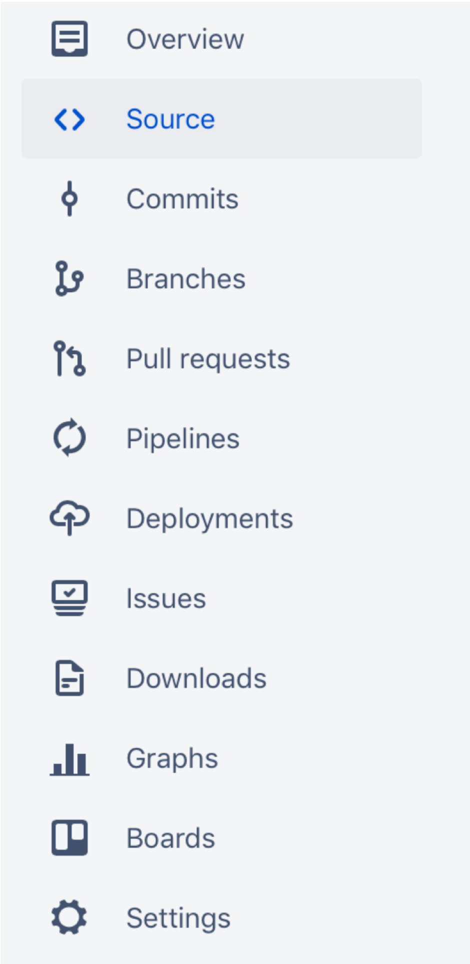
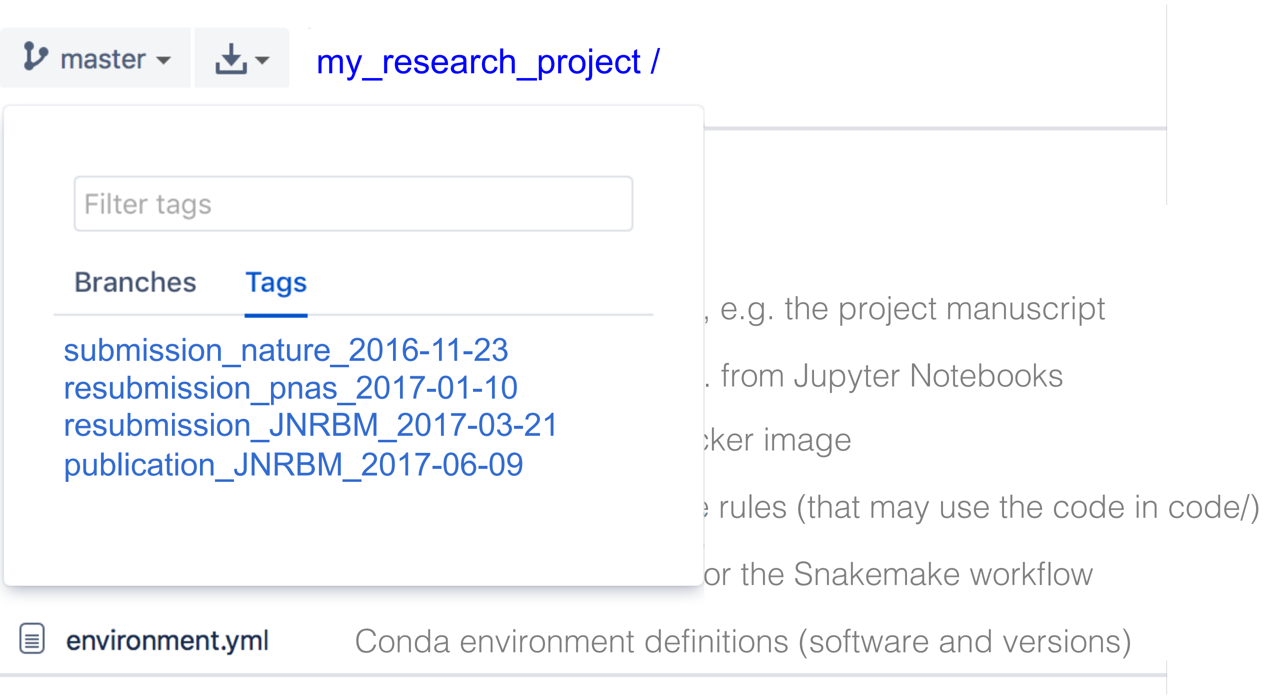
Putting it all together

my_research_project/





submission_nature_2016-11-23
resubmission_pnas_2017-01-10
resubmission_JNRBM_2017-03-21
publication_JNRBM_2017-06-09

, e.g. the project manuscript

. from Jupyter Notebooks

ker image

rules (that may use the code in code/)

or the Snakemake workflow

environment.yml Conda environment definitions (software and versions)

Options for reproducing:

- Git clone and run workflow.
- Git clone, activate conda env, and run workflow.
- Git clone, docker build, and run workflow in container.
- Docker pull and run workflow in container.

