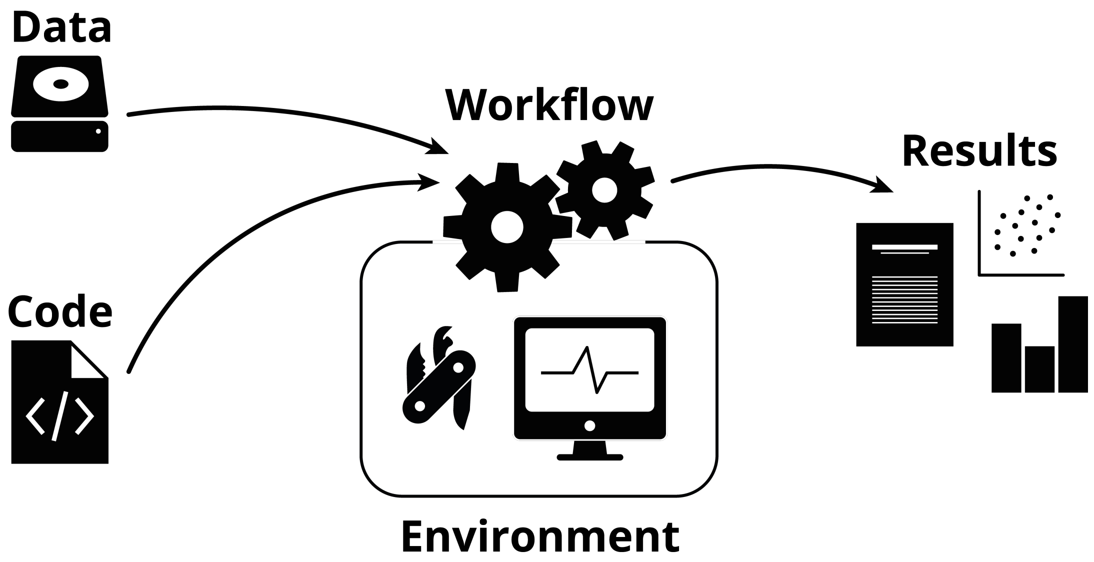
Putting it all together

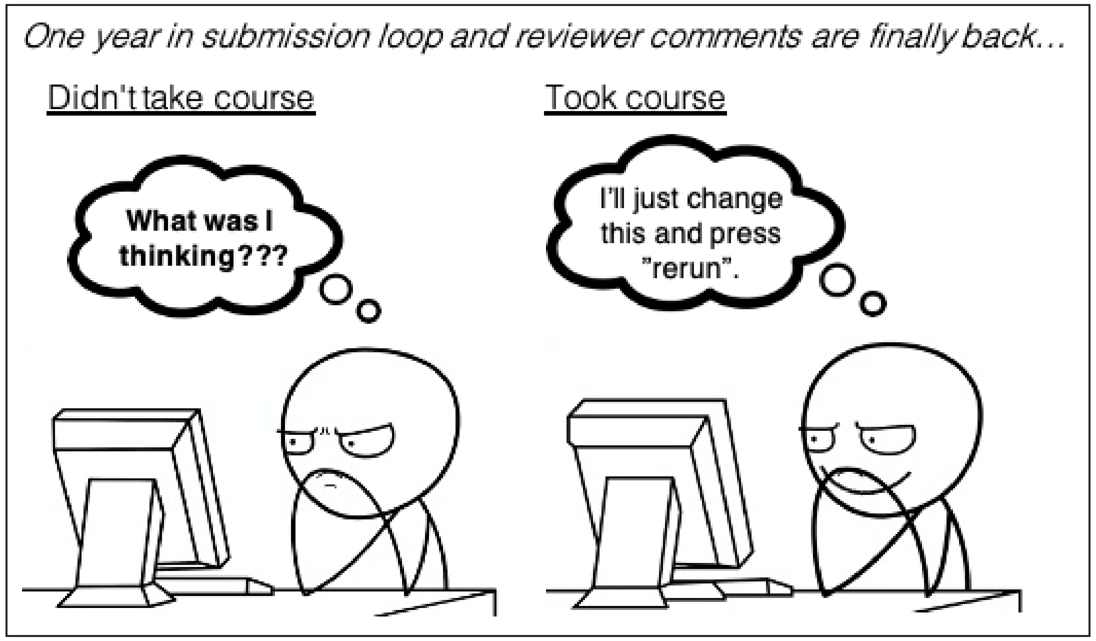
Take control of your research project
by making its different components reproducible



