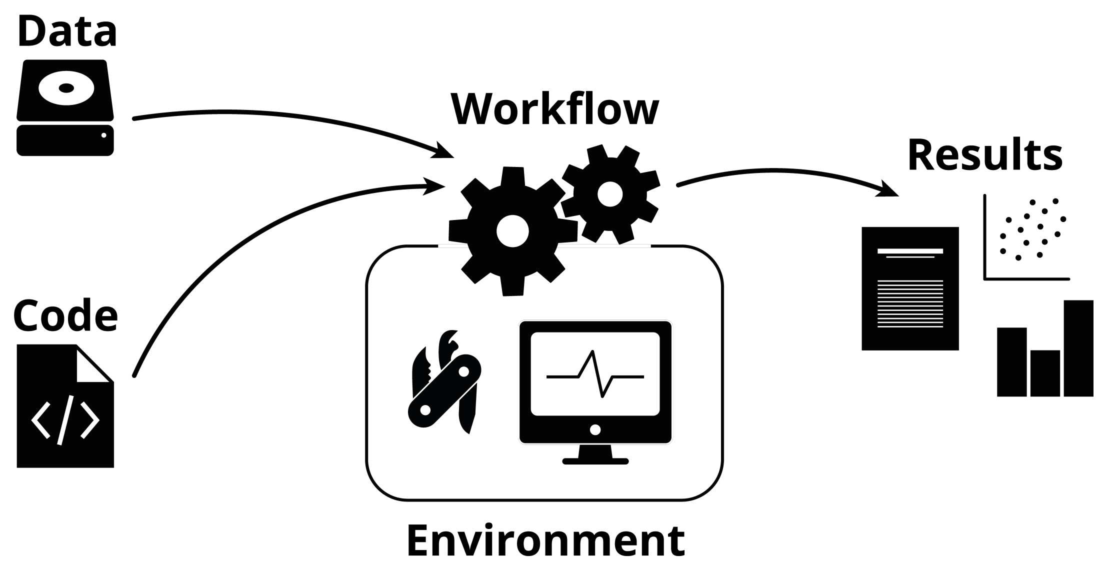
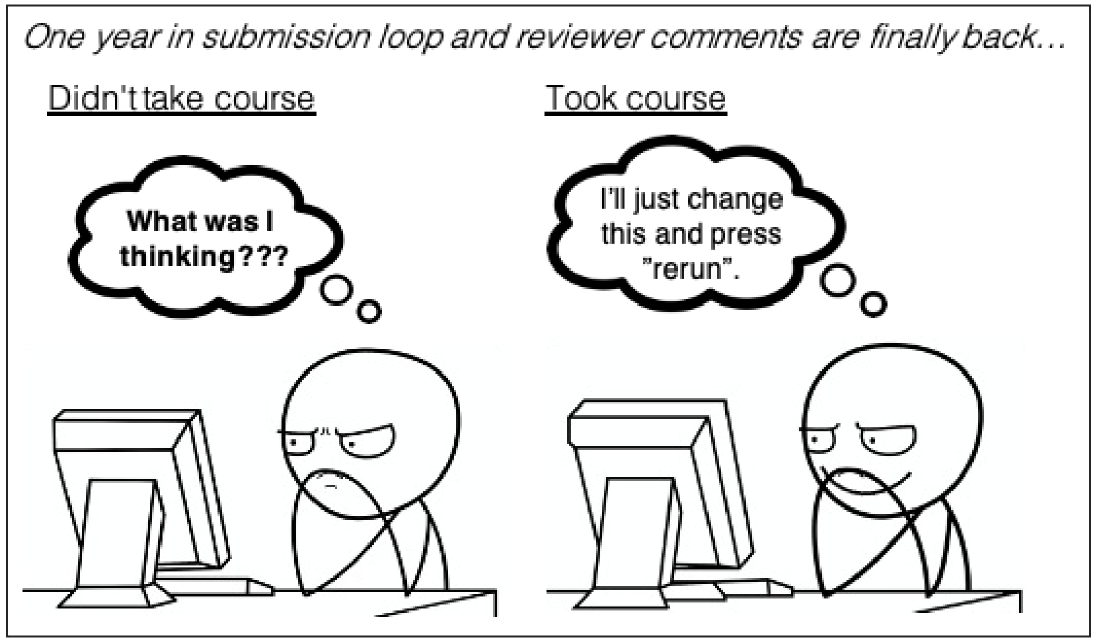
Putting it all together

