Combining Tools for Reproducible Research with Snakemake

SciLifeLab

NBIS

Reproducibility is rarer than you think

The results of only 26% out of 204 randomly selected papers in the journal Science could be reproduced.[1]

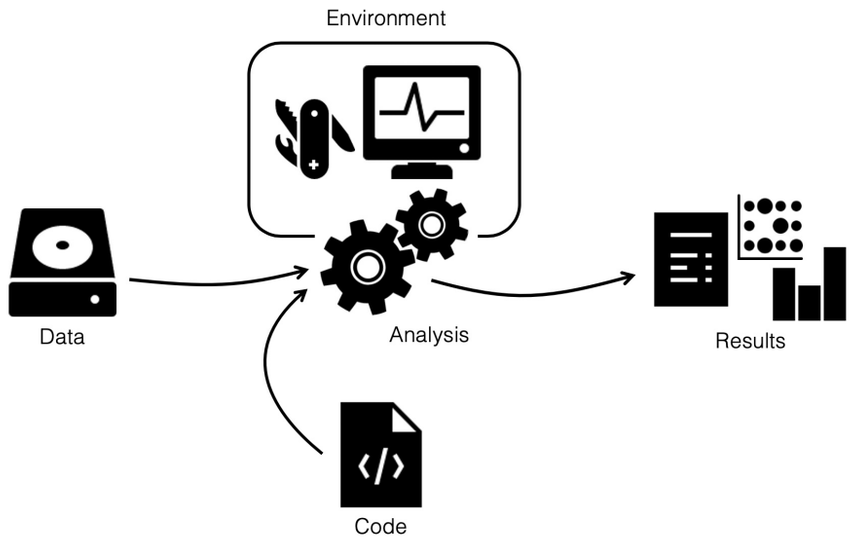
[1] Stodden et. al (2018). "An empirical analysis of journal policy effectiveness for computational reproducibility". PNAS. 115 (11): 2584-2589

> Many journals are revising author guidelines to include data and code availability.
>
> (...) an improvement over no policy, but currently insufficient for reproducibility.

SciLifeLab

NBIS

Combining Tools for Reproducible Research with Snakemake



- Track your Snakemake code with Git and share it in a remote repository on GitHub or BitBucket (not covered in this lecture)
- Combine Snakemake with Conda and/or containers to make the compute environment reproducible
- Integrate foreign workflow management systems such as Nextflow pipelines into your Snakemake workflow

Conda

- Is a package, dependency, and environment manager

  > packages: any type of program (e.g. bowtie2, snakemake etc.)
  >
  > dependency: other software required by a package
  >
  > environment: a distinct collection of packages

- Keeps track of the dependencies between packages in each environment

Conda

1. Running a Snakemake rule with a Conda environment

- Make sure you have Conda installed (Miniconda or Anaconda)
- Find your Conda package on http://anaconda.org
- Create a Conda environment file (e.g. `bwa.yaml`)

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

source: best practice example

- Store your `yaml` files in a directory for environments
- For reproducibility, it is important to keep include package versions in your environment file

Conda

1. Running a Snakemake rule with a Conda environment

- Add the path to the Conda environment `yaml` file to your rule using `conda`

```
rule map_bwa_index:
    output: expand("{{ref}}{ext}", ext=[".amb", ".ann", ".bwt", ".pac", ".sa"])
    input: config["ref"]
    log: "logs/bwa/index/{ref}.log"
    conda: "../envs/bwa.yaml"
    shell:
        "bwa index {input}"
```

modified from: best practice example

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

```
$ snakemake --use-conda
```

- This doesn't work if you use `run` (instead of `shell` or `script`)

Conda

2. Using one Conda environment for the entire workflow

- Write a Conda environment file that includes all tools used by the workflow (save it as e.g. `environment.yaml`)

```
name: best-practice-smk
channels:
  - conda-forge
  - bioconda
  - default
dependencies:
  - snakemake=6.8.0
  - python=3.8
  - pandas=1.3.3
  - jupyter=1.0
  - jupyter_contrib_nbextensions=0.5.1
  - jupyterlab_code_formatter=1.4
  - bwa=0.7.17
  - multiqc=1.11
  - r-ggplot2=3.3.5
  - samtools=1.13
```

source: best practice example

Conda

2. Using one Conda environment for the entire workflow

- Create the environment

```
$ conda env create -f environment.yml
```

- Activate your Conda environment

```
$ conda activate best-practice-smk
```

- Start your Snakemake workflow

```
(best-practice-smk) [...] $ snakemake
```

Containers

What can I use containers for?

