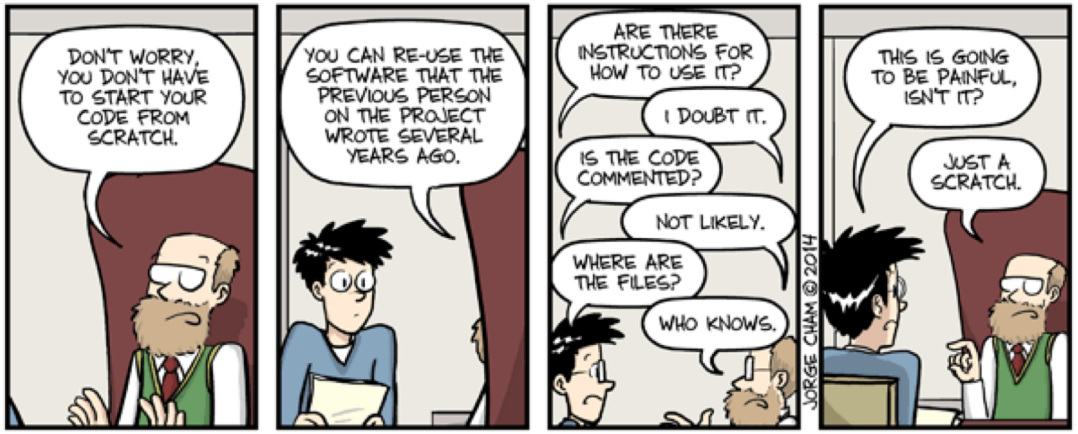
# Tools for Reproducible Research

1 ECTS

Göteborg, November 28-29




"Piled Higher and Deeper" by Jorge Cham

NATIONAL BIOINFORMATICS INFRASTRUCTURE SWEDEN

SciLifeLab

# Teachers

Rasmus

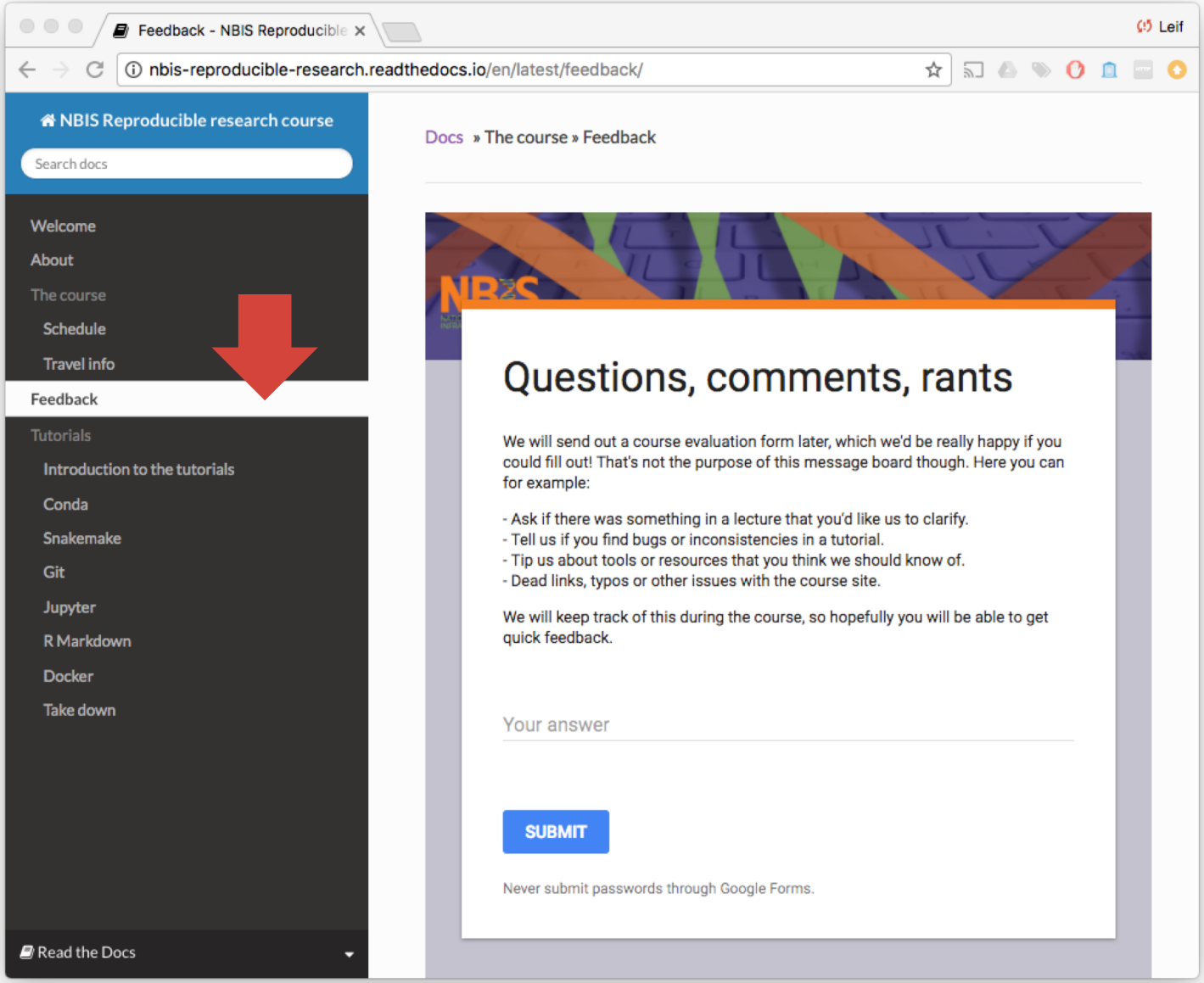
Leif

Viktor

Course content:

- good practices for data analysis and management
- how to use the version control system git to track edits and collaborate on coding
- how to use the package and environment manager Conda
- how to use the workflow manager Snakemake
- how to use R Markdown to generate automated reports
- how to use Jupyter notebooks to document your ongoing analysis
- how to use Docker to distribute containerized computational environments

NBIS
NATIONAL BIOINFORMATICS INFRASTRUCTURE SWEDEN




http://nbis-reproducible-research.readthedocs.io

# Introduction to Reproducible Research

Why all the talk about

# reproducible research?

RESEARCH ARTICLE

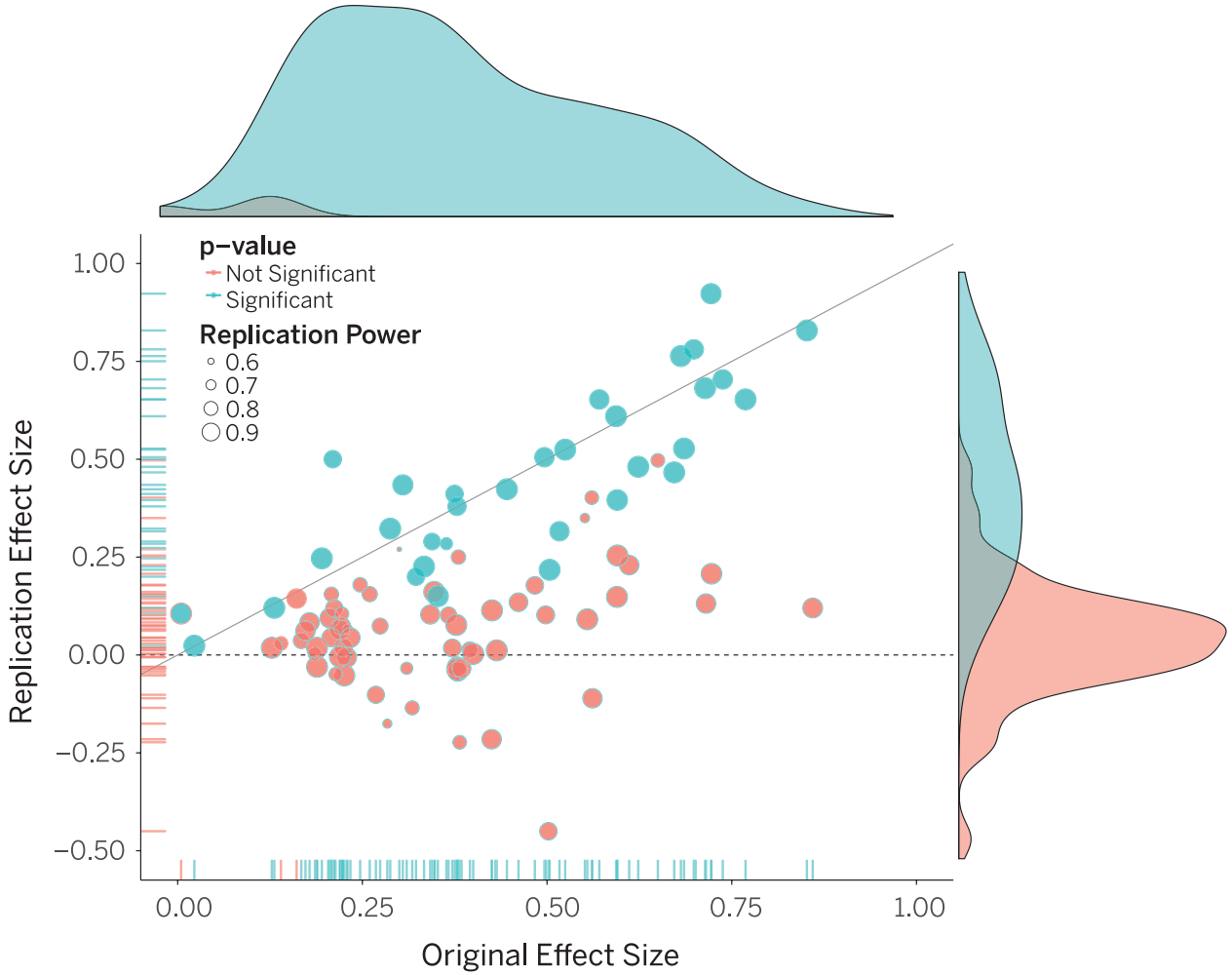
## Estimating the reproducibility of psychological science

Open Science Collaboration*,†
+ See all authors and affiliations

*Science* 28 Aug 2015:
Vol. 349, Issue 6251, aac4716
DOI: 10.1126/science.aac4716

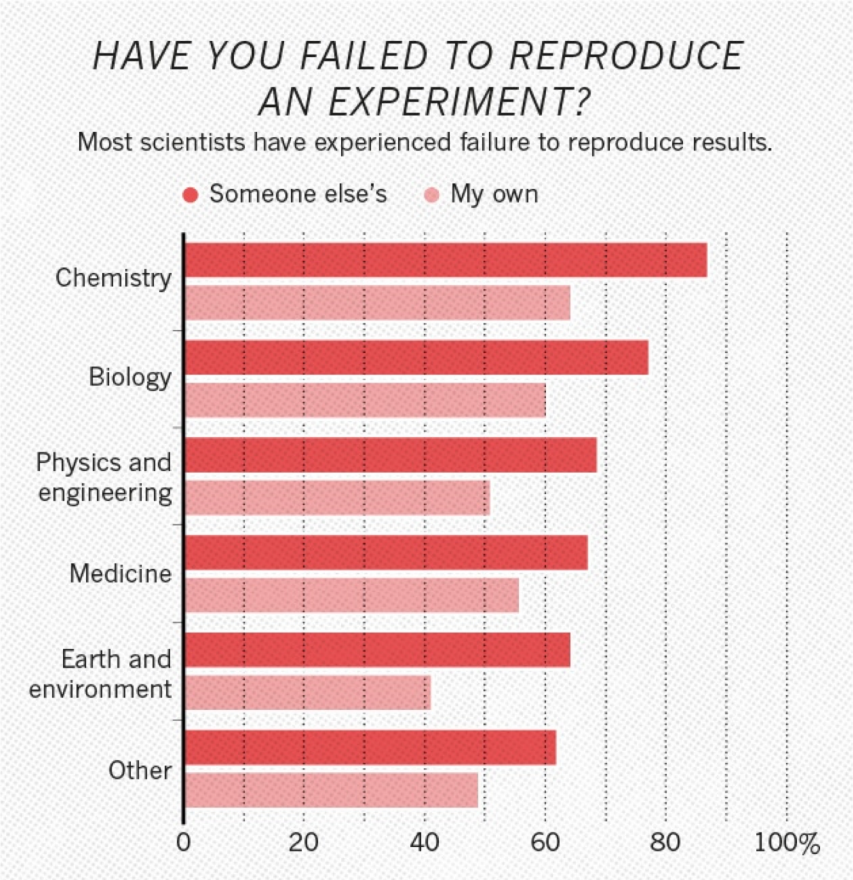
The *Reproducibility project* set out to replicate 100 experiments published in high-impact psychology journals.

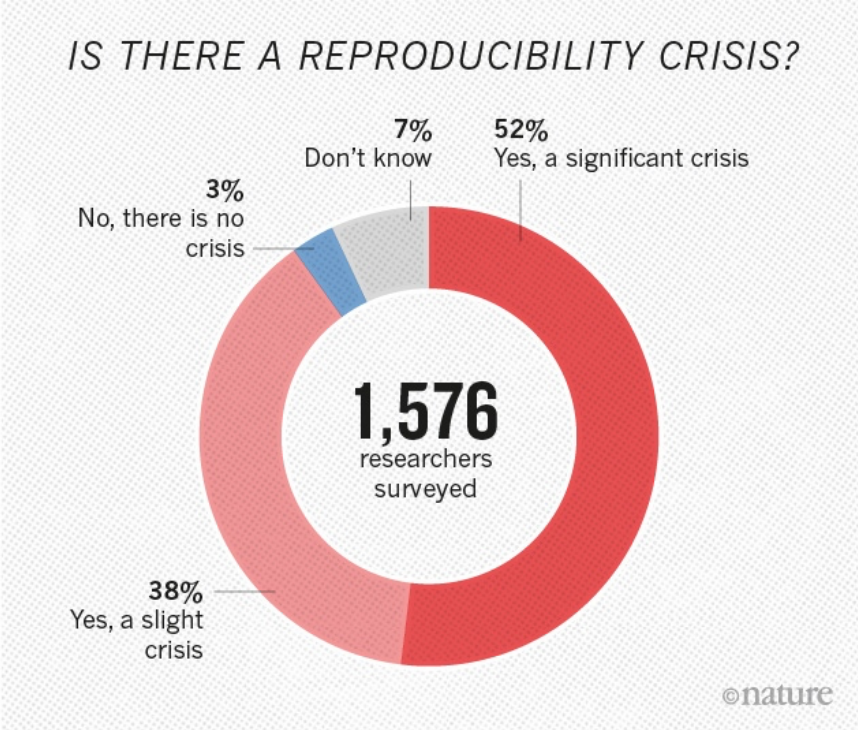
About one-half to two-thirds of the original findings could not be observed in the replication study.



Why all the talk about

# reproducible research?




A survey in Nature revealed that irreproducible experiments are a problem across all domains of science[1].

Medicine is among the most affected research fields. A study in Nature found that 47 out of 53 medical research papers focused on cancer research were irreproducible[2].

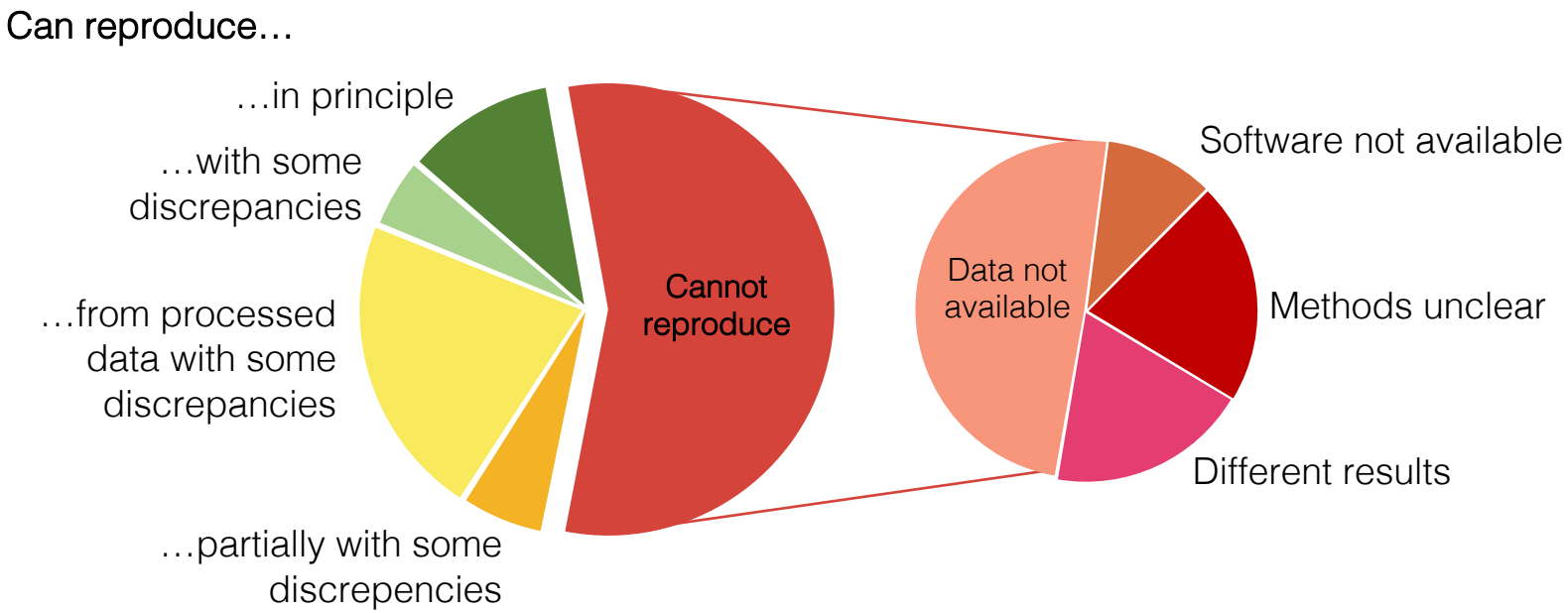
Common features were failure to show all the data and inappropriate use of statistical tests.

[1] "1,500 scientists lift the lid on reproducibility". Nature. 533: 452–454
[2] Begley, C. G.; Ellis, L. M. (2012). "Drug development: Raise standards for preclinical cancer research". Nature. 483 (7391): 531–533.

Why all the talk about

# reproducible research?

Replication of data analyses in 18 articles on microarray-based gene expression profiling published in Nature Genetics in 2005–2006:



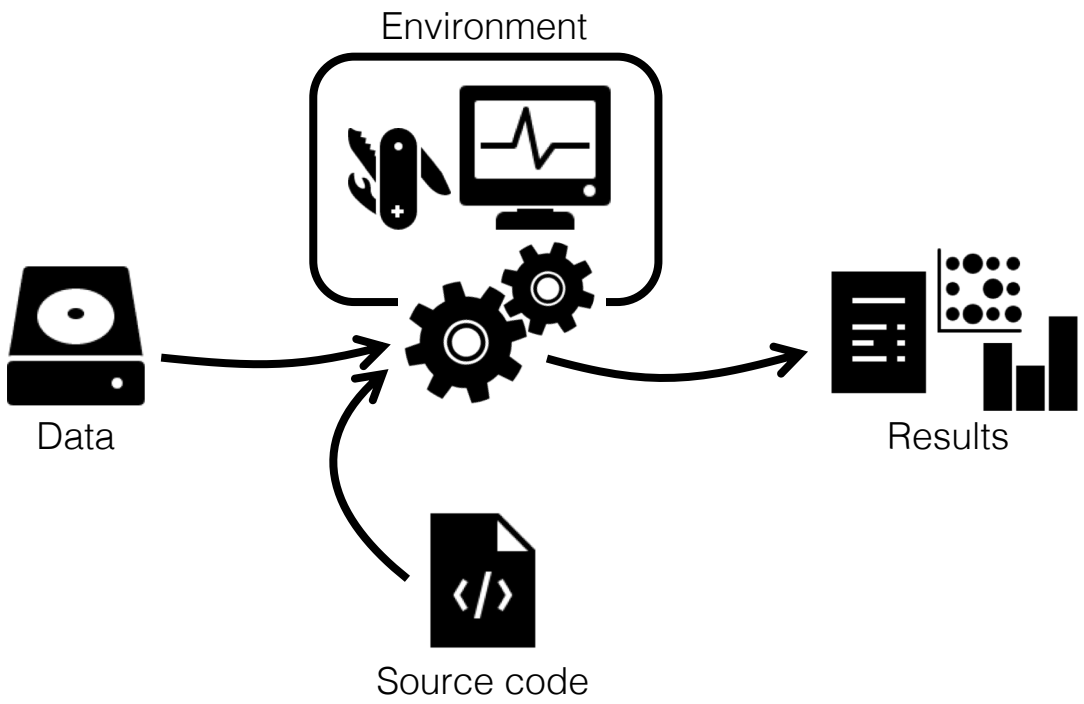

Summary of the efforts to replicate the published analyses.
Adopted from: Ioannidis et al. Repeatability of published microarray gene expression analyses. *Nature Genetics* **41** (2009) doi:10.1038/ng.295

What do we mean with

# reproducible research?

| | | Data | |
|---|---|---|---|
| | | Same | Different |
| Code | Same | Reproducible | Replicable |
| | Different | Robust | Generalisable |

All parts of a bioinformatics analysis have to be reproducible:




”The foundations of knowledge should be constituted by experimentally produced facts, which can be made believable to a scientific community by their reproducibility."

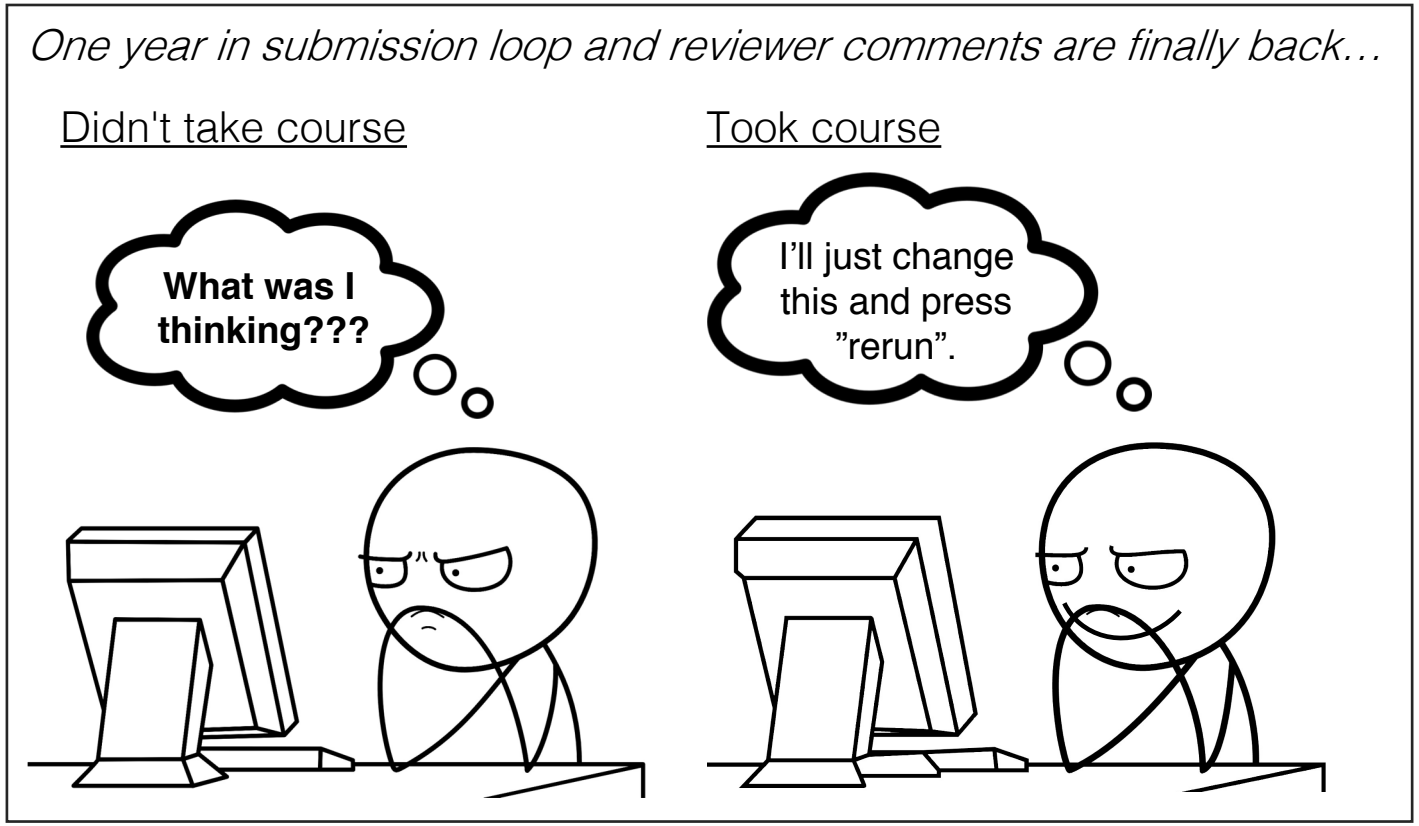
- Robert Boyle, 1627-1691

# Where does your latest publication fit?

| Decent | Getting there… | Well done! |
|---|---|---|
| • Data available on request.<br>• All meta data required for generating the results available. | • Data deposited in public repositories.<br>• Raw data available in unedited form.<br>• If the raw data needed preprocessing, scripts were used rather than modifying it manually. | • Section in the paper to aid in reproduction.<br>• Used non-proprietary and machine-readable formats, e.g. .csv rather than .xls. |
| • All code for generating results from processed data available on request. | • All code for generating results from raw data is available.<br>• The code is publically available with timestamps/tags. | • All code for generating results from *publically available* raw data is available.<br>• Code is documented and contains instructions for reproducing results.<br>• Seeds were used and documented for heuristic methods. |
| • Key programs used are mentioned in the methods section. | • List of all programs used, and their respective versions, available. | • Instructions for reproducing the environment publically available. |

Read more: Wilson et al. (2017) Good enough practices in scientific computing. PLoS Comput Biol 13(6)

# What's in it for me?




*"It takes some effort to organize your research to be reproducible. We found that although the effort seems to be directed to helping other people stand up on your shoulders, the principal beneficiary is generally the author herself. This is because time turns each one of us into another person, and by making effort to communicate with strangers, we help ourselves to communicate with our future selves."*

Schwab et al. Making scientific computations reproducible. Computing in Science Engineering (2000).

Before project

- Improved structure and organization.
- Forced to think about scope and limitations.

During project

- Easier to rerun analyses and generate results after updating data, tools, parameters, etc.
- Closer interaction between collaborators.
- Much of the manuscript "writes itself".

After project

- Faster resumption of research by others (or your future self), thereby increasing the impact of your work.
- Increased visibility in the scientific community.

# Data management

# Data (mis)management in practice

| | Data acquisition | Analysis | First submission | Review | Second submission | Publication |
|---|---|---|---|---|---|---|
| **Raw data** | Data arrives in cumbersome and proprietary format. | Gets converted to format of choice. Original files (and conversion settings) are lost. | | Leads a quiet life on the HPC cluster, until the project expires and the data has to be urgently retrieved. | Ends its days on an external hard drive on the researcher's desk. | "Data available upon request". |
| **Meta data** | In researcher's lab journal. | Hard-coded in various analysis scripts. | Mailed back and forth between collaborators in ever-changing (but nicely colored) Excel sheets. | | Reformatted and included as PDF in the supplementary. | |

# FAIR

Strive to make your data **FAIR** – Findable, Accessible, Interoperable, and Reusable *for both machines and humans*.

