Snakemake

A framework for reproducible data analysis

Microbiome group
Workshop August 2021

Cristina Leal

COPSAC

COPENHAGEN PROSPECTIVE STUDIES
ON ASTHMA IN CHILDHOOD

UNIVERSITY OF
COPENHAGEN

REGION H
Herlev og Gentofte
Hospital

Agenda

Why for and what is Snakemake

Rules: the lego pieces of your workflow

The Snakefile: the file that rules them all

Run your workflow

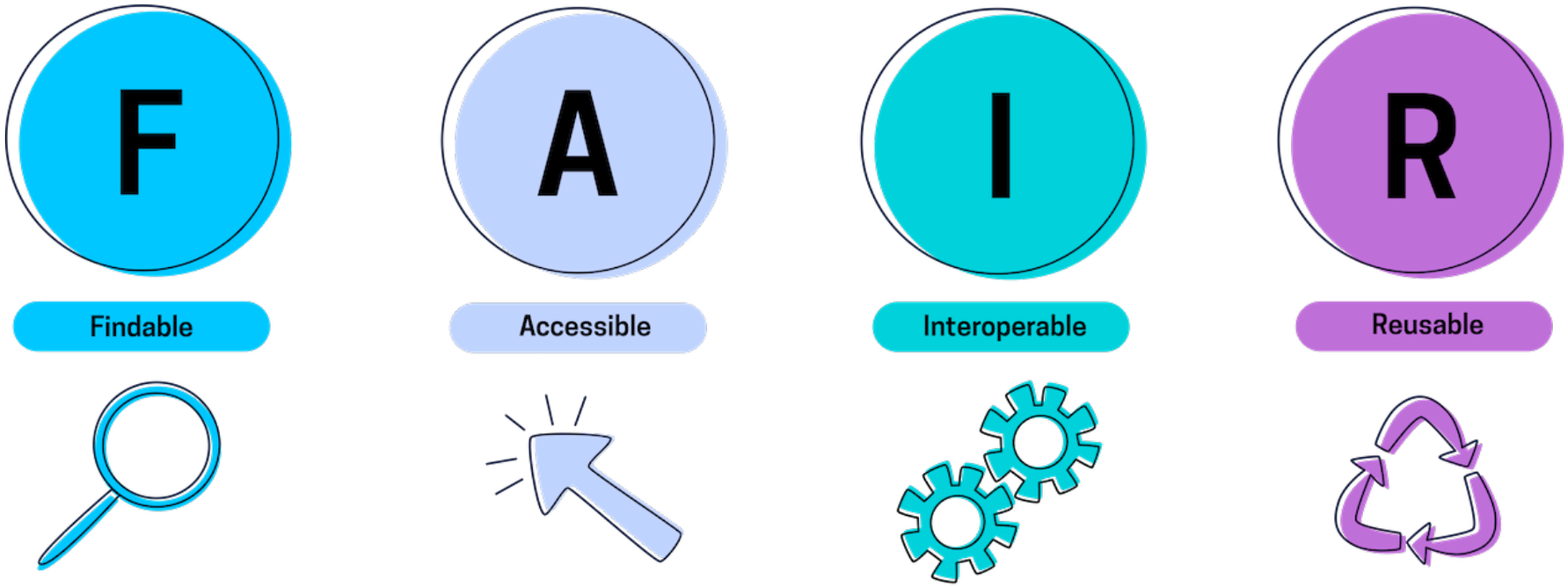
Visualise your workflow

Benchmarking and report

Tips for distribution and reproducibility: the folder structure

… hands-on and play around with some snakemake features I have prepared

All components of the research process must be available to ensure transparency, reproducibility, and reusability



Wilkinson, Mark D., et al. "The FAIR Guiding Principles for scientific data management and stewardship." *Scientific data* 3.1 (2016): 1-9.
https://flur.ee/2020/07/08/making-data-fair/

Snakemake

What is it?

