Reproducible bioinformatics

from a user's perspective

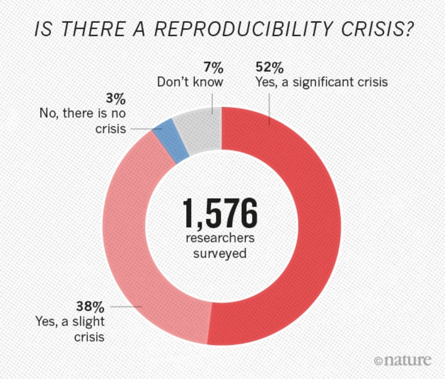
Tom Harrop

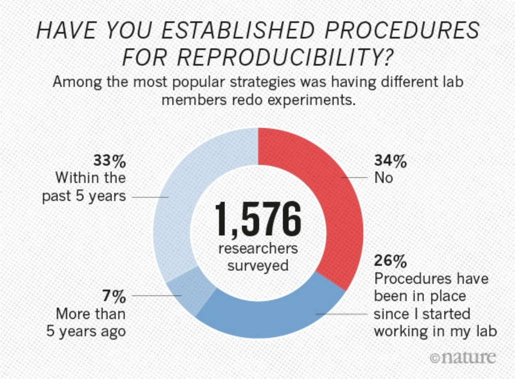
The University of Otago

tom.harrop@otago.ac.nz

@tharrop_

2020-02-12





Jenny Bryan https://youtu.be/7oyiPBjLAWY & Nature News, 10.1038/533452a

What is reproducibility?

Reproduce: under identical conditions to the previous result, repeat the analysis and get the **exact** same result

In bioinformatics:

- **same data**
- **same methodology** (code)
- **same result**

Guidelines for reproducible analysis:

1. Don’t modify raw data
2. Record the code
3. Capture the computing environment

Adapted from Adam Labadorf’s slides

Interactive analysis may be hard to reproduce

Examples:

- install software locally
- use software installed by the admin
- paste commands from a text file into the console
- save a set of scripts to run in order

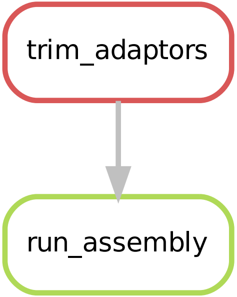
Possible issues:

- will it run again?
- are all the steps documented?
- is the recorded code exactly what was run?
- are the steps in the right order?

Workflow managers force you to record every step

Define steps in `my_workflow.txt`

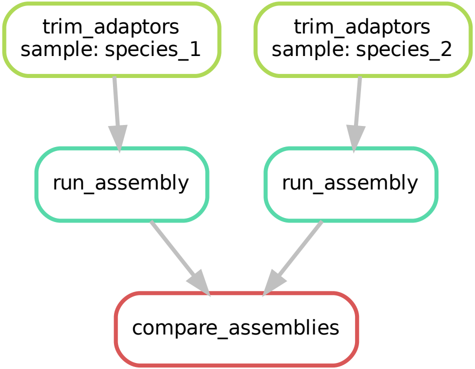
```
step trim_adaptors:
    input:  'data/raw_reads/{sample}.fastq'
    output: 'output/trimmed/{sample}.fastq'
    shell:  'trim_adaptors --raw_reads={input} > {output}'

step run_assembly:
    input:  'output/trimmed/{sample}.fastq'
    output: 'output/assemblies/{sample}.fasta'
    shell:  'choice_assembler --reads={input} > {output}'
```



Run:

```
workflow_manager my_workflow.txt run_assembly
```

Reproducibility and convenience



- The code *is* the documentation
- Scale the same code to different data
- Version control → versioned results

Lots of good options:

```
snakemake : python3
 nextflow : java
      CWL : ‘vendor-neutral specification’
    drake : R
     make : DIY
```

Reproducible computing environment

Software has

- a **version**,
- other software **dependencies** (with versions)
- all with **system dependencies**

e.g. `DESeq2`

`DESeq2_1.26.0`

`Bioconductor 3.10.1`

`libblas3 3.8.0`, `libc6 2.30`, *etc.*

Adapted from Adam Labadorf's slides

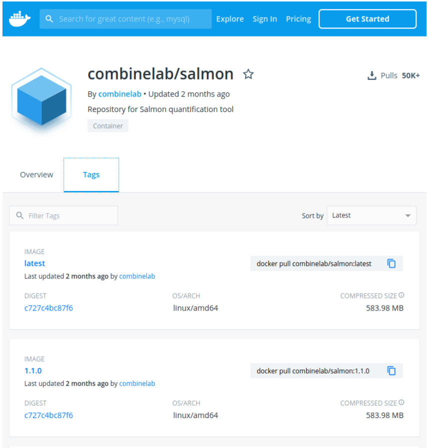
Software containers

- Isolated, complete environment (a mini OS)
- Contain specific version of software with dependencies
- Mobility of compute
- Reproducibility
- `Singularity` can run on traditional HPC

