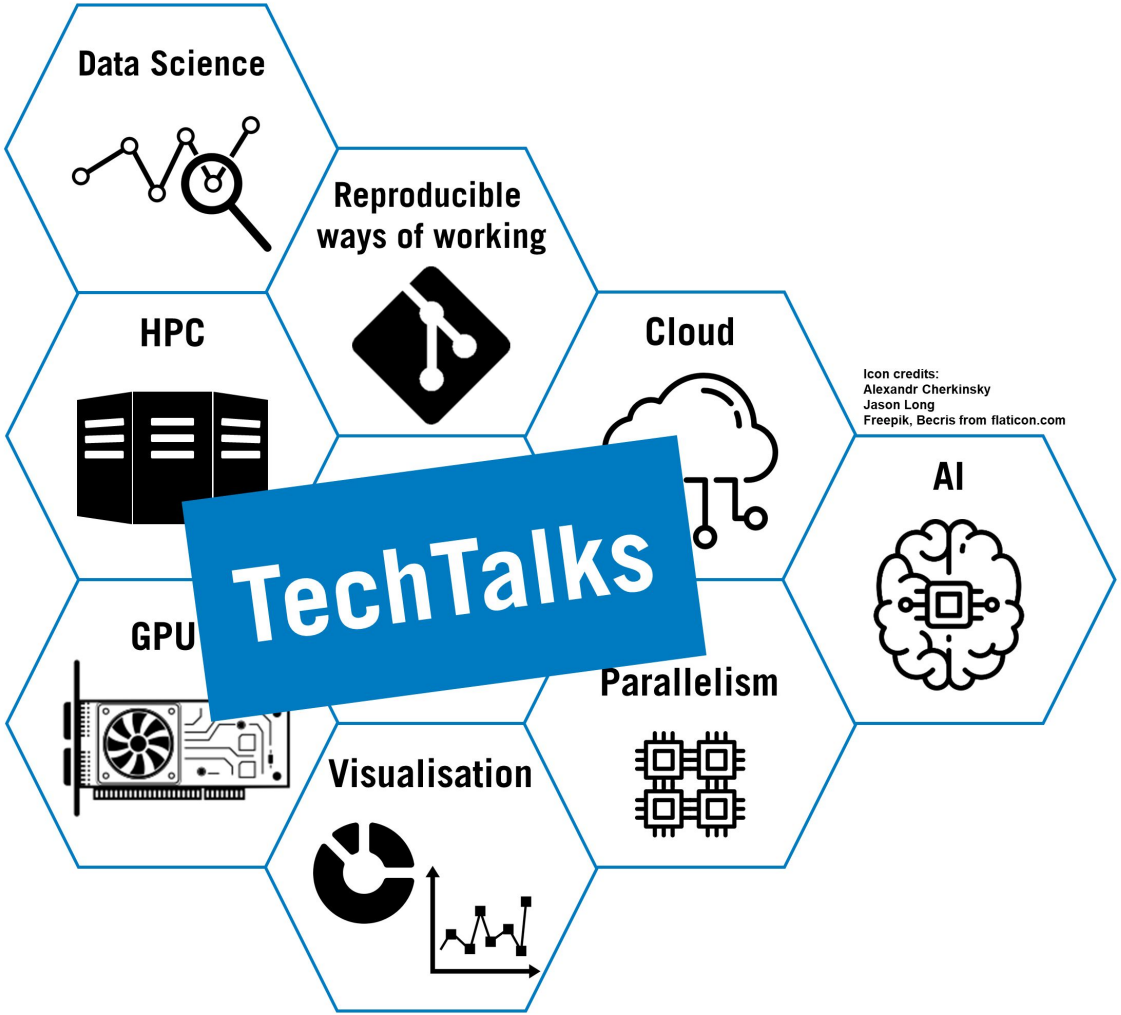


http://bit.ly/tt-repres

TechTalk:
Tools and Techniques for Reproducible Research

Martin Callaghan
Research Computing

http://bit.ly/tt-repres

What are these talks?