## **Box 2** | The FAIR Guiding Principles

**To be Findable:**
F1. (meta)data are assigned a globally unique and persistent identifier
F2. data are described with rich metadata (defined by R1 below)
F3. metadata clearly and explicitly include the identifier of the data it describes
F4. (meta)data are registered or indexed in a searchable resource

**To be Accessible:**
A1. (meta)data are retrievable by their identifier using a standardized communications protocol
A1.1 the protocol is open, free, and universally implementable
A1.2 the protocol allows for an authentication and authorization procedure, where necessary
A2. metadata are accessible, even when the data are no longer available

**To be Interoperable:**
I1. (meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation.
I2. (meta)data use vocabularies that follow FAIR principles
I3. (meta)data include qualified references to other (meta)data

**To be Reusable:**
R1. meta(data) are richly described with a plurality of accurate and relevant attributes
R1.1. (meta)data are released with a clear and accessible data usage license
R1.2. (meta)data are associated with detailed provenance
R1.3. (meta)data meet domain-relevant community standards

www.nature.com/scientificdata

SCIENTIFIC DATA

OPEN

SUBJECT CATEGORIES
» Research data
» Publication characteristics

Received: 10 December 2015
Accepted: 12 February 2016
Published: 15 March 2016

## Comment: The FAIR Guiding Principles for scientific data management and stewardship

Mark D. Wilkinson *et al.*#

**There is an urgent need to improve the infrastructure supporting the reuse of scholarly data. A diverse set of stakeholders—representing academia, industry, funding agencies, and scholarly publishers—have come together to design and jointly endorse a concise and measureable set of principles that we refer to as the FAIR Data Principles. The intent is that these may act as a guideline for those wishing to enhance the reusability of their data holdings. Distinct from peer initiatives that focus on the human scholar, the FAIR Principles put specific emphasis on enhancing the ability of machines to automatically find and use the data, in addition to supporting its reuse by individuals. This Comment is the first formal publication of the FAIR Principles, and includes the rationale behind them, and some exemplar implementations in the community.**

**Supporting discovery through good data management**
Good data management is not a goal in itself, but rather is the key conduit leading to knowledge discovery and innovation, and to subsequent data and knowledge integration and reuse by the community after the data publication process. Unfortunately, the existing digital ecosystem surrounding scholarly data publication prevents us from extracting maximum benefit from our research investments (e.g., ref. 1). Partially in response to this, science funders, publishers and governmental agencies are beginning to require data management and stewardship plans for data generated in publicly funded experiments. Beyond proper collection, annotation, and archival, data stewardship includes the notion of 'long-term care' of valuable digital assets, with the goal that they should be discovered and re-used for downstream investigations, either alone, or in combination with newly generated data. The outcomes from good data management and stewardship, therefore, are high quality digital publications that facilitate and simplify this ongoing process of discovery, evaluation, and reuse in downstream studies. What constitutes 'good data management' is, however, largely undefined, and is generally left as a decision for the data or repository owner. Therefore, bringing some clarity around the goals and desiderata of good data management and stewardship, and defining simple guideposts to inform those who publish and/or preserve scholarly data, would be of great utility.

This article describes four foundational principles—Findability, Accessibility, Interoperability, and Reusability—that serve to guide data producers and publishers as they navigate around these obstacles, thereby helping to maximize the added-value gained by contemporary, formal scholarly digital publishing. Importantly, it is our intent that the principles apply not only to 'data' in the conventional sense, but also to the algorithms, tools, and workflows that led to that data. All scholarly digital research objects[2]—from data to analytical pipelines—benefit from application of these principles, since all components of the research process must be available to ensure transparency, reproducibility, and reusability.

There are numerous and diverse stakeholders who stand to benefit from overcoming these obstacles: researchers wanting to share, get credit, and reuse each other's data and interpretations; professional data publishers offering their services; software and tool-builders providing data analysis and processing services such as reusable workflows; funding agencies (private and public) increasingly

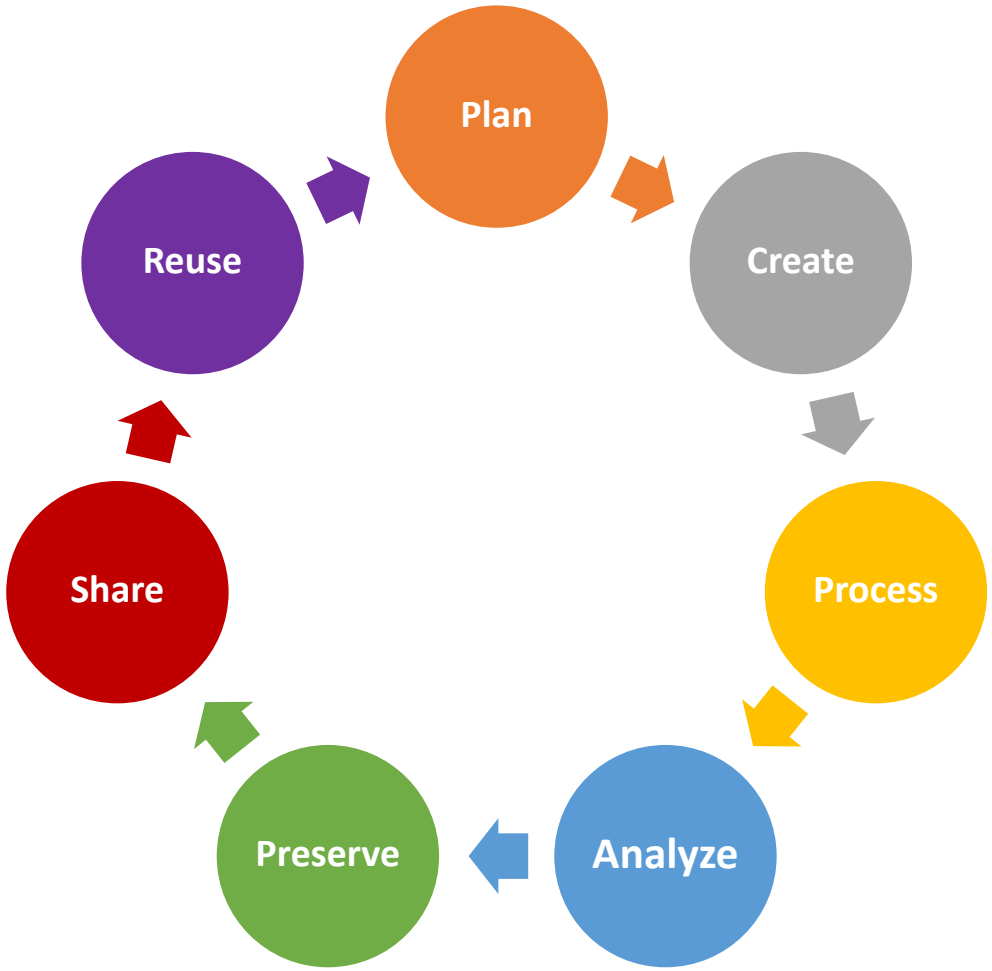
Correspondence and requests for materials should be addressed to B.M. (email: barend.mons@dtls.nl).
#A full list of authors and their affiliations appears at the end of the paper.



Wilkinson, Mark et al. "The FAIR Guiding Principles for scientific data management and stewardship". Scientific Data 3, 160018 (2016) doi:10.1038/sdata.2016.18

# Data management plan

- Check the requirements of your funding agency and field of research.
- List the types of data that you expect to produce.
- Decide what data require archiving, and determine how much storage space you will need (short and long term).
- Provide metadata that allows others to understand, cite and reuse your data files.
- Make clear how and when your data can be shared with scientists outside your group.
- If your research involves sensitive data, explain any legal and ethical restrictions on data access and reuse.
- Look for suitable data repositories used by your research community.
- Check what data format and structure the chosen repository might request.




Life cycle for scientific data

Adapted from Nature 555, 403-405 (2018), doi: 10.1038/d41586-018-03071-1

# Pair up and discuss!

- Does your group have a data management plan in place?
- Do you know "your" repositories and how to submit data to them?

# Data acquisition and deposit

- Find the right repository for your data, and strive towards uploading data to its final destination already at the beginning of a project.
- Structure metadata in the format needed by the repository already as the experiments are being performed.
- Stick to non-proprietary and widely used file formats.

- Only 12% of articles from NIH funded research mention data deposited in international repositories
- Estimated 200000+ “invisible” data sets / year

Read et al. (2015) PLoS ONE 10(7) doi:10.1371/journal.pone.0132735

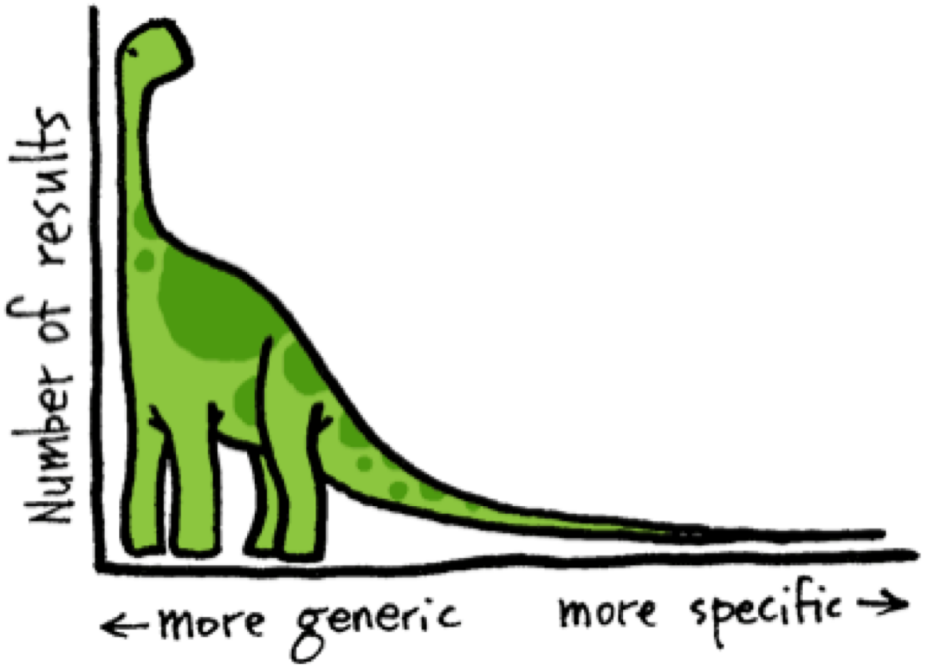
---

Scientific Data (Springer Nature) maintains a list of recommended repositories at www.nature.com/sdata/policies/repositories.

Dedicated repositories:
e.g. SRA, GEO, GenBank, UniProt etc.

Generalist ("long-tail data") repositories:
Research data that doesn’t fit in structured data repositories, e.g. Data Dryad, Figshare, Zenodo.

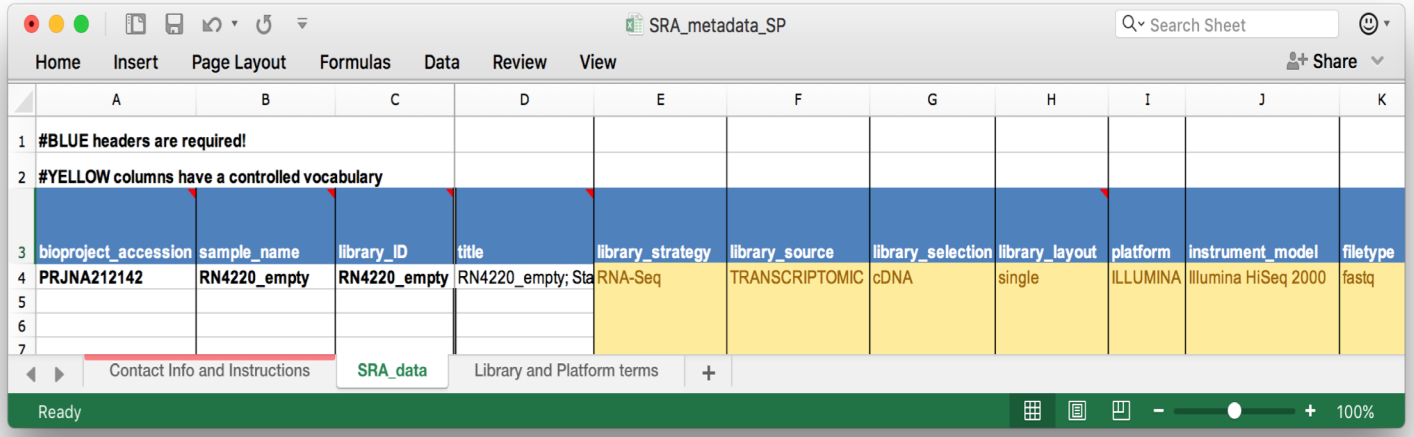
Each dataset can be assigned a Digital Object Identifier (doi); a persistent identifier used to uniquely identify objects.



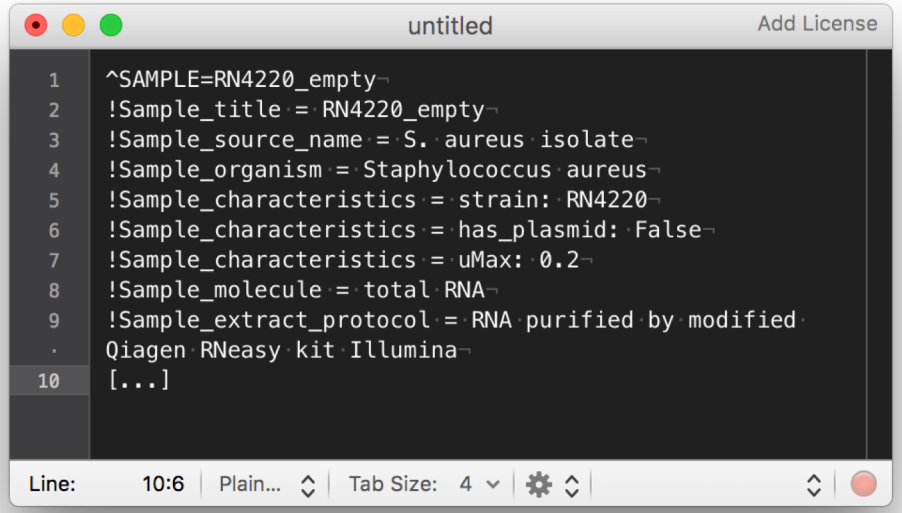
# Data acquisition and deposit

- Find the right repository for your data, and strive towards uploading data to its final destination already at the beginning of a project.
- Structure metadata in the format needed by the repository already as the experiments are being performed.
- Stick to non-proprietary and widely used file formats.

---




SRA (Sequence Read Archive) uses a template Excel sheet for metadata.




GEO (Gene Expression Omnibus) uses text files in SOFT format.

# Data acquisition and deposit

- Find the right repository for your data, and strive towards uploading data to its final destination already at the beginning of a project.
- Structure metadata in the format needed by the repository already as the experiments are being performed.
- Stick to non-proprietary and widely used file formats.

---

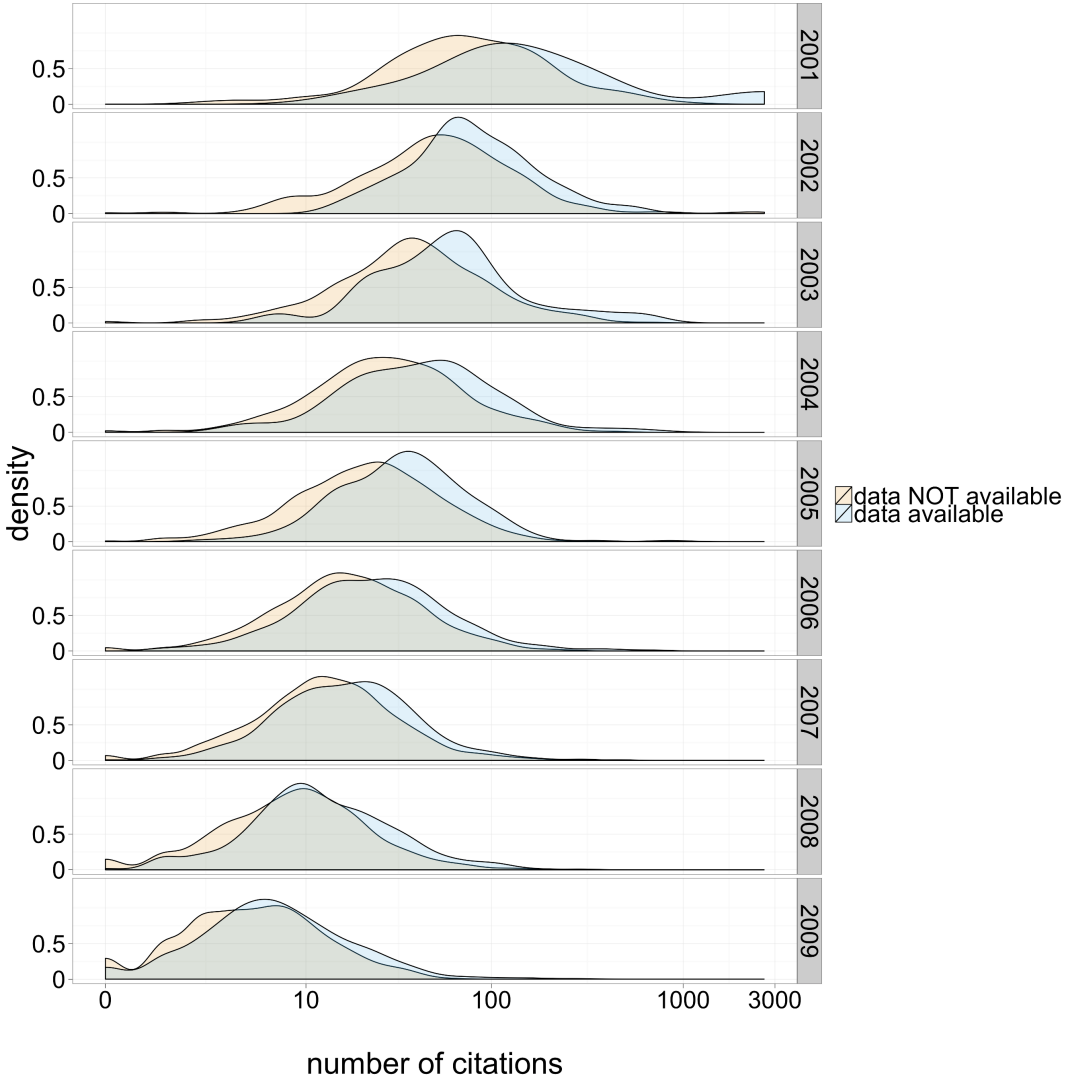
| - | + |
|---|---|
| Binary | Text-based |
| Proprietary | Open |
| New kid on the block | Old as the hills |
| Compressed/encrypted | Uncompressed/unencrypted |
| Platform dependent | Interoperable |
| Complex | Simple |

| | 🤢 | 🙄 | 😀 |
|---|---|---|---|
| Raster graphic | wmf, psd | bmp, gif | tiff, png, jpeg |
| Vector graphic | ai, eps | pdf | svg |
| Document | doc | docx, tex | odt, utf-8, md |
| Archive | rar | 7z | zip, tar, gz |
| Tabular data | xls, rds, mat | xlsx, ods | csv |

# Data sharing

From 10,555 studies with gene expression microarray data:

- Studies that shared data received 9% more citations (after accounting for other covariates).
- Data reuse by other researchers continued for >6 years.
- A very conservative estimate found that 20% of the datasets deposited between 2003 and 2007 had been reused at least once by third parties.




Piwowar and Vision (2013), Data reuse and the open data citation advantage, PeerJ 1:e175, doi:10.7717/peerj.175

# Data sharing – Open access

- Democracy and transparency
  - Publicly funded research data should be accessible to all free of charge.
  - Published results and conclusions should be possible to check by others.
- Research
  - Enables others to combine data, address new questions, and develop new analytical methods.
  - Reduce duplication and waste.
- Innovation and utilization outside research
  - Public authorities, companies, and individuals outside academia can make use of the data.
- Citation
  - Citation of data will be a merit for the researcher that produced it.

# Data sharing – Ontologies

lauroyl-CoA

dodecanoyl-CoA

C12:0-CoA

lauroyl coenzyme A

coenzyme A, S-dodecanoate

dodecanoyl coenzyme A

C12:0 coenzyme A

dodecanoic acid coenzyme A

lauroylic acid CoA

Dodecanethioic acid, S-ester with coenzyme A

Coenzyme A, S-laurate (7CI,8CI)

12:0, lauroyl-CoA

1-undecanecarboxylic acid CoA

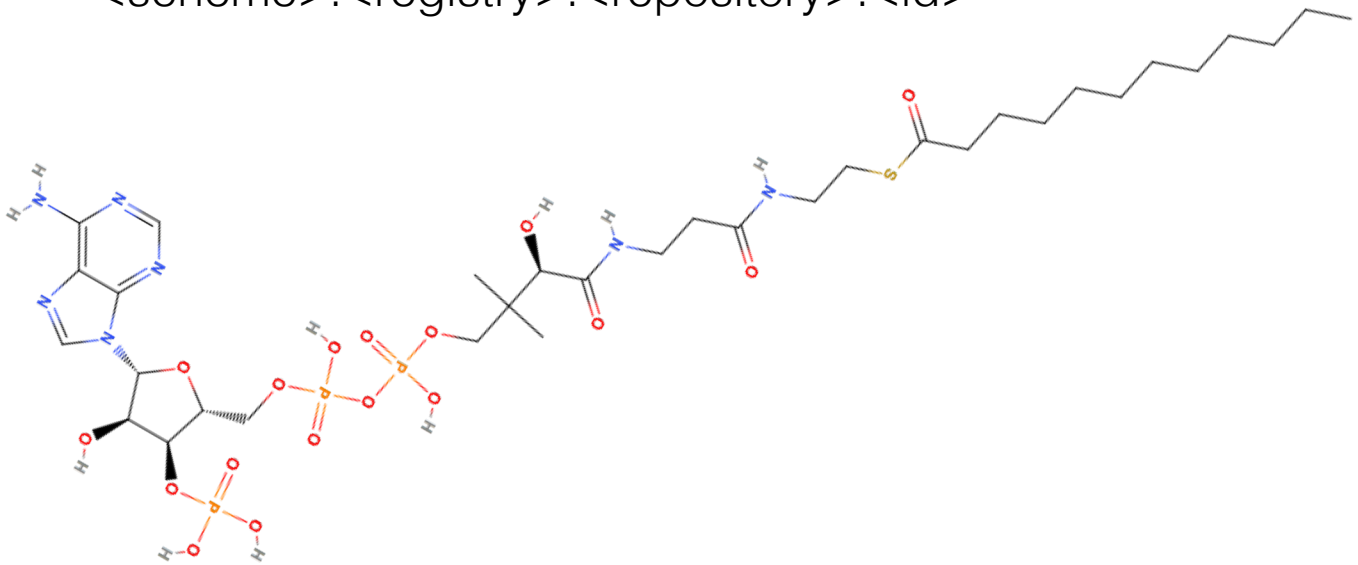
vulvic acid CoA

3'-phosphoadenosine 5'-(3-{(3R)-4-[(3-[[2-(dodecanoylsulfanyl)ethyl]amino]-3-oxopropyl)amino]-3-hydroxy-2,2-dimethyl-4-oxobutyl} dihydrogen diphosphate)

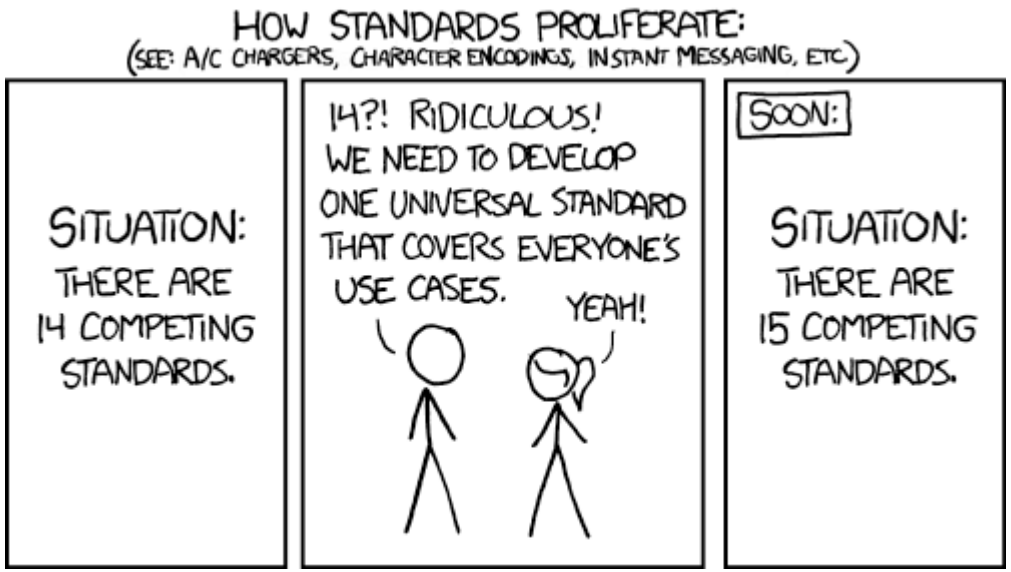
## Who am I?

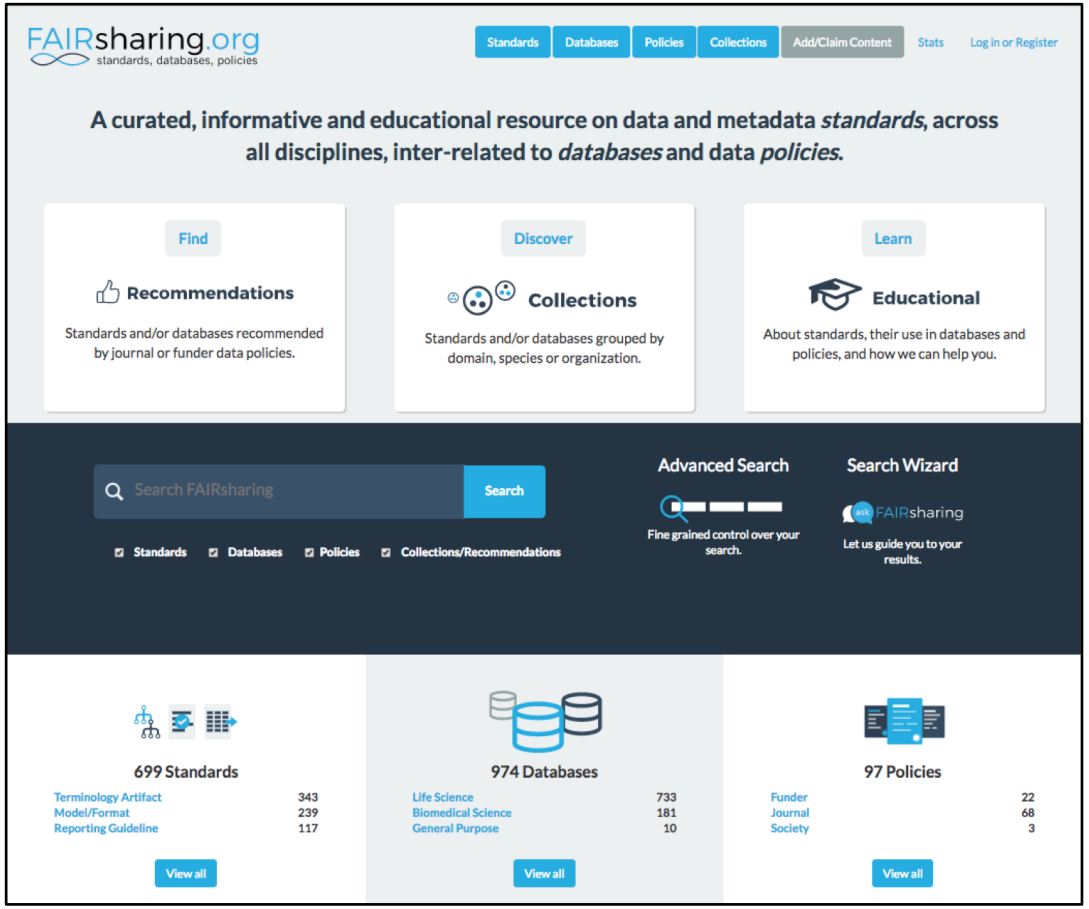
**urn.miriam.chebi:15521** of course!