What is reasonable for your project?

Choose the right ambition level…

Snakemake

R Markdown

from R Studio

jupyter

Minimal: code for reproducible results

Good: versioned and structured repository

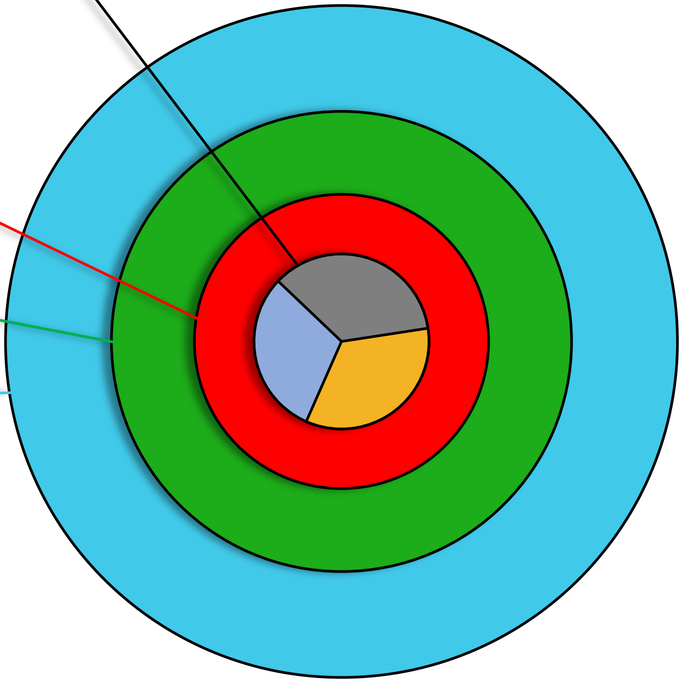
git

Better: ambition to organize dependencies

CONDA

Best: export everything!

docker



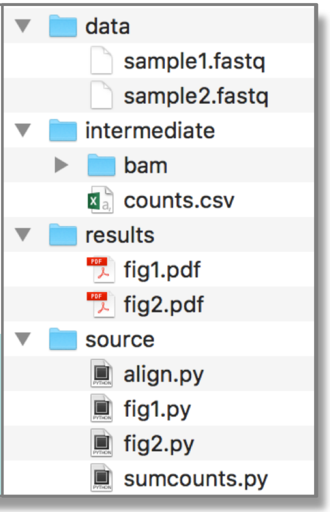
Reproducible research for bioinformatics projects

Leif Wigge (leif.wigge@scilifelab.se)
Rasmus Ågren (rasmus.agren@scilifelab.se)
Bioinformatics long-term support (LTS)

Everything can be a project

Divide your work into distinct projects and keep all files needed to go from raw data to final results in a dedicated directory with relevant subdirectories (see example).

Many software support the “project way of working”, e.g. Rstudio and the text editors Sublime Text and Atom.



```
▼ data
    sample1.fastq
    sample2.fastq
▼ intermediate
  ▶ bam
    counts.csv
▼ results
    fig1.pdf
    fig2.pdf
▼ source
    align.py
    fig1.py
    fig2.py
    sumcounts.py
```

Tip! Learn how to use git, a widely used system (both in academia and industry) for version controlling and collaborating on code.

