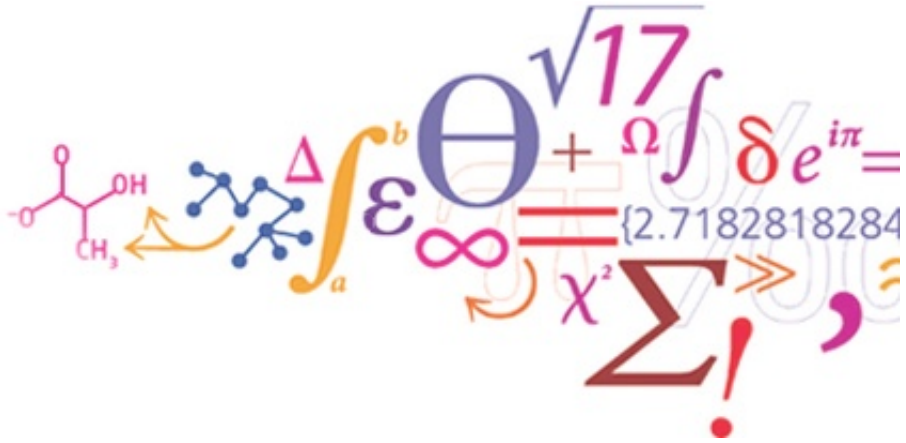


Practice What You Preach

Reproducible Research at the Front Lines of Science

Kai Blin
Novo Nordisk Foundation Center for Biosustainability

Tools for Reproducability in Bioinformatics
2016-04-21

Hej

Who is that guy?

- Scientific Software Engineering
- Microbiology
- antiSMASH
- CRISPy

More info:

- phdops.kblin.org
- github.com/kblin
- @kaiblin
- orcid.org/0000-0003-3764-6051

DTU

Overview

- Reproducible Research
- Reproducible Research in Practice
  - Problems

Overview

- Reproducible Research
- Reproducible Research in Practice
    - Challenges

Overview

- Reproducible Research
- Reproducible Research in Practice
  - Challenges
- What does actually work (for me)

There is no such thing as "reproducible science", there is just "science" and "not science".
– Someone on Twitter

Reproducible Research

Reproducibility

Ideally:

- Lab notebook available
- Data on Figshare
- Code on GitHub/Bitbucket (and Figshare)
- Preprint on a preprint server

Reproducibility

- Reviewers check and reproduce results
- Fame and glory (and grants)

In theory, there is no difference between theory and practice.
But, in practice, there is.
– Jan L. A. van de Snepscheut

Reproducible Research in Practice

Challenges

Reproducibility isn't free

- Making sure your research is 100% reproducible is a lot of work.
- This takes time and effort. (see Reproducibility isn't free by FitzJohn *et al.*)
- Even if you are convinced, is your PI / supervisor? Their boss?

Challenges

Reproducibility isn't compelling

Nice post by Greg Wilson in the context of Software Carpentry.

- ~ 5 mio articles published between 1990-2000
- Of these ~ 100 retracted for "computational irreproducibility"
- Chances that your paper is retracted: 1 : 5 000 000
- Assuming ~ 8 months to write paper and 48 hour work week, can spend **115 seconds** on reproducibility