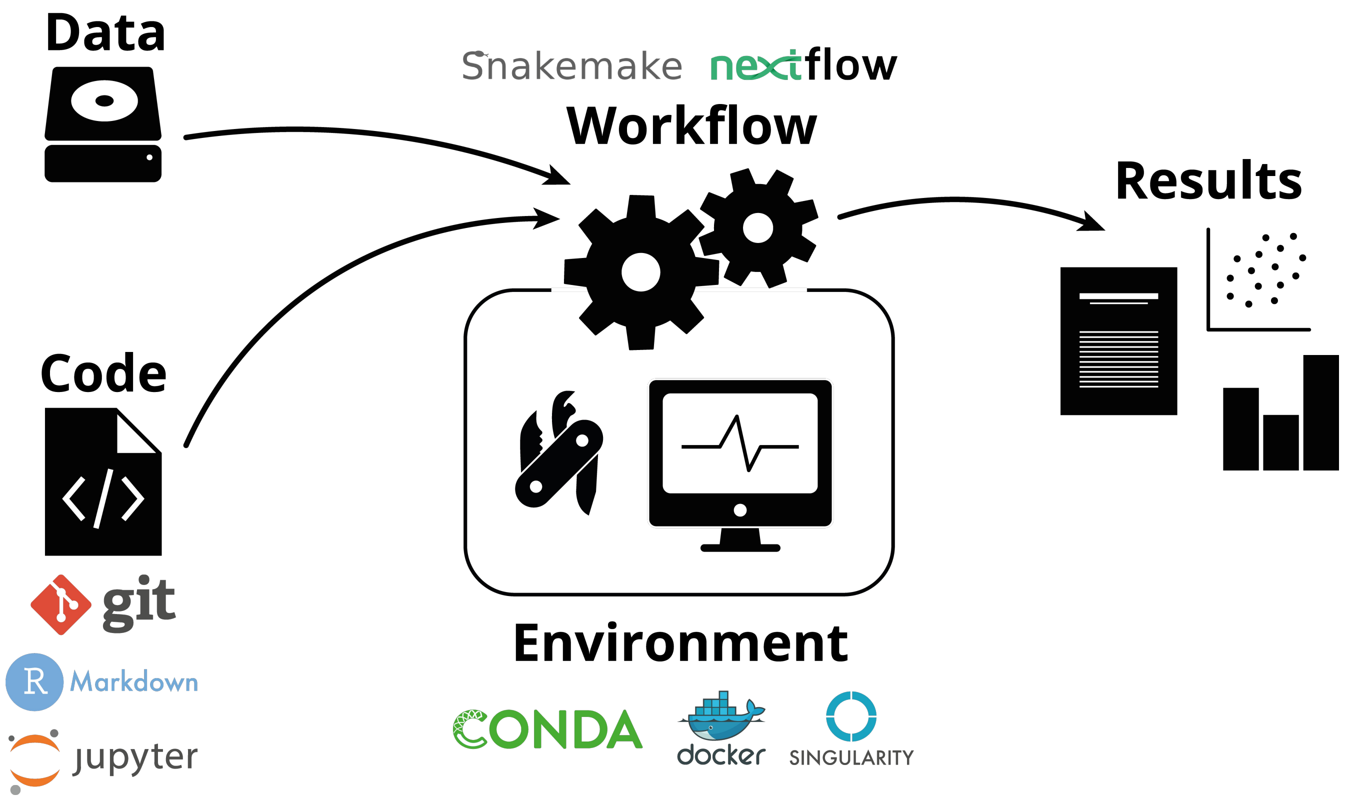
SciLifeLab

NBIS

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
Took course
What was I thinking???
I'll just change this and press "rerun".