We’re hoping a monthly series of talks:

- On interesting technical topics relevant to a wide range of researchers
- Some news about what we’re up to in Research Computing…
  - Hint: It’s not just HPC
- Given by us, by some of you and you’ll also get to hear from some of our RSE colleagues from across the UK.

http://bit.ly/tt-repres

This first series of talks

- Some of the tools and techniques we use (and perhaps you do too) that help make the computational aspects of the research process more straightforward and assist in **reproducibility**.

- Some upcoming events on the same theme:
  - Reprohack (February 14th 2020)
  - RSS Meeting (April 21st 2020)

http://bit.ly/tt-repres

ROYAL STATISTICAL SOCIETY

DATA | EVIDENCE | DECISIONS

@RSSLeedsBrad

Up-coming MEETING

Tuesday 21st April 2020

Anna Krystalli + Michael Croucher

Research reproducibility and good practice in statistical software

Room MALL 2, School of Mathematics

Refreshments 14:30-15:00, Presentation 15:00-16:30

Website: https://sites.google.com/site/rssleedsbradford/home/2019---2020-session

http://bit.ly/tt-repres

Who’s the ‘we’: Research Computing...

Mark Conmy
Martin Callaghan
Alex Coleman
Nick Rhodes
Ollie Clark
John Hodrien
Sam Crossfield
Phil Chambers
Adam Keeley

http://bit.ly/tt-repres

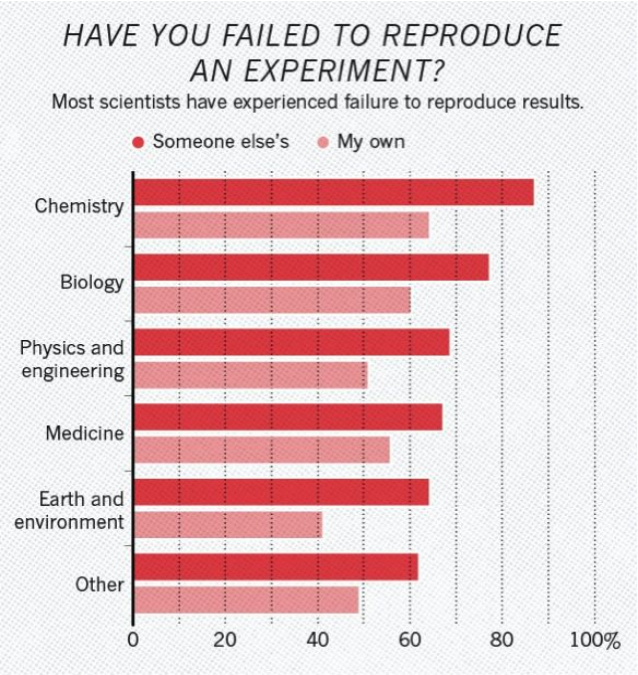
On with the show...