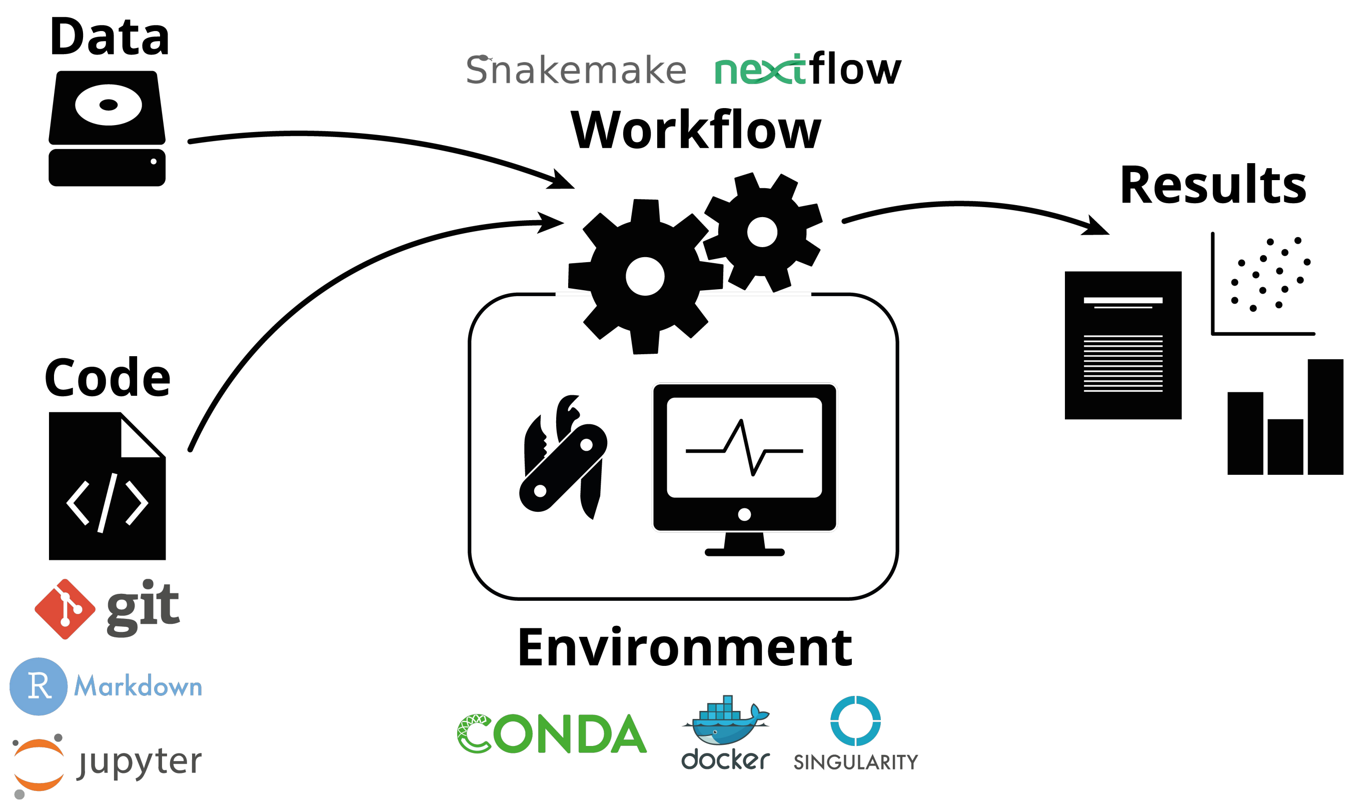
Take control of your research project
by making its different components reproducible



By working reproducibly you will also make your life a lot easier!


One year in submission loop and reviewer comments are finally back...
Didn't take course
What was I thinking???
Took course
I'll just change this and press "rerun".

What have we learned?