<scheme>.<registry>.<repository>:<id>



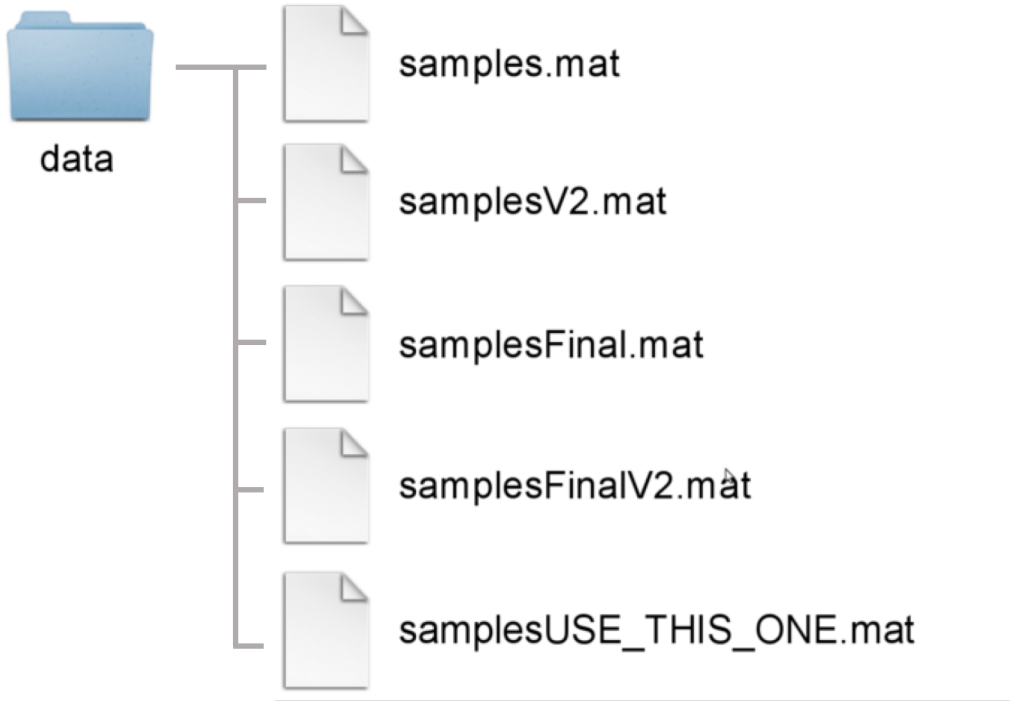
# Data sharing – Ontologies

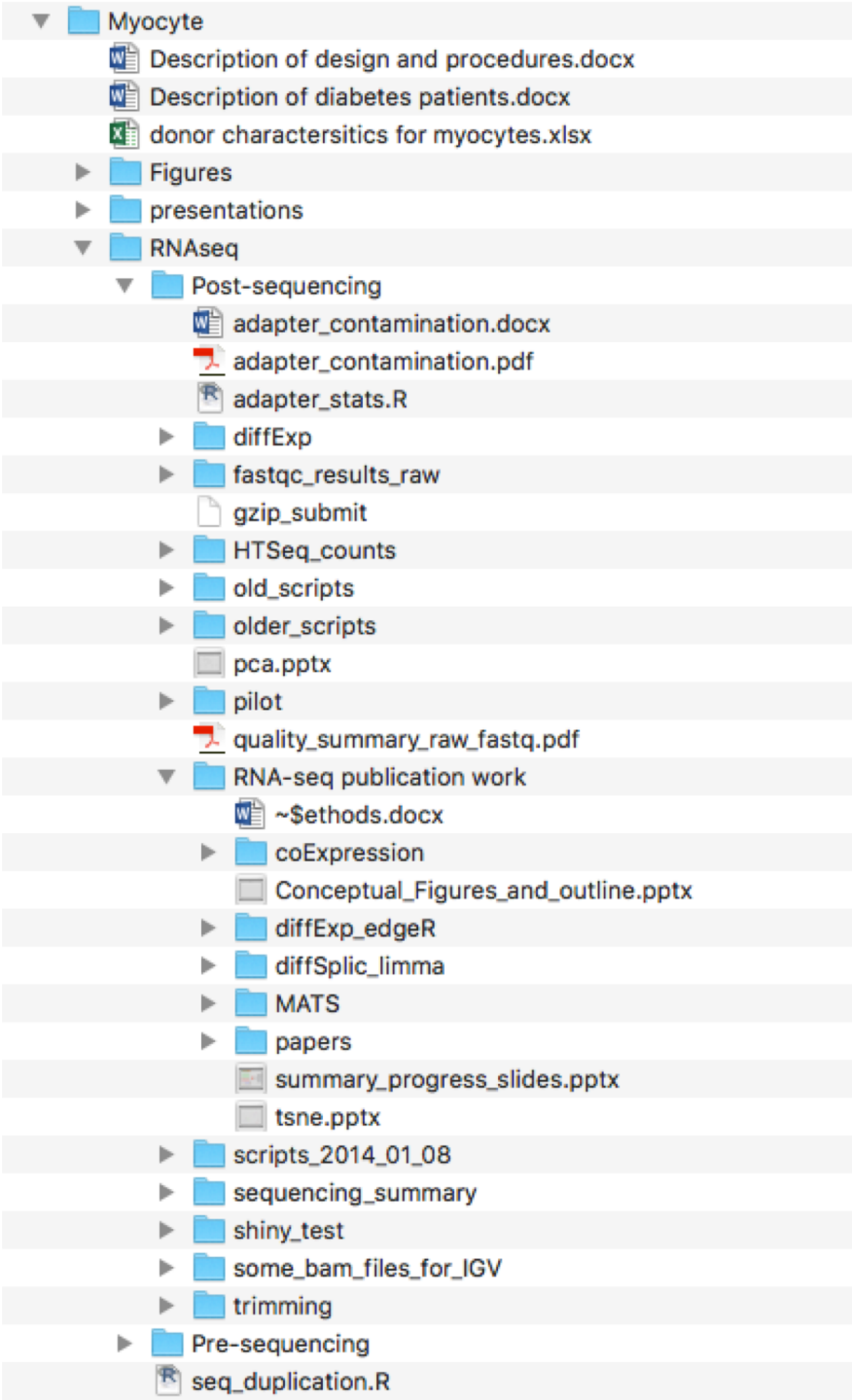


# Project organization

# The project directory




The first step towards working reproducible: Get organized!

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.

# Pair up and discuss!

- Do you organize your work in distinct projects?
- How do you organize your files in this context?
- Are you happy with the way you work today?

# The project directory

```
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/           notebooks that document your day-to-day work
|
|- 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/
|
|- 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
```

Working in projects

RStudio

ATOM

Sublime Text

git

Noble WS (2009) A Quick Guide to Organizing Computational Biology Projects. PLoS Comput Biol 5(7): e1000424.
http://journals.plos.org/ploscompbiol/article?id=info:doi/10.1371/journal.pcbi.1000424

https://github.com/NBISweden/project_template

# File naming system

## Machine readable

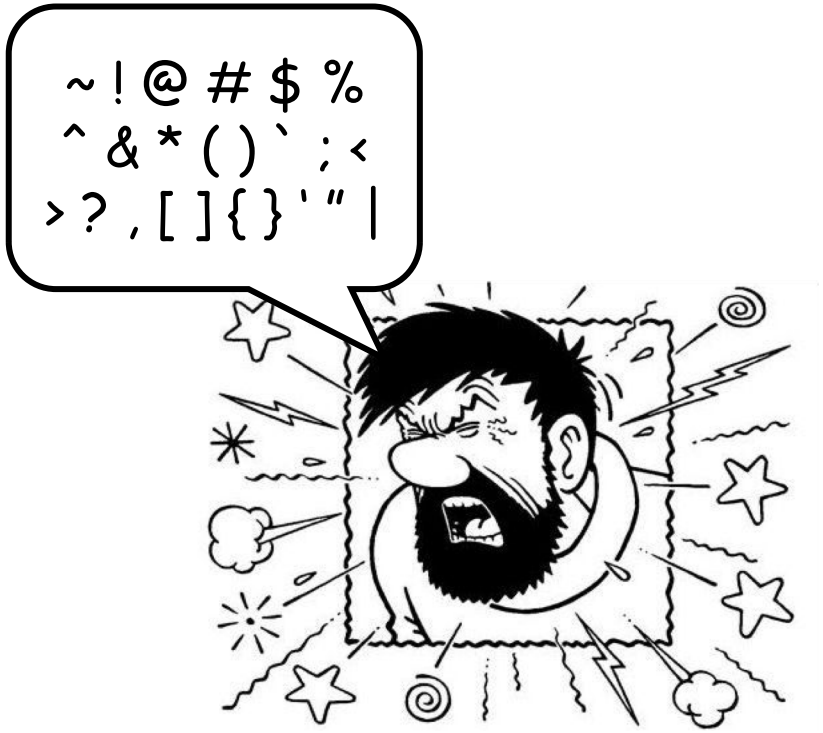
- Avoid special characters, e.g.: `~!@#$%^&*()` `;<>?,[]{}‘”|`
- Avoid spaces, alternatives:
  - `file_name.txt`
  - `file-name.txt`
  - `filename.txt`
  - `FileName.txt`

## Human readable

- Know the content of a file without opening it, e.g.: `SRR1234.hg19.sorted.trimmed.bam`

## Control file ordering

- Use dates if appropriate
- Use 01, 02, rather than 1, 2




Bad examples:

- reproducible%20research.pptx
- suppl fig 10.png
- Supplementary Figure 9.png

Good examples:

- 2018-11-28_Gothenburg_Reproducible_research.pptx
- suppl_fig_09_barplot_alignment_stats.png
- suppl_fig_10_samples_PCA_count_data.png

# A project in Atom

Dockerfile — ~/Documents/projects/project_template

Project | Remote

- project_template
  - code
  - data
  - doc
  - intermediate
  - logs
  - notebooks
  - results
  - scratch
  - .ftpconfig
  - .gitignore
  - config.yml
  - Dockerfile
  - environment.yml
  - README.md
  - Snakefile

Dockerfile

```
FROM ubuntu:16.04
LABEL description = "Lightweight image with Conda, Jupyter Notebook and Snakemake"

# Install Miniconda3 and prerequisites
RUN apt-get update && \
    apt-get install -y --no-install-recommends bzip2 curl ca-certificates
RUN curl https://repo.continuum.io/miniconda/Miniconda3-4.5.11-Linux-x86_64.sh -O && \
    bash Miniconda3-4.5.11-Linux-x86_64.sh -bf -p /opt/miniconda3/ && \
    rm Miniconda3-4.5.11-Linux-x86_64.sh

# Add Conda to PATH
ENV PATH="/opt/miniconda3/bin:${PATH}"

# Use bash as shell
SHELL ["/bin/bash", "-c"]

# Set up the Conda environment
COPY environment.yml .
RUN conda env update -n root -f environment.yml && \
    conda clean --all

# Install Jupyter Notebook and set default user to UID 1000
RUN pip install --no-cache-dir notebook==5.*
ENV NB_USER nbuser
ENV NB_UID 1000

RUN adduser --disabled-password --no-create-home \
    --gecos "Default user" \
    --uid ${NB_UID} \
```

```
liv433l:project_template arasmus$ pwd
/Users/arasmus/Documents/projects/project_template
liv433l:project_template arasmus$
```

Dockerfile 22:46 | LF UTF-8 Dockerfile | git+ master | Fetch | 1 file

Syntax highlighting, indentation, and autocomplete

# A project in Atom

Dockerfile — ~/Documents/projects/project_template

Project | Remote

- project_template
  - code
  - data
  - doc
  - intermediate
  - logs
  - notebooks
  - results
  - scratch
  - .ftpconfig
  - .gitignore
  - config.yml
  - Dockerfile
  - environment.yml
  - README.md
  - Snakefile

Dockerfile

```
FROM ubuntu:16.04
LABEL description = "Lightweight image with Conda, Jupyter Notebook and Snakemake"

# Install Miniconda3 and prerequisites
RUN apt-get update && \
    apt-get install -y --no-install-recommends bzip2 curl ca-certificates
RUN curl https://repo.continuum.io/miniconda/Miniconda3-4.5.11-Linux-x86_64.sh -O && \
    bash Miniconda3-4.5.11-Linux-x86_64.sh -bf -p /opt/miniconda3/ && \
    rm Miniconda3-4.5.11-Linux-x86_64.sh

# Add Conda to PATH
ENV PATH="/opt/miniconda3/bin:${PATH}"

# Use bash as shell
SHELL ["/bin/bash", "-c"]

# Set up the Conda environment
COPY environment.yml .
RUN conda env update -n root -f environment.yml && \
    conda clean --all

# Install Jupyter Notebook and set default user to UID 1000
RUN pip install --no-cache-dir notebook==5.*
ENV NB_USER nbuser
ENV NB_UID 1000

RUN adduser --disabled-password --no-create-home \
    --gecos "Default user" \
    --uid ${NB_UID} \
```

```
liv433l:project_template arasmus$ pwd
/Users/arasmus/Documents/projects/project_template
liv433l:project_template arasmus$
```

Dockerfile 22:46 | LF UTF-8 Dockerfile git+ master Fetch 1 file

Integrated version control with Git

# A project in Atom

Dockerfile — ~/Documents/projects/project_template

Project | Remote

- project_template
  - code
  - data
  - doc
  - intermediate
  - logs
  - notebooks
  - results
  - scratch
  - .ftpconfig
  - .gitignore
  - config.yml
  - Dockerfile
  - environment.yml
  - README.md
  - Snakefile

Dockerfile

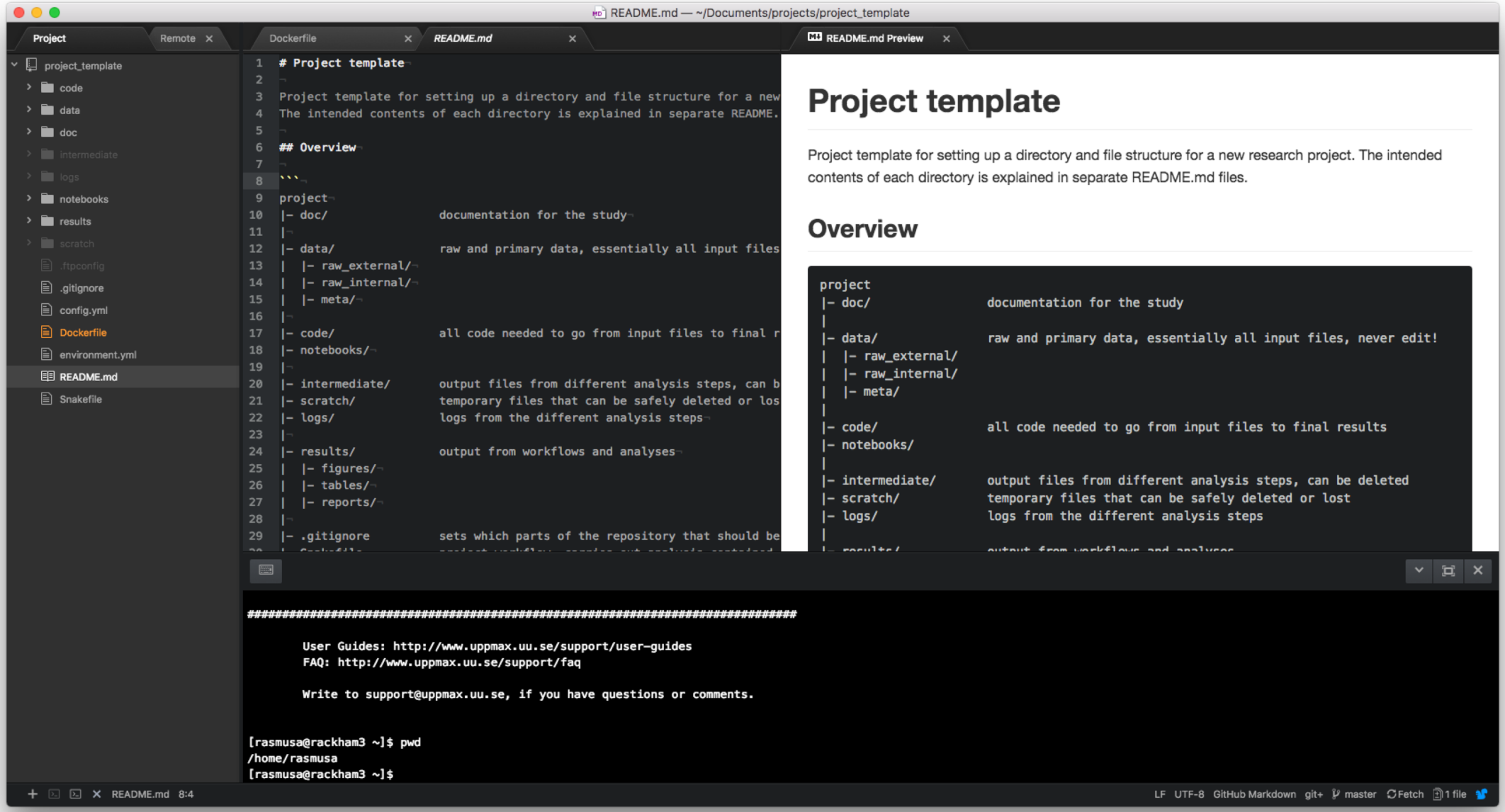
```
FROM ubuntu:16.04
LABEL description = "Lightweight image with Conda, Jupyter Notebook and Snakemake"

# Install Miniconda3 and prerequisites
RUN apt-get update && \
    apt-get install -y --no-install-recommends bzip2 curl ca-certificates
RUN curl https://repo.continuum.io/miniconda/Miniconda3-4.5.11-Linux-x86_64.sh -O && \
    bash Miniconda3-4.5.11-Linux-x86_64.sh -bf -p /opt/miniconda3/ && \
    rm Miniconda3-4.5.11-Linux-x86_64.sh

# Add Conda to PATH
ENV PATH="/opt/miniconda3/bin:${PATH}"

# Use bash as shell
SHELL ["/bin/bash", "-c"]

# Set up the Conda environment
COPY environment.yml .
RUN conda env update -n root -f environment.yml && \
    conda clean --all

# Install Jupyter Notebook and set default user to UID 1000
RUN pip install --no-cache-dir notebook==5.*
ENV NB_USER nbuser
ENV NB_UID 1000

RUN adduser --disabled-password --no-create-home \
    --gecos "Default user" \
    --uid ${NB_UID} \
```

```
##############################################################################

        User Guides: http://www.uppmax.uu.se/support/user-guides
        FAQ: http://www.uppmax.uu.se/support/faq

        Write to support@uppmax.uu.se, if you have questions or comments.


[rasmusa@rackham3 ~]$ pwd
/home/rasmusa
[rasmusa@rackham3 ~]$
```

Dockerfile 22:46 | LF UTF-8 Dockerfile git+ master Fetch 1 file

Automatically sync files between local/remote

# A project in Atom




Tons of plugins, e.g. for viewing different file formats

# A project in RStudio

~/Work/Teaching/Reproducible_research/project_template - master - RStudio

Go to file/function | Addins | project_template

**Files** | Git

New Folder | Delete | Rename | More

Home > Work > Teaching > Reproducible_research > project_template

| Name | Size | Modified |
|---|---|---|
| .. | | |
| .gitignore | 256 B | Nov 23, 2018, 9:14 AM |
| code | | |
| config.yml | 20 B | Mar 19, 2018, 10:33 AM |
| data | | |
| doc | | |
| Dockerfile | 818 B | Mar 20, 2018, 1:23 PM |
| environment.yml | 61 B | Mar 19, 2018, 10:33 AM |
| intermediate | | |
| logs | | |
| notebooks | | |
| README.md | 1.2 KB | Mar 20, 2018, 1:23 PM |
| results | | |
| scratch | | |
| Snakefile | 262 B | Mar 19, 2018, 10:33 AM |
| README.html | 213.5 KB | Nov 23, 2018, 9:16 AM |

README.md | Preview | Insert | Run

```
# Project template

Project template for setting up a directory and file structure for a new research project.
The intended contents of each directory is explained in separate README.md files.

##Overview

```
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
```
```

1:1 | Markdown

Console | Terminal | R Markdown

Terminal 1 | ~/Work/Teaching/Reproducible_research/project_template

```
[base][master] >> ll
total 48
drwxr-xr-x   4 varemo  NET\Domain Users   128B Nov 23 09:11 .Rproj.user/
drwxr-xr-x  15 varemo  NET\Domain Users   480B Nov 23 09:14 .git/
-rw-r--r--   1 varemo  NET\Domain Users   256B Nov 23 09:14 .gitignore
-rw-r--r--   1 varemo  NET\Domain Users   818B Mar 20  2018 Dockerfile
-rw-r--r--   1 varemo  NET\Domain Users   1.2K Mar 20  2018 README.md
-rw-r--r--   1 varemo  NET\Domain Users   262B Mar 19  2018 Snakefile
drwxr-xr-x   3 varemo  NET\Domain Users    96B Mar 19  2018 code/
-rw-r--r--   1 varemo  NET\Domain Users    20B Mar 19  2018 config.yml
drwxr-xr-x   6 varemo  NET\Domain Users   192B Mar 20  2018 data/
drwxr-xr-x   4 varemo  NET\Domain Users   128B Mar 20  2018 doc/
-rw-r--r--   1 varemo  NET\Domain Users    61B Mar 19  2018 environment.yml
drwxr-xr-x   3 varemo  NET\Domain Users    96B Mar 20  2018 intermediate/
drwxr-xr-x   3 varemo  NET\Domain Users    96B Mar 19  2018 logs/
drwxr-xr-x   3 varemo  NET\Domain Users    96B Mar 19  2018 notebooks/
drwxr-xr-x   6 varemo  NET\Domain Users   192B Mar 20  2018 results/
drwxr-xr-x   3 varemo  NET\Domain Users    96B Mar 19  2018 scratch/
---------------------------------------------------------------------------
Fri Nov 23, 09:14:52 | MacBook ~/Work/Teaching/Reproducible_research/project_t
[base][master] >> git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

Environment | History | Plots | Connections | Packages | Help | Viewer

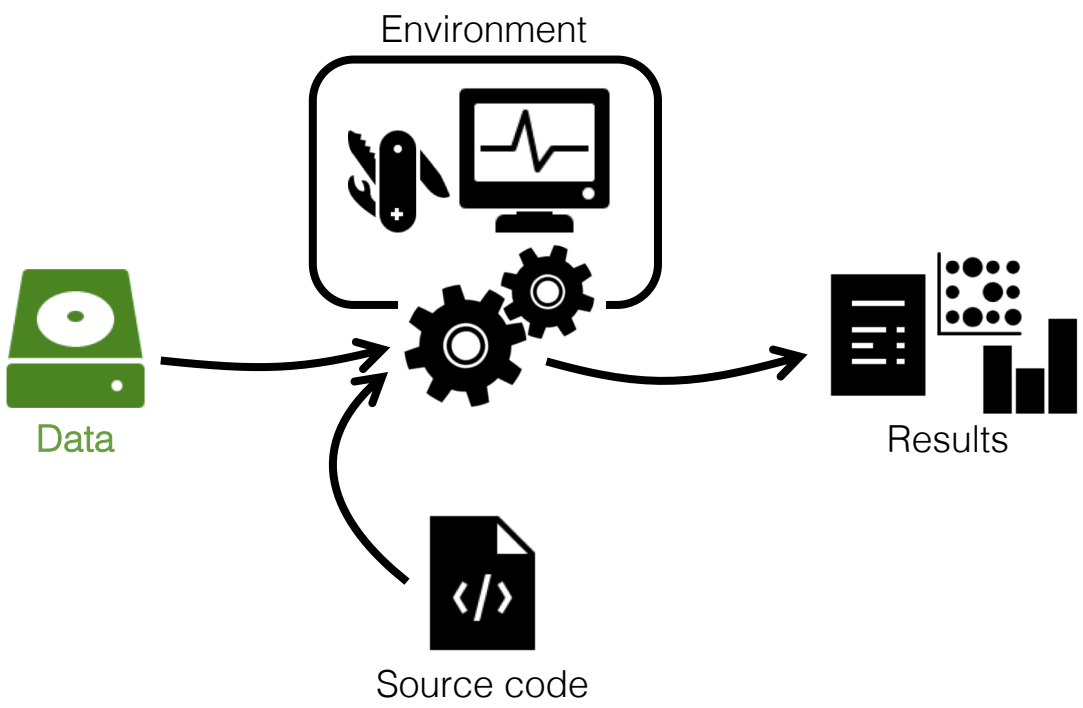
Publish

# Project template

Project template for setting up a directory and file structure for a new research project. The intended contents of each directory is explained in separate README.md files.

## Overview

```
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
```




```
project
|- doc/
|
|- data/
|   |- raw_external/
|   |- raw_internal/
|   |- meta/
|
|- code/
|- notebooks/
|
|- intermediate/
|- scratch/
|- logs/
|
|- results/
|   |- figures/
|   |- tables/
|   |- reports/
|
|- Snakefile
|- config.yml
|- environment.yml
|- Dockerfile
```

# Tools for Reproducible Research

CONDA

Managing dependencies

Snakemake

Managing and executing analysis workflow

git Bitbucket

Versioning and collaborating on code (and some other files)

jupyter / R Markdown from RStudio

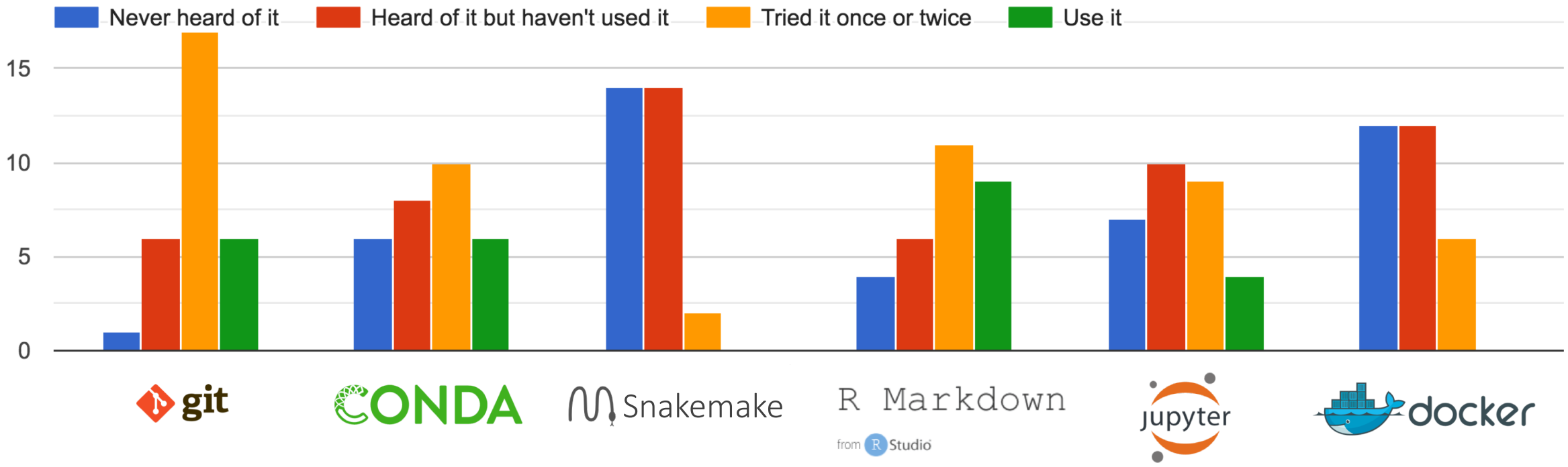
Connecting code and reporting

and…

docker

Isolating and exporting environment

# Student experience



CONDA

Managing dependencies

Snakemake

Managing and executing analysis workflow

git

Bitbucket

Versioning and collaborating on code (and some other files)

jupyter / R Markdown

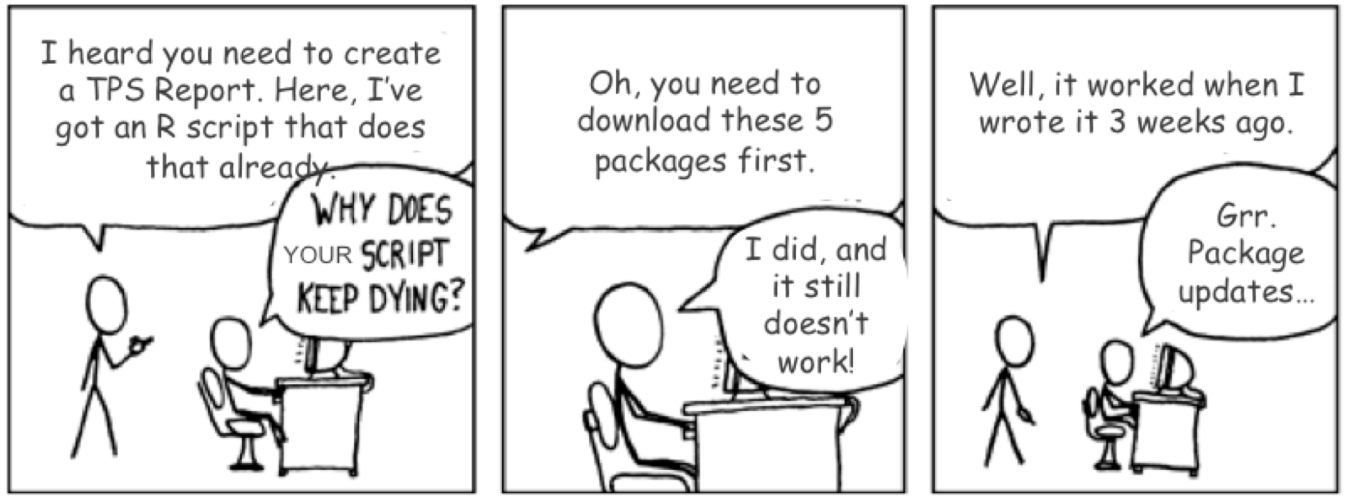
from RStudio

Connecting code and reporting

and…

docker

Isolating and exporting environment




Full reproducibility requires the possibility to recreate the system that was originally used to generate the results.

# What is Conda?

- Conda is a package, dependency, and environment manager.
- Works for software developed in any programming language.

| Decent | Getting there… | Well done! |
|---|---|---|

Manuscript.docx

Methods

We used:

Bowtie2
Samtools
HTSeq

Manuscript.docx

Methods

We used:

Bowtie2 (v2.3)
Samtools (v1.6)
HTSeq (v0.9)

environment.yml

```
channels:
- conda-forge
- bioconda
dependencies:
- bowtie2=2.3
- samtools=1.6
- htseq=0.9
```

User install manually (including all needed dependencies).

Conda creates an environment with the listed packages and dependencies automatically.

# Package manager

- Conda package: compressed tarball (system-level libraries, Python or other modules, executable programs, or other components).
- Conda keeps track of the dependencies between packages and platforms.
- Conda packages are downloaded from remote channels.