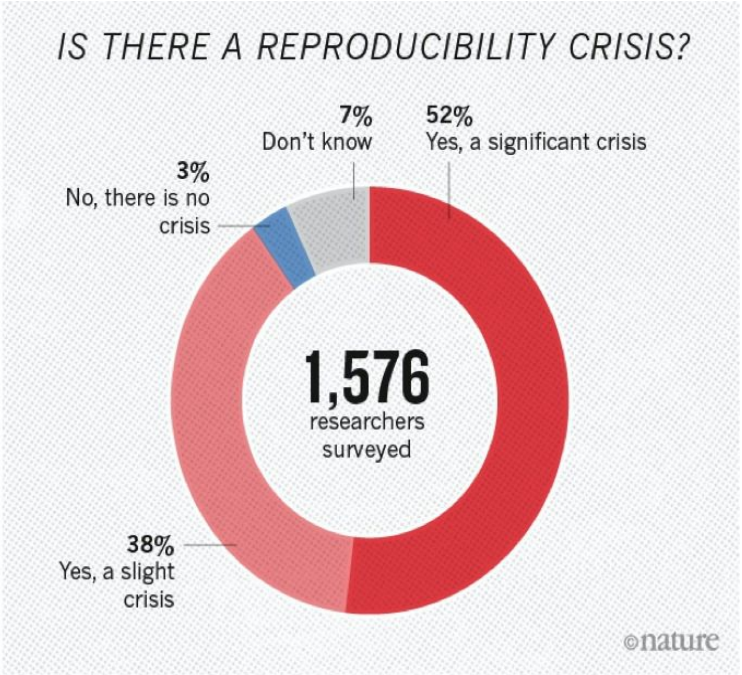
http://bit.ly/tt-repres

Our motivation...

http://bit.ly/tt-repres

Why all the fuss?





Number of respondents from each discipline:
Biology **703**, Chemistry **106**, Earth and environmental **95**, Medicine **203**, Physics and engineering **236**, Other **233**

©nature

http://bit.ly/tt-repres

Today we’re going to concentrate on code:

- Data is the topic for another TechTalk
- We’re going to give an overview of

- **Conda** to manage dependencies
- **Containers** to record environments
- **Workflow Tools** (SnakeMake) to record computational steps
- A mention of **Virtual Machines** to record environments

Dangerously, there will be live demonstrations.

http://bit.ly/tt-repres

Starting points

Good project practice

Version Control (!)

http://bit.ly/tt-repres

Project management

- A good starting point is to keep all files associated with a project in a single folder
- Different projects should have separate folders
- Use consistent and informative directory structure
- If you need to separate public/private/secret, separate these by folder (and Git repo)
- Add a README file to describe the project and instructions on reproducing the results
- Talk to others in the project about what you do and write it down
- When software is reused in several projects it can make sense to put them in own repo

http://bit.ly/tt-repres

Directory organisation

```
project_name/
├── README.md              # overview of the project
├── data/                  # data files used in the project
│   ├── README.md          # describes where data came from
│   └── sub-folder/        # may contain subdirectories
├── processed_data/        # intermediate files from the analysis
├── manuscript/            # manuscript describing the results
├── results/               # results of the analysis (data, tables, figures)
├── src/                   # contains all code in the project
│   ├── LICENSE            # license for your code
│   ├── requirements.txt   # software requirements and dependencies
│   └── ...
└── doc/                   # documentation for your project
    ├── index.rst
    └── ...
```

epres

Conda

Our codes often depend on other codes that in turn depend on other codes …

- **Reproducibility**: We can control our code but how can we control dependencies (packages, libraries)?
- **10-year challenge**: Try to build/run your own code that you have created 10 (or fewer) years ago. Will your code from today work in 5 years if you don't change it?
- **Dependency hell**: Different codes on the same environment can have conflicting dependencies.