- How to use the version control system Git to track changes to code
- How to use the package and environment manager Conda
- How to use the workflow managers Snakemake and Nextflow
- How to use R Markdown and Jupyter to generate automated reports and to document your analyses
- How to use Docker and Singularity to distribute containerized computational environments

Divide your work into distinct projects

- Keep all files needed to go from raw data to final results in a dedicated directory
- Use relevant subdirectories
- Use Git to version control your projects
- Do not store data and results/output in your Git repository
- When in doubt, commit often instead of seldom

Find your own project structure

An example Snakemake-based project:

```
project/
├──── code/
├──── data/
│     ├──── meta/
│     ├──── raw_external/
│     └──── raw_internal/
├──── doc/
├──── intermediate/
├──── logs/
├──── notebooks/
│     └──── Untitled.ipynb
├──── results/
│     ├──── figures/
│     ├──── reports/
│     └──── tables/
├──── scratch/
├──── .gitignore
├──── config.yml
├──── environment.yml
├──── Dockerfile
├──── README.md
└──── Snakefile
```

An example Nextflow-based project:

```
project/
├──── bin/
│     └──── report.qmd
├──── data/
│     └──── metadata.csv
├──── doc/
├──── env/
│     ├──── Dockerfile
│     └──── environment.yml
├──── results/
├──── .gitignore
├──── main.nf
├──── nextflow.config
└──── README.md
```

- https://github.com/NBISweden/project_template
- https://github.com/fasterius/nbis-support-template
- https://github.com/snakemake-workflows/snakemake-workflow-template

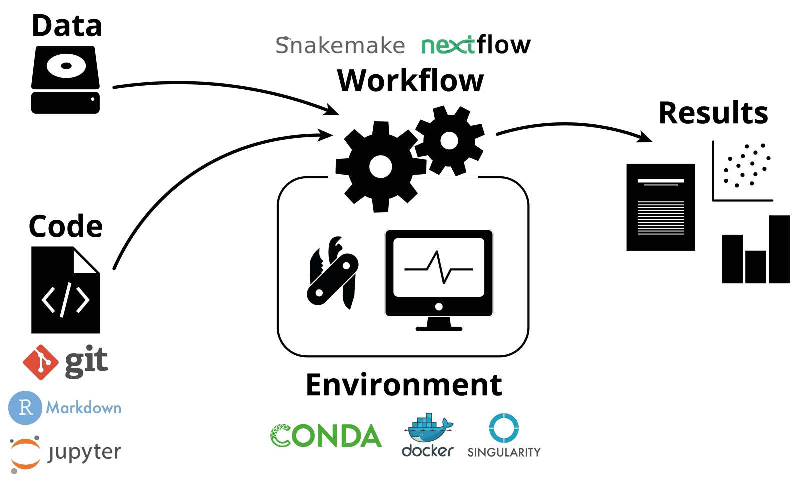
Treasure your data

- Keep your input data read-only - consider it static
- Don't create different versions of the input data - write a script, R Markdown document, Jupyter notebook or a Snakemake / Nextflow workflow if you need to pre-process your input data so that the steps can be recreated
- Backup! Keep redundant copies in different physical locations
- Upload your raw data as soon as possible to a public data repository

Organize your coding

- Avoid generating files interactively or doing things by hand
  - There is no way to track how they were made
- Write scripts, R Markdown documents, Jupyter notebooks or Snakemake / Nextflow workflows for reproducible results to connect raw data to final results
- Keep the parameters separate (e.g. at top of file or in a separate configuration file)

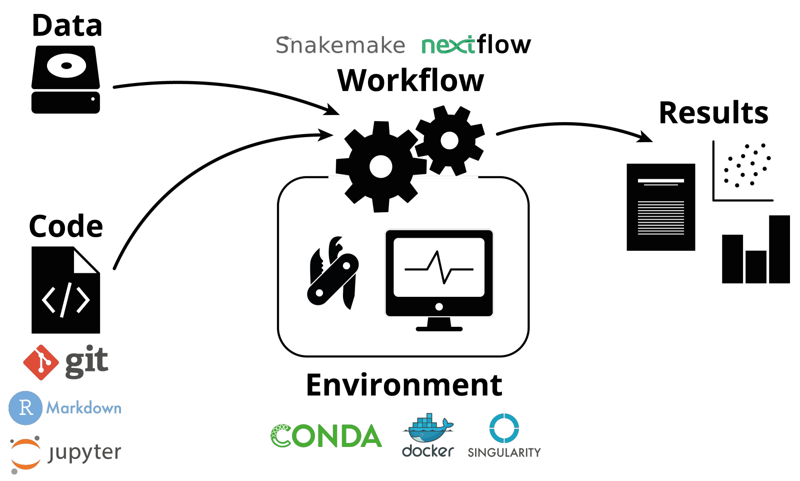
What is reasonable for your project?