- **Workflow management system (WMS)**

  Provides an infrastructure for the set-up, performance and monitoring of a defined sequence of tasks arranged as a workflow application

- Based on **Python**
- **Reproducible** because…
  1. **Scalable**

     Optimizes the number of processes that can be run in parallel wrt CPU cores and needed threads

  2. **Portable**

     Runs across many different platforms

  3. **Automated**

  Makes sure everything is **up-to-date**

https://snakemake.readthedocs.io/

Snakemake is popular

Volume 28, Issue 19
1 October 2012

Article Contents

Abstract
1 INTRODUCTION
2 SNAKEMAKE LANGUAGE
3 SNAKEMAKE ENGINE
ACKNOWLEDGEMENTS
REFERENCES
Author notes

< Previous Next >

Snakemake—a scalable bioinformatics workflow engine FREE

Johannes Köster, Sven Rahmann Author Notes

Bioinformatics, Volume 28, Issue 19, 1 October 2012, Pages 2520–2522,
https://doi.org/10.1093/bioinforma
Published: 20 August 2012 **Arti**

A correction has been published:
Bioinformatics, Volume 34, Issue
https://doi.org/10.1093/bioinfor

PDF Split View Cite Per

Abstract

Summary: Snakemake is a workflow engine
based workflow definition language and a p
scales from single-core workstations to compute clusters without modifying the workflow. It is the first system to support the use of automatically inferred multiple named wildcards (or variables) in input and output filenames.

Availability: http://snakemake.googlecode.com.

Contact: johannes.koester@uni-due.de

Issue Section: GENOME ANALYSIS

Accessed 10/08/2021

Snakemake—a scalable bioinformatics workflow engine
J Köster, S Rahmann - Bioinformatics, 2012 - academic.oup.com
Snakemake is a workflow engine that provides a readable Python-based workflow definition language and a powerful execution environment that scales from single-core workstations to compute clusters without modifying the workflow. It is the first system to support the use of …
☆ Citado por 1402 Artículos relacionados Las 16 versiones

Widely used and accepted for reproducible data science. On average, it has **6 new citations per week** !!

Köster, Johannes, and Sven Rahmann. "Snakemake—a scalable bioinformatics workflow engine." *Bioinformatics* 28.19 (2012): 2520-2522.

Rules: the lego pieces of your workflow

Snakemake defines workflows in terms of rules

Rule name

```
rule mytask:
        input:
                sample_data = "data/{sample}.txt"
        output:
                outfile = "outputs/{result}.txt"
    shell:
        "some-tool {input} > {output}"
```

Generalizable wildcard

Define

Input

Output

Log files

Parameters

Resources

…

How to create output from input

Rules: the lego pieces of your workflow

Snakemake defines workflows in terms of rules

Boilerplate-free integration of scripts

```
rule mytask:
        input:
                sample_data = "data/{sample}.txt"
        output:
                outfile = "outputs/{result}.txt"
        script:
                "script/myscript.R"
```

R/Python/Julia ←

Boilerplate-free integration of R and python scripts

R scripts

```r
for (p in c("data.table", "dplyr")) {
        library(p, character.only = TRUE)
}

data <- fread(snakemake@input$sample_data) %>%
        arrange(desc(id))
fwrite(data, snakemake@output$outfile)
```

Python scripts

```python
import pandas as pd

data = pd.read_table(snakemake.input["sample_file"])
data = data.sort_values("id")
data.to_csv(snakemake.output["outfile"], sep="\t")
```


Rules: the lego pieces of your workflow

Snakemake defines workflows in terms of rules

…and more!

Jupyter notebook integration

```
rule mytask:
        input:
                sample_data = "data/{sample}.txt"
        output:
                outfile = "outputs/{result}.txt"
        notebook:
                "notebooks/mynotebook.ipynb"
```

Reusable wrappers from central snakemake repo

```
rule sort_reads:
        input:
                sample_data = "data/{sample}.bam"
        output:
                outfile = "outputs/{sample}.sorted.bam"
        wrapper:
                "0.22.0/bio/samtools/sort"
```

e.g. DADA2

https://snakemake-wrappers.readthedocs.io/en/stable/

The snakefile

The file that rules them all

```
configfile: config.yaml

rule all:
        input:
                "a.tsv"
                "b.tsv"
                "b.png"

rule mytask_1:
        input: config["input_data"]
        output: "a.tsv"
        threads: 1
        script: "scripts/01_mytask.py"

rule mytask_2:
        input: config["input_data"]
        output:
                outfile = "b.tsv",
                fig = "b.png"
        threads: 1
        script: "scripts/02_mytask.R"
```