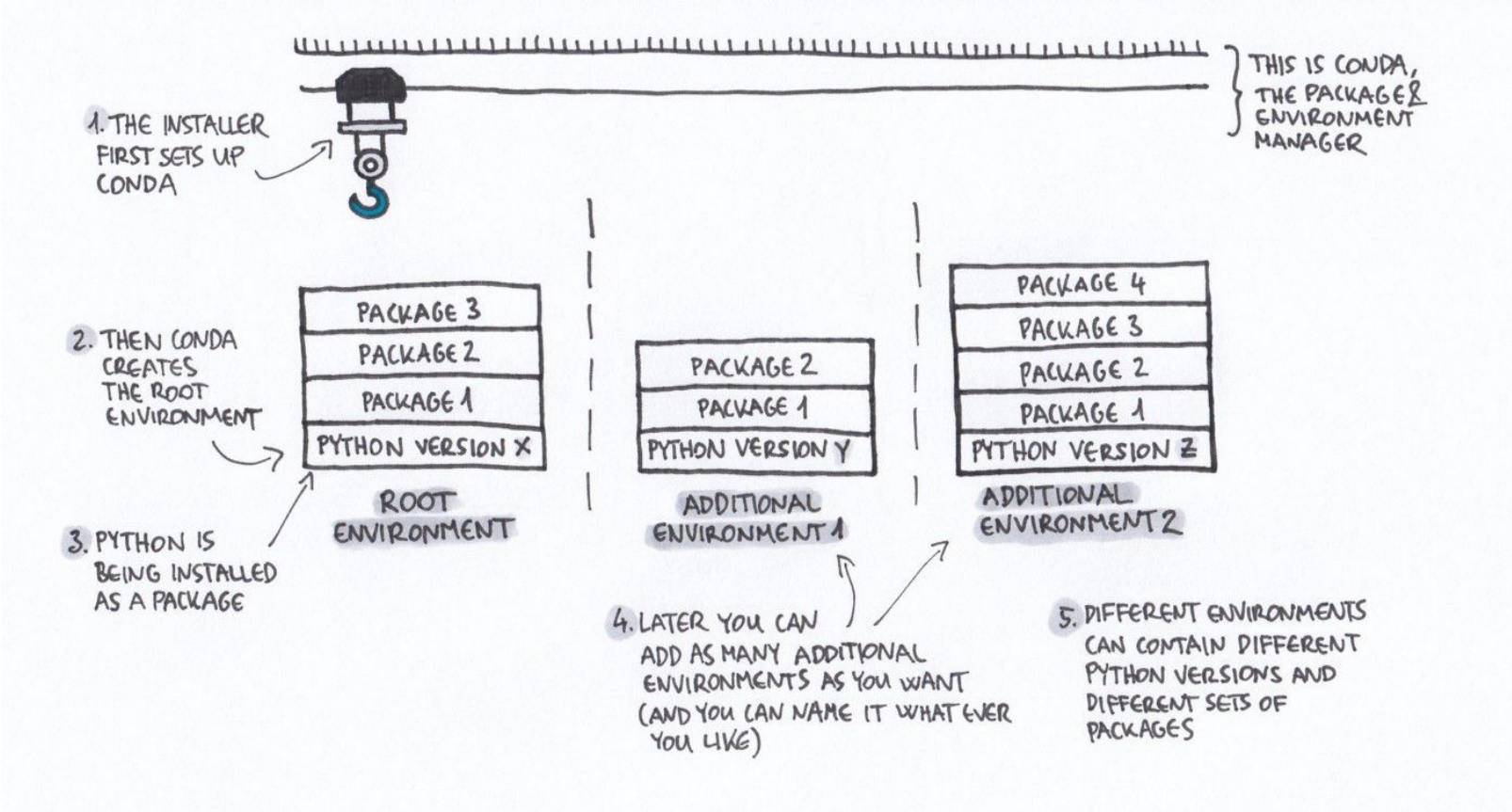
http://bit.ly/tt-repres

Conda

Tools like Conda try to solve the following problems:

- Install a **specific set of dependencies**, usually with well defined versions
- **Recording those versions** for all dependencies
- **Isolate environments** on your computer for projects that have conflicting dependencies
- Isolate environments on computers with many users (eg. HPC)
- Using **different Python/R versions** per project
- Provide tools and services to **share packages**

Conda



http://bit.ly/tt-repres

Conda

You might be using it already, remember:

- Not only for Python: any language, also binaries.
- Created by Continuum Analytics, part of Anaconda/Miniconda, but can be installed standalone.
- Open source BSD license.
- Manages isolated software environments.
- Allows you to create and share conda packages.
- Miniconda is a lightweight alternative to Anaconda.
- Can be installed on your computer (Windows, Mac, Linux) without Admin rights

Let’s have a go...

http://bit.ly/tt-repres

Summary

Capturing software dependencies is a must for reproducibility.

Files like requirements.txt, environment.yml, should be part of the source repository.

Be sceptical (or even annoyed) when you see dependency lists without versions.

http://bit.ly/tt-repres

Containers

You might have heard of tools like

- Docker
- Singularity

Containers can be built to bundle **all** the necessary ingredients (data, code, environment).

A container provides operating-system-level virtualisation, sharing the host system's kernel with other containers. It's an easy way to run (eg) Linux applications on a Mac/ Windows, Ubuntu on Centos, etc.