Minimal: write code in a reproducible way and track your environment

- Track your projects with a Git repository each; publish code along with your results on e.g. GitHub
- Use Conda to install software in environments that can be exported and installed on a different system; also publish your `environment.yml` file along with your code

What is reasonable for your project?

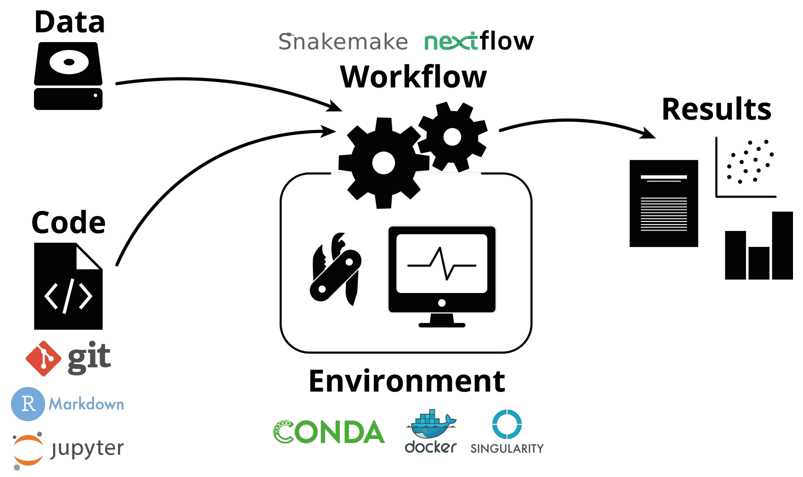


Minimal: write code in a reproducible way and track your environment

Good: structure and document your code with notebooks

- Use R Markdown or Jupyter notebooks to better keep track of and document your code

What is reasonable for your project?



Minimal: write code in a reproducible way and track your environment

Good: structure and document your code with notebooks

Great: track the full environment and connect your code in a workflow

- Go one step beyond in tracking your environment using Docker and/or Singularity/Apptainer
- Convert your code into a Snakemake / Nextflow workflow

Alternatives

Version control

- Git – Widely used and a lot of tools available + GitHub/BitBucket.
- Mercurial – Distributed model just like Git, close to sourceforge.
- Subversion – Centralized model unlike git/mercurial; no local repository on your computer and somewhat easier to use.

Alternatives

Environment / package managers

- Conda – General purpose environment and package manager. Community-hosted collections of tools at bioconda or conda-forge.
- Pip – Package manager for Python, has a large repository at pypi.
- Apt/yum/brew – Native package managers for different OS. Integrated in OS and might deal with e.g. update notifications better.
- Virtualenv – Environment manager used to set up semi-isolated python environments.

Alternatives

Workflow managers

- Snakemake – Based on Python, easily understandable format, relies on file names.
- Nextflow – Based on Groovy, uses data pipes rather than file names to construct the workflow.
- Make – Used in software development and has been around since the 70s. Flexible but notoriously obscure syntax.
- Galaxy - attempts to make computational biology accessible to researchers without programming experience by using a GUI.

Alternatives

Literate programming

- Jupyter – Create and share notebooks in a variety of languages and formats by using a web browser.
- R Markdown – Developed by Posit (previously Rstudio), focused on generating high-quality documents.
- Quarto - Dveloped by Posit (previously RStudio), command-line tool focused on generating high-quality documents in a language-agnostic way
- Zeppelin – Developed by Apache. Closely integrated with Spark for distributed computing and Big Data applications.
- Beaker – Newcomer based on Ipython, just as Jupyter. Has a focus on integrating multiple languages in the same notebook.

Alternatives

Containerization / virtualization

- Docker – Used for packaging and isolating applications in containers. Dockerhub allows for convenient sharing. Requires root access.
- Singularity/Apptainer – Simpler Docker alternative geared towards high performance computing. Does not require root.
- Podman - open source daemonless container tool similar to docker in many regards
- Shifter – Similar ambition as Singularity, but less focus on mobility and more on resource management.
- VirtualBox/VMWare – Virtualization rather than containerization. Less lightweight, but no reliance on host kernel.

"What's in it for me?"

MY PAST SELF IS CORRESPONDING WITH MY FUTURE SELF.
TOO BAD YOU CAN'T WRITE BACK.

NBIS Bioinformatics drop-in

Any questions related to reproducible research tools and concepts? Talk to an NBIS expert!

- Online (Zoom)
- Every Tuesday, 14.00-15.00 (except public holidays)
- Check www.nbis.se/events for Zoom link and more info