Run your workflow

Portability and flexibility in many different platforms

Perform a dry-run

```
snakemake -n
```

Execute workflow locally with 16 CPU cores

```
snakemake --cores 16
```

Execute on cluster

```
snakemake --cluster qsub --jobs 100
```

Execute in the cloud

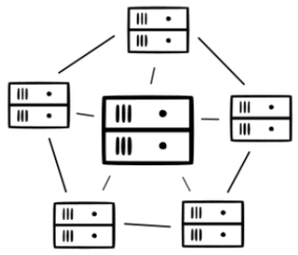
```
snakemake --k
```

workstation

compute server

cluster

grid computing

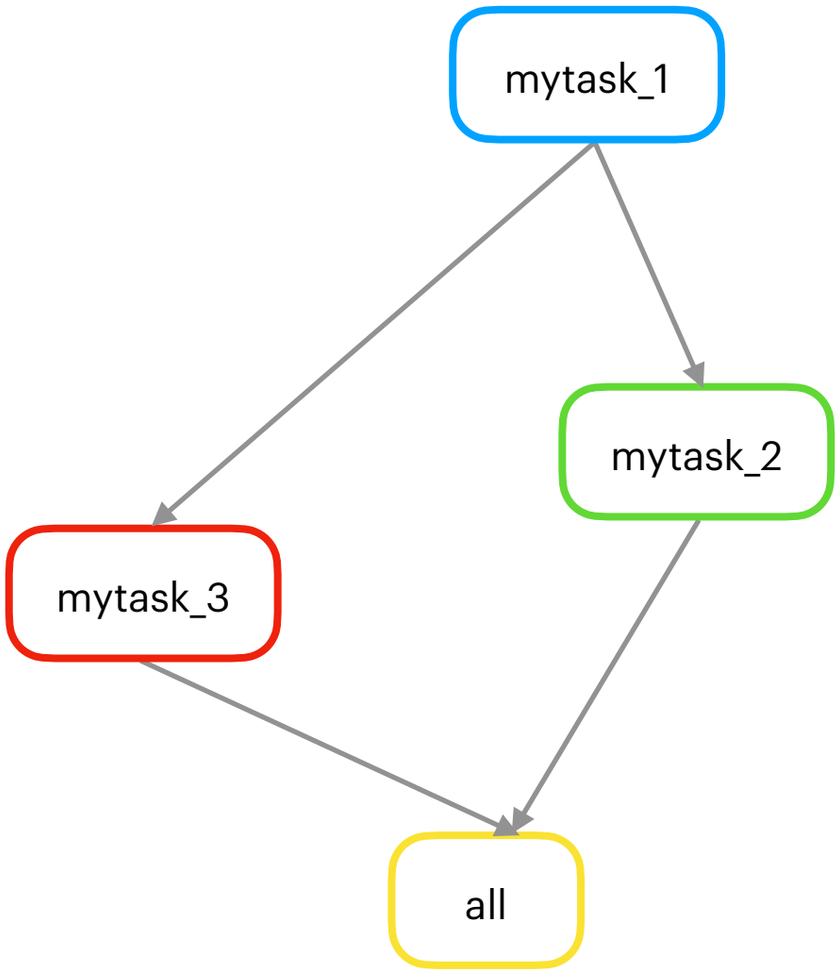


cloud computing

Visualise your workflow

The **rulegraph** diagram

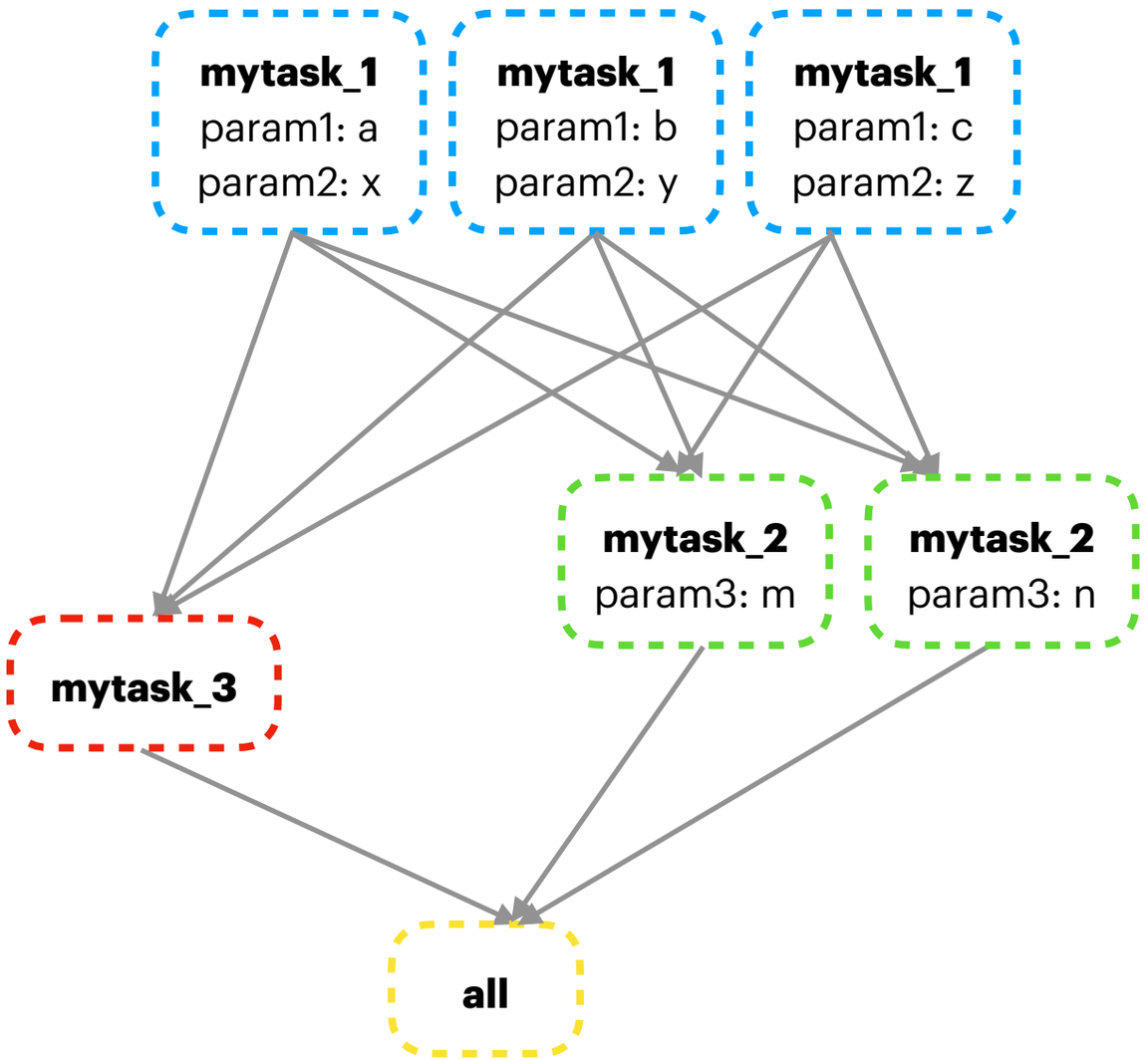
```
snakemake --rulegraph | dot -Tpng > workflow/rulegraph.png
```