What have we learned?



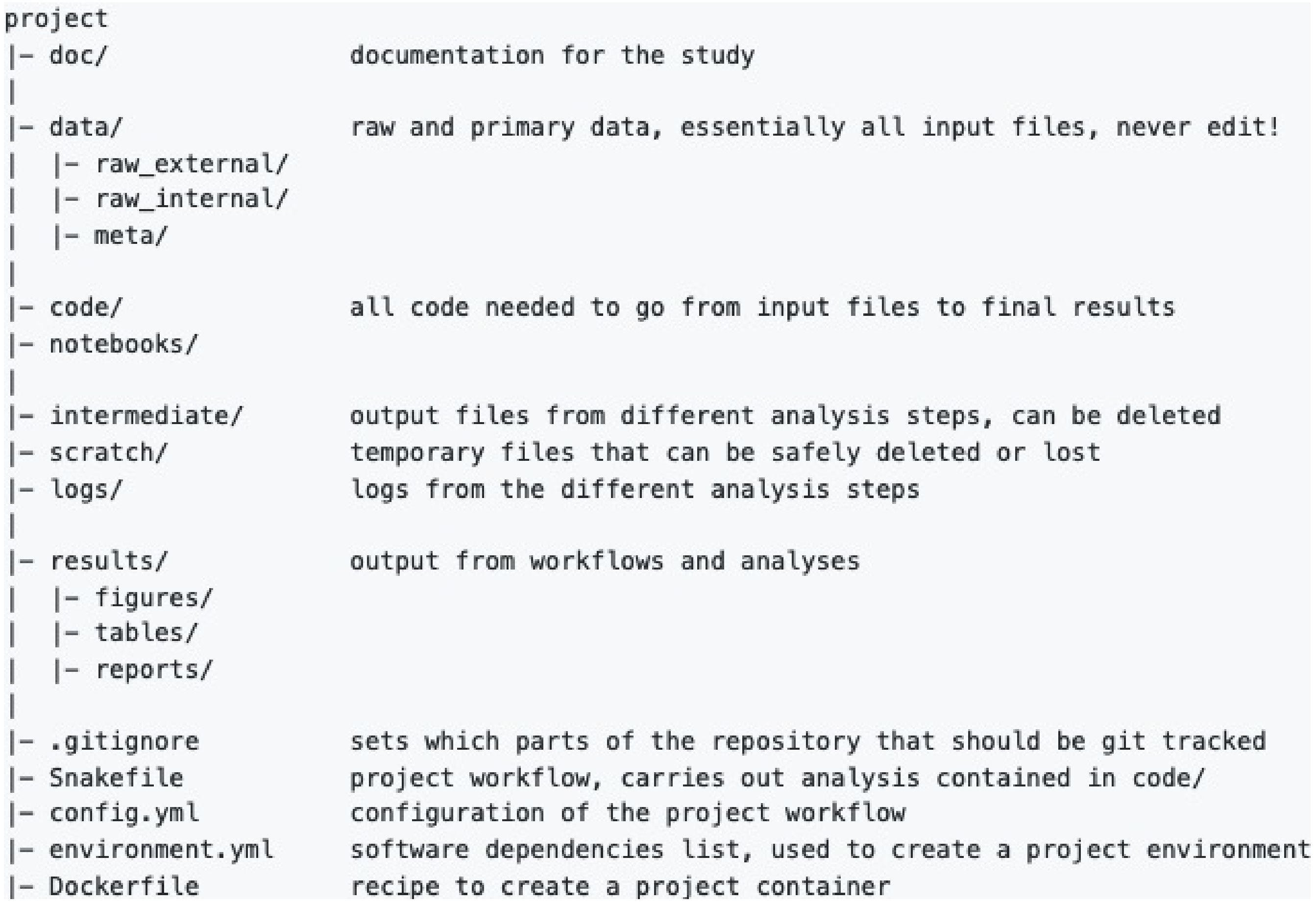
- How to use the version control system Git to track changes to code
- How to use the package and environment manager Conda
- How to use the workflow managers Snakemake and Nextflow
- How to use R Markdown and Jupyter to generate automated reports and to document your analyses
- How to use Docker and 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
- Many software support the “project way of working”, e.g. Rstudio and the text editors Sublime Text and Atom
- Use Git to create structured and version controlled project repositories

