UNIVERSITÄT BERN

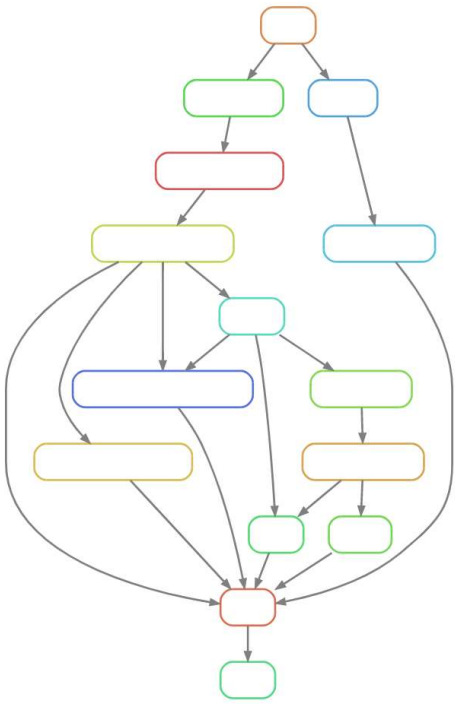
An Introduction to Snakemake

Irene Keller, Department for BioMedical Research DBMR

10.05.2019 Vetsuisse

Swiss Institute of Bioinformatics

Workflows in bioinformatics



Option 1:
Run each step separately

Option 2:
Write a shell script that executes the different steps

Option 3:
Use a workflow management tool, e.g.

- Common workflow language CWL
- Snakemake

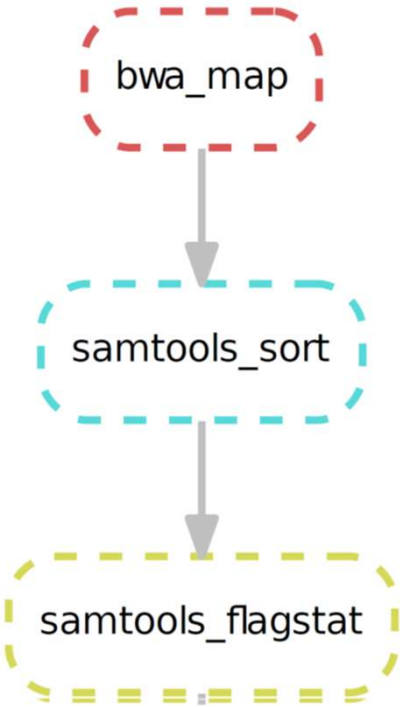
Snakemake

UNIVERSITÄT BERN

- Goal is to create **reproducible** and **scalable** analyses
- Workflows can run in server, cluster, grid and cloud environments
- Can automatically deploy required software dependencies of a workflow using Conda or Singularity
- Python-based language

- Introductory slides: https://slides.com/johanneskoester/snakemake-short#/
- Tutorial: https://snakemake.readthedocs.io/en/stable/tutorial/tutorial.html#tutorial
- Original publication: Köster, Johannes and Rahmann, Sven. “Snakemake - A scalable bioinformatics workflow engine”. Bioinformatics 2012.

Our example workflow



Data are for individuals HG002, HG003 and HG004 (AJ Trio, 1000 Genomes Project)

For the sake of speed, the fastq files and the reference genome sequence contain only the first 1MB of chromosome 1.

Exercises

1) Map 1 sample
2) Extend mapping to multiple samples and add a target rule
3) Add additional steps: Sorting, indexing and flagstat
4) Add a config file
5) Communicate with Slurm

General principle

- The workflow is defined in a file called «**Snakefile**»
- This contains **rules** describing how to produce the output files from the input files
- We specify the final output, and Snakemake will automatically determine in which order the rules have to be executed

```
rule bwa_map:
    input:
        "data/genome.fa",
        "data/samples/A.fastq"
    output:
        "mapped_reads/A.bam"
    shell:
        "bwa mem {input} | samtools view -Sb - > {output}"
```