```
$ conda install -c conda-forge matplotlib

Fetching package metadata ...........
Solving package specifications: ..........

Package plan for installation in environment /Users/varemo/Applications/miniconda2/envs/test-r2:

The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    sqlite-3.13.0              |                1         1.4 MB  conda-forge
    libpng-1.6.24              |                0         338 KB  conda-forge
    python-2.7.12              |                1        11.8 MB  conda-forge
    certifi-2016.8.31          |           py27_0         218 KB  conda-forge
    freetype-2.6.3             |                1         782 KB  conda-forge
    functools32-3.2.3.2        |           py27_1          16 KB  conda-forge
    numpy-1.11.1               |           py27_0         3.1 MB  defaults
    pyparsing-2.1.8            |           py27_0          89 KB  conda-forge
    pytz-2016.6.1              |           py27_0         183 KB  conda-forge
    six-1.10.0                 |           py27_0          18 KB  conda-forge
    cycler-0.10.0              |           py27_0          13 KB  conda-forge
    python-dateutil-2.5.3      |           py27_0         236 KB  conda-forge
    setuptools-26.1.1          |           py27_0         346 KB  conda-forge
    matplotlib-1.5.3           |      np111py27_0         4.1 MB  conda-forge
    wheel-0.29.0               |           py27_0          81 KB  conda-forge
    pip-8.1.2                  |           py27_0         1.5 MB  conda-forge
    ------------------------------------------------------------
$ python                                   Total:        24.2 MB

The following NEW packages will be INSTALLED:
>>> import matplotlib
```

# Environment manager

- Conda environment: directory that contains a specific collection of Conda packages that you have installed.
- Packages are symlinked between environments to avoid duplication.

```
$ conda create --name env1 -c bioconda fastqc
$ fastqc --version
-bash: fastqc: command not found
$ source activate env1
$(env1)fastqc --version
FastQC v0.11.5
$(env1)source deactivate
$ conda create --name env2 -c bioconda python=3 snakemake
$ python --version
Python 2.7.12 :: Continuum Analytics, Inc.
$ snakemake --version
-bash: snakemake: command not found
$ source activate env2
$(env2)python --version
Python 3.4.3 :: Continuum Analytics, Inc.
$(env2)snakemake --version
3.7.1
$(env2)
```

# Defining and sharing environments

environment.yml

```
channels:
- conda-forge
- bioconda
dependencies:
- fastqc=0.11
- sra-tools=2.8
- snakemake=4.3.0
- multiqc=1.3
- bowtie2=2.3
- samtools=1.6
- htseq=0.9
- graphviz=2.38.0
```

- Create an environment from specifications in a file.
- All additional dependencies will be included.
- The environment.yml file can be shared with others and used to recreate the environment on other systems.

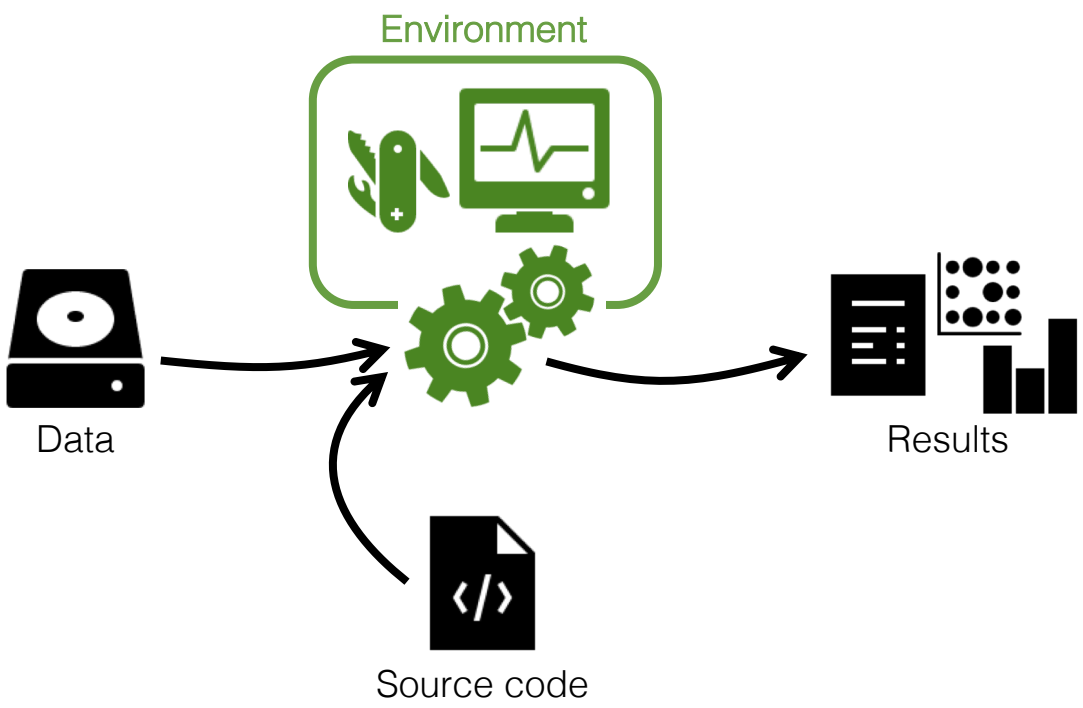
```
$ conda env create --name project_a -f environment.yml
```

- Update existing environment after adding new packages to environment.yml:

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

- Export existing environment as new yaml file (also includes dependencies):

```
$ conda env export > environment_full.yml
```




```
project
|- doc/
|
|- data/
|   |- raw_external/
|   |- raw_internal/
|   |- meta/
|
|- code/
|- notebooks/
|
|- intermediate/
|- scratch/
|- logs/
|
|- results/
|   |- figures/
|   |- tables/
|   |- reports/
|
|- Snakefile
|- config.yml
|- environment.yml
|- Dockerfile
```

# The tutorials

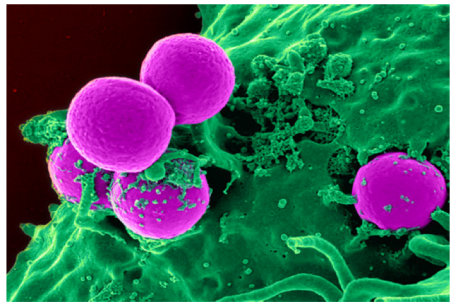
A few practical notes…

OPEN ACCESS Freely available online PLOS | ONE

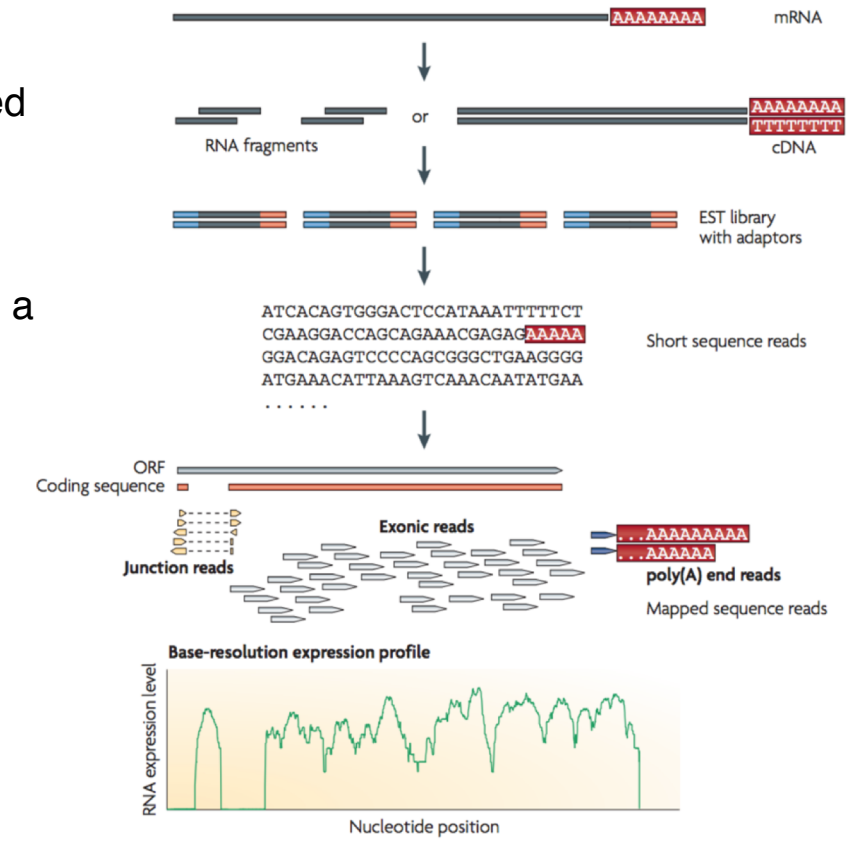
# RNA-Seq Reveals Differential Gene Expression in *Staphylococcus aureus* with Single-Nucleotide Resolution

Joseph Osmundson[1*], Scott Dewell[2], Seth A. Darst[1]

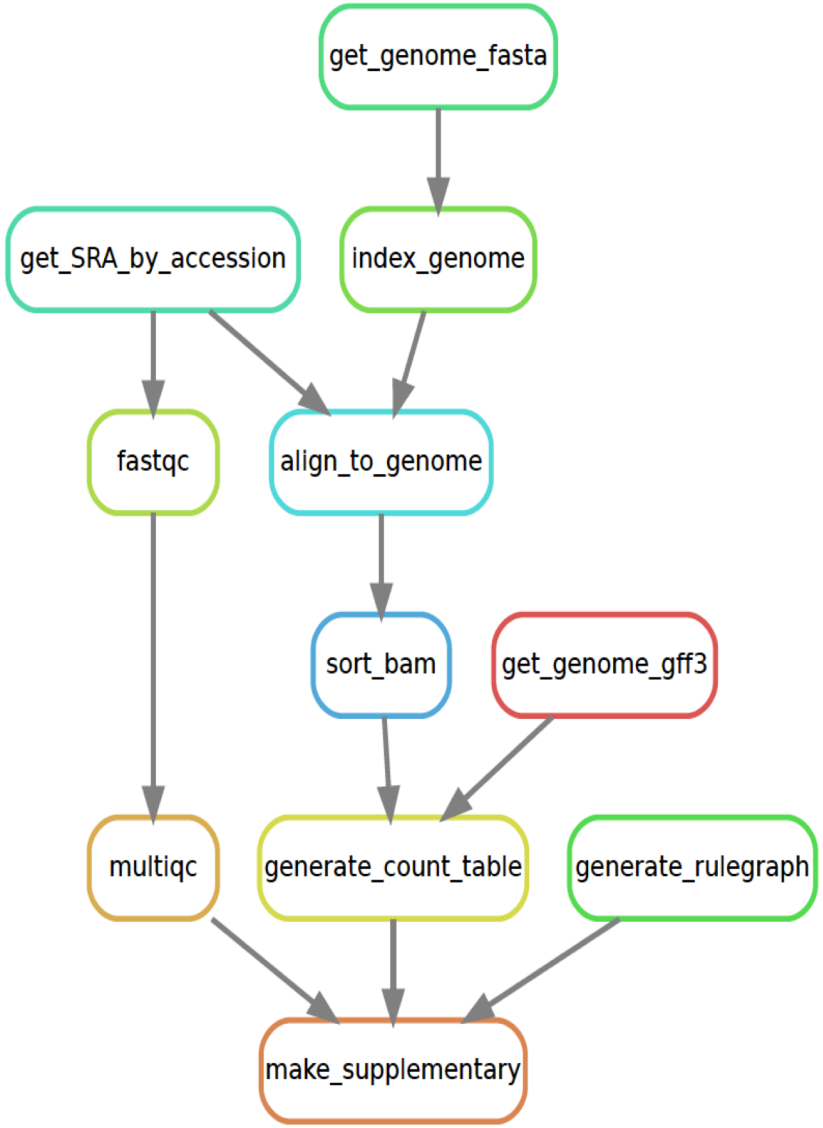
- Methicillin-resistant *Staphylococcus aureus* (MRSA):
  - is resistant to broad spectrum beta-lactam antibiotics
  - lead to difficult-to-treat infections in humans
- Lytic bacteriophages have been suggested as potential therapeutic agents, or as the source of novel antibiotic proteins or peptides.
- One such protein, gp67, was identified as a transcription-inhibiting transcription factor with an antimicrobial effect.
- To identify *S. aureus* genes repressed by gp67, the authors expressed gp67 in *S. aureus* cells.
- RNA-seq was performed on *S. aureus* strains:
  - RN4220 with pRMC2 with gp67
  - RN4220 with empty pRMC2
  - NCTC8325-4



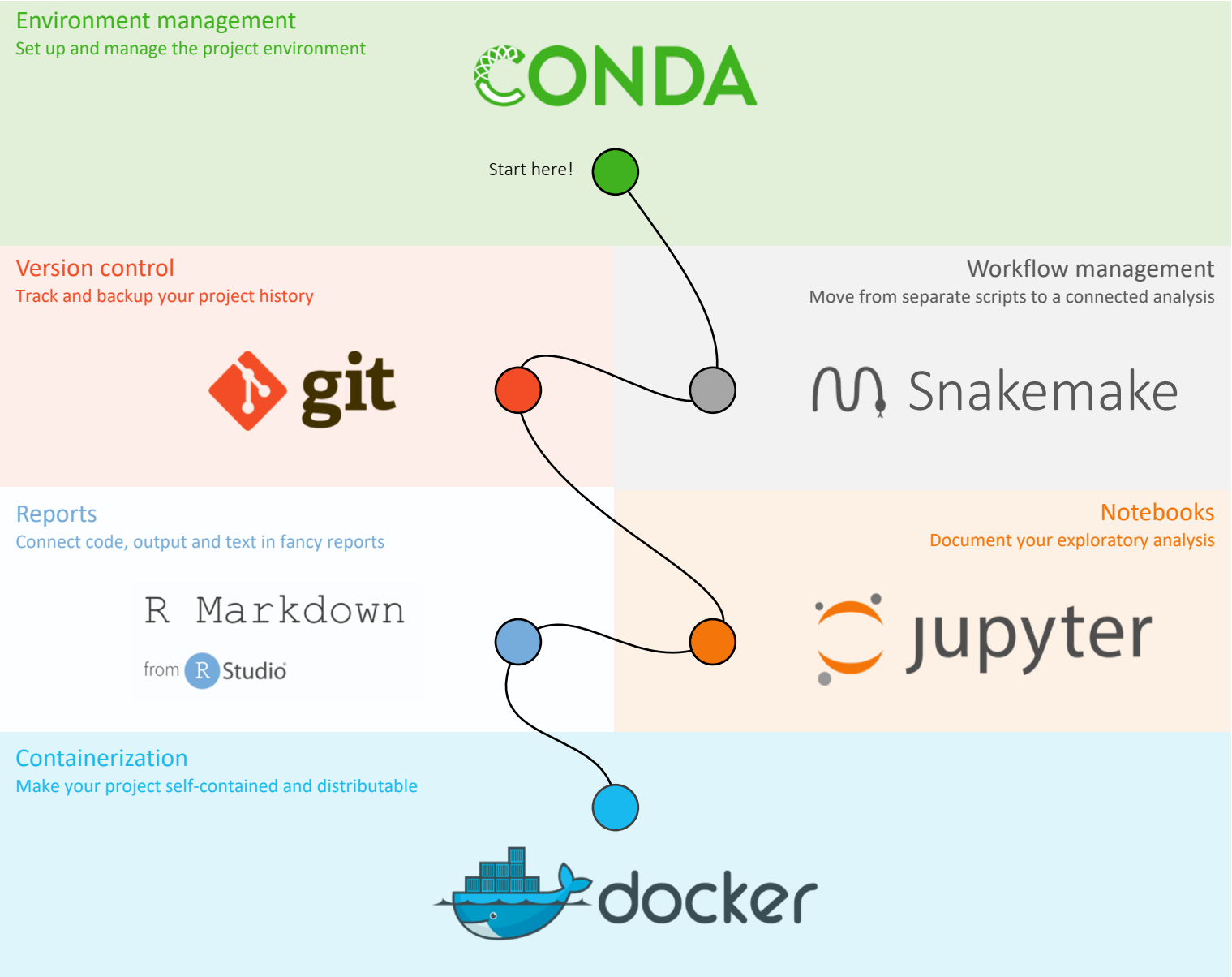
Scanning electron micro-graph of a human neutrophil ingesting MRSA

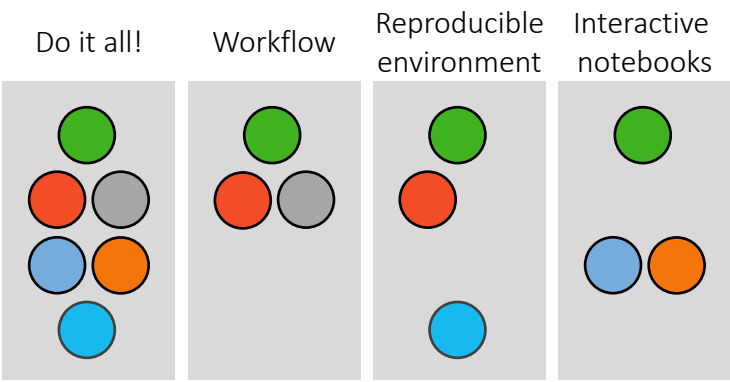



## The analysis workflow



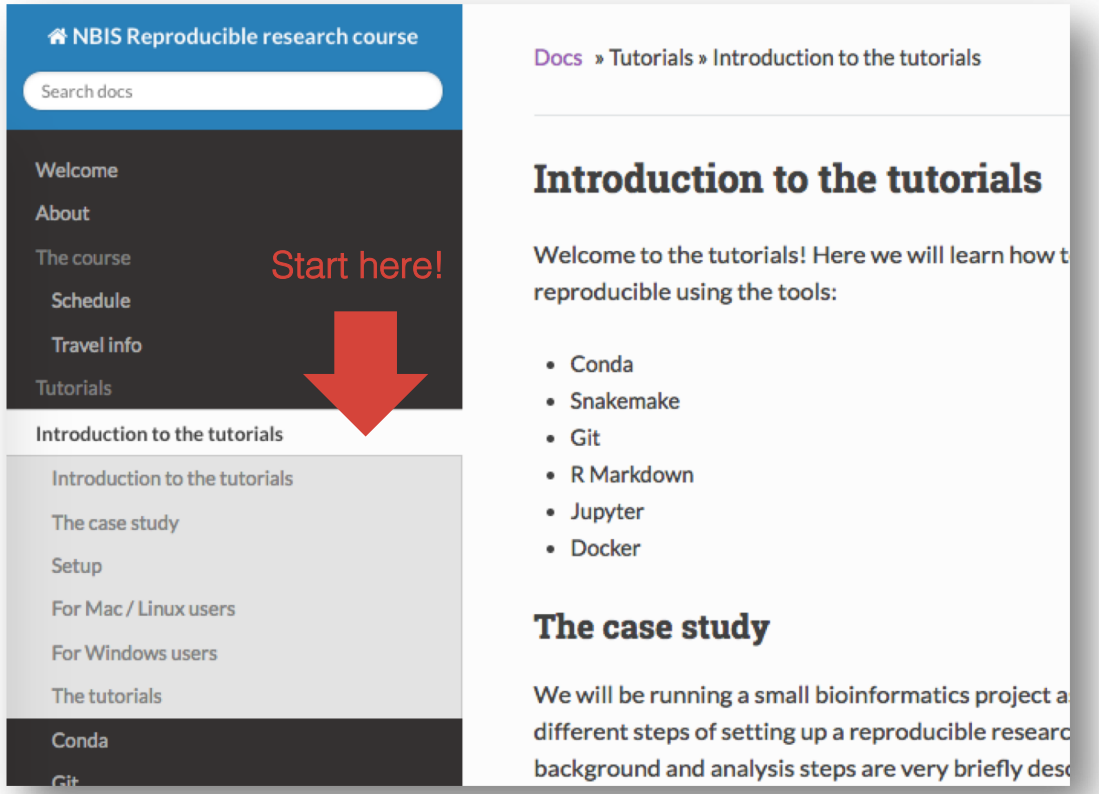
# The tutorials



# Getting started

- Clone course git repository to get all files needed for tutorials!
- Each tutorial will run in a specific subdirectory within `reproducible_research_course`, make sure you are running from the right place!
- Exception: the git tutorial will be run in a user-created directory outside of `reproducible_research_course`.




http://nbis-reproducible-research.readthedocs.io

CONDA

Managing dependencies

Snakemake

Managing and executing analysis workflow

git

Bitbucket

Versioning and collaborating on code (and some other files)

jupyter / R Markdown

from RStudio

Connecting code and reporting

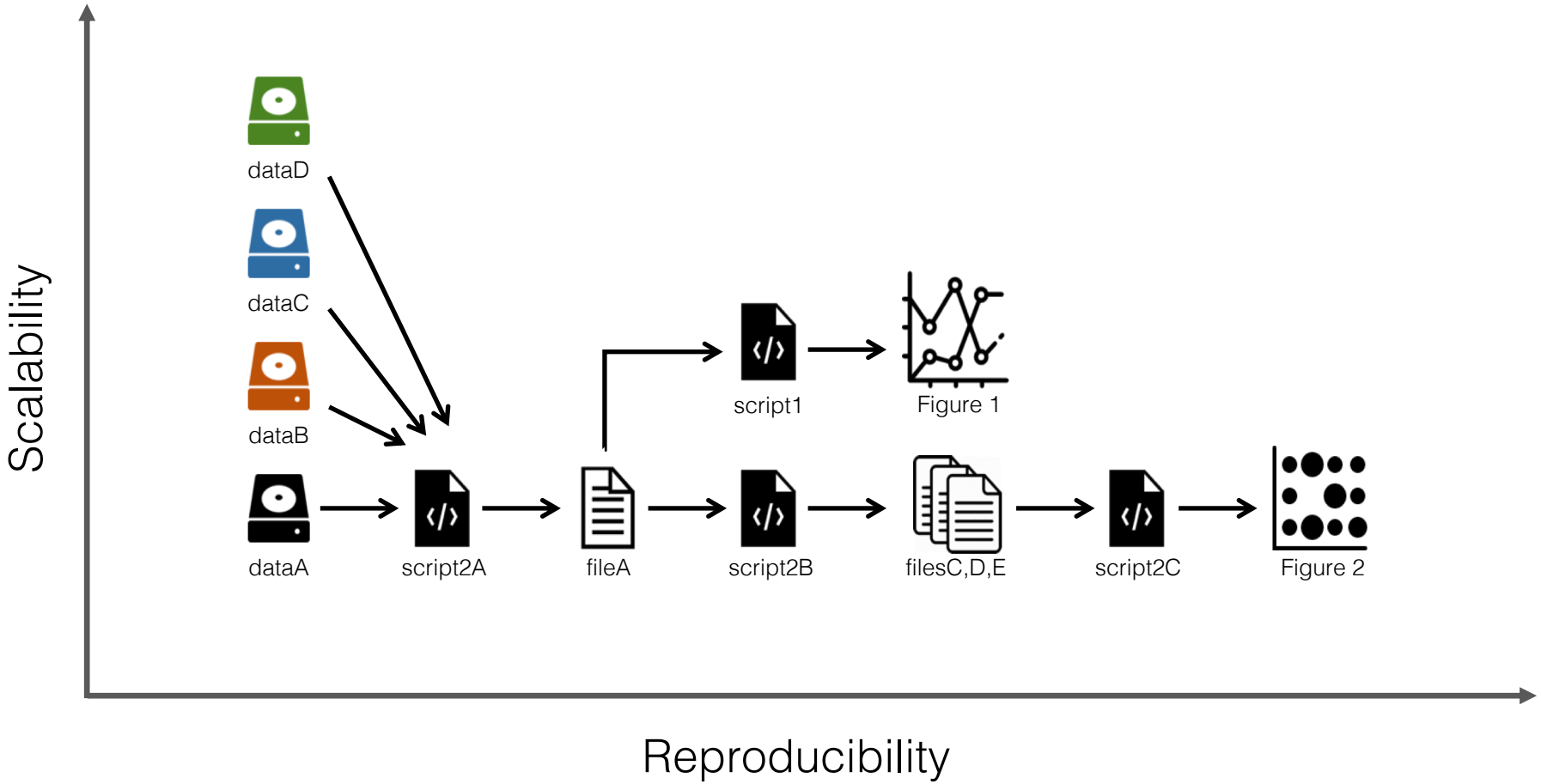
and…

docker

Isolating and exporting environment

As projects grow or age, it becomes
increasingly difficult to keep track of all the parts
and how they fit together.

“Snakemake is a workflow management system that
aims to reduce the complexity of creating workflows by
providing a fast and comfortable execution environment,
together with a clean and modern specification
language in python style.”


Scalability
dataD
dataC
dataB
dataA
script2A
fileA
script1
Figure 1
script2B
filesC,D,E
script2C
Figure 2
Reproducibility

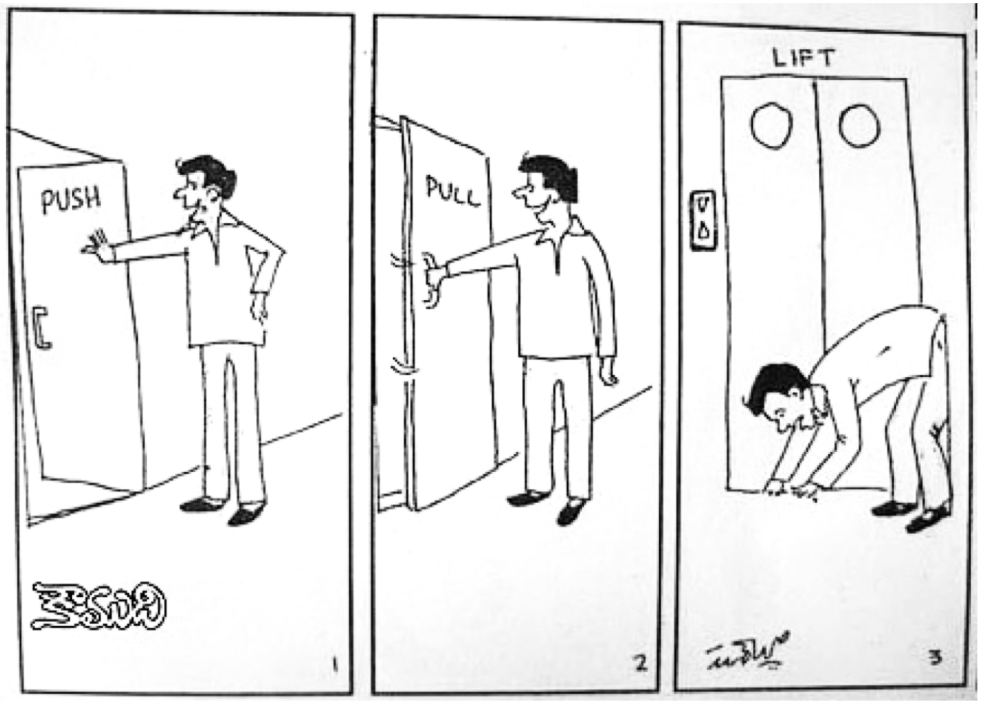
Workflow management systems come in different flavors.

## Explicit syntax ("Push")

"Here are my inputs, please perform these operations in this order on them."

## Implicit syntax ("Pull")

"I need this output, could you please figure out which operations to perform and in which order?"



# Explicit approach using Bash

trim_and_zip.sh