Getting software in containers

- Some developers provide docker containers

```
singularity pull \
    --name salmon_1.1.0.sif \
    docker://combinelab/salmon:1.1.0
```



combinelab/salmon ☆

Pulls 50K+

By combinelab • Updated 2 months ago

Repository for Salmon quantification tool

Container

IMAGE
latest
Last updated **2 months ago** by combinelab

```
docker pull combinelab/salmon:latest
```

DIGEST	OS/ARCH	COMPRESSED SIZE
c727c4bc87f6	linux/amd64	583.98 MB

IMAGE
1.1.0
Last updated **2 months ago** by combinelab

```
docker pull combinelab/salmon:1.1.0
```

DIGEST	OS/ARCH	COMPRESSED SIZE
c727c4bc87f6	linux/amd64	583.98 MB

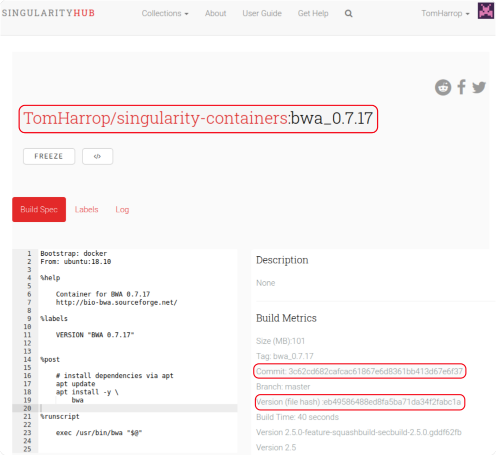
hub.docker.com

Getting software into containers

- Often have to build our own containers

Singularity.bwa_0.7.17

```
Bootstrap: docker
From: ubuntu:18.10

%labels
    VERSION "BWA 0.7.17"
%post
    apt-get update
    apt-get install -y bwa
%runscript
    exec /usr/bin/bwa "$@"
```



TomHarrop/singularity-containers:bwa_0.7.17

```
Bootstrap: docker
From: ubuntu:18.10

%help

    Container for BWA 0.7.17
    http://bio-bwa.sourceforge.net/

%labels

    VERSION "BWA 0.7.17"


%post

    # install dependencies via apt
    apt update
    apt install -y \
        bwa

%runscript

    exec /usr/bin/bwa "$@"
```

Description

None

Build Metrics

Size (MB):101

Tag: bwa_0.7.17

Commit: 3c62cd682cafcac61867e6d8361bb413d67e6f37

Branch: master

Version (file hash) :eb49586488ed8fa5ba71da34f2fabc1a

Build Time: 40 seconds

Version 2.5.0-feature-squashbuild-secbuild-2.5.0.gddf62fb

Version 2.5

singularity-hub.org

Workflow managers support containers

```
step trim_adaptors:
    input:          'data/raw_reads/{sample}.fastq'
    output:         'output/trimmed/{sample}.fastq'
    singularity:    'docker://my_repos/trim_adaptors:2.9'
    shell:          'trim_adaptors --raw_reads={input} > {output}'

step run_assembly:
    input:           'output/trimmed/{sample}.fastq'
    output:          'output/assemblies/{sample}.fasta'
    singularity:     'shub://my_repos/choice_assembler:1.5'
    shell:           'choice_assembler --reads={input} > {output}'
```

Some barriers to container usage

- **Building containers can be painful** if the dependencies are disorganised
- **Duplication of effort**
- **Some software shouldn't go in a container** because of "unfortunate licensing issues"
  - DTU software *e.g.* `rnammer`, `tmhmm`
  - GATech: `GeneMark`
  - GIRInst's `RepBase`
- **Getting `Singularity` installed**

See the DTU's license here

Reproducible analysis stack

Guidelines:

1. Don't modify raw data
2. Record the code (with version control)
3. Capture the computing environment

Stack:

`md5sum raw_reads.fastq`? `chmod 444`?
+ Workflow manager (`snakemake`, `nextflow`)
+ VCS (`git`)
+ Software containers (`Singularity`)

Getting started

Reproducibility for bioinformatics:

- Joep de Ligt: *Scalable workflows and reproducible data analysis for genomics*
- plenty of online talks e.g. Adam Labadorf of Boston Uni

Workflow managers:

- Snakemake Tutorial
- Nextflow: Get started

Software containers:

- Blair Bethwaite: *Containers in HPC* Tutorial
- Singularity Quick Start

Version control:

- memorise a handful of `git` commands

These slides: https://github.com/TomHarrop/eresearch2020