General principle

```
rule bwa_map:
    input:
        "data/genome.fa",
        "data/samples/A.fastq"
    output:
        "mapped_reads/A.bam"
    shell:
        "bwa mem {input} | samtools view -Sb - > {output}"
```

ref="data/genome.fa",
fastq="data/samples/A.fastq"

{input.ref} {input.fastq}

General principle

UNIVERSITÄT BERN

```
snakemake --dryrun --printshellcmds mapped/myfile.bam
```

Do test run

Print the command that is run

The final output produced by the workflow

Exercises

UNIVERSITÄT BERN

Fastq files are here: /data9/projects/Snakemake_Tutorial/reads

Reference fasta and bwa index files: /data9/projects/Snakemake_Tutorial/reference

Please work here: /data9/projects/Snakemake_Tutorial/working

If you are stuck, solutions are here: /data9/projects/Snakemake_Tutorial/solutions

Exercises

I suggest you open a shell in interactive mode for the first 4 exercises so that we can ignore Slurm for now:

```
srun --partition=pcourse80 --time=3:0:0 --mem 10G --pty /bin/bash
```

Exercise 1

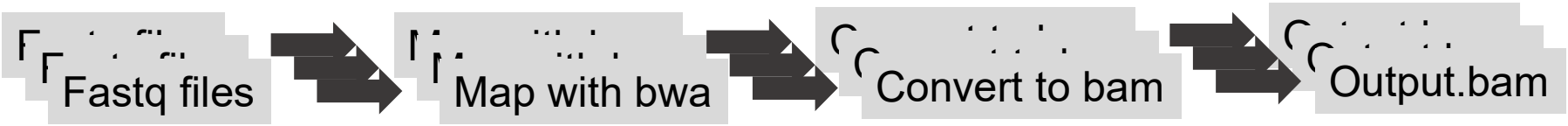
UNIVERSITÄT BERN

Map reads for HG002 using bwa

```
module add UHTS/Aligner/bwa/0.7.17;
module add module add UHTS/Analysis/samtools/1.8;
bwa mem <reference> <R1.fastq.gz> <R2.fastq.gz> | samtools view -bh - > <outputfile>
```

Fastq files → Map with bwa → Convert to bam → Output.bam

Exercise 2 – Extend to multiple samples



- Generalise rules by including **wildcards**

```
rule bwa_map:
    input:
        "data/genome.fa",
        "data/samples/{sample}.fastq"
    output:
        "mapped_reads/{sample}.bam"
    shell:
        "bwa mem {input} | samtools view -Sb - > {output}"
```

Exercise 2 – Add a target rule

UNIVERSITÄT BERN

snakemake ~~mapped/myfile.bam~~

Snakefile

```
rule all:
    input:
        "mapped/myfile.bam"
```

Exercise 2 – Add a target rule

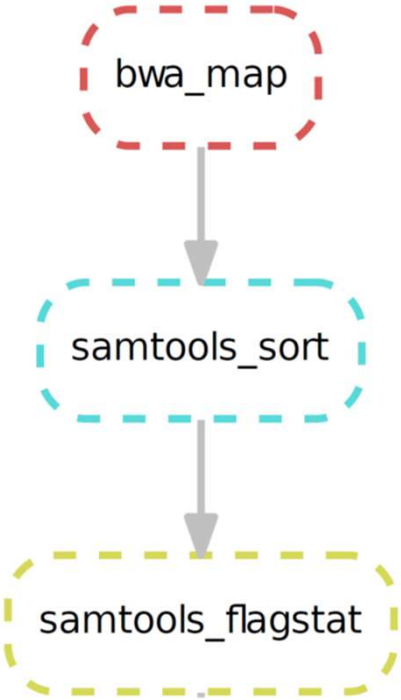
Snakefile

```
sample = ["HG002", "HG003", "HG004"]

rule all:
    input:
        expand("mapped/{sample}.bam", sample=sample)
```

["mapped/HG002.bam", "mapped/HG003.bam", "mapped/HG004.bam"]