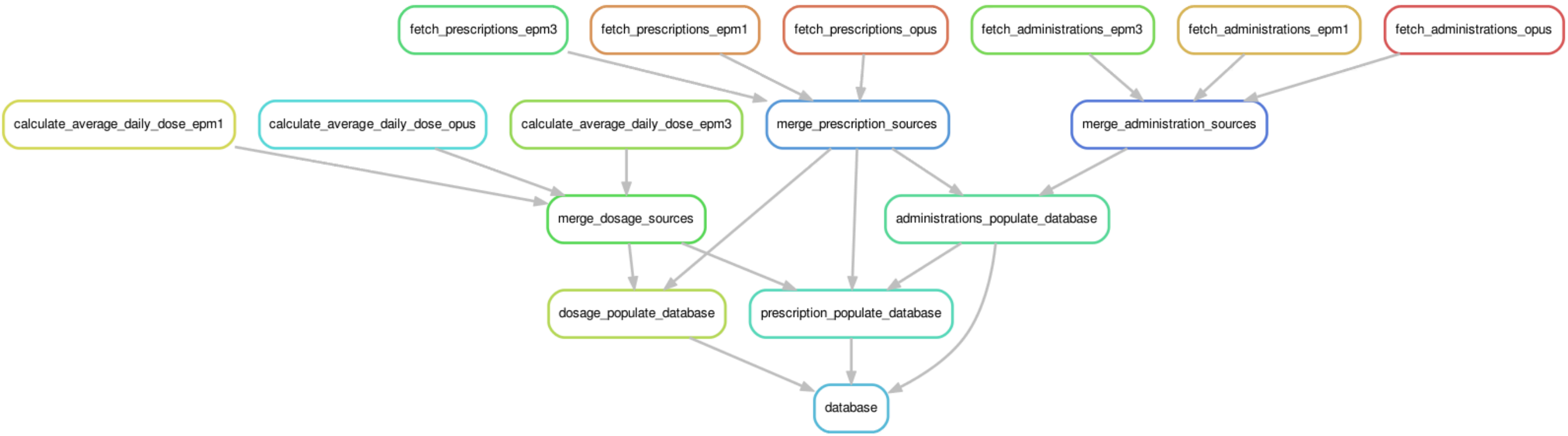


Visualise your workflow

The **dag** diagram

```
snakemake --dag | dot -Tpng > workflow/dag.png
```




fetch_prescriptions_epm3
fetch_prescriptions_epm1
fetch_prescriptions_opus
fetch_administrations_epm3
fetch_administrations_epm1
fetch_administrations_opus
calculate_average_daily_dose_epm1
calculate_average_daily_dose_opus
calculate_average_daily_dose_epm3
merge_prescription_sources
merge_administration_sources
merge_dosage_sources
administrations_populate_database
dosage_populate_database
prescription_populate_database
database

Workflow benchmarking and report

Workflow statistics, runtime, params, etc.

Keep **control** of the time and resources used, parameters, and the outputs

- Benchmarks: define it within your rules
- A beautiful and interactive report

```
snakemake --report workflow/report.html
```

https://snakemake.readthedocs.io/en/stable/snakefiles/reporting.html

Distribution and reproducibility

Your workflow folder structure

tutorial/structure.md

```
my_project
├── .gitignore
├── README.md
├── config
│   ├── config.yaml
├── workflow
│   ├── envs
│   │   ├── environment.yaml
│   ├── benchmarks
│   ├── scripts
│   │   ├── script1.py
│   │   └── script2.R
│   ├── notebooks
│   │   ├── notebook1.py.ipynb
│   │   └── notebook2.r.ipynb
│   ├── report
│   │   ├── report.html
│   └── Snakefile
├── results
├── logs
└── resources
```


Hands-on

https://github.com/crlero/snakemake-tutorial

Then, download this repository to your favourite local computer. If you already have `git` installed, you can download it with:

```
git clone https://github.com/crlero/snakemake-tutorial.git
# or
git clone git@github.com:crlero/snakemake-tutorial.git
```

... otherwise download the `ZIP` repository from the `Code` button on the up-right side of this page.

First, we start by creating our project environment

tutorial/create_your_environment.md

Install miniconda3

If you don't have conda (miniconda, conda or Anaconda) yet, install miniconda3.

```
bash Miniconda3-latest-MacOSX-x86_64.sh
```

Follow the prompts on the installer screens and test the installation with:

```
conda list
```

Set up your conda channels

```
conda config --add channels defaults
conda config --add channels bioconda
conda config --add channels conda-forge
```


First, we start by creating our project environment

tutorial/create_your_environment.md

Create a new environment

```
conda create -n tutorial -c conda-forge mamba
conda activate tutorial
```

Installing needed packages and dependencies

From conda (recommended)

```
conda install snakemake
conda install r-essentials
conda install bioconductor-microbiome
conda install bioconductor-phyloseq
conda install bioconductor-metagenomeseq
conda install r-vegan
conda install r-cowplot
conda install r-ggthemes
conda install r-ggsci
```

First, we start by creating our project environment

tutorial/create_your_environment.md

Installing needed packages and dependencies

From R (not recommended, and if so add a rule to your workflow to download the packages)

```
$ R
>> if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")

>> BiocManager::install("phyloseq")
>> BiocManager::install("microbiome")
>> install.packages("vegan")
```

When your project env is ready, you are ready to start writing your scripts

tutorial/create_your_environment.md

Testing and scripting using RStudio with your environment

Open a `tmux` screen, activate your environment and launch your RStudio. Now you are all set up for start writing testing your own code.

```
tmux new-session -s rstudio
conda activate tutorial
/Applications/RStudio.app/Contents/MacOS/RStudio
```


Recreate this tutorial environment

In order to be able to run this workflow example, we must ensure we are running it within a defined environment. To do so, you can either create a new conda environment using the `yaml` file and download the specific dependencies defined in it or you can activate the environment from the zipped folder that I have placed in the NAS server to speed up the slow stuff :-).