Docker

- Available for most common operating systems.
- A mechanism to “send the computer to the data” when data is too large or too sensitive to travel over network.
- DockerHub is a platform to share Docker images (stored in repositories - similar to a Git repository).
- A wide range of public Docker images available on DockerHub.

http://bit.ly/tt-repres

Singularity

- Singularity is aimed at the scientific community and to run scientific workflows on (mainly) HPC resources.
- Docker images can be converted into Singularity images.
- There’s a Singularity Hub to share images and recipes

http://bit.ly/tt-repres

Container vs. image vs. recipe (Dockerfile)

Image is like a blueprint. It is immutable.

Container is an instance of an image.

Dockerfile is a recipe which creates a *container* based on an *image* and potentially applies small changes to it.

http://bit.ly/tt-repres

Warning!

Check the provenance of images on Dockerhub and Singularity Hub!

- Do you know where the image or recipe comes from?
- Do you trust the developer?
- Can you read the recipe and see what was built inside the image?

Pros...

- Allow for seamlessly moving workflows across different platforms.
- Much more lightweight than virtual machines.
- Eliminates the “works on my machine” situation.
- For software with many dependencies with in turn it’s own dependencies possibly the only (?) way to preserve the computational experiment for future reproducibility.

http://bit.ly/tt-repres

Cons …

- Containers can have security vulnerabilities which can be exploited.
- Can be used to hide away software installation problems and thereby discourage good software development practices.
- It may not be clear whether to record the environment in the image part or the recipe part.

Scientific Workflow management systems

To reproduce a number of steps in a computational workflow, we could:

- Remember what buttons we clicked in a GUI
- Type in a set of commands over and over to reproduce the steps
- Write a (BASH) script- a list of our commands
- Use GNU Make (which is great, but better for building software…)
- Use a specialist tool like Snakemake

http://bit.ly/tt-repres

Snakemake

- Gentle learning curve.
- Free, open-source, and installs easily via conda.
- Cross-platform (Windows, MacOS, Linux) and compatible with HPC
- Same workflow works without modification and scales whether on a laptop or cluster.
- Heavily used in bioinformatics, but is completely general.

http://bit.ly/tt-repres

How does it work?

Tell Snakemake what files you want to be created

```
rule:
  input: "A.txt", "B.txt", "C.txt"
```

Produce the files you want to have from some intermediate result

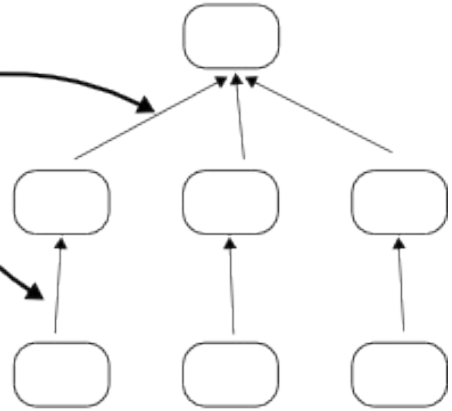
```
rule:
  input: "{sample}.inter"
  output: "{sample}.txt"
  shell: "somecommand {input} {output}"
```

Create a needed intermediate result

```
rule:
  input:  "{sample}.in"
  output: "{sample}.inter"
  run:
    somepythoncode()
```