Exercise 3 – Add additional steps



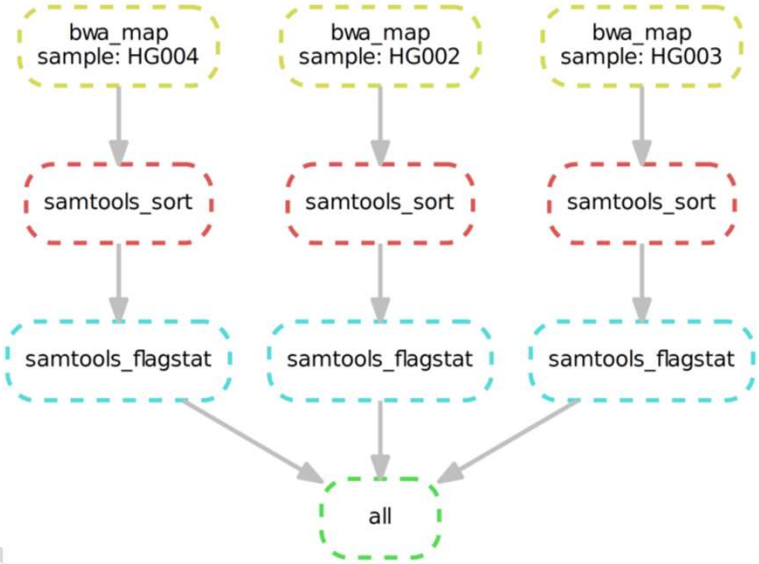

```
module add UHTS/Analysis/samtools/1.8;
samtools sort -o <out.bam> <in.bam>;
samtools index <out.bam>"
```

```
module add UHTS/Analysis/samtools/1.8;
samtools flagstat <in.bam> > <out.txt>
```

Visualise the DAG

- From input and output files of each rule, Snakemake knows how the rules have to be connected
- We can visualise the directed acyclic graph (DAG) of our workflow:

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



Use a configuration file

- Configuration file to make workflow more flexible / to make it clear what has to be adjusted for different analysis runs
- json or yaml format

Snakefile

```
configfile: "path/to/config.json"
```

config.json

```
{
        "Var1": "MyValue",
        "Var2": "MyOtherValue"
}
```

- Values from config file will be available in a dictionary called config and can be accessed like this

```
config["Var1"]
```

Rule parameters

- We can have a **params** directive in addition to input, output and shell
- Can be extremely useful if parameters need to be set that depend on wildcards:

```
rule bwa_map:
    input:
        "data/genome.fa",
        lambda wildcards: config["samples"][wildcards.sample]
    output:
        "mapped_reads/{sample}.bam"
    params:
        rg=r"@RG\tID:{sample}\tSM:{sample}"
    threads: 8
    shell:
        "bwa mem -R '{params.rg}' -t {threads} {input} | samtools view -Sb - > {output}"
```

- Or simply to make definition of parameters more explicit

Exercise 4

UNIVERSITÄT BERN

Modify your workflow from exercise 3 as follows:

- Put samtools and bwa version in a config file
- Load this file from your Snakefile
- Add to rules via the params section

Submit to Slurm

UNIVERSITÄT BERN

- We can specify the number of threads as well as other resources in each rule by adding new sections:

```
threads:
        1
resources:
        mem_mb=1000,
        hours=1
```

- Submit the job to Slurm as follows:

/opt/cluster/software/Conda/miniconda/3/bin/snakemake --printshellcmds --drmaa " --ntasks=1 --mem={resources.mem_mb} --cpus-per-task={threads} --time={resources.hours}:0" --latency-wait 300 --jobs 1 --jobname <myJobName>_{jobid}

Important: There must be a whitespace after the quote

How many jobs should be submitted in parallel?

Exercise 5

UNIVERSITÄT BERN

- Modify your Snakefile from exercise 4 to include threads, memory and time requirements for each rule
- Submit to Slurm