```
for sample in *.fastq
do
   id=$(echo ${sample} | sed 's/.fastq//')

   # Trim fastq file
   echo “Trimming ${id}“
   seqtk trimfq -b 5 -e 10 $sample > \
   ${id}.trimmed.fastq

   # Compress fastq file
   echo “Compressing ${id}“
   gzip -c ${id}.trimmed.fastq > \
   ${id}.trimmed.fastq.gz

   # Remove intermediate files
   rm ${id}.trimmed.fastq
done
```

```
$bash trim_and_zip.sh
Trimming sample: a
Compressing sample: a
Trimming sample: b
Compressing sample: b
```

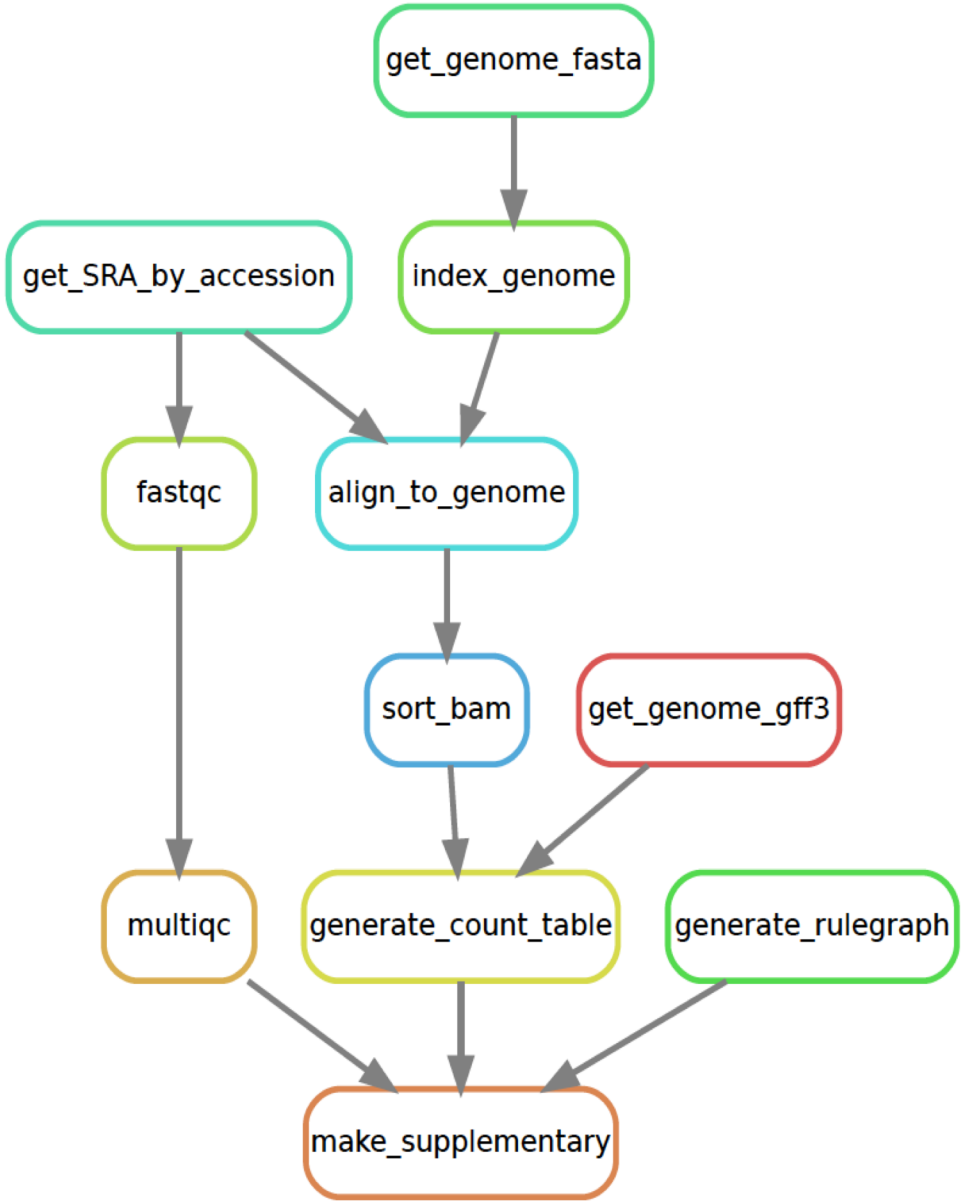
Snakemake

# Implicit approach using Snakemake

Snakefile

```
rule trim_fastq:
    input: "{prefix}.fastq"
    output: temp("{prefix}.trimmed.fastq")
    shell:
        "seqtk trimfq -b 5 -e 10 {input} > {output}"

rule gzip:
    input: "{prefix}"
    output: "{prefix}.gz"
    shell:
        "gzip -c {input} > {output}"
```

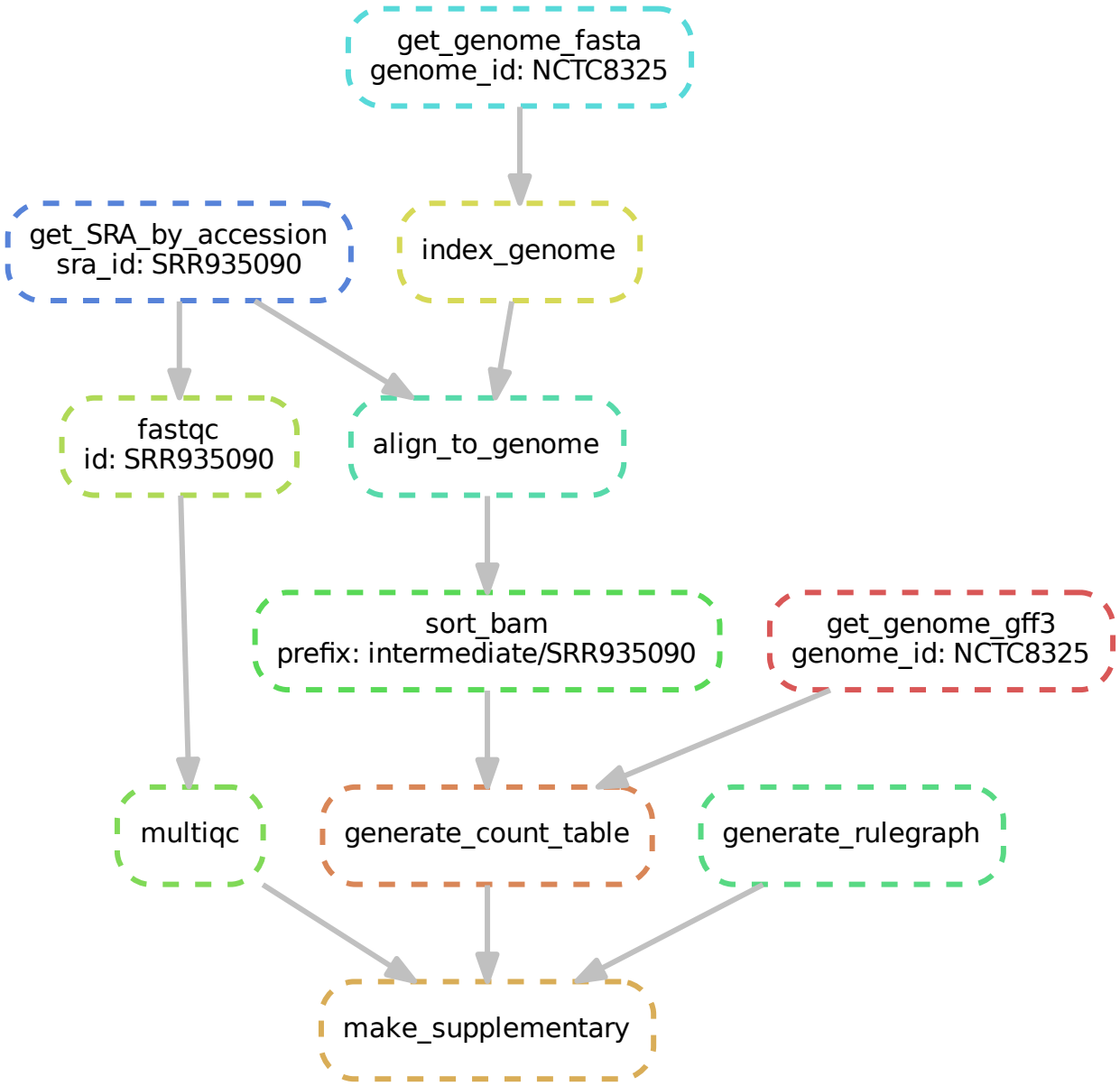
```
$snakemake {a,b}.trimmed.fastq.gz
Provided cores: 1
Rules claiming more threads will be
scaled down.
Job counts:
count   jobs
2       gzip
2       trim_fastq
4
rule trim_fastq:
    input: a.fastq
    output: a.trimmed.fastq
    wildcards: prefix=a
1 of 4 steps (25%) done
rule gzip:
    input: a.trimmed.fastq
    output: a.trimmed.fastq.gz
    wildcards: prefix=a.trimmed.fastq
Removing temporary output file a.trimmed.fastq.
2 of 4 steps (50%) done
rule trim_fastq:
    input: b.fastq
    output: b.trimmed.fastq
    wildcards: prefix=b
3 of 4 steps (75%) done
rule gzip:
    input: b.trimmed.fastq
    output: b.trimmed.fastq.gz
    wildcards: prefix=b.trimmed.fastq
Removing temporary output file b.trimmed.fastq.
4 of 4 steps (100%) done
```




Snakemake figures out how rules can be pieced together to generate some requested output.

Here we ask for `supplementary.pdf`, which is an R Markdown report generated by the rule `make_supplementary`.

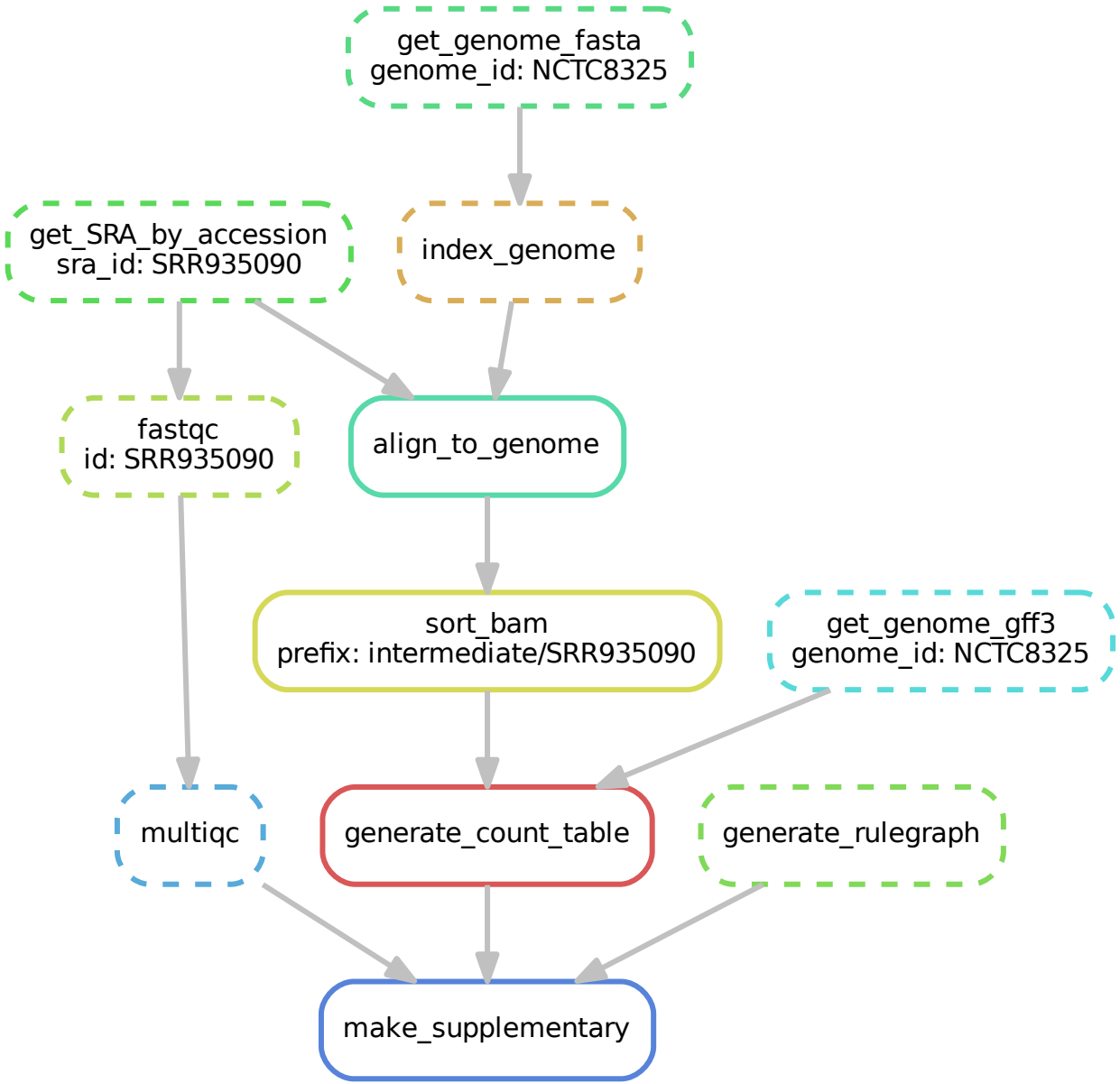
```
$snakemake supplementary.pdf --rulegraph | dot -Tpdf > rulegraph.pdf
```




Snakemake keeps track of when files were generated and by which rules.

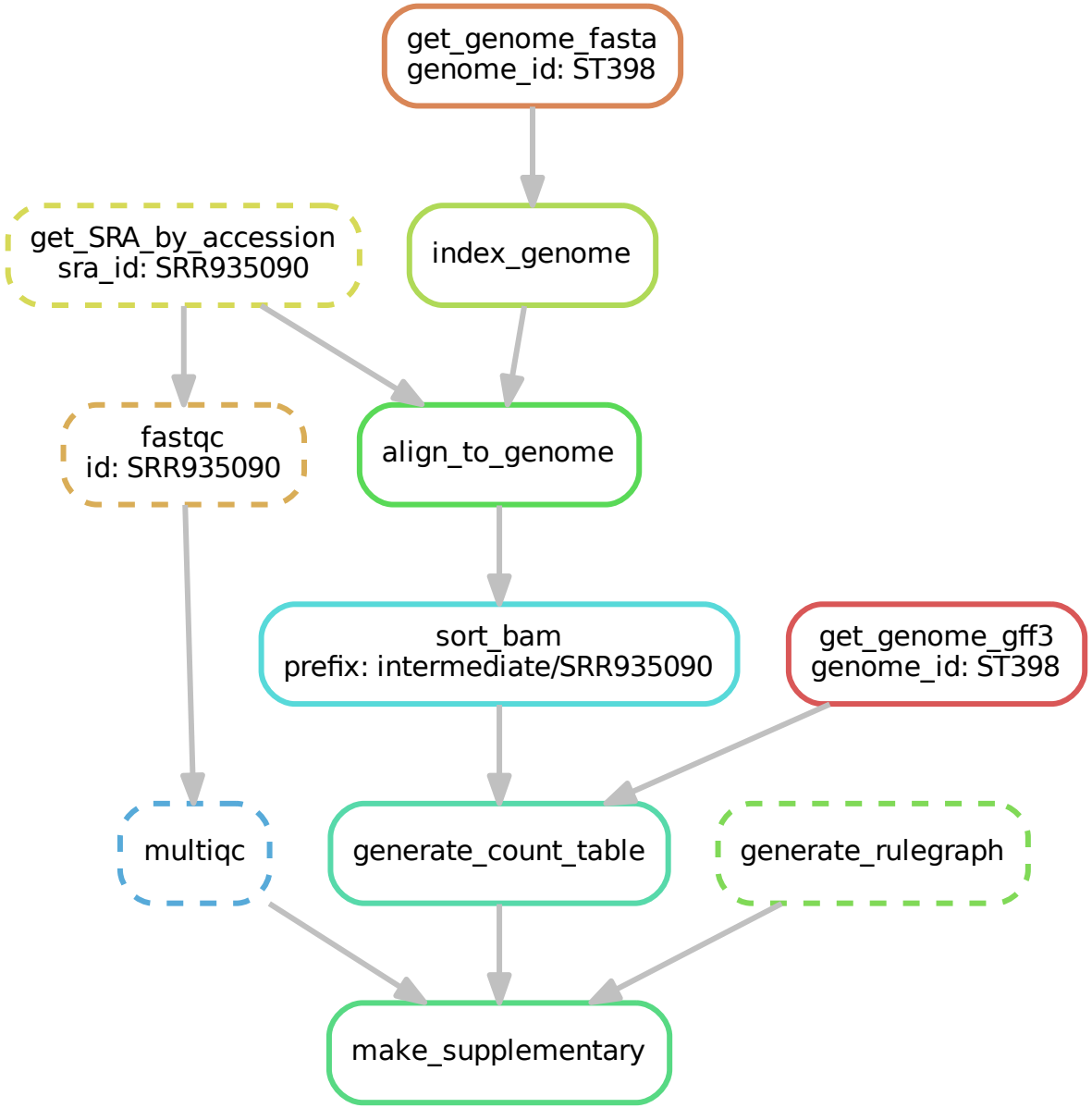
Dotted rule boxes show that `supplementary.pdf` already exists and that it's newer than its dependencies (recursively).

```
$snakemake supplementary.pdf --dag | dot -Tpdf > dag.pdf
```




Here Snakemake detects that a file used in `align_to_genome` is newer than downstream files, so it reruns the necessary rules.

```
$touch intermediate/NCTC8325.1.bt2
$snakemake supplementary.pdf --dag | dot -Tpdf > dag.pdf
```




Forcing a rule (`get_genome_fasta` here) to be rerun also leads to rerunning all rules that depend on it.

Note that we also change the parameter "genome_id" to use another genome to align to. This causes `get_genome_gff3` to be rerun as well.

```
$snakemake supplementary.pdf --config genome_id=ST398
        -f get_genome_fasta --dag | dot -Tpdf > dag.pdf
```

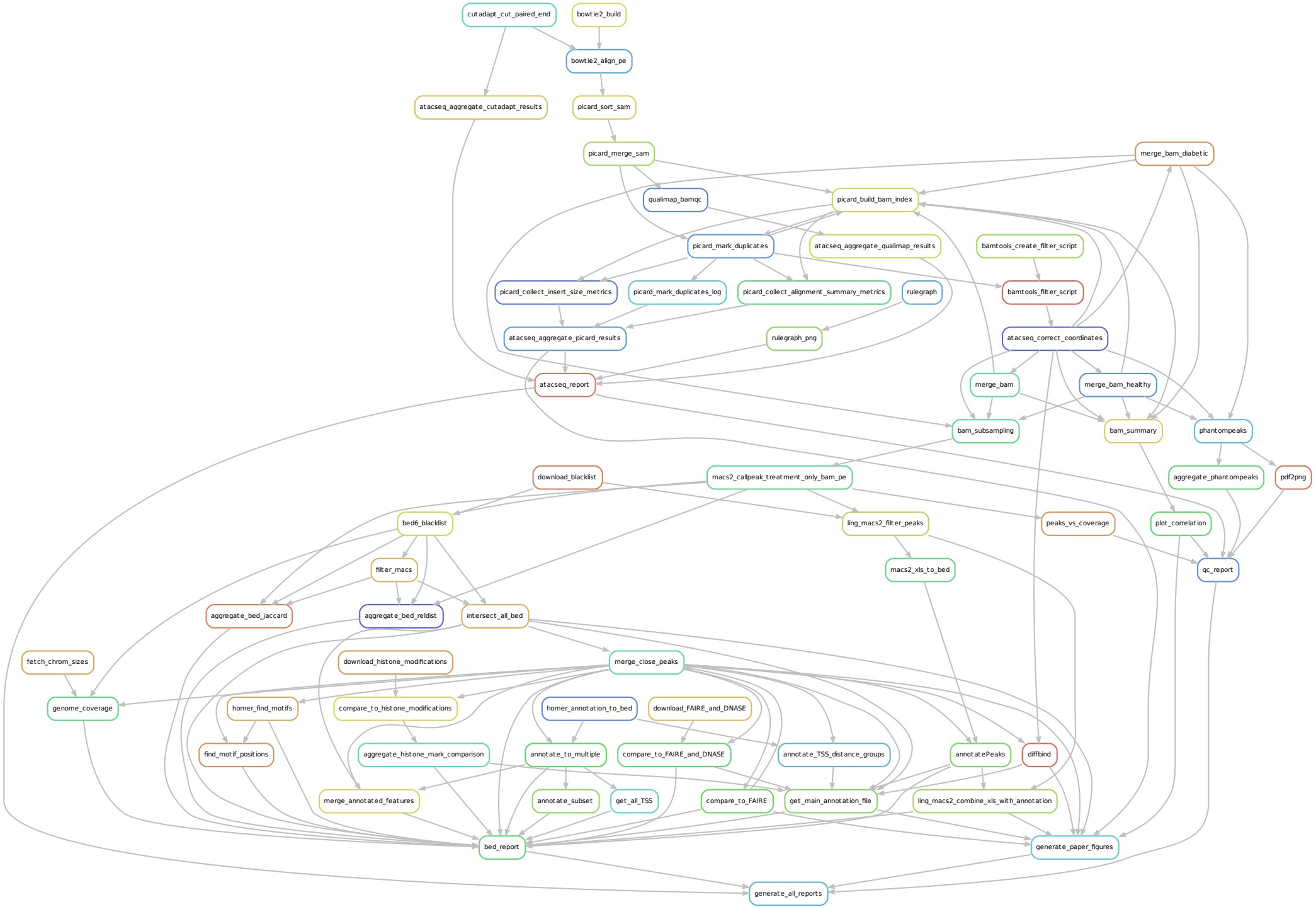
# Anatomy of a Snakemake rule

```
import os

rule trim_fastq:
    input: "{prefix}.fastq"
    output: temp("{prefix}.trimmed.fastq")
    params:
        leftTrim=5,
        rightTrim=10
    log: "logs/trim_fastq.log"
    version: "0.1"
    message: "Trimming {input[0]}."
    shadow: True
    threads: 8
    priority: 90
    resources: mem=64
    conda: "envs/seqtk.yaml”
    singularity: "docker://quay.io/biocontainers/seqtk"
    run:
        if (os.stat(input[0]).st_size > 0):
            shell("seqtk trimfq -t {threads} –b {params.leftTrim}
                   -e {params.rightTrim} {input} > {output} 2> {log}")
        else:
            raise IOError(input[0]+" is empty.")
```

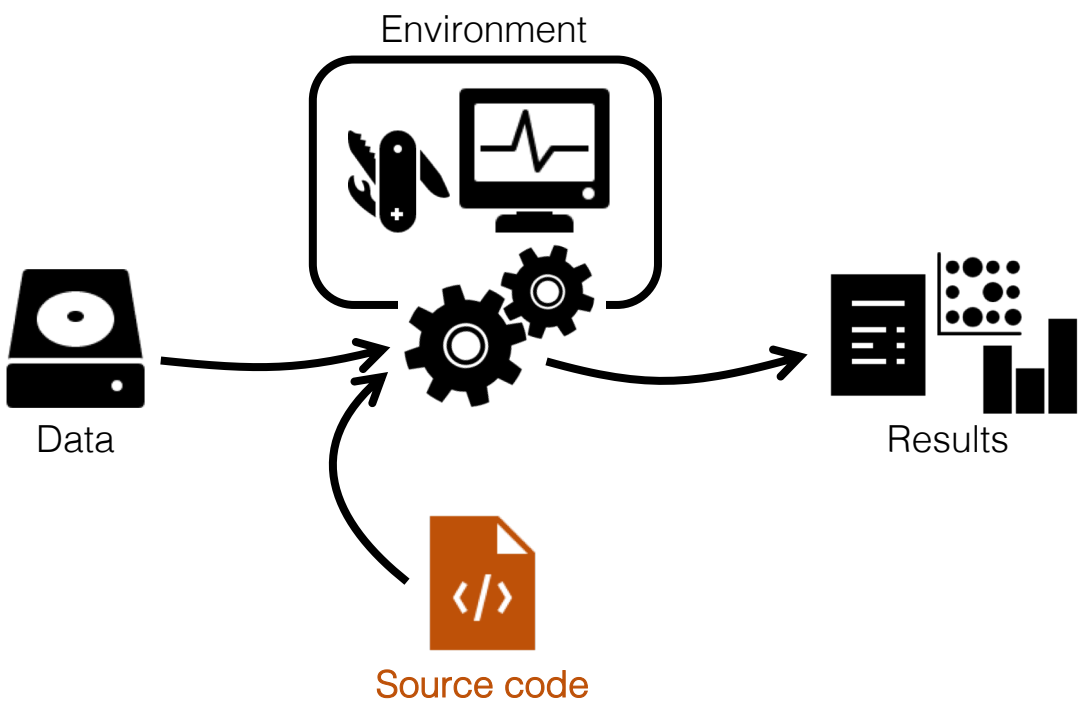
# Command line interface

```
# execute the workflow with target a.trimmed.fastq.gz
snakemake a.trimmed.fastq.gz

# execute the workflow with the first rule as target
snakemake

# dry-run, print shell commands and reason for execution
snakemake -n —p -r

# visualize the DAG of jobs using the Graphviz dot command
snakemake --dag | dot -Tsvg > dag.svg

# execute the workflow with 8 cores
snakemake --cores 8

# run the workflow on a SLURM cluster
snakemake --cluster-config cluster.yml --cluster \
      "sbatch -A {cluster.account} -t {cluster.time}"
```

Snakemake




Things can get rather complex...




