Reproducible Research and Snakemake

SciLifeLab

NBIS

Reproducibility

- Reproducible research is about being able to replicate the results of a study
- It is an important aspect of the scientific method
- Computational reproducibility is one part of it
- Ideally, given the same data and the same code, there are identical outcomes

Code encompasses

- The workflow itself (→ `Snakefile`)
- The helper scripts you are calling (→ `scripts/`)
- The 3rd-party tools you are running (→ this lecture)

Computational reproducibility

Why the effort?

M. Schwab et al. Making scientific computations reproducible. https://dx.doi.org/10.1109/5992.881708

> Because many researchers typically forget details of their own work, they are not unlike strangers when returning to projects after time away. Thus, efforts to communicate your work to strangers can actually help you communicate with yourself over time.

→ You are also in the target audience

Don’t be that person

The journal Science implemented a replication policy in 2011. A study in 2018 requested raw data and code in accordance with the policy. Some answers:

> When you approach a PI for the source codes and raw data, you better explain who you are, whom you work for, why you need the data and what you are going to do with it.

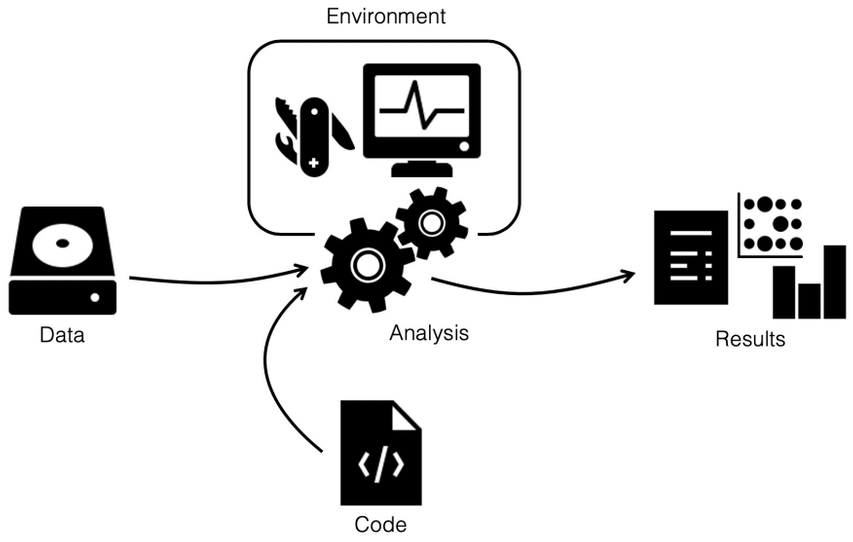
.

> I have to say that this is a very unusual request without any explanation! Please ask your supervisor to send me an email with a detailed, and I mean detailed, explanation.

(26% out of 204 randomly selected papers in the journal could be reproduced.)

Stodden et. al (2018). An empirical analysis of journal policy effectiveness for computational reproducibility https://doi.org/10.1073/pnas.1708290115

Combine tools to make research reproducible



- Track code changes over time with Git and share it on GitHub (not this talk)
- Make your workflow reproducible with a workflow manager (Snakemake, Nextflow, WDL)
- Make the execution environment reproducible with Conda environments and/or containers

Conda: a package, dependency, and environment manager

- Conda installs packages
- Packages come from a central repository at https://anaconda.org/
- Users can contribute their own packages via channels
- Highly recommended: The Bioconda channel

Using Conda

Prerequisites

- Install Conda, for example with Miniconda
- Set up the Bioconda channel
- Install Samtools and BWA into a new Conda environment named `mapping`:

  ```
  $ conda create -n mapping samtools bwa
  ```

- Conda also installs also the dependencies – other software required by Samtools and/or BWA.

To use the tools in the environment, activate it:

```
$ conda activate mapping
$ samtools --version
samtools 1.15.1
```

- Install a tool into an existing environment:

  ```
  conda install -n mapping bowtie2
  ```

  (Leaving out `-n mapping` installs into the currently active environment.)