git https://git-scm.com/

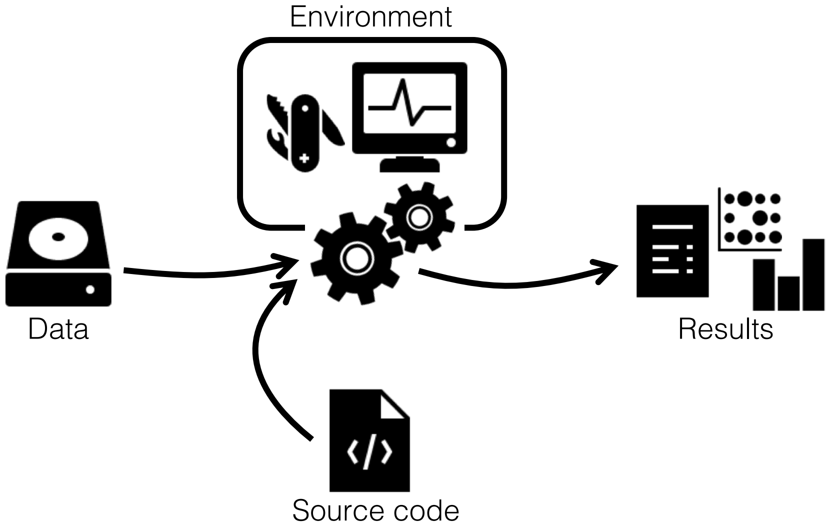
Treasure your data

- Consider your input data static. Keep it read-only!
- Don’t make *different* versions. If you need to preprocess it in any way, script it so you can recreate the steps (see box below).
- Backup! Keep redundant copies in different physical locations.
- Strive towards uploading it to its final destination already at the beginning of a project (e.g. specific repositories such as SRA, GEO, or GenBank, or general repositories such as Dryad or Figshare).

Take control of your research by making it reproducible!

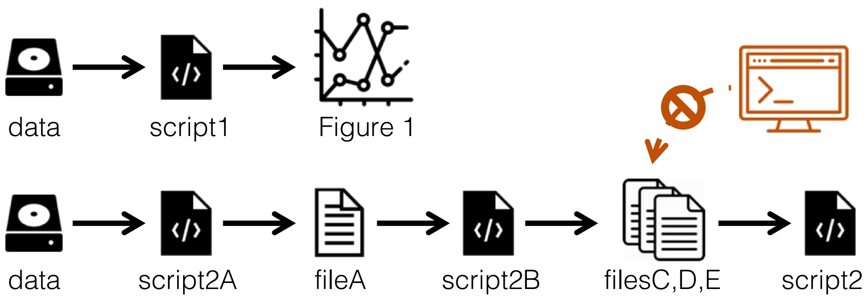
By moving towards a reproducible way of working you will quickly realize that you at the same time make your own life a lot easier! The added effort pays off by gain in control, organization and efficiency.

Below are all the components of a bioinformatics project that have to reproducible.



Organize your coding

- Write scripts/functions/notebooks for specific tasks (connect raw data to final results)
- Keep parameters separate (e.g. top of file, or input arguments)



Avoid generating files interactively on the fly or doing things by hand (no way to track how they were made).

For the advanced

As projects grow, it becomes increasingly difficult to keep track of all the parts and how they fit together. Snakemake is a workflow management system that keeps track of how your files tie together, from raw data and scripts to final figures. If anything changes (script code, parameters, software version, etc) it will know what parts to rerun in order to have up to date and reproducible results.

Snakemake https://snakemake.readthedocs.io/

Connect your results with the code

Rmarkdown and Jupyter notebooks blur the boundaries between code and its output. They allow you to add non-code text (markdown) to your code. This generates a report containing custom formatted text, as well as figures and tables together with the code that generated them.

R Markdown http://rmarkdown.rstudio.com/

jupyter http://jupyter.org/

Master your dependencies

- Full reproducibility requires the possibility to recreate the system that was originally used to generate the results.
- Conda is a package, dependency, and environment manager that makes it easy to install (most) software that you need for your project.
- Your environment can be exported in a simple text format and reinstalled by Conda on another system.