```
project
|- doc/
|
|- data/
|   |- raw_external/
|   |- raw_internal/
|   |- meta/
|
|- code/
|- notebooks/
|
|- intermediate/
|- scratch/
|- logs/
|
|- results/
|   |- figures/
|   |- tables/
|   |- reports/
|
|- Snakefile
|- config.yml
|- environment.yml
|- Dockerfile
```

CONDA

Managing dependencies

Snakemake

Managing and executing analysis workflow

git

Bitbucket

Versioning and collaborating on code (and some other files)

jupyter / R Markdown

from RStudio

Connecting code and reporting

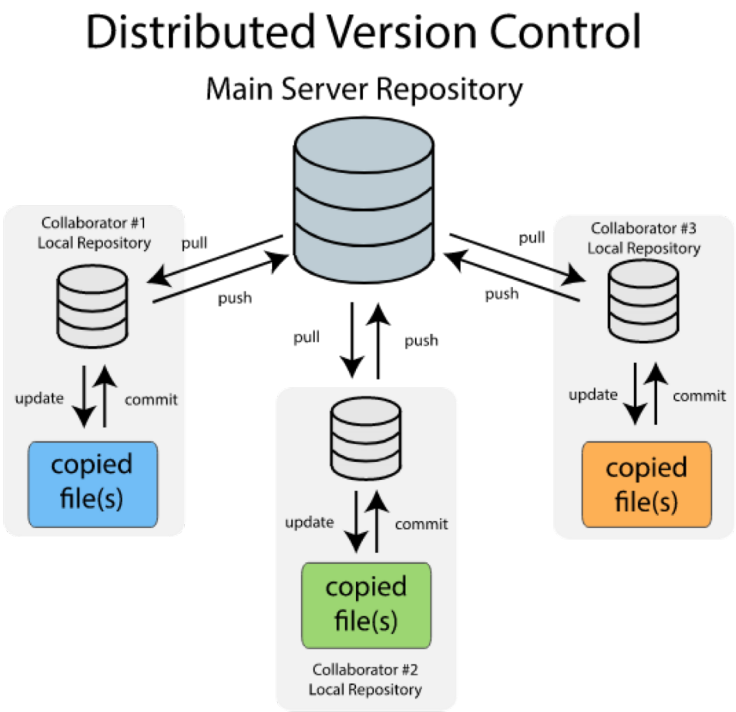
and…

docker

Isolating and exporting environment

# What is Git?

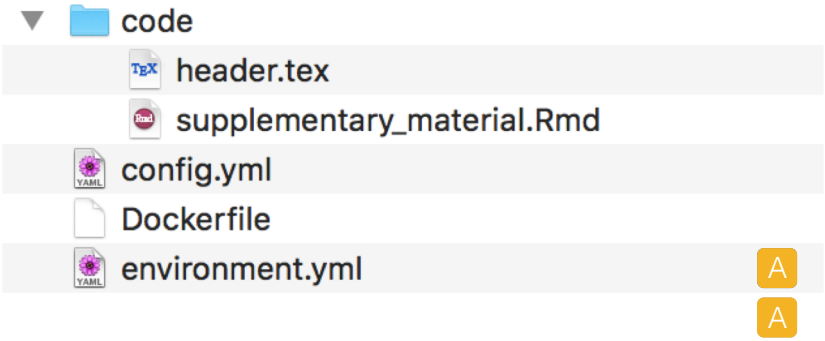
- Widely used system for file version control.
- Keeps copies of files and code from every stages in their lifecycle.
- Kind of like Dropbox, but you decide when each version is saved (and a lot more advanced features).
- Each file has a distinct history with specific incremental changes (each with a reference code).
  - Revert files to previous state.
  - Compare changes over time.
  - See who modified what.
- Makes you fearless.
- Runs on command line, but there also exists GUI and integration in e.g. text editors.
- Mainly for text files, not for binary files or large files.
- Versioning, backup, and sharing!



# How does it work in practice?

**Nomenclature**

| | |
|---|---|
| Repository | *Directory with all files, will include a .git folder* |
| Commit | *A specific version of the repository* |
| Push | *Upload local changes to remote repository* |
| Pull | *Download changes from remote repository* |



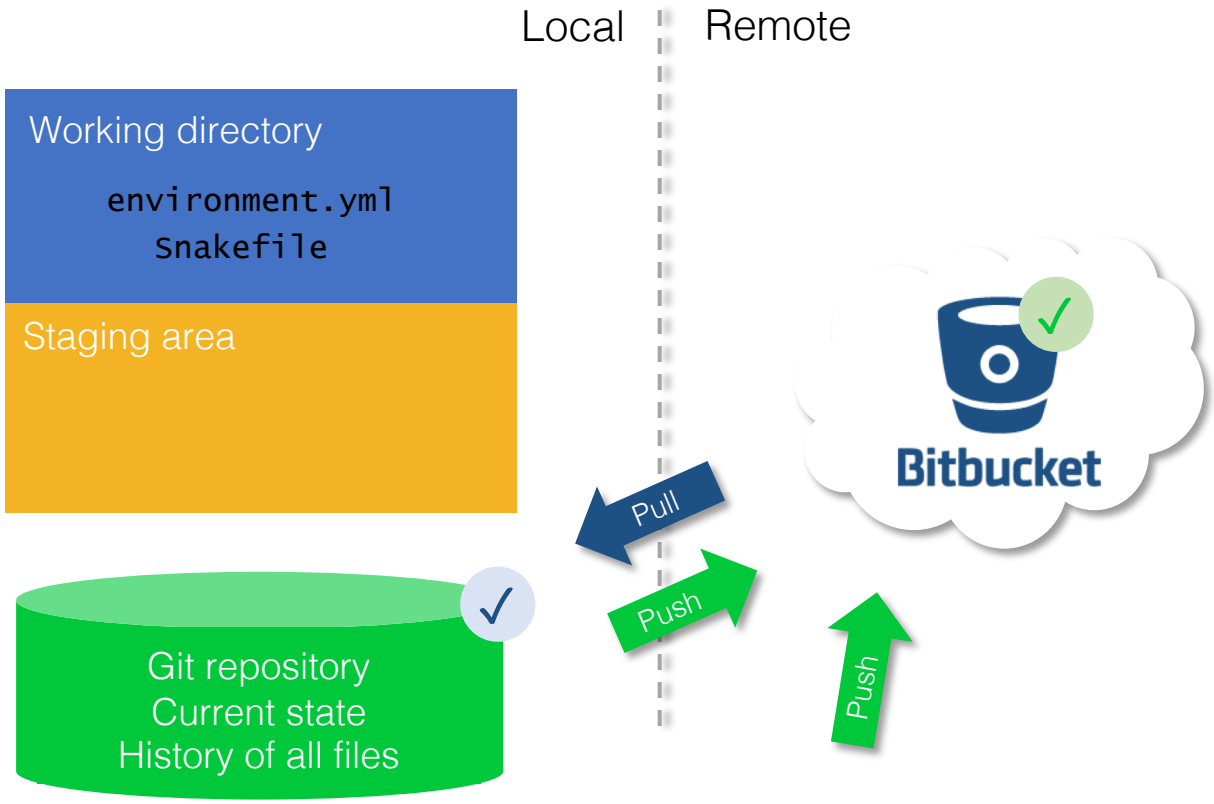

- Edit environment.yml
- Add a new file Snakefile
- `git add environment.yml`
- `git commit -m "Add snakemake 4.4.0"`
- `git add Snakefile`
- `git commit -m "Add Snakefile"`
- `git push`
- `git pull`

**git log**

```
a2c36bs Add heatmap figure
6152ff6 Format figure label
0abd0cb Update multiqc version
8dhfls8 Add snakemake 4.4.0
kfhs7s6 Add Snakefile
2kd7f0f Fix alignment command
```

**git log**

```
a2c36bs Add heatmap figure
6152ff6 Format figure label
0abd0cb Update multiqc version
8dhfls8 Add snakemake 4.4.0
kfhs7s6 Add Snakefile
2kd7f0f Fix alignment command
```



reproducible_resear...

- Overview
- Source
- Commits
- Branches
- Pull requests
- Pipelines
- Issues
- Downloads
- Boards
- Settings

SciLifeLab Bioinformatics LTS / Bioinformatics LTS / reproducible_research_course / Source

# docker/

master | reproducible_research_course /

| | | | |
|---|---|---|---|
| .. | | | |
| code | | | |
| Dockerfile | 1.1 KB | 5 hours ago | split git_jupyter_docker |
| Snakefile | 6.5 KB | 4 hours ago | moved counts table |
| config.yml | 1.5 KB | 5 hours ago | split git_jupyter_docker |
| environment.yml | 202 B | 5 hours ago | split git_jupyter_docker |

| | | | |
|---|---|---|---|
| Rasmus Ågren | 232ffc0 | language and spelling in git tutorial | 5 hours ago |
| varemo | a81f6e8 | remove Where to next sections | 6 hours ago |
| varemo | d8e30df M | Merge branch 'master' of https://bitbucket.org/scilifelab-lts/reproducible_researc... | 6 hours ago |
| varemo | 449b4b9 | remove todo | 6 hours ago |
| Rasmus Ågren | bfa0edd M | Merge branch 'master' of https://bitbucket.org/scilifelab-lts/reproducible_researc... | 7 hours ago |
| Rasmus Ågren | 851e43e | language and spelling in git tutorial | 7 hours ago |
| varemo | cc2bbcb | fix figure fonts | 7 hours ago |
| varemo | 19c67d4 M | Merge branch 'master' of https://bitbucket.org/scilifelab-lts/reproducible_researc... | 7 hours ago |
| varemo | 3a1806f | fix nicer figure | 7 hours ago |
| Rasmus Ågren | 1487f3b | test anchoring | 8 hours ago |
| Rasmus Ågren | 3397711 | split snakemake env for speed | 8 hours ago |
| varemo | 8f265b0 | tutorial text | 9 hours ago |
| varemo | 414de52 | typo | 9 hours ago |
| varemo | 208e1d9 | restructure pages | 10 hours ago |
| varemo | 9fca2f7 | remove take down part | 10 hours ago |
| varemo | 831a44e | revert back to project a from project new | 10 hours ago |
| varemo | 3575b9c M | Merge branch 'master' of https://bitbucket.org/scilifelab-lts/reproducible_researc... | 11 hours ago |
| Rasmus Ågren | 0e4296a | typo | 11 hours ago |
| varemo | aeffaa3 | restructure pages | 11 hours ago |
| Rasmus Ågren | 14c60cb M | Merge branch 'master' of https://bitbucket.org/scilifelab-lts/reproducible_researc... | 11 hours ago |
| Rasmus Ågren | 9795b0f | language and spelling in the conda tutorial | 11 hours ago |

Diff from 8baf4a1 2017-11-22 to d797810 2017-11-23

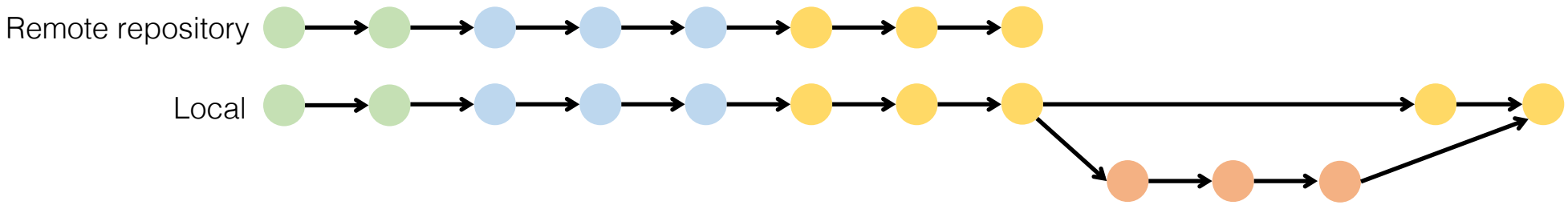
snakemake/Snakefile_mrsa MODIFIED

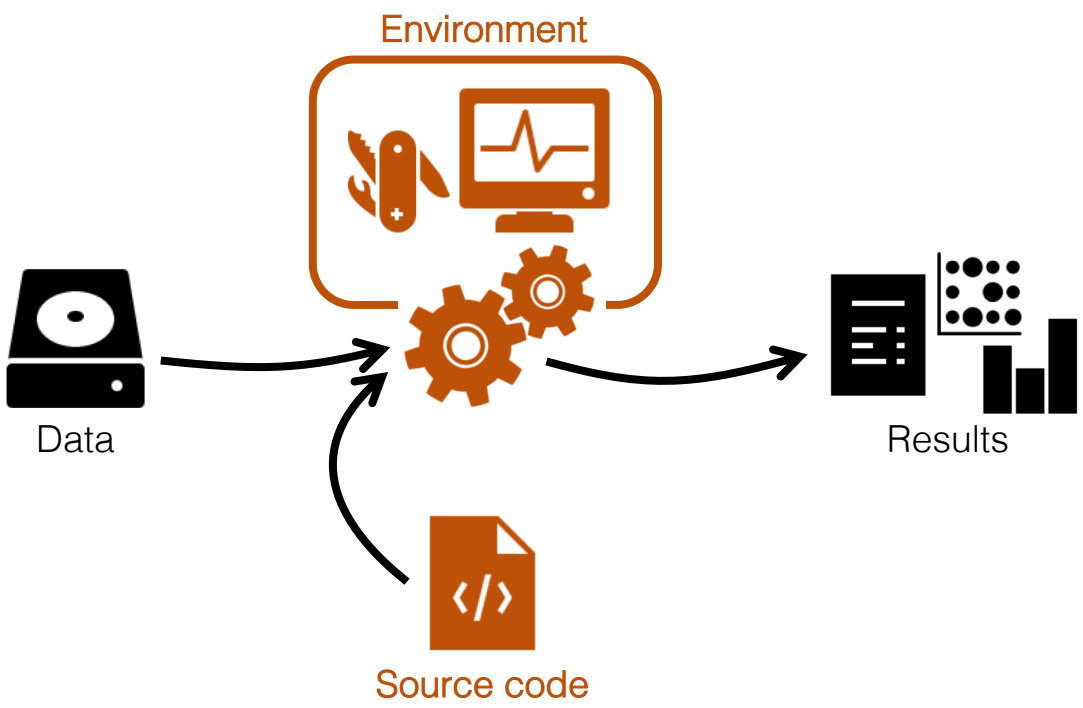
Side-by-side diff

```diff
156 156
157 157         # Save the count table as a temporary file and then prepend a header line
158 158         # with the sample names
159     -       htseq-count --format bam --type gene --idattr gene_id {input.bams} tempfile > tempfile2
    159 +       htseq-count --format bam --type gene --additional-attr description --idattr gene_id {input.bams} tempfile > tempfile2
160 160         echo '{input.bams}' | tr ' ' '\t' | cat - tempfile2 > {output}
161 161
162 162         # Remove the temporary files
```

# During the working day…

- Pull collaborator's latest work to get your local repository up to date.
- Carry on with your work and edit files.
- Commit often!
  - Each commit should be related to a distinct change/addition/task.
  - Write descriptive commit messages.
- Push your changes to the remote repository.
- If you know several people are actively working on the same repository, push and pull often!






```
project
|- doc/
|
|- data/
|   |- raw_external/
|   |- raw_internal/
|   |- meta/
|
|- code/
|- notebooks/
|
|- intermediate/
|- scratch/
|- logs/
|
|- results/
|   |- figures/
|   |- tables/
|   |- reports/
|
|- Snakefile
|- config.yml
|- environment.yml
|- Dockerfile
```

CONDA

Managing dependencies

Snakemake

Managing and executing analysis workflow

git

Bitbucket

Versioning and collaborating on code (and some other files)

jupyter / R Markdown

from RStudio

Connecting code and reporting

and…

docker

Isolating and exporting environment

# Typical guidelines for keeping a notebook of wet-lab work

Record everything you do in the lab, even if you are following a published procedure.

If you make a mistake, put a line through the mistake and write the new information next to it.

Use a ball point pen so that marks will not smear nor will they be erasable.

Use a bound notebook so that tear-out would be visible.

If you're testing a specific hypothesis, write it down beforehand.

When you finish a page, put a corner-to corner line through any blank parts that could still be used for data entry.

All pages must be pre-numbered.

Use a ball point pen so that marks will not smear nor will they be erasable.

Write a title for each and every new set of entries.

Each page should be numbered and dated consistently.

It is critical that you enter all procedures and data directly into your notebook in a timely manner.

The investigator and supervisor must sign each page.

Properly introduce and summarize each experiment.

# Typical guidelines for keeping a notebook of dry-lab work

# Literate programming

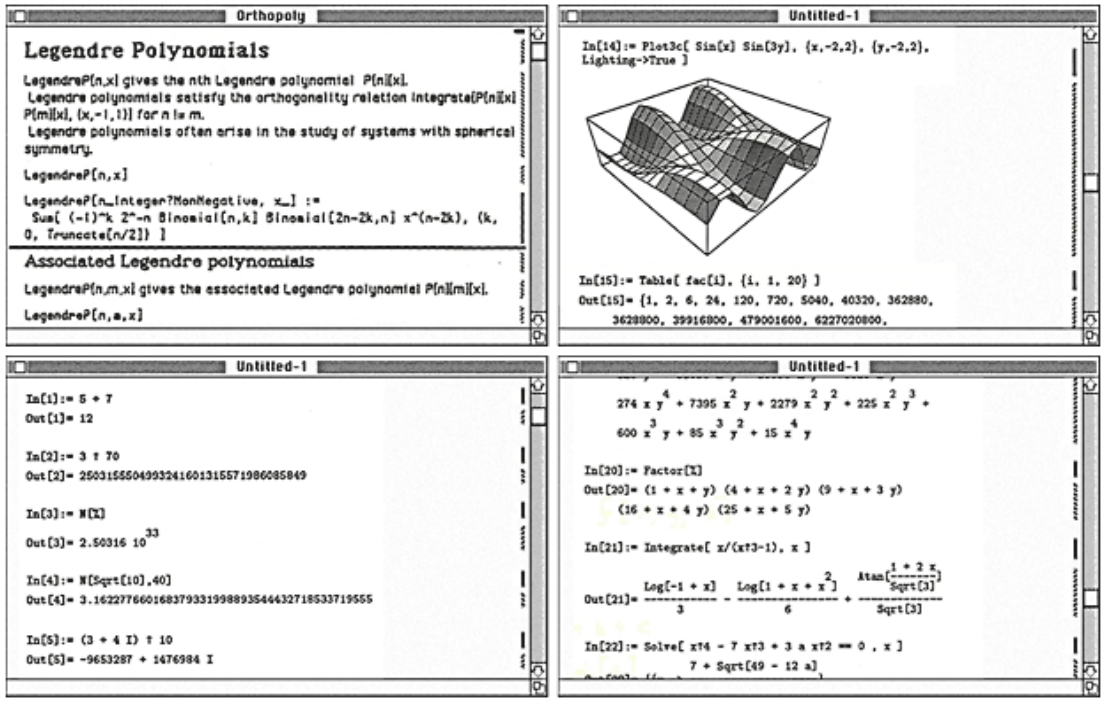
*Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do.*

Donald Knuth (1984)

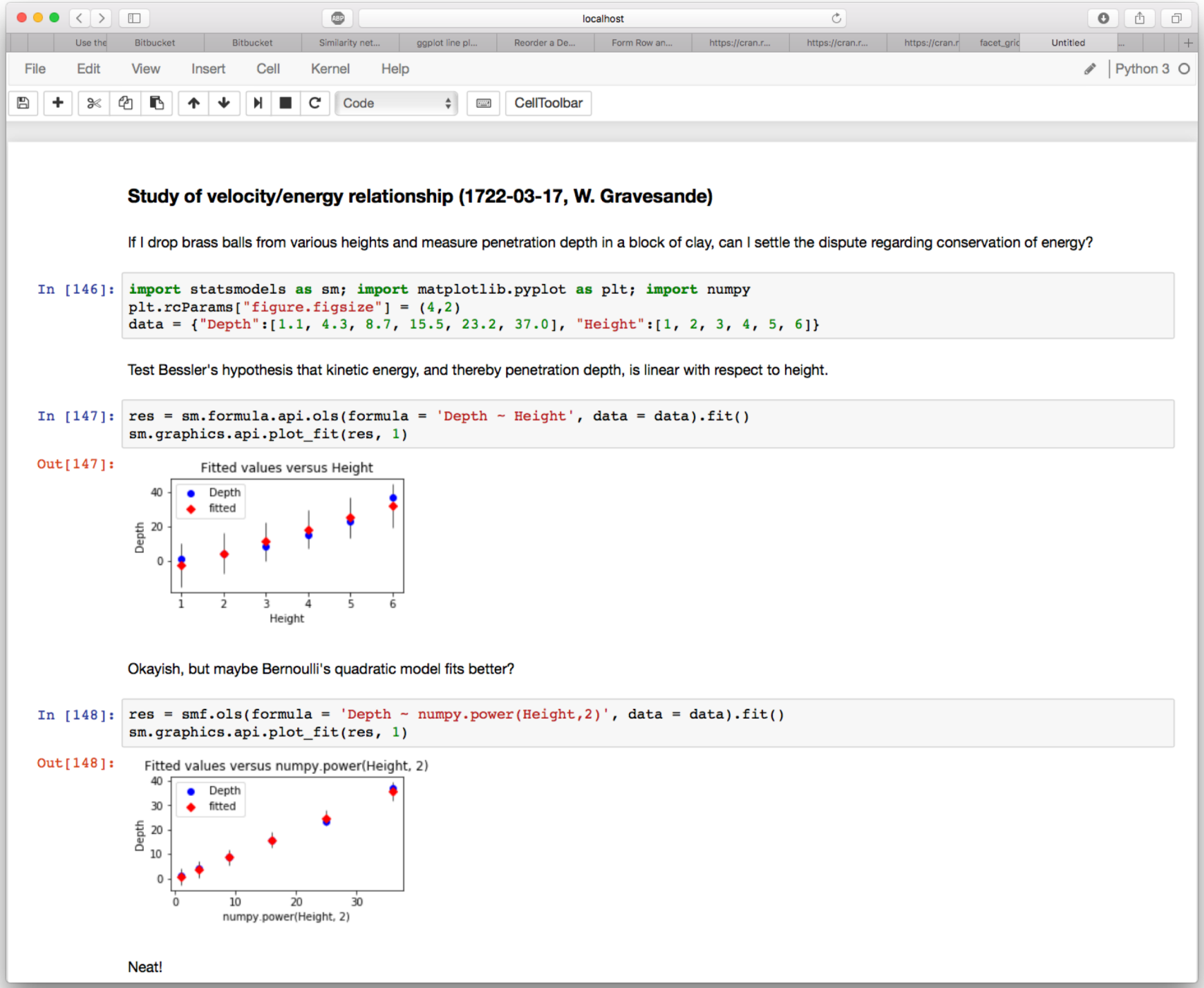
# Literate computing

*A literate computing environment is one that allows users not only to execute commands interactively, but also to store in a literate document the results of these commands along with figures and free-form text.*

Millman KJ and Perez F (2014)




Wolfgang Mathematica notebook (1988)




- The Jupyter Notebook is a web application for *interactive* data science and scientific computing.
- In-browser editing for code, with automatic syntax highlighting, indentation, and tab completion/introspection.
- The ability to execute code from the browser, with the results of computations attached to the code which generated them.
- Mix and match languages to suit your needs (e.g. scikit-learn + ggplot2).

Runs as a local web server →

Load/save/manage notebooks →

Markdown cell with a header →

Code cell with some Python code →

Run shell command to list files →

The notebook itself is a JSON file →

You can define and call functions →

localhost

Home | Untitled

jupyter Untitled Last Checkpoint: 2 hours ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python [conda env:jupyter_exercise]

Run | Code

# My notebook

In [4]:
```
print("Hello world!")
```

```
Hello world!
```

In [5]:
```
!ls
```

```
Snakefile       Untitled.ipynb  code            config.yml      environment.yml
```

In [6]:
```
!head Untitled.ipynb
```

```
{
 "cells": [
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# My notebook"
   ]
  },
  {
```

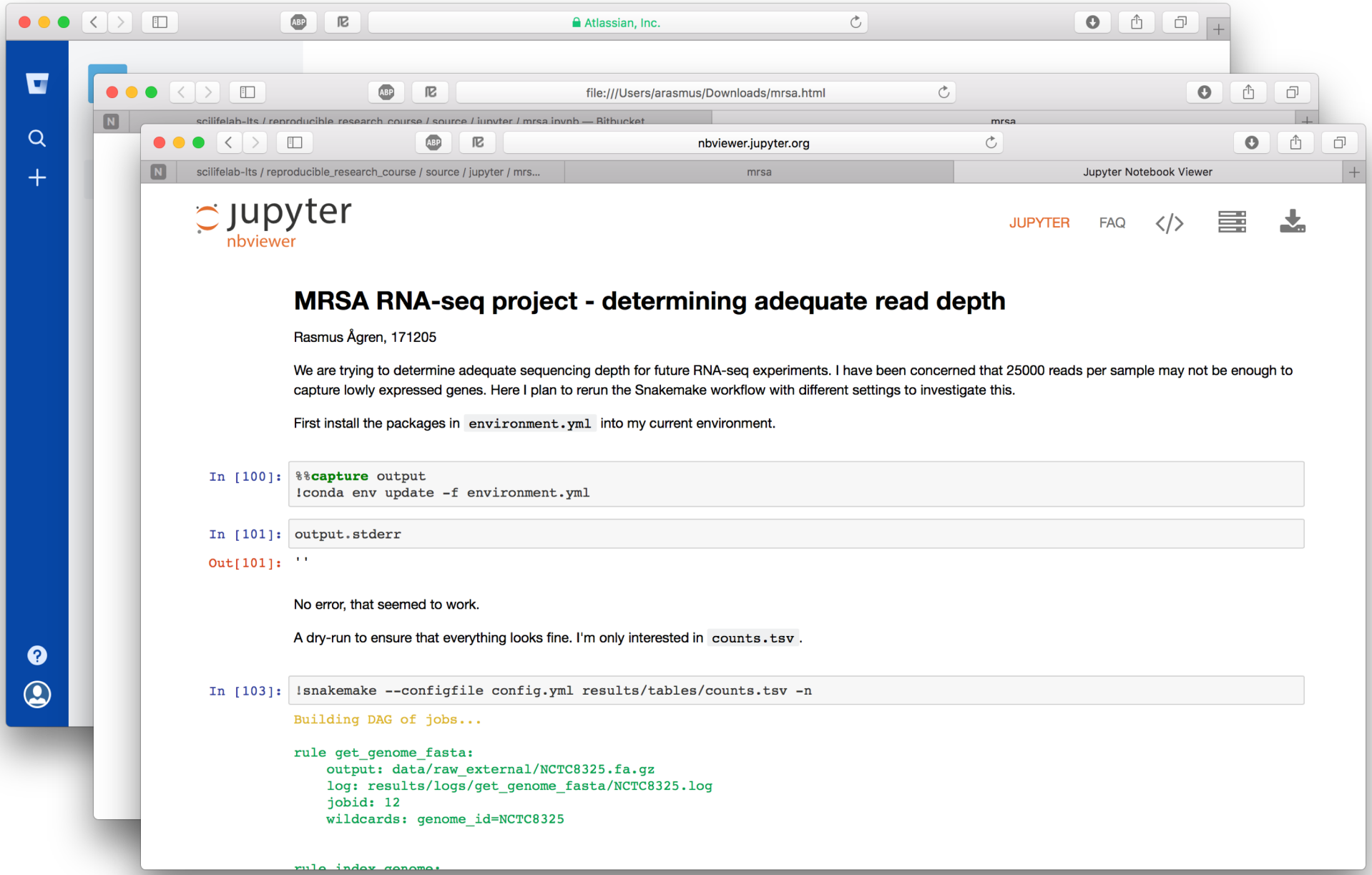
In [7]:
```
def print_me(str):
    print(str)
```

In [8]:
```
print_me("Hi!")
```

