Reproducible bioinformatics

from a user's perspective

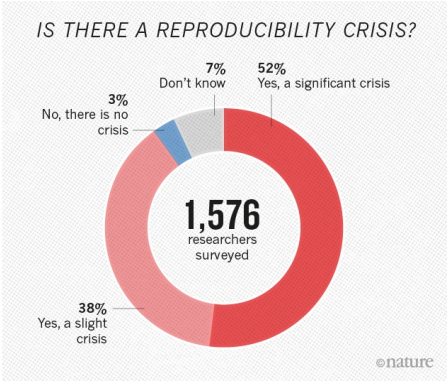
Tom Harrop

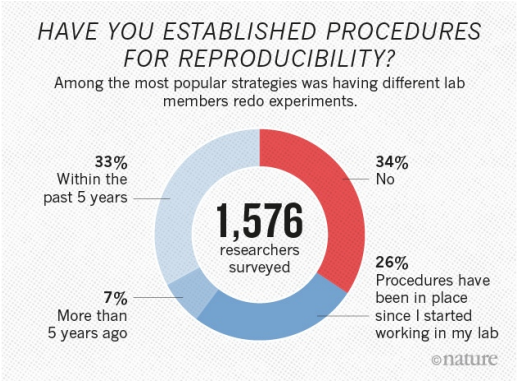
The University of Otago

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

1. Take care peeking at the data

Ziemann *et al. Genome Biology* (2016) 17:177
DOI 10.1186/s13059-016-1044-7

Genome Biology

COMMENT Open Access

Gene name errors are widespread in the scientific literature

Mark Ziemann[1], Yotam Eren[1,2] and Assam El-Osta[1,3*]

Abstract

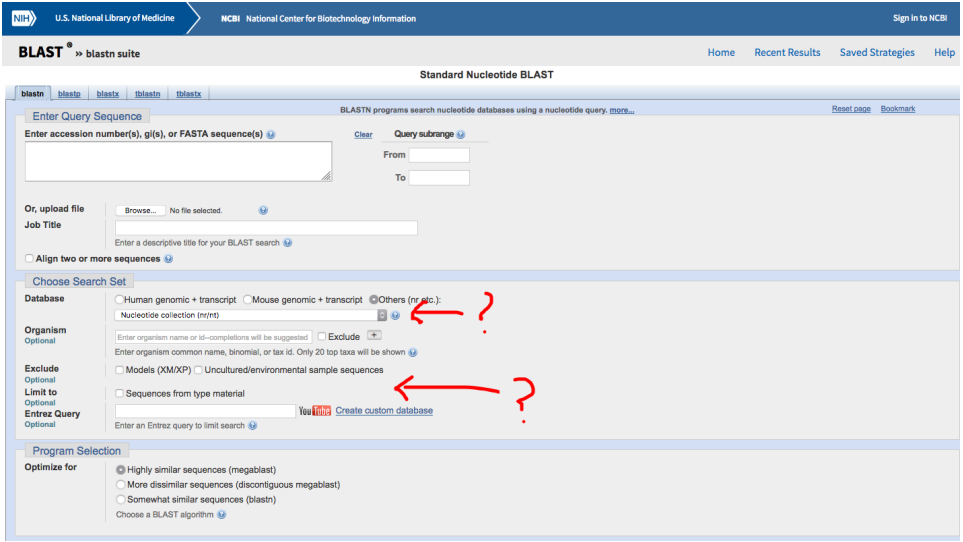
The spreadsheet software Microsoft Excel when used with default settings, is known to convert gene names to dates and floating-point numbers. A programmatic scan of leading genomics journals reveals that approximately one-fifth of papers with supplementary Excel gene lists contain erroneous gene name conversions.

Keywords: Microsoft Excel, Gene symbol, Supplementary data

Abbreviations: GEO, Gene Expression Omnibus; JIF, journal impact factor

Ziemann *et al.*, 2016. 10.1186/s13059-016-1044-7

2. Point-and-click software is less likely to be reproducible



2. Running on-the-fly probably won't be reproducible

Examples:

- install software locally
- use software installed by the admin
- type your commands directly into the console and hit enter!
- save a set of scripts to run in order

Possible issues:

- will it run again?
- are **all** the steps documented?
- is the code you recorded the same as the code you ran?
- did you correctly record the order of steps?