CONDA https://conda.io

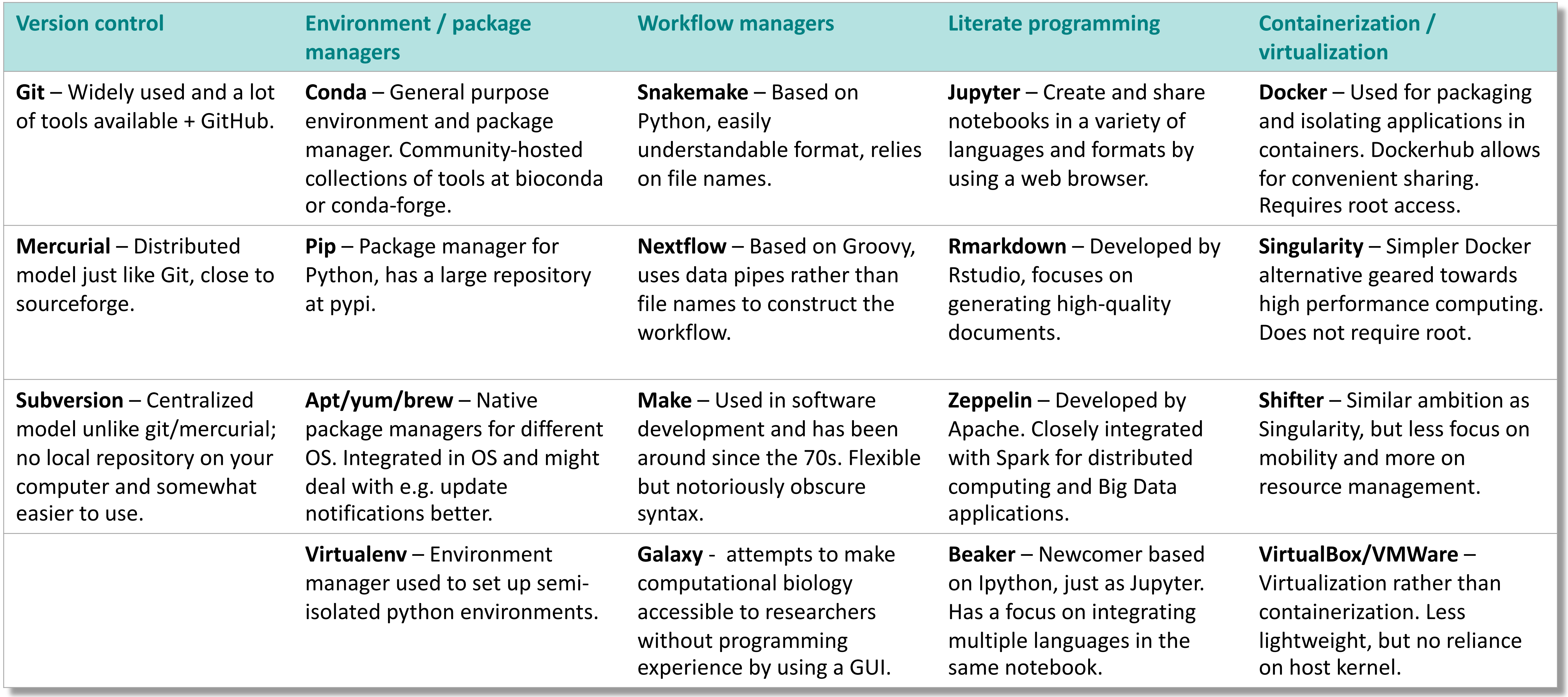
For the advanced

- Conda cannot always *completely* recreate the system, which is required for proper reproducibility.
- A solution is to package your project in an isolated Docker container, together with all its dependencies and libraries.
- A vision is that every new bioinformatics publication is accompanied by a publically available Docker container!
- Singularity is an alternative to Docker which runs better on HPC clusters.

docker https://www.docker.com/

Singularity http://singularity.lbl.gov/

alternatives



Version control	Environment / package managers	Workflow managers	Literate programming	Containerization / virtualization
Git – Widely used and a lot of tools available + GitHub.	**Conda** – General purpose environment and package manager. Community-hosted collections of tools at bioconda or conda-forge.	**Snakemake** – Based on Python, easily understandable format, relies on file names.	**Jupyter** – Create and share notebooks in a variety of languages and formats by using a web browser.	**Docker** – Used for packaging and isolating applications in containers. Dockerhub allows for convenient sharing. Requires root access.
Mercurial – Distributed model just like Git, close to sourceforge.	**Pip** – Package manager for Python, has a large repository at pypi.	**Nextflow** – Based on Groovy, uses data pipes rather than file names to construct the workflow.	**Rmarkdown** – Developed by Rstudio, focuses on generating high-quality documents.	**Singularity** – Simpler Docker alternative geared towards high performance computing. Does not require root.
Subversion – Centralized model unlike git/mercurial; no local repository on your computer and somewhat easier to use.	**Apt/yum/brew** – Native package managers for different OS. Integrated in OS and might deal with e.g. update notifications better.	**Make** – Used in software development and has been around since the 70s. Flexible but notoriously obscure syntax.	**Zeppelin** – Developed by Apache. Closely integrated with Spark for distributed computing and Big Data applications.	**Shifter** – Similar ambition as Singularity, but less focus on mobility and more on resource management.
	Virtualenv – Environment manager used to set up semi-isolated python environments.	**Galaxy** - attempts to make computational biology accessible to researchers without programming experience by using a GUI.	**Beaker** – Newcomer based on Ipython, just as Jupyter. Has a focus on integrating multiple languages in the same notebook.	**VirtualBox/VMWare** – Virtualization rather than containerization. Less lightweight, but no reliance on host kernel.