```
Hi!
```

In [ ]:

# Sharing is caring

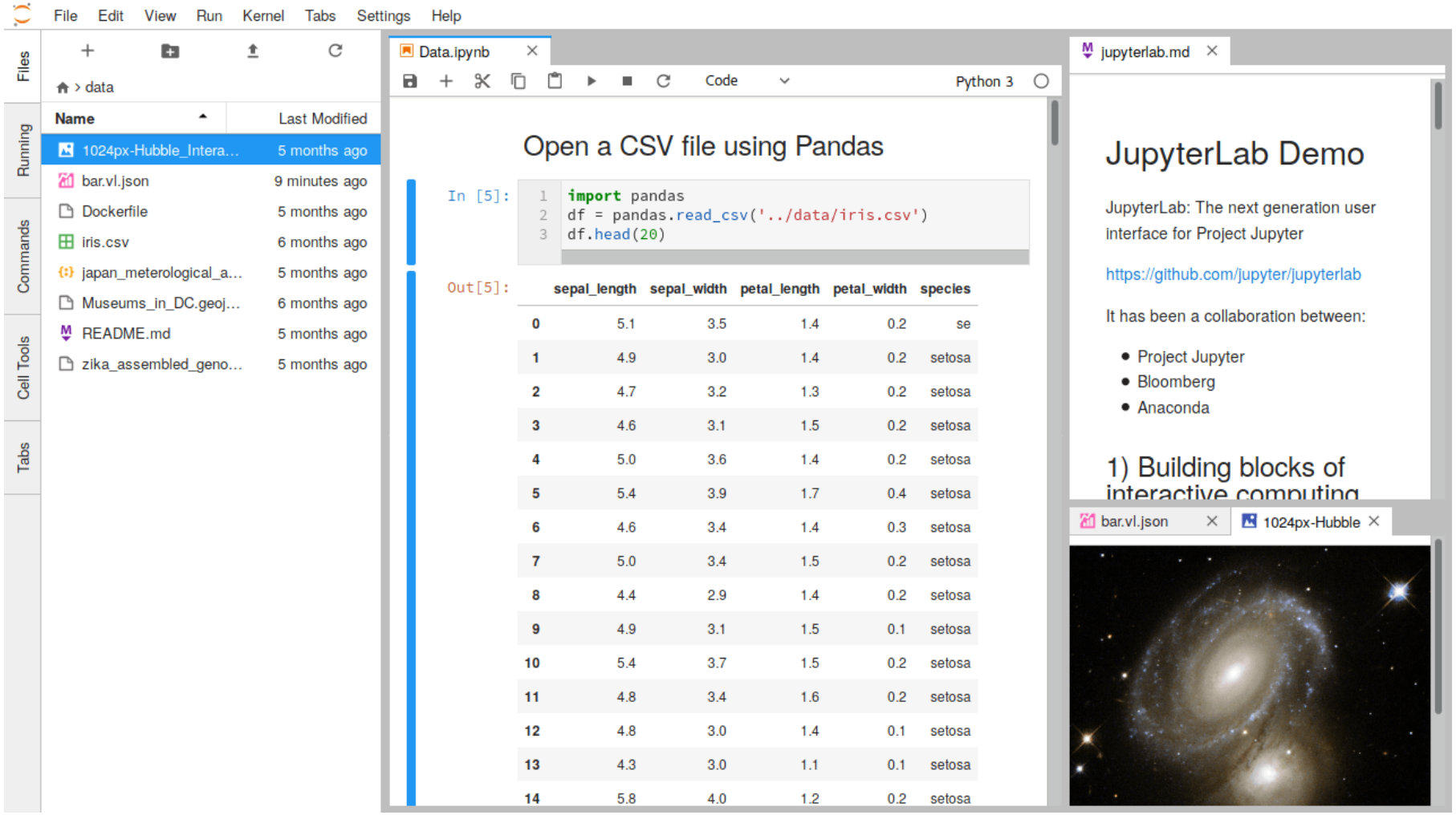



Put the notebook on GitHub/Bitbucket and it will be rendered there..

.. or export to one of many different formats, including HTML and PDF ..

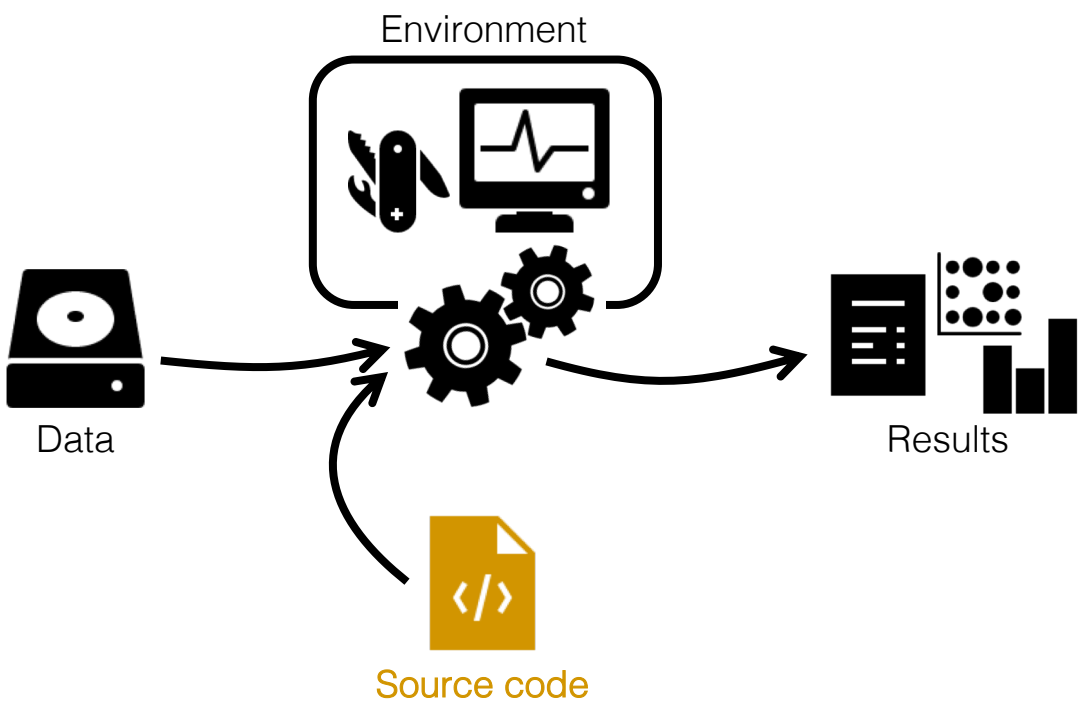
.. or paste a link to any Jupyter notebook at [nbviewer.jupyter.org](nbviewer.jupyter.org) and they will render it for you.

# JupyterLab




JupyterLab is a full-fledged IDE, similar to e.g. Rstudio.

```
conda install -c conda-forge jupyterlab
```




```
project
|- doc/
|
|- data/
|   |- raw_external/
|   |- raw_internal/
|   |- meta/
|
|- code/
|- notebooks/
|
|- intermediate/
|- scratch/
|- logs/
|
|- results/
|   |- figures/
|   |- tables/
|   |- reports/
|
|- Snakefile
|- config.yml
|- environment.yml
|- Dockerfile
```

CONDA

Managing dependencies

Snakemake

Managing and executing analysis workflow

git

Bitbucket

Versioning and collaborating on code (and some other files)

jupyter / R Markdown

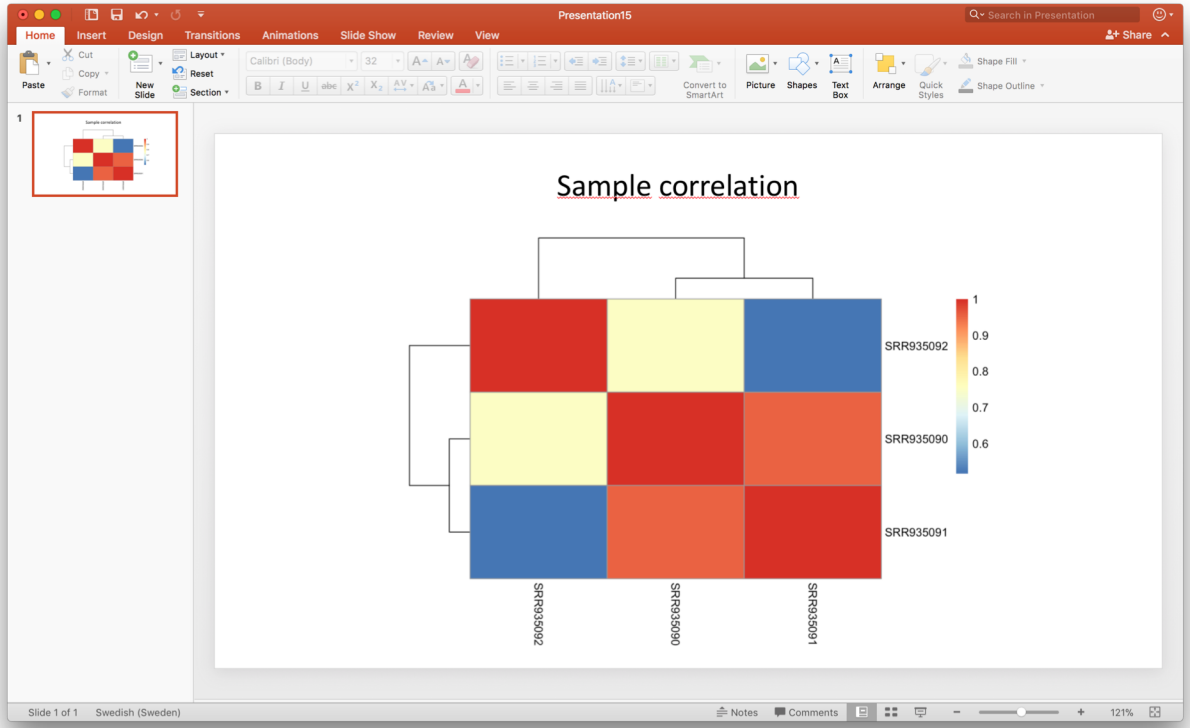
from RStudio

Connecting code and reporting

and…

docker

Isolating and exporting environment




- R Markdown makes your analysis more reproducible by connecting your code, figures and descriptive text.
- You can use it to make reproducible reports, rather than e.g. copy-pasting figures into a Word document.
- You can also use it as a notebook, in the same way as lab notebooks are used in a wet lab setting.

R Markdown

from RStudio

# Supplementary material

*John Doe, Joan Dough, Jan Doh, Dyon Do*

*18 March, 2018*

## Read in the data

We have *count data* for three samples:

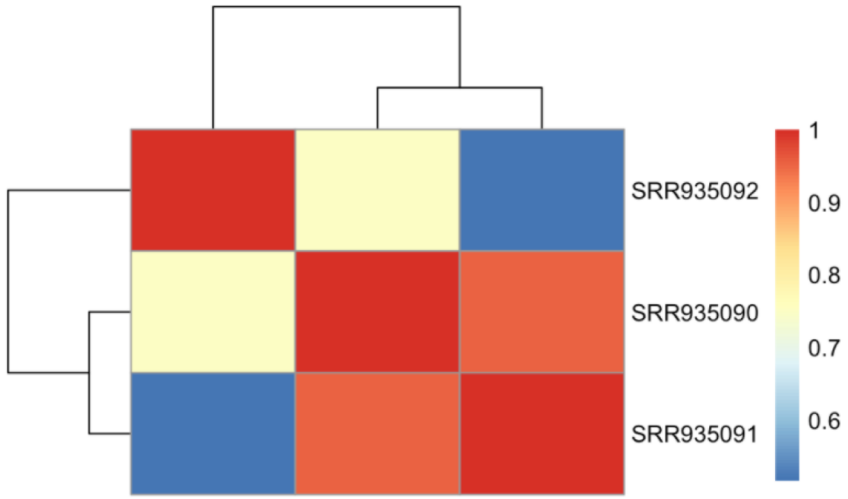
- SRR935090
- SRR935091
- SRR935092

```
# Read counts:
counts_file <- "results/tables/counts.tsv"
counts <- read.delim(counts_file, skip=1, header=F, row.names=1)
sample_names <- t(read.delim(counts_file, nrow=1, header=F))
colnames(counts) <- c("description",gsub(".*(SRR[0-9]*)\\..*","\\1", sample_names))
```

## Plot sample correlation

Next, we will plot the sample correlation of the *count data*:

```
pheatmap(cor((counts[,-1])), show_colnames=F)
```



```
---
title: "Supplementary material"
author: John Doe, Joan Dough, Jan Doh, Dyon Do
date: "`r format(Sys.time(), '%d %B, %Y')`"
output: html_document
---

```{r, include=FALSE}
library("pheatmap")
```

# Read in the data

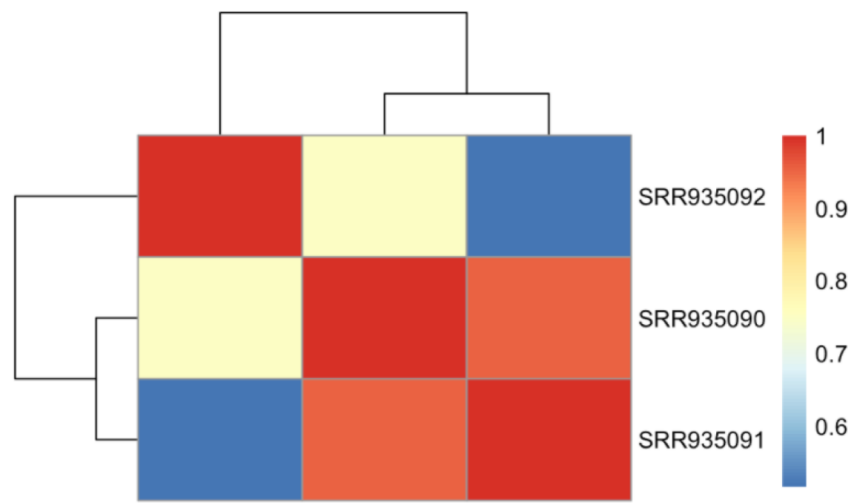
We have *count data* for three samples:

- SRR935090
- SRR935091
- SRR935092

```{r}
# Read counts:
counts_file <- "results/tables/counts.tsv"
counts <- read.delim(counts_file, skip=1, header=F, row.names=1)
sample_names <- t(read.delim(counts_file, nrow=1, header=F))
colnames(counts) <- c("description",
                      gsub(".*(SRR[0-9]*)\\..*","\\1", sample_names))
```

# Plot sample correlation

Next, we will plot the sample correlation of the *count data*:

```{r, fig.height=3, fig.width=5}
pheatmap(cor((counts[,-1])), show_colnames=F)
```
```

R Markdown
from RStudio

# Supplementary material

*John Doe, Joan Dough, Jan Doh, Dyon Do*

*18 March, 2018*

# Read in the data

We have *count data* for three samples:

- SRR935090
- SRR935091
- SRR935092

```
# Read counts:
counts_file <- "results/tables/counts.tsv"
counts <- read.delim(counts_file, skip=1, header=F, row.names=1)
sample_names <- t(read.delim(counts_file, nrow=1, header=F))
colnames(counts) <- c("description",gsub(".*(SRR[0-9]*)\\..*","\\1", sample_names))
```

# Plot sample correlation

Next, we will plot the sample correlation of the *count data*:

```
pheatmap(cor((counts[,-1])), show_colnames=F)
```



```
---
title: "Supplementary material"
author: John Doe, Joan Dough, Jan Doh, Dyon Do
date: "`r format(Sys.time(), '%d %B, %Y')`"
output: html_document
---

```{r, include=FALSE}
library("pheatmap")
```

# Read in the data

We have *count data* for three samples:

- SRR935090
- SRR935091
- SRR935092

```{r}
# Read counts:
counts_file <- "results/tables/counts.tsv"
counts <- read.delim(counts_file, skip=1, header=F, row.names=1)
sample_names <- t(read.delim(counts_file, nrow=1, header=F))
colnames(counts) <- c("description",
                      gsub(".*(SRR[0-9]*)\\..*","\\1", sample_names))
```

# Plot sample correlation

Next, we will plot the sample correlation of the *count data*:

```{r, fig.height=3, fig.width=5}
pheatmap(cor((counts[,-1])), show_colnames=F)
```
```

## Header in YAML format

- Document-wide options
- Output format
- Parameters

```
---
title: "Supplementary material"
author: John Doe, Joan Dough, Jan Doh, Dyon Do
date: "`r format(Sys.time(), '%d %B, %Y')`"
output: html_document
---

```{r, include=FALSE}
library("pheatmap")
```

# Read in the data

We have *count data* for three samples:

- SRR935090
- SRR935091
- SRR935092

```{r}
# Read counts:
counts_file <- "results/tables/counts.tsv"
counts <- read.delim(counts_file, skip=1, header=F, row.names=1)
sample_names <- t(read.delim(counts_file, nrow=1, header=F))
colnames(counts) <- c("description",
                      gsub(".*(SRR[0-9]*)\\..*","\\1", sample_names))
```

# Plot sample correlation

Next, we will plot the sample correlation of the *count data*:

```{r, fig.height=3, fig.width=5}
pheatmap(cor((counts[,-1])), show_colnames=F)
```
```

## Header in YAML format

- Document-wide options
- Output format
- Parameters

## Code chunks

- Evaluate R code and show output
- Also Bash, Python, Rcpp, SQL, Stan
- Chunk options

```
---
title: "Supplementary material"
author: John Doe, Joan Dough, Jan Doh, Dyon Do
date: "`r format(Sys.time(), '%d %B, %Y')`"
output: html_document
---

```{r, include=FALSE}
library("pheatmap")
```

# Read in the data

We have *count data* for three samples:

- SRR935090
- SRR935091
- SRR935092

```{r}
# Read counts:
counts_file <- "results/tables/counts.tsv"
counts <- read.delim(counts_file, skip=1, header=F, row.names=1)
sample_names <- t(read.delim(counts_file, nrow=1, header=F))
colnames(counts) <- c("description",
                      gsub(".*(SRR[0-9]*)\\..*","\\1", sample_names))
```

# Plot sample correlation

Next, we will plot the sample correlation of the *count data*:

```{r, fig.height=3, fig.width=5}
pheatmap(cor((counts[,-1])), show_colnames=F)
```
```

## Header in YAML format

- Document-wide options
- Output format
- Parameters

## Code chunks

- Evaluate R code and show output
- Also Bash, Python, Rcpp, SQL, Stan
- Chunk options

## Markdown text

- Freely add and format text using markdown

```
---
title: "Supplementary material"
author: John Doe, Joan Dough, Jan Doh, Dyon Do
date: "`r format(Sys.time(), '%d %B, %Y')`"
output: html_document
---

```{r, include=FALSE}
library("pheatmap")
```

# Read in the data

We have *count data* for three samples:

- SRR935090
- SRR935091
- SRR935092

```{r}
# Read counts:
counts_file <- "results/tables/counts.tsv"
counts <- read.delim(counts_file, skip=1, header=F, row.names=1)
sample_names <- t(read.delim(counts_file, nrow=1, header=F))
colnames(counts) <- c("description",
                      gsub(".*(SRR[0-9]*)\\..*","\\1", sample_names))
```

# Plot sample correlation

Next, we will plot the sample correlation of the *count data*:

```{r, fig.height=3, fig.width=5}
pheatmap(cor((counts[,-1])), show_colnames=F)
```
```

R Markdown
from RStudio

# Supplementary material

*John Doe, Joan Dough, Jan Doh, Dyon Do*

*18 March, 2018*

# Read in the data

We have *count data* for three samples:

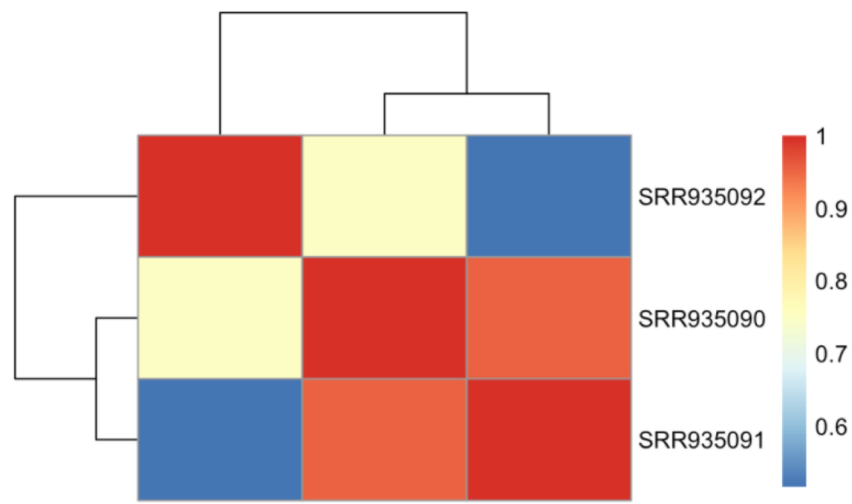
- SRR935090
- SRR935091
- SRR935092

```
# Read counts:
counts_file <- "results/tables/counts.tsv"
counts <- read.delim(counts_file, skip=1, header=F, row.names=1)
sample_names <- t(read.delim(counts_file, nrow=1, header=F))
colnames(counts) <- c("description",gsub(".*(SRR[0-9]*)\\..*","\\1", sample_names))
```

# Plot sample correlation

Next, we will plot the sample correlation of the *count data*:

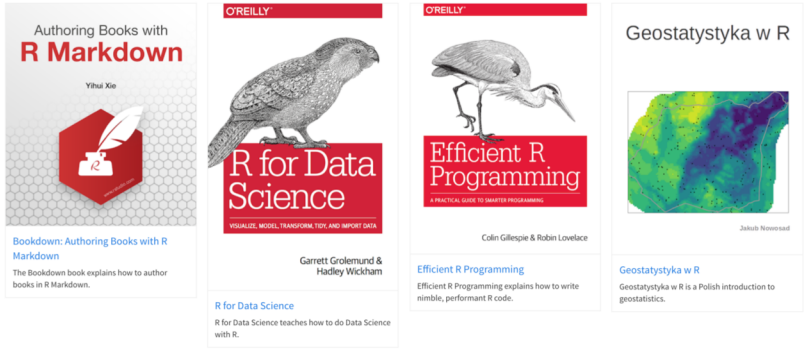
```
pheatmap(cor((counts[,-1])), show_colnames=F)
```

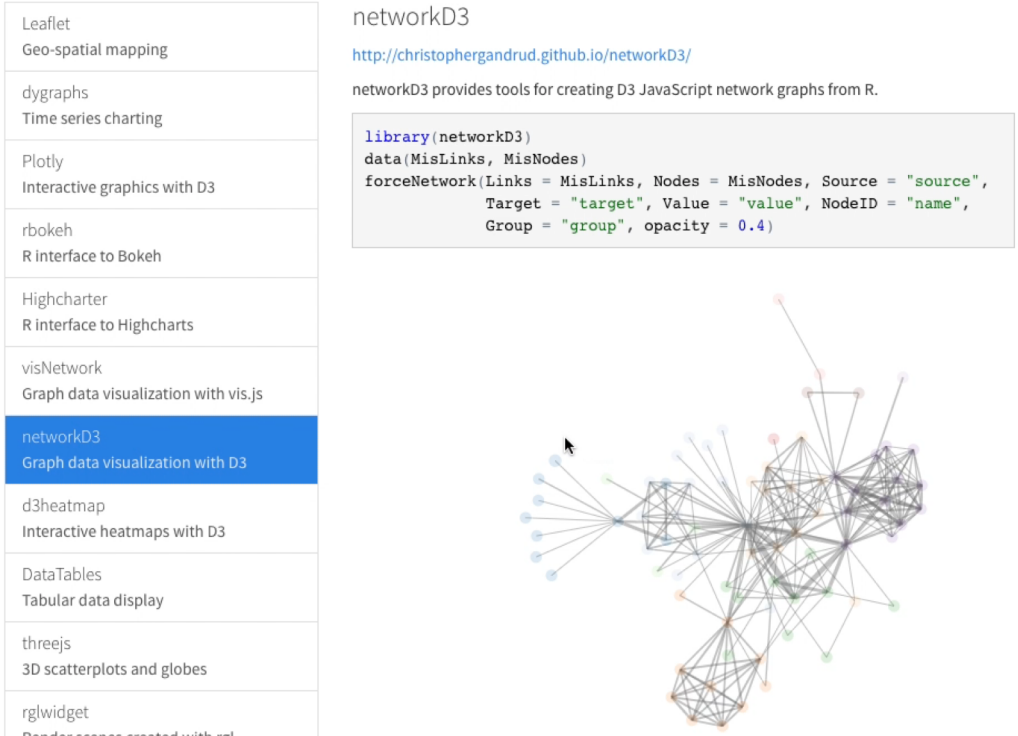


# Output formats

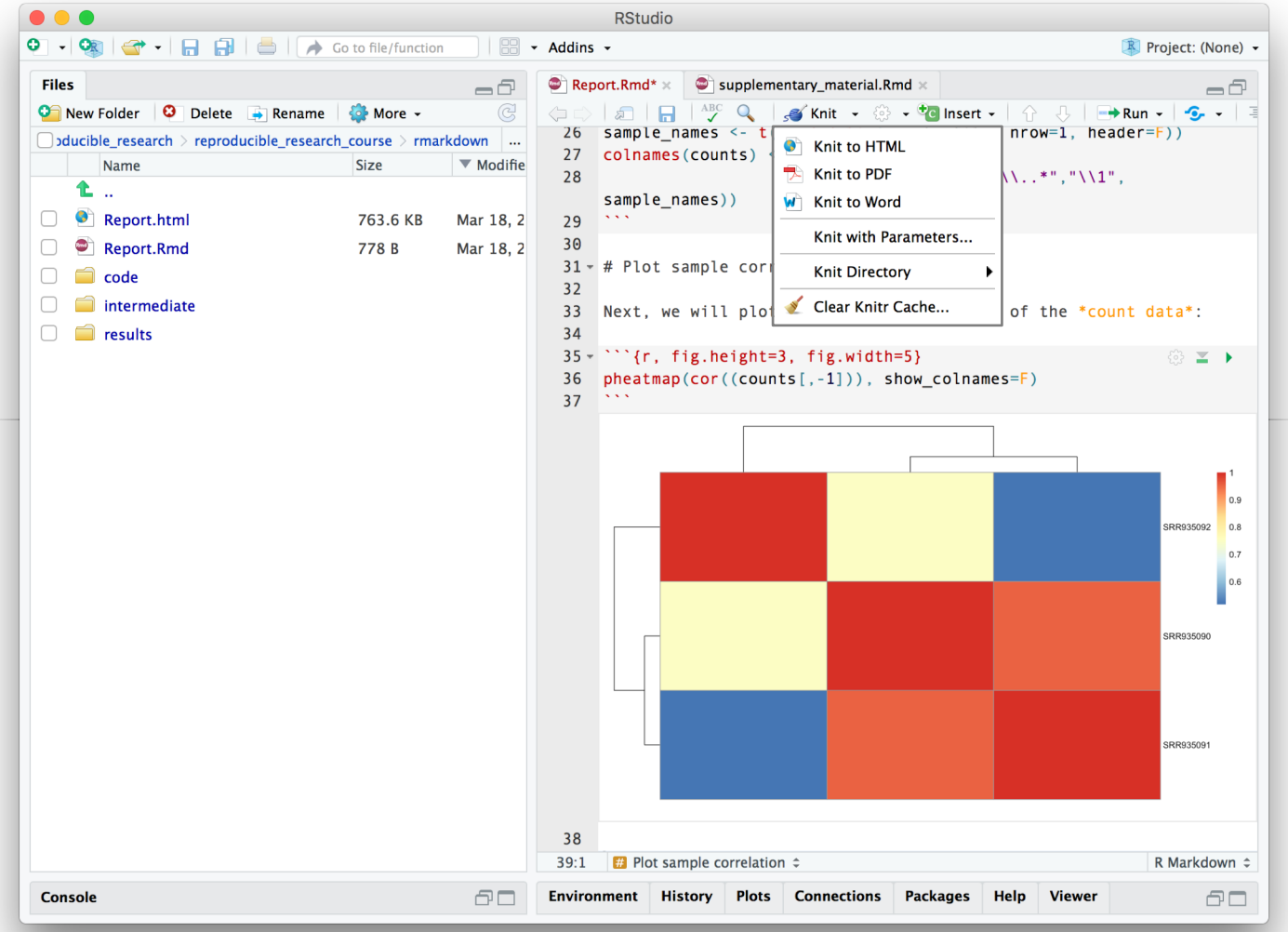
- Documents/reports (HTML, PDF, MS Word, Tufte handouts)
- Presentations (Powerpoint, Beamer, Slidy, ioslides, reveal.js)
- Interactive documents and dashboards (HTML widgets, Shiny)
- Books and websites
- Other templates…

Can require different markdown syntax depending on output!



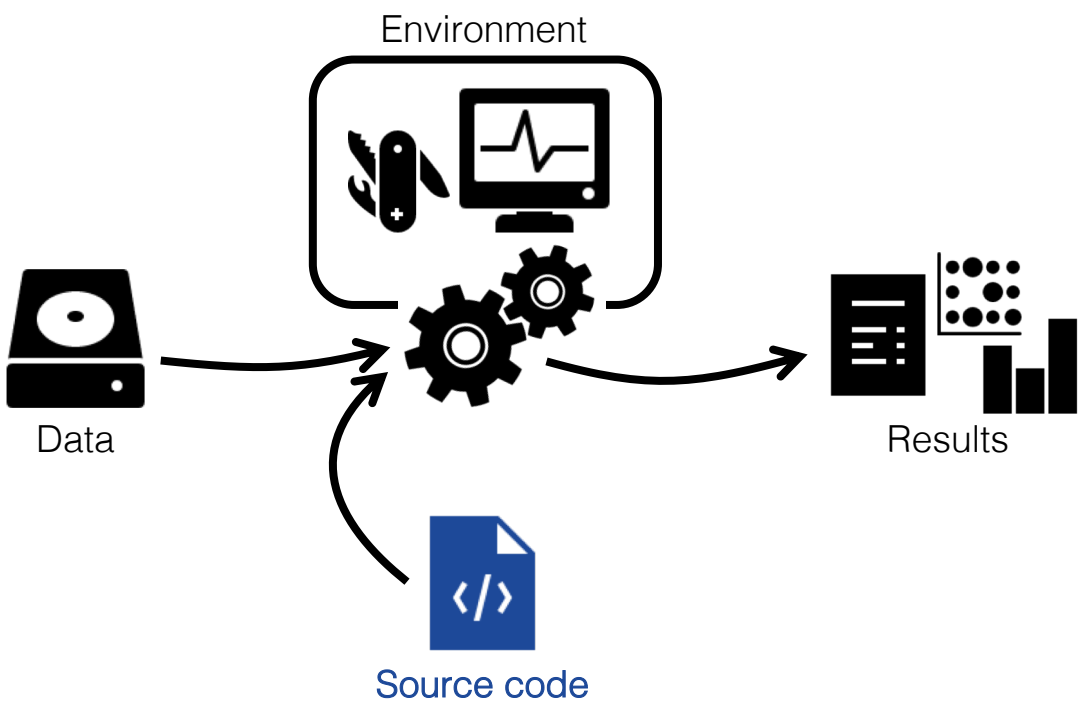



# R Markdown in RStudio




- Evaluate inline
- Render from menu
- Render from R console or terminal