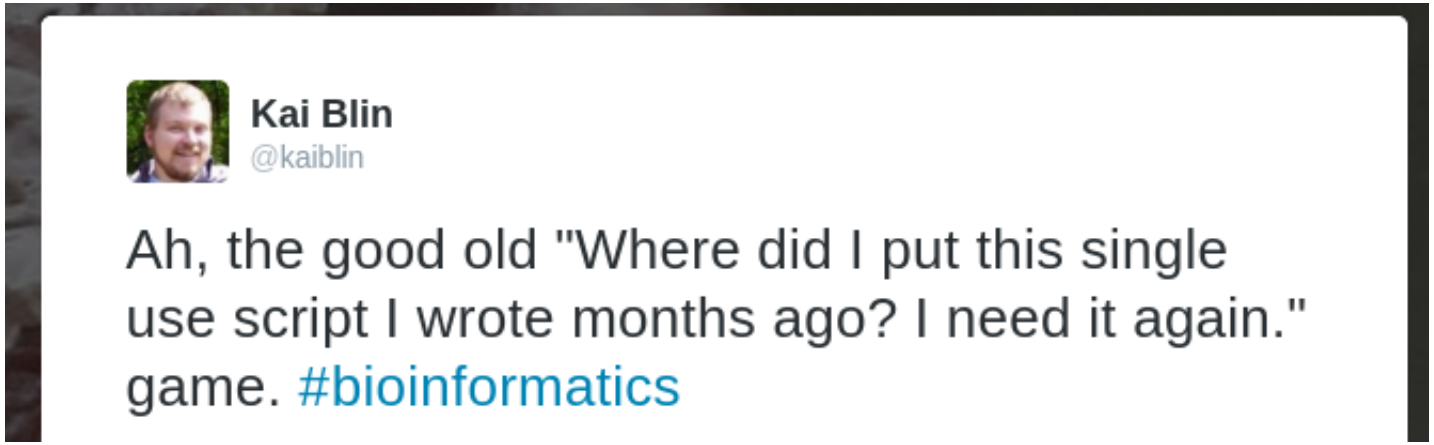
Challenges

Chicken and egg

- Reviewers don't ask for it
- Researchers don't provide it
- Catching this at publication stage is too late

Challenges

It's not a reflex yet



But…

Some points raised by C. Titus Brown

- start small
  - provide raw data
  - provide code
  - provide what version of what tool you called with what parameters
- any reproducibility is better than no reproducibility

Reproducible Research != Open Science

Disclaimer: I like Open Science

- Can work reproducibly even in closed science
- Maybe easier to get buy-in from senior scientists for RR
- Many selfish reasons to do RR

It works for me.
– Christopher Walken

What works (for me)

What Works (for me)

- Reduce special cases
- Document remaining special cases
- less surprises == better work
- learn from past mistakes

Directory Layout

As regular as possible, at least per project type.

```
2016-04-21_e.xamplis_de_novo/
|-- fastqc/
    |-- post/
    \-- pre/
|-- output/
|-- reads/
|-- scripts/
|-- trimmed/
|-- Makefile
\-- README.md
```

Directory Layout

Use a script to create project layout

```
#!/bin/bash

mkdir -p reads fastqc/{pre,post} output scripts trimmed
touch README.md Makefile
git init

cat >> .gitignore <<EOF
reads/*gz
fastqc/*
*.swp
EOF

git add .gitignore
git commit -m "Initial commit"
```

Lab Notebook

- `README.md` file per project
  - Have an explanation what the project is about for future self
  - Then add whatever you're doing to it as it happens
- copy & paste `README.md` into ELN
- Keep in git

git

all the things! (almost)

- Whenever possible, keep stuff in git
- `README.md`, `Makefile`, scripts, etc.
- Commit when something is changed
- Don't commit reads / generated data
- (Maybe use git-annex for large files)

git

part deux

- Have some git repo for random stuff unrelated to projects
  - Avoid "where did I put this script?"
- Make small commits of logical units of change
- Write good commit messages

git

commit messages

I suggest the following format:

```
Short summary line < 60 characters

A longer explanation of what the change is about.
This is what your'll read to figure out what this change is about.
Make this count, your future self will thank you.
```

Don't Run Commands Manually

- Every project has a `scritps` directory or `Makefile`.
- Data manipulation is driven from there
- Also applies to (most) other things I do
  - If more elaborated than `ls`, put it in a script
  - Not perfect yet, but getting there

Software Management

- Package software
- Use fpm & aptly / `createrepo`
- Install locally or put into Docker containers

Docker

- Great to deal with software with many dependencies

BUT

- Clumsy for CLI tools

Docker CLI example

```
#!/bin/bash

readonly INPUT_FILE=$(basename $1)
readonly INPUT_DIR=$(dirname $(readlink -f $1)); shift
readonly OUTPUT_DIR=$(readlink -f $1); shift
readonly CONTAINER_SRC_DIR="/input"
readonly CONTAINER_DST_DIR="/output"

if [ ! -d ${OUTPUT_DIR} ]; then
    mkdir ${OUTPUT_DIR}
fi
docker run \
    --volume ${INPUT_DIR}:${CONTAINER_SRC_DIR}:ro \
    --volume ${OUTPUT_DIR}:${CONTAINER_DST_DIR}:rw \
    --detach=false --rm --user=$(id -u):$(id -g) \
    antismash/standalone ${INPUT_FILE} $@
```

Workflow Management Systems

- Good idea for repetitive workflows
- Check what your colleagues are using
- Maybe use Common Workflow Language (CWL)

Workflow Management Systems

- Keep track of inputs and parameters
- Easily run a workflow 5, 10, 100 times
- Lots of overhead for one-off analyses

Further reading

- Git can facilitate greater reproducibility and increased transparency in science
- Reproducible research is still a challenge
- Some myths of reproducible computational research

Thanks for your attention.

Questions?