Create conda environment from the .yaml file

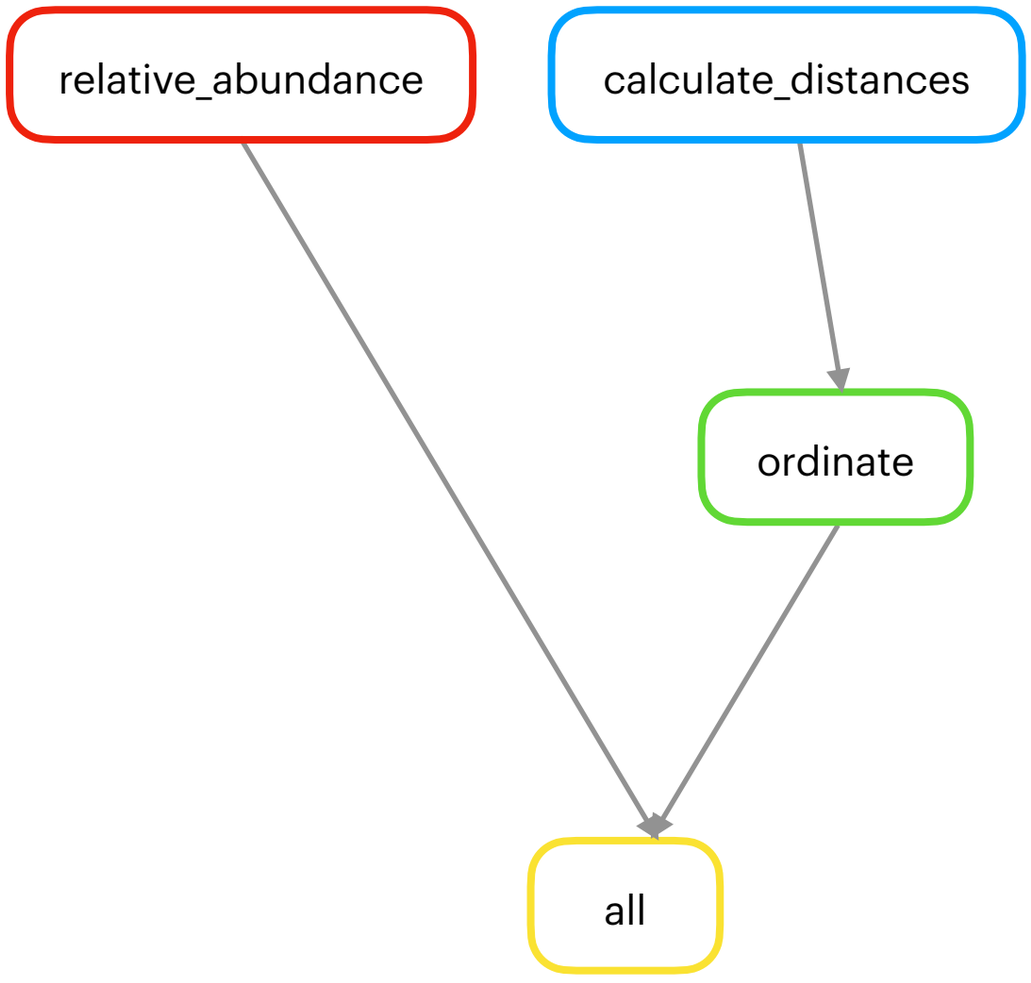
```
cd snakemake-tutorial/handson/
conda env create --name tutorial --file=workflow/environment.yaml
conda activate tutorial
```

Activate folder with the environment

```
mkdir environment_tutorial
tar xzvf /Volumes/UserFolders/cristina.leal/snakemake-tutorial/environment_tutorial.tar.gz -C enviro
source environment_tutorial/bin/activate # activates the environment
source environment_tutorial/bin/deactivate # deactivates the environment
```