Use wildcards to write general rules for all samples

Snakemake determines the dependencies for you



http://bit.ly/tt-repres

Let’s have a go…

http://bit.ly/tt-repres

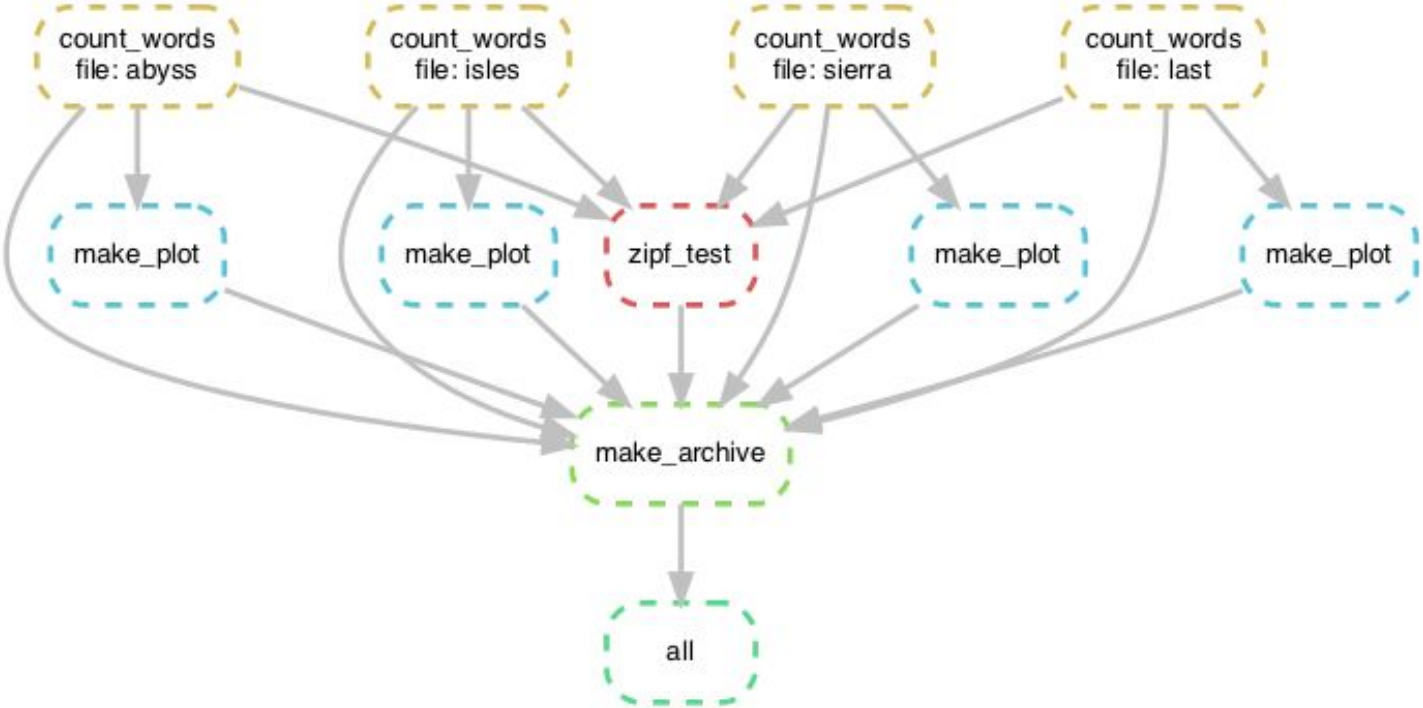
Visualising the workflow

Snakemake can conveniently create a DAG to help visualise the workflow.

snakemake --dag | dot -Tpng > dag.png

Rules yet to be completed have solid outlines, completed rules have dotted outlines.

A Snakemake DAG



http://bit.ly/tt-repres

Summary

- GUIs may or may not be reproducible.
- Some GUIs can be automated, many cannot.
- Typing the same series of commands for 100 similar inputs is tedious and error prone.
- Imperative scripts are reproducible and great for automation.
- Declarative workflows such as Snakemake are great for longer multi-step analyses.
- Declarative workflows are often easy to parallelise without you changing anything.
- With declarative workflows it is straightforward to add/change things late on in the project.
- Interesting modern alternative to Make/Snakemake: https://taskfile.dev

http://bit.ly/tt-repres

Any questions?

http://bit.ly/tt-repres

More information:

The Turing Way:
https://www.turing.ac.uk/research/research-projects/turing-way-handbook-reproducible-data-science

Coderefinery:
https://coderefinery.org

http://bit.ly/tt-repres

Feedback

Please spare a few minutes now or later to give us a bit of feedback:

http://bit.ly/ttfbjan20