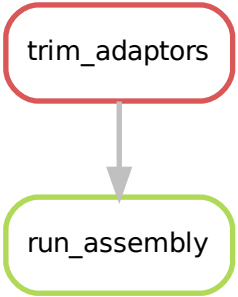
2. Workflow managers force you to record every step

Define:

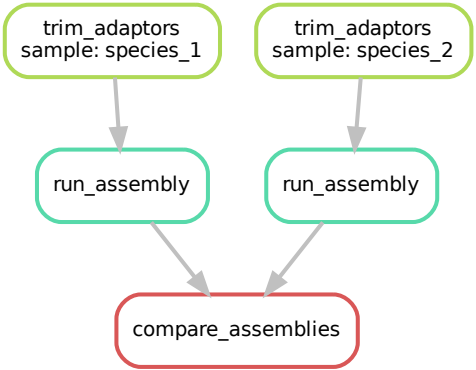
```
rule trim_adaptors:
    input:  'data/raw_reads/{sample}.fastq',
    output: 'output/trimmed/{sample}.fastq'
    shell:  'trim_adaptors --raw_reads={input} > {output}'

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

Run:

```
workflow_manager run_assembly
```



2. 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

3. 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

3. Reproducible computing environment

On our department's hardware:

```
salmon --version
```

```
salmon 0.9.1
```

e.g. **Ubuntu 19.10**:

```
apt policy salmon
```

```
salmon:
  Installed: (none)
  Candidate: 0.12.0+ds1-1
  Version table:
     0.12.0+ds1-1 500
        500 http://nz.archive.ubuntu.com/ubuntu eoan/universe amd64 Packages
```

3. Software containers

- Isolated, complete environment (a mini OS)
- Contain specific version of software with dependencies

Singularity:

- Mobility of compute
- Reproducibility
- Support on existing traditional HPC

sylabs.io

3. Singularity containers

Running directly:

```
salmon --help
```

```
Error in running command bash
```

Running with Singularity:

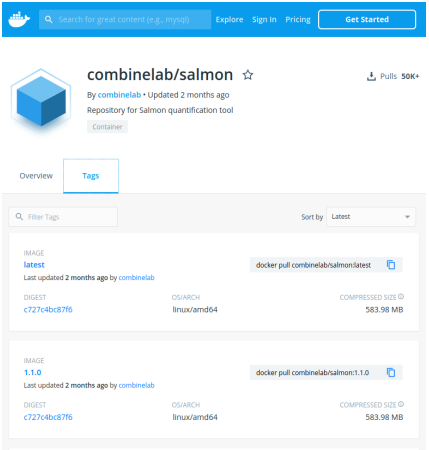
```
singularity exec \
  salmon_1.1.0.sif \
  salmon --help
```

```
Usage:  salmon -h|--help or
        salmon -v|--version or
        salmon -c|--cite or
        salmon [--no-version-check] <COMMAND> [-h | options]
```

3. Getting software in containers

- Some developers provide docker containers

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



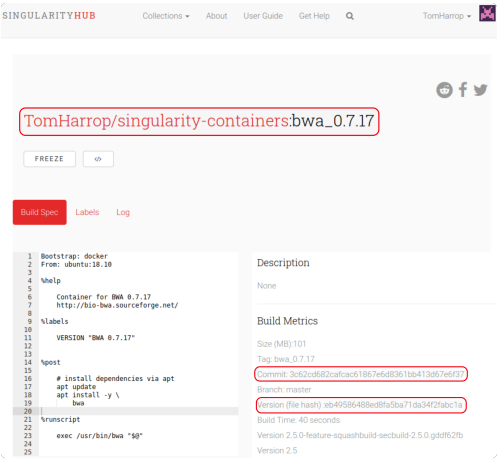
hub.docker.com

3. Getting software into containers

- Usually have to build it yourself

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 "$@"
```



singularity-hub.org

3. Some barriers to container usage

- **Building containers can be painful** if the dependencies are disorganised
- **Duplication of effort**
- **Some software ~~can't~~ 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

2 & 3. Workflow managers support containers and clusters

```
rule 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}'

rule 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}'
```

Cluster execution, *e.g.*:

```
snakemake --drmaa " -q username" -j 32
```

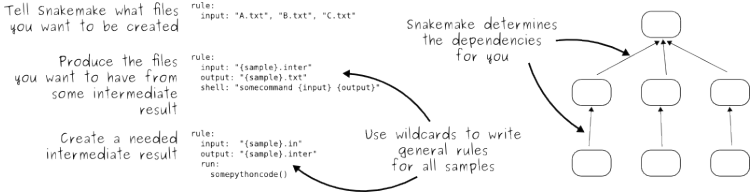
Reproducible analysis stack

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`)



snakemake.readthedocs.io

Getting started

Reproducibility for bioinformatics:

- online lectures e.g. Adam Labadorf of Boston Uni

Workflow managers:

- Snakemake Tutorial
- Nextflow: Get started

Software containers:

- Singularity Quick Start

Version control:

- memorise a handful of `git` commands

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