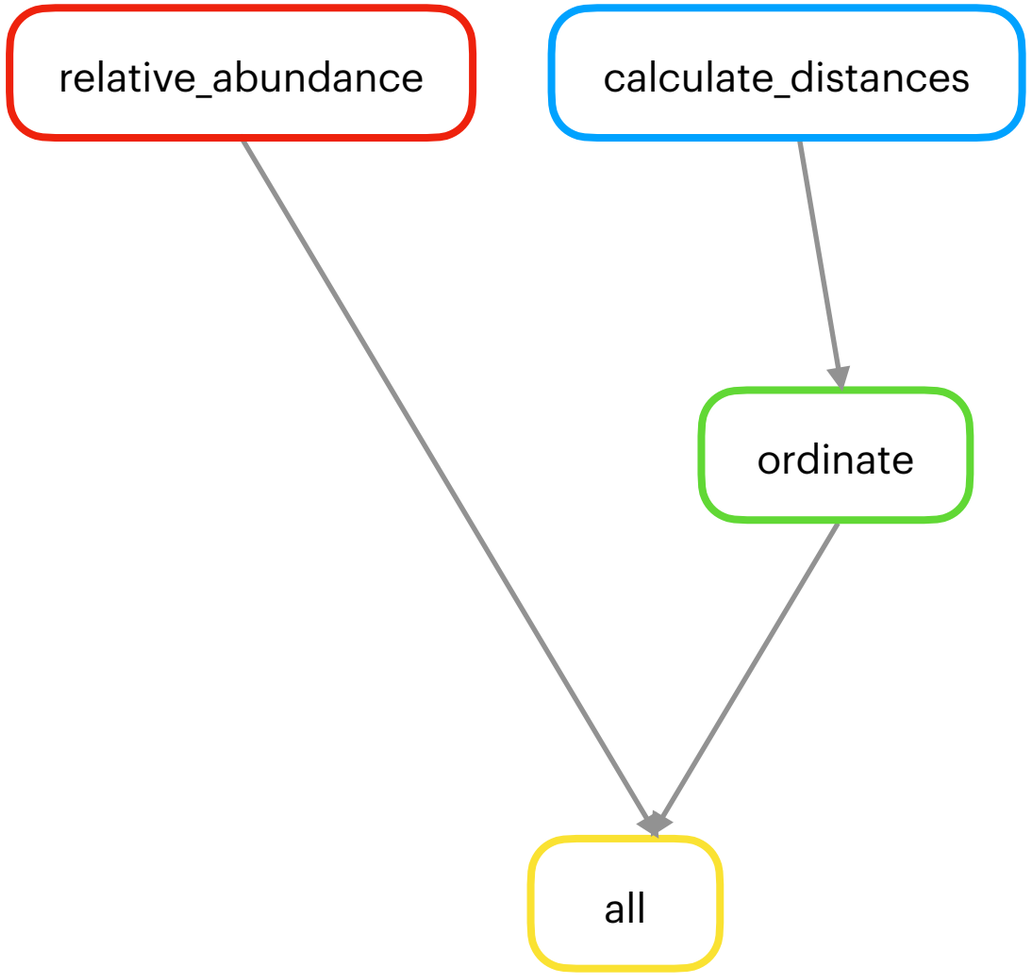

Agenda of the hands-on

- Activate the environment
- Open the **handson/** folder in your preferred editor (mine is visual studio code)
- Look at the folder structure
- Open the Snakefile
- First rule: **relative_abundances**
  - Try a dry run
  - The config.yaml file
- Second rule: **calculate distances**
  - Wildcards and the expand function
- Third rule: **ordinate**
- Run it all
- Check logs, benchmarks
- Create a report



Try yourself!

- Can we shorten the script for the rule relative abundances that takes a list of outcomes of interest? How?
- Add a new outcome parameter to the ordination plots
- Change the threshold_abundance parameter and rerun snakemake. Which rules are affected?
- Generate a rulegraph
- Generate a dag
- Generate a report with all the outputs



References

This introduction https://github.com/crlero/snakemake-tutorial

These slides in NAS /Volumes/UserFolders/cristina.leal/snakemake-tutorial/

Paper Köster, Johannes, and Sven Rahmann. "Snakemake—a scalable bioinformatics workflow engine." Bioinformatics 28.19 (2012): 2520-2522.

Snakemake official docs https://snakemake.readthedocs.io/

Snakemake official tutorial https://snakemake.readthedocs.io/en/stable/tutorial/tutorial.html#tutorial

Best practices https://snakemake.readthedocs.io/en/stable/snakefiles/best_practices.html#snakefiles-best-practices

YOUTUBE

https://www.youtube.com/watch?v=_dG9b3a9zkk

https://www.youtube.com/watch?v=NNPBDOBHlxo&t=655s