```
$ R -e "rmarkdown::render('Report.Rmd')"
```




```
project
|- doc/
|
|- data/
|   |- raw_external/
|   |- raw_internal/
|   |- meta/
|
|- code/
|- notebooks/
|
|- intermediate/
|- scratch/
|- logs/
|
|- results/
|   |- figures/
|   |- tables/
|   |- reports/
|
|- Snakefile
|- config.yml
|- environment.yml
|- Dockerfile
```

CONDA

Managing dependencies

Snakemake

Managing and executing analysis workflow

git

Bitbucket

Versioning and collaborating on code (and some other files)

jupyter / R Markdown

from RStudio

Connecting code and reporting

and…

docker

Isolating and exporting environment

Results should be possible to reproduce
regardless of platform and with minimal effort.


“Docker provides a way to run applications securely
isolated in a container, packaged with all its
dependencies and libraries."

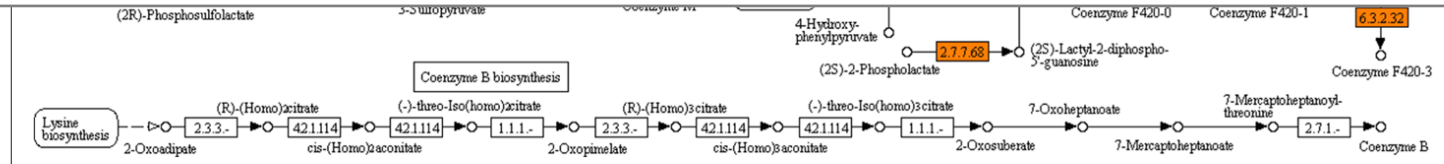



**Fig. 1** Methane metabolism pathway analysis. Genes reconstructed in our assembly that are involved in the methane metabolism [PATH:ko00680, (http://www.genome.jp/kegg-bin/show_pathway?ko00680)], are highlighted: genes with only metagenomic support are in yellow and genes with metatranscriptomic support as well, suggesting active gene expression, are in orange. Methane is synthesized from $CO_2$, methanol or acetate. KEGG pathway map courtesy of Kanehisa Laboratories

## Discussion

We report extensive metagenomic and metatranscriptomic profiling of the microbial community from a production-scale biogas plant. Given the unprecedented sequencing depth and established bioinformatics, our data will be of great interest to the biogas research community in general and microbiologists working on biogas-producing microbial communities in particular. In a first applied study, our metagenome assembly was used to improve the characterization of a metaproteome generated from biogas plant fermentation samples and to investigate the metabolic activity of the microbial community [17].

By sharing our data, we want to actively encourage its reuse. This will hopefully result in novel biological and biotechnological insights, eventually enabling a more efficient biogas production.

## Availability of supporting data

### Data accession

Raw sequencing data are available in the European Nucleotide Archive (ENA) under study accession PRJEB8813 (http://www.ebi.ac.uk/ena/data/view/PRJEB8813). The datasets supporting the results of this article are available in *GigaScience*'s GigaDB [2].

### Reproducibility

The complete workflow is organized in a single GNU Makefile and available on GitHub [18]. All data and results can be reproduced by a simple invocation of *make*. To further support reproducibility, we bundled all tools and dependencies into one Docker container available on DockerHub [19]. *docker run* executes the aforementioned Makefile inside the container. Reproduction

Bremges et al., "*Deeply sequenced metagenome and metatranscriptome of a biogas-producing microbial community from an agricultural production-scale biogas plant*", GigaScience (2015) 4:33, doi:10.1186/s13742-015-0073-6

```
$uname –a
Darwin dhcp-140-26.vpn.chalmers.se 15.6.0 Darwin Kernel Version 15.6.0:
Thu Sep  1 15:01:16 PDT 2016; root:xnu-3248.60.11~2/RELEASE_X86_64
x86_64

$docker pull ubuntu:16.04
16.04: Pulling from library/ubuntu
22dc81ace0ea: Pull complete
1a8b3c87dba3: Pull complete
91390a1c435a: Pull complete
07844b14977e: Pull complete
b78396653dae: Pull complete
Digest:
sha256:e348fbbea0e0a0e73ab0370de151e7800684445c509d46195aef73e090a49bd6
Status: Downloaded newer image for ubuntu:16.04

$docker run -it ubuntu:16.04
root@407b0fd13fe5:/# uname -a
Linux 407b0fd13fe5 4.9.60-linuxkit-aufs #1 SMP Mon Nov 6 16:00:12 UTC
2017 x86_64 x86_64 x86_64 GNU/Linux
```

Dockerfile

```
FROM ubuntu:16.04

# Install prerequisites
RUN apt-get update && \
    apt-get install -y --no-install-recommends \
    bzip2 curl ca-certificates

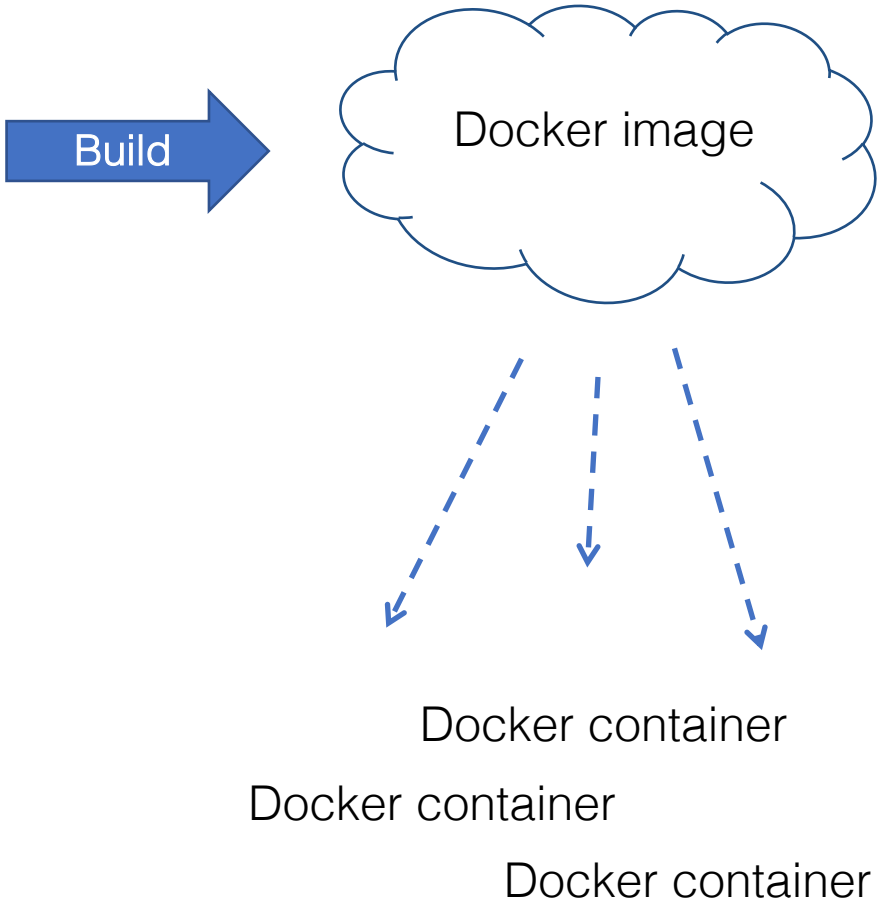
# Install Conda
RUN curl https://repo.continuum.io/miniconda.sh -O && \
    bash miniconda.sh -bf -p /opt/miniconda3/ && \
    rm miniconda.sh

# Add conda to PATH
ENV PATH="/opt/miniconda3/bin:${PATH}”

# Install git and nano from conda-forge
RUN conda install -c conda-forge git nano

# Use bash as shell
SHELL ["/bin/bash", "-c"]

# Set workdir
WORKDIR /home
```



# Mounting volumes

## Local project directory

```
project
|- doc/
|
|- data/
|   |- raw_external/
|   |- raw_internal/
|   |- meta/
|
|- code/
|- notebooks/
|
|- intermediate/
|- scratch/
|- logs/
|
|- results/
|   |- figures/
|   |- tables/
|   |- reports/
|
|- Snakefile
|- config.yml
|- environment.yml
|- Dockerfile
```

## Docker image file system

```
|- bin/
|- boot/
|- dev/
|- etc/
|- home/
|   |- data/
|       |- raw_external/
|       |- raw_internal/
|       |- meta/
|- lib/
|- lib64/
|- media/
|- opt/
|- proc/
|- root/
|- run/
|- sys/
|- tmp/
|- usr/
|- var/
```

```
$docker run -it -v $PWD/data:/home/data ubuntu:16.04
```

# What can I use Docker for?

As an advanced environment manager.


```
$docker run -it
   -v $PWD:/home
   my_image /bin/bash
```

To package your code with the environment it needs.


```
$docker run
   -v $PWD/data:/home/data
   -v $PWD/results:/home/results
   my_image snakemake report.pdf
```

To package a whole workflow, e.g. to accompany a manuscript.


```
$docker run
   -v $PWD/results:/home/results
   my_image snakemake report.pdf
```

# A Docker image

Dockerfile

```
FROM debian:latest

RUN conda install star

ADD config.yml

…
```

me$ docker run image_id

OS kernel — From host system

ob3b79cf: Base image

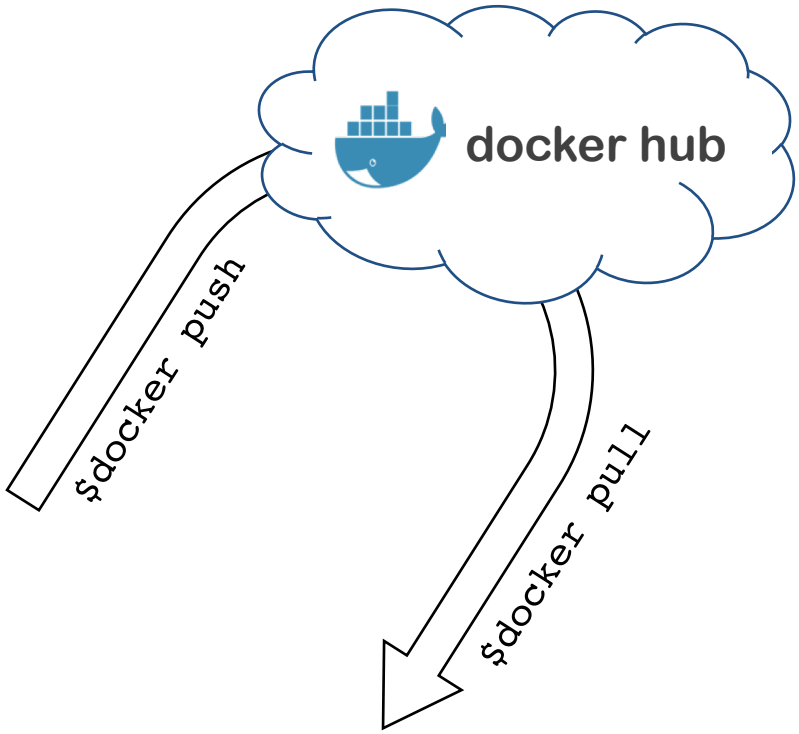
6f0d2f5e: Star installed

1904ae78: Config added

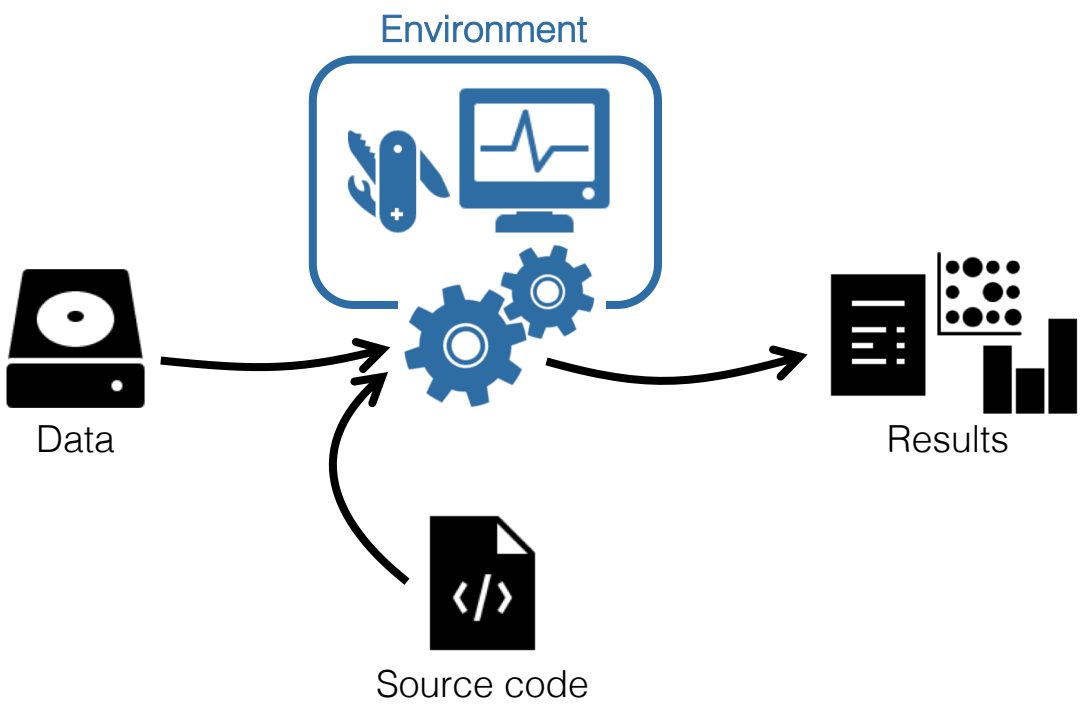
C471fd60: …

Read-only layers of diffs

9c774b63ce: Container x — Writable layer



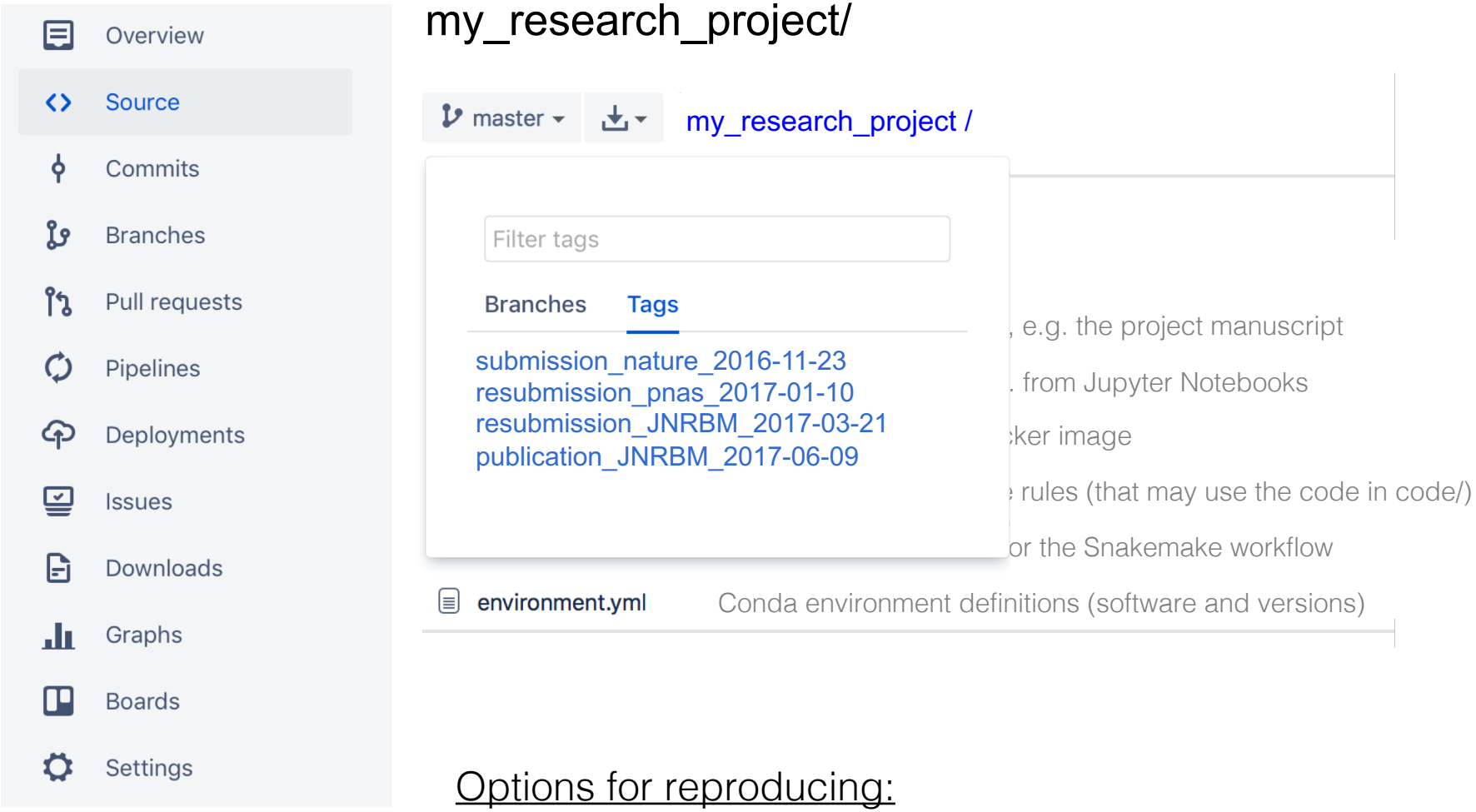

you$ docker run image_id




```
project
|- doc/
|
|- data/
|   |- raw_external/
|   |- raw_internal/
|   |- meta/
|
|- code/
|- notebooks/
|
|- intermediate/
|- scratch/
|- logs/
|
|- results/
|   |- figures/
|   |- tables/
|   |- reports/
|
|- Snakefile
|- config.yml
|- environment.yml
|- Dockerfile
```

# Putting it all together

# my_research_project/




<u>Options for reproducing:</u>

- 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…

Minimal: code for reproducible results

Snakemake

R Markdown

from RStudio

jupyter

Good: versioned and structured repository

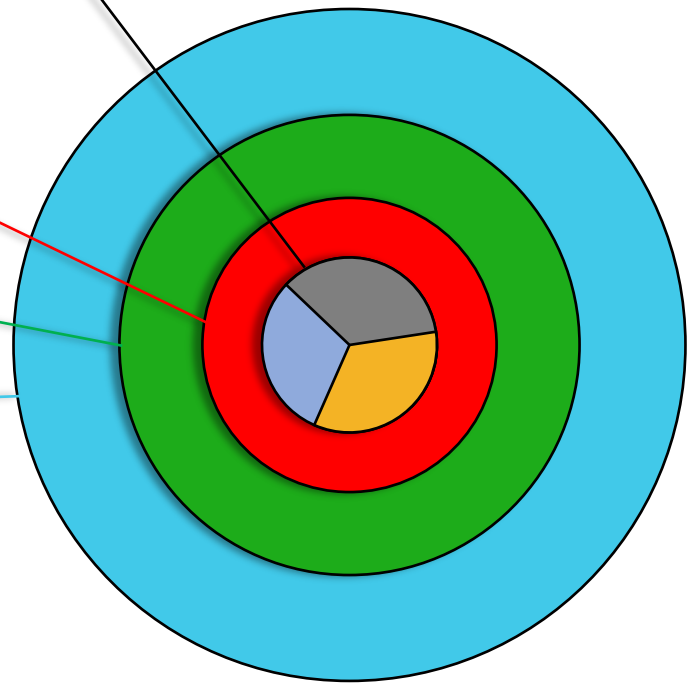
git

Better: ambition to organize dependencies

CONDA

Best: export everything!

docker



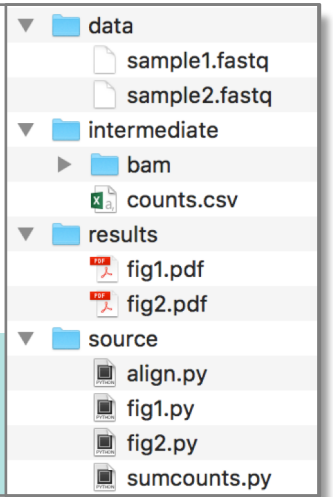
# Reproducible research for bioinformatics projects

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.



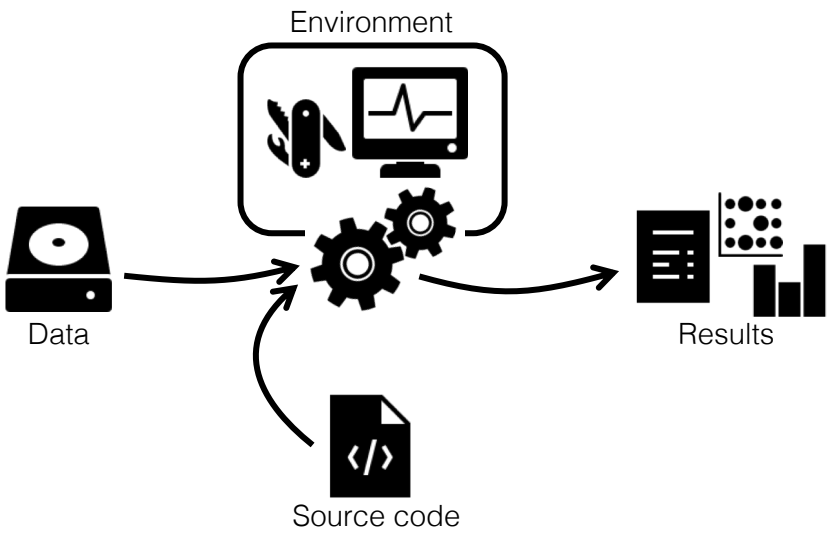

**Tip!** Learn how to use git, a widely used system (both in academia and industry) for version controlling and collaborating on code.

https://git-scm.com/

## 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.




## 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/

## 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).

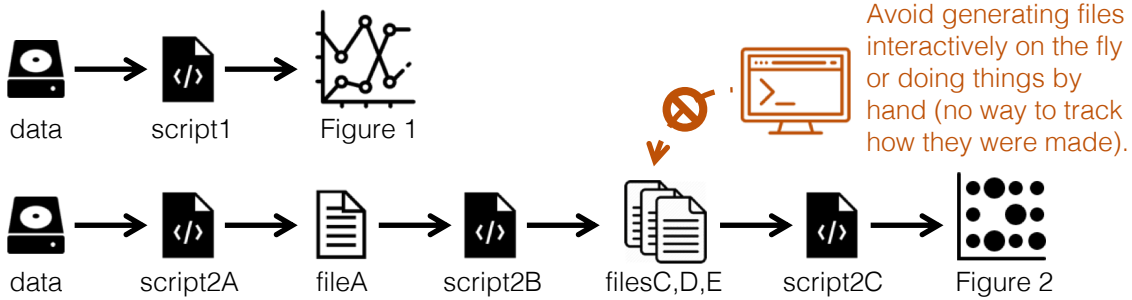
## Master your dependencies

- 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

## 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)




## 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/

## 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. |