Everything can be a project

Project directory templates, e.g. NBIS project template:



```
project
|- doc/                documentation for the study
|
|- data/               raw and primary data, essentially all input files, never edit!
|  |- raw_external/
|  |- raw_internal/
|  |- meta/
|
|- code/               all code needed to go from input files to final results
|- notebooks/
|
|- intermediate/       output files from different analysis steps, can be deleted
|- scratch/            temporary files that can be safely deleted or lost
|- logs/               logs from the different analysis steps
|
|- results/            output from workflows and analyses
|  |- figures/
|  |- tables/
|  |- reports/
|
|- .gitignore          sets which parts of the repository that should be git tracked
|- Snakefile           project workflow, carries out analysis contained in code/
|- config.yml          configuration of the project workflow
|- environment.yml     software dependencies list, used to create a project environment
|- Dockerfile          recipe to create a project container
```

- https://github.com/NBISweden/project_template
- https://github.com/snakemake-workflows/cookiecutter-snakemake-workflow

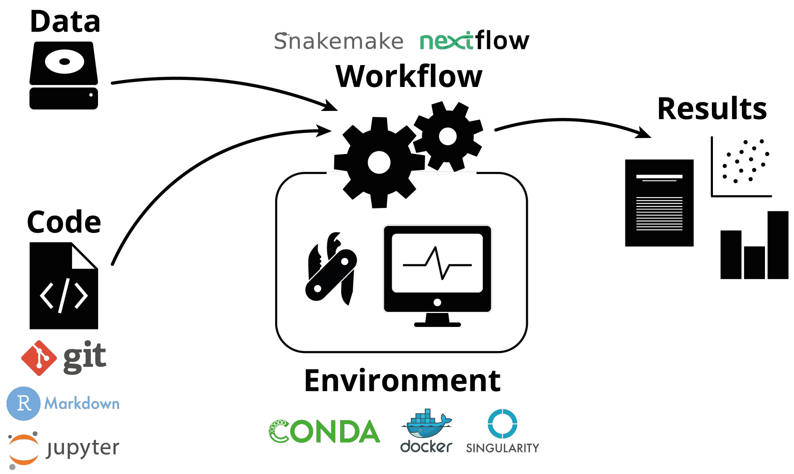
Treasure your data

- 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

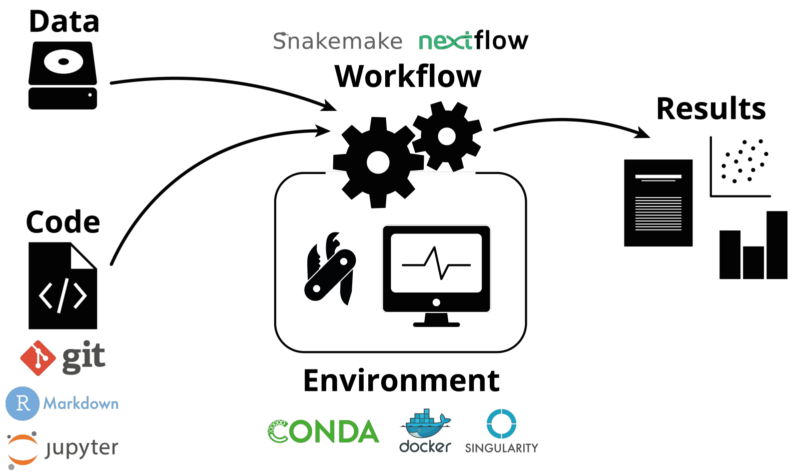
Connect your results with the code:

- Use R Markdown documents or Jupyter notebooks

Take another step:

- Convert your code into a Snakemake / Nextflow workflow

What is reasonable for your project?