- Run applications securely isolated in a container, packaged with all dependencies and libraries
- As advanced environment manager
- To package your code with the environment it needs
- To package a whole workflow (e.g. to accompany a manuscript)
- And much more

Docker vs. Singularity

- Docker was developed for any operating system except high-performance computing (HPC) clusters
- Singularity is an open source container platform suitable for HPC clusters

Containers

Docker nomenclature

- A Docker file is a recipe used to build a Docker image
- A Docker image is a standalone executable package of software
- A Docker container is a standard unit of software run on the Docker Engine
- DockerHub is an online service for sharing Docker images
- Docker images can be converted into Singularity images

Containers

1. Running Snakemake rules with Singularity

- Snakemake can run a rule isolated in a container, using Singularity
- All Conda packages are available as Docker and Singularity images, e.g. on http://biocontainers.pro (bioconda channel)
- Many other Docker images are available on DockerHub
- Or build your own Docker or Singularity images

Containers

1. Running Snakemake rules with Singularity

- Make sure your system has Singularity installed
- Find the Docker or Singularity image in which you want to run the rule
- Add the link to the container image (or the path to a Singularity `*.sif` file) to your rule using the `container` directive

```
rule NAME:
    input:
        "table.txt"
    output:
        "plots/myplot.pdf"
    container:
        "docker://joseespinosa/docker-r-ggplot2"
    script:
        "scripts/plot-stuff.R"
```

source: Snakemake documentation

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

```
$ snakemake --use-singularity
```

Containers

2. Packaging your Snakemake workflow in a Docker container

- Make sure your system has Docker installed
- Write a Docker file, e.g. see this example
    - Start with the official `Ubuntu` image
    - Install Miniconda and other required tools (e.g. Snakemake)
    - Add the project files (e.g. `Snakefile`, `config.yaml`, `environment.yaml`)
    - Install the Conda environment containing all packages run by the workflow

Containers

2. Packaging your Snakemake workflow in a Docker container

- Create a Docker image from your Docker file (e.g. called `my_workflow`)

```
$ docker build -t my_workflow .
```

- Run your container, e.g.

```
$ docker run my_workflow
```

- Share your Docker file on GitHub or BitBucket, or your Docker image on DockerHub

Combinations of Conda and Containers

Combine Conda-based package management with running jobs in containers

- A container can be specified globally (for the entire workflow) for a workflow with rule-specific Conda environments
- Snakemake then runs each job in this container with its corresponding Conda environment when run with `--use-conda --use-singularity`

More info: Snakemake documentation & best practice example

Combinations of Conda and Containers

Containerization of Conda-based workflows

- Snakemake can automatically generate a Docker file that contains all Conda environments used by the rules of the workflow using the flag `--containerize`

More info: Snakemake documentation

Integrating foreign workflow management systems

- From version 6.2 on, Snakemake can run workflows written in other workflow management systems such as Nextflow

- The workflow runs in Snakemake until a rule to run the foreign workflow is reached

- In this rule, Snakemake hands over to the other workflow manager

- Afterwards, Snakemake continues to run rules processing the output files of the foreign workflow

```
rule chipseq_pipeline:
    input:
        input="design.csv",
        fasta="data/genome.fasta",
        gtf="data/genome.gtf",
    output:
        "multiqc/broadPeaks/multiqc_report.html",
    params:
        pipeline="nf-core/chipseq",
        revision="1.2.1",
        profile=["conda"],
    handover: True
    wrapper:
        "0.74.0/utils/nextflow"
```

More info & source: Snakemake documentation

Summary

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

- Use Git to version control, backup and share your code
- Run rules or your entire workflow in Conda environments
- Run your rules in isolated Singularity containers
- Package your entire workflow in a Docker container
- Run pipelines written in other workflow management systems in your Snakemake workflow

Questions?

SciLifeLab

NBIS

SciLifeLab

NBIS