubscribe

You're receiving notifications because you authored the thread.

1 participant



Lock conversation

Pin issue

Transfer issue
Beta

Delete issue