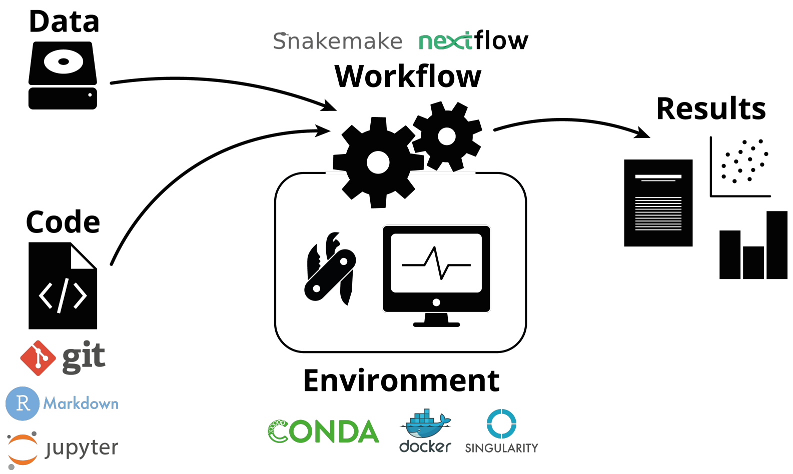
Minimal: write code in a reproducible way

Good: versioned and structured repository

Use Git for version controlling and collaboration:

- Create one Git repository per project
- Track your changes with Git
- Publish your code along with your results on e.g. GitHub

What is reasonable for your project?



Minimal: write code in a reproducible way

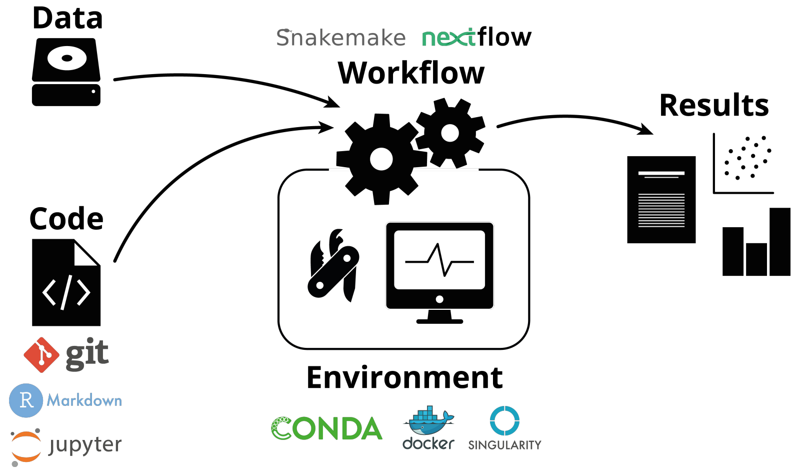
Good: versioned and structured repository

Better: organize software dependencies

Manage your depencencies:

- Use Conda to install software in environments that can be easily exported and installed on a different system

What is reasonable for your project?



Minimal: write code in a reproducible way

Good: versioned and structured repository

Better: organize software dependencies

Best: export everything!

Completely recreate the compute system:

- Consider packaging your project inside a or together with a Docker or Singularity container

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 Rstudio, focuses on generating high-quality documents.
- 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 – Simpler Docker alternative geared towards high performance computing. Does not require root.
- 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.
WATTERSON

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