Conda environments

- You can have as many environments as you wish
- Environments are independent from each other
- If something is broken, simply delete the environment and start over
- To test a new tool, install it into a Conda environment. Delete the environment to uninstall.
- Find tools to install by searching anaconda.org or with `conda search`

Conda environment files

- Conda environments can be created from environment files in YAML format.
- Example `bwa.yaml`:

```
channels:
  - conda-forge
  - bioconda
  - defaults
dependencies:
  - bwa=0.7.17
```

- Create the environment:

```
$ conda env create -n bwa -f bwa.yaml
```

Snakemake + Conda

Option one: A single environment for the entire workflow

- Write an environment file (`environment.yaml`) that includes all tools used by the workflow:

```
name: best-practice-smk
channels:
  - conda-forge
  - bioconda
  - default
dependencies:
  - snakemake=6.8.0   # ← Snakemake is part of the environment
...
  - multiqc=1.11   # ← Version numbers for reproducibility
  - samtools=1.13
```

- Create the environment, activate it and run the workflow within it:

```
$ conda env create -f environment.yml
$ conda activate best-practice-smk
$ snakemake
```

- Possibly helpful: `conda export -n envname > environment.yaml`

source: best practice example

Snakemake + Conda

Option two: Rule-specific environments

You can let Snakemake create and activate Conda environments for you.

1. Create the environment file, such as `envs/bwa.yaml` (`envs/` is best practice)
2. Add the `conda:` directive to the rule:

```
rule create_bwa_index:
    output: ...
    input: ...
    conda: "envs/bwa.yaml"  # ← Path to environment YAML file
    shell:
        "bwa index {input}"
```

1. Run `snakemake --use-conda`

- Snakemake creates the environment for you and re-uses it next time
- If the YAML file changes, the environment is re-created
- `conda:` does not work if you use `run:` (instead of `shell:` or `script:`)

modified from: best practice example

Using a "module" system

- Conda environments can be large and slow to create
- Some cluster operators frown upon using it
- UPPMAX and other clusters have a module command for getting access to software:

```
$ module load bioinfo-tools bwa
```

- Snakemake supports this with the `envmodules:` directive:

```
rule create_bwa_index:
    output: ...
    input: ...
    envmodules:
      "bioinfo-tools",
      "bwa",
    conda: "envs/bwa.yaml"  # ← Fallback
    shell:
        "bwa index {input}"
```

- Run with `snakemake --use-envmodules`
- For reproducibility, the Snakemake documentation recommends to also include a `conda:` section

Containers

- Containers represent another way of packaging applications
- Each container contains the application itself and all dependencies and libraries (that is, a functional Linux installation)
- It is fully isolated from the other software on the machine: By default, the tools in the container can only access what is in the container.
- The most common software for managing containers is Docker

Containers

Docker nomenclature

- A Docker image is a standalone executable package of software (on disk)
- A Dockerfile is a recipe used to build a Docker image
- A Docker container is a standard unit of software run on the Docker Engine (running an image gives a container)
- DockerHub is an online service for sharing Docker images

Docker vs Singularity

- On high-performance clusters (HPC), Docker is often not installed due to security concerns. Singularity is often available as an alternative.
- Docker images can be converted into Singularity images
- → Singularity can be used to run Docker containers

Running Snakemake jobs in containers

Snakemake can run a jobs in a container using Singularity

- Ensure your system has Singularity installed
- Find a Docker or Singularity image with the tool to run (https://biocontainers.pro/ or DockerHub)
- Add the `container:` directive to your rule:

```
rule minimap2_version:
    container: "docker://quay.io/biocontainers/minimap2:2.24--h5bf99c6_0"   # ← "docker://" is needed
    shell:
        "minimap2 --version"
```

- Start your workflow on the command line with `--use-singularity`

```
$ snakemake --use-singularity -j 1
...
Activating singularity image .../.snakemake/singularity/342e6ddbac7e5929a11e6ae9350454c0.simg
INFO:    Converting SIF file to temporary sandbox...
2.24-r1122
INFO:    Cleaning up image...
...
```

Containers – advanced topics

- A Docker image to use for all rules can be specified
- You can package your entire workflow into a Docker image by writing a Dockerfile. See this example
  - Snakemake runs inside the container.
  - To run the workflow, only Docker or Singularity is needed
- Conda and containers can be combined: Specify a global container, run with `--use-conda --use-singularity`, and Snakemake creates the Conda environment within the container.
- Snakemake can automatically generate a Dockerfile that contains all Conda environments used by the rules of the workflow using the flag `--containerize`.

Summary

There are many ways to use other tools for reproducible research together with Snakemake:

- Use Git for version control, backup and share your code
- Run rules or your entire workflow in Conda environments
- Run your rules in isolated Docker/Singularity containers
- Package your